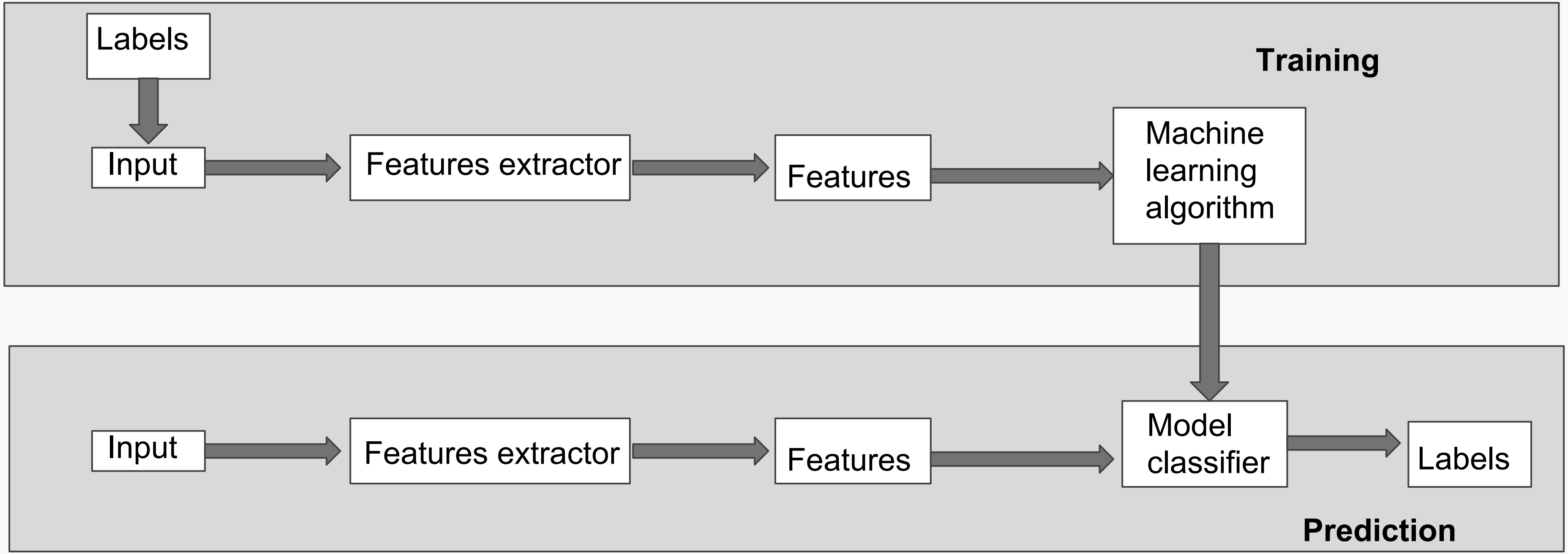
Introduction to Machine Learning

Machine learning gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959

Workflow


Labels
Training
Input
Features extractor
Features
Machine learning algorithm
Input
Features extractor
Features
Model classifier
Labels
Prediction

Categories

- Supervised learning
- Unsupervised learning
- Semi-supervised learning

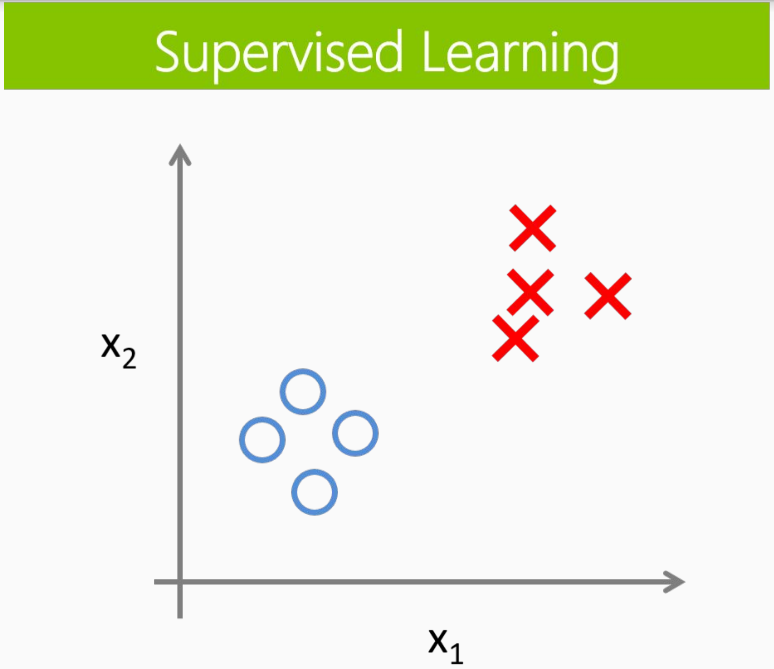
Supervised learning

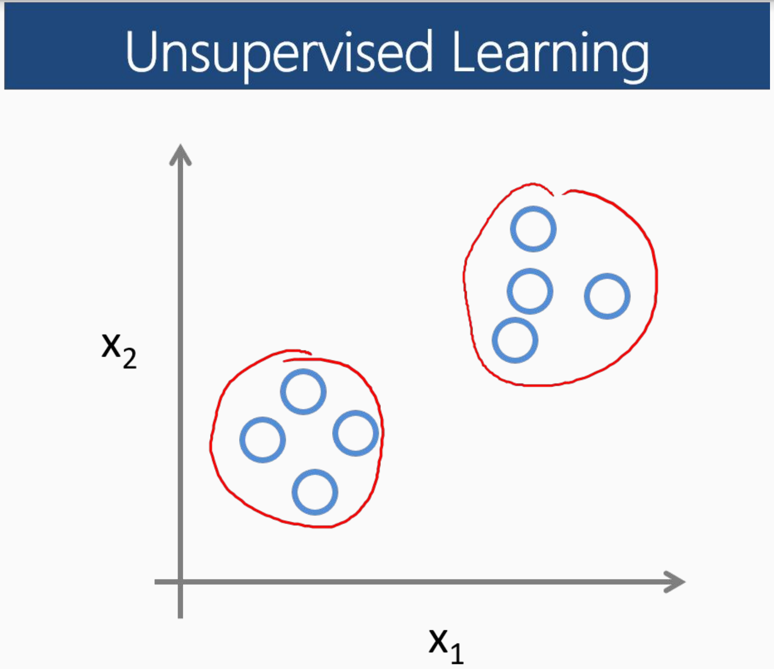
- A dataset X comes with labels y
- Training: fit a function f such as f(X) = y
- Prediction: for a new data X', we predict f(X')
- We looked for a f which minimizes a metric error M
- Model selection can be automated
- Sentiment analysis, churn prediction, credit risk

Unsupervised learning

- A dataset X with no label
- Find hidden pattern in the structure of the data
- Clustering, anomaly detection, similarity detection
- No error metric, manual verification

Supervised VS unsupervised





Semi-supervised learning

- A dataset X with label y and a dataset X’ with no label
- Perform supervised learning on X and use information of X’ to improve the supervised model

Regression VS Classification

Regression:

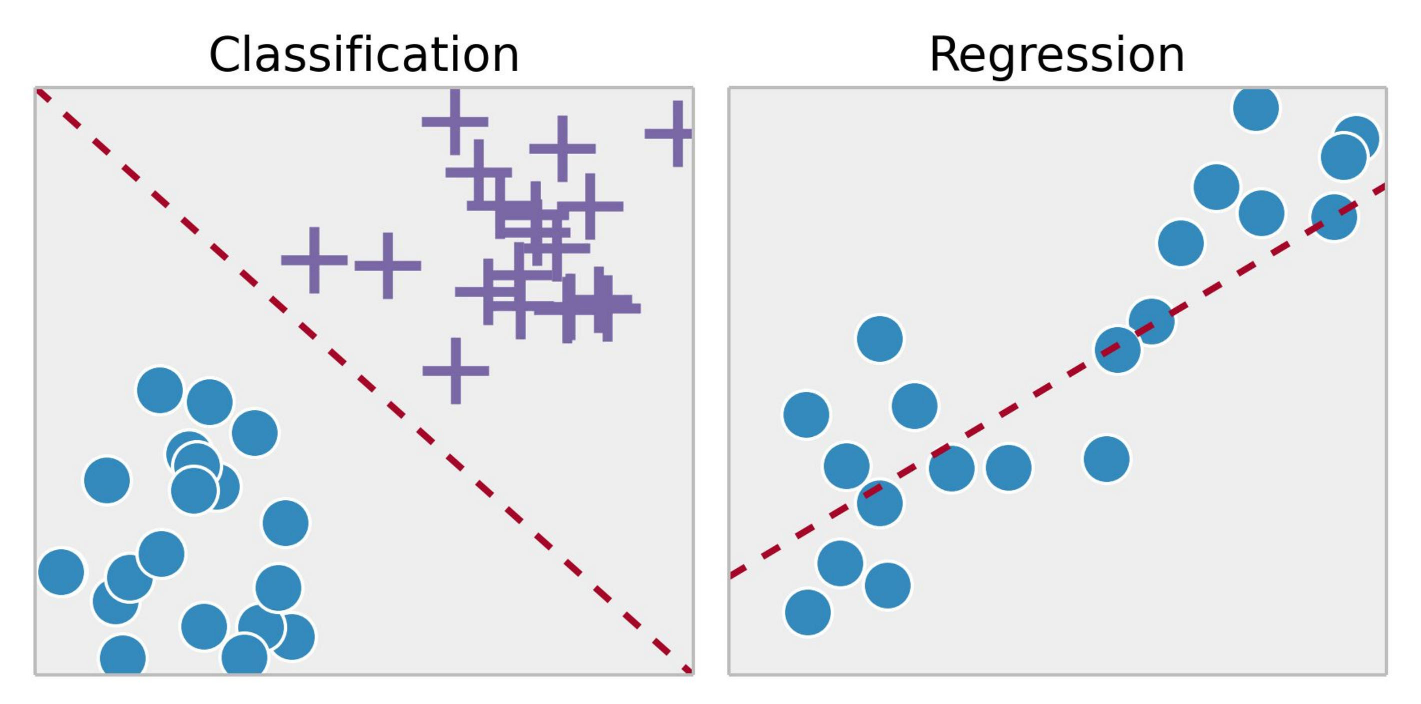
- The output are real number/continuous values
- For an input x, we predict a number n
- Example: price house

Regression VS Classification

Classification:

- The output are class
- For an input x, we predict the class y it belongs to
- Example: sentiment analysis, typed

Regression VS Classification



How do we train a model

Case: single variable linear regression

Input: size of the house $x_1, \dots, x_n$ with the associated price $y_1, \dots, y_n$

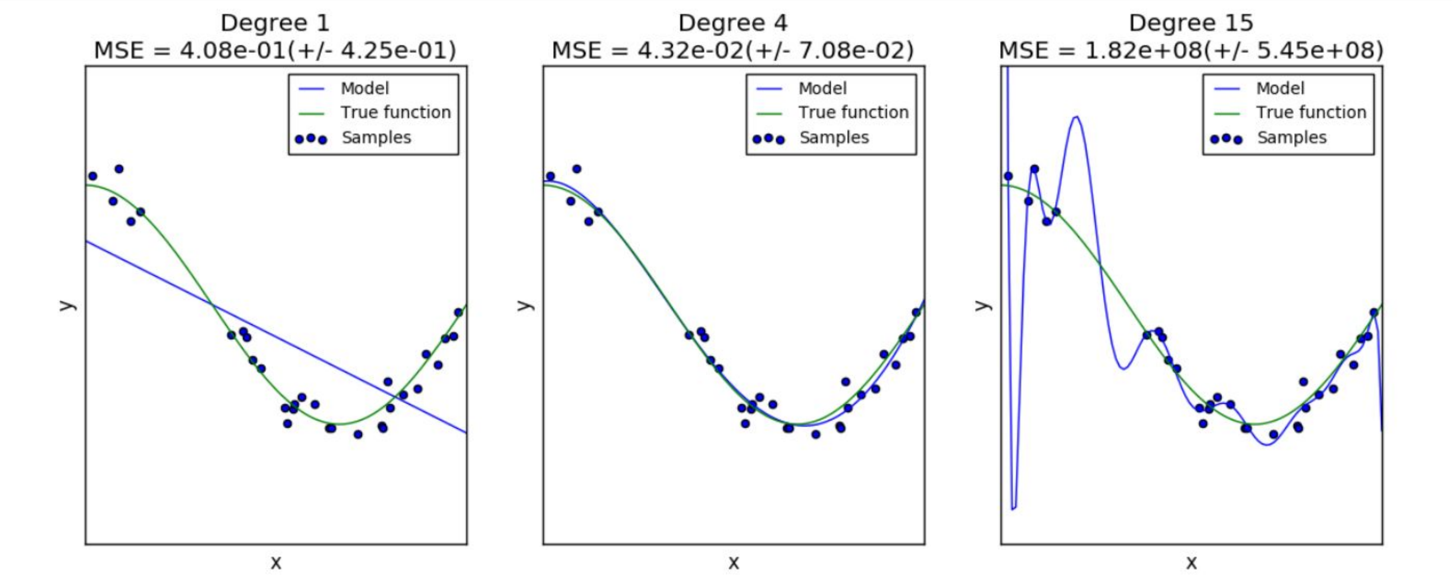
We look for a price function $f(x)=\alpha.x+\beta$ such as $f(x_i)$ as close as possible of y_i

In other word we want to find α and β which minimizes $(\alpha.x_i+\beta - y_i)^2$

So we want to reduce the **cost function** $J(\alpha, \beta) = \sum_{i=1..n}(\alpha.x_i+\beta - y_i)^2$

How do we train a model

Is finding the model with the minimal cost function enough?




I OVERFIT MY
DATA
NOW I DON'T FIT IN
ANYWHERE
memegenerator.net

How to avoid overfitting

- Add regularization parameter(s) to the model
  - $J(\alpha, \beta) = \sum_{i=1..n}(\alpha.x_i + \beta - y_i)^2$ would become $J(\alpha,\beta;\lambda) = \sum_{i=1..n}(\alpha.x_i + \beta - y_i)^2 + \lambda.\alpha^2$

- Optimize hyperparameters
- Reduce the number of features

How to avoid overfitting

- Add regularization parameter(s) to the model
  - $J(\alpha, \beta) = \sum_{i=1..n}(\alpha.x_i + \beta - y_i)^2$ would become $J(\alpha,\beta;\lambda) = \sum_{i=1..n}(\alpha.x_i + \beta - y_i)^2 + \lambda.\alpha^2$

- Optimize hyperparameters
- Reduce the number of features

How to select a model

We need to determine:

- The best regularisation parameters
- The best hyperparameters
- The best number of features
- The best size of the data set

And then the best algorithm once each is optimized

How to select a model

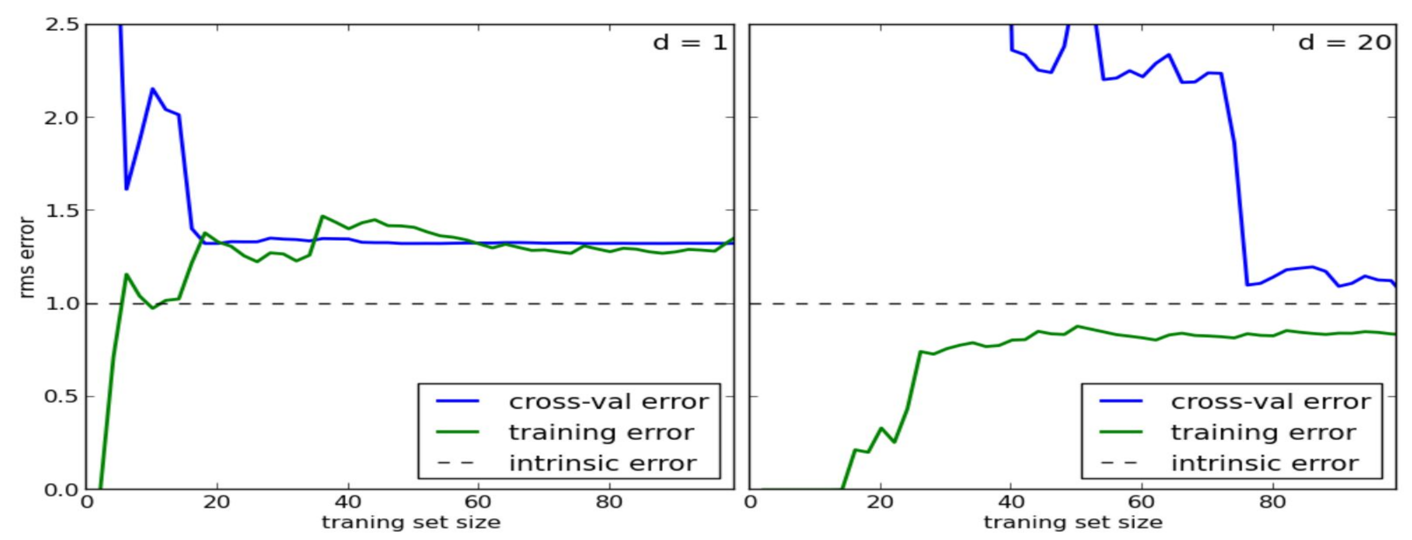
1) Split the data set into training/validation/test sets
    a) The training set is used to train the model
    b) The validation set is used to validate the model and parameters and must not be used to train the model
    c) The test set is used to have a benchmark, it must not be used to train the model or validate the parameter
    d) We keep the model which optimizes the error metric over the validation set

How to select a model

2) Perform cross validation (k-folds):
    a) Split the the data set into a training and a test set
    b) Split the training set into k folds
    c) For i in 1..k, train the model with all the folds but i and measure the prediction error on the fold i.
    d) Average the error over the k folds
    e) Keep the model which optimizes the average error

How to select a model

Learning curves



How to select a model

- Cross-validation more robust but more time consuming
- Ideally we should try all the parameters combinations and keep the best (grid search)
- Can be very complex, so we may only test a random subsets of the combinations
- Some algorithms might help to optimize the hyperparameter search

Type of models

- Linear models
- Tree based models
- Neural network models
- Blending of models: Boosting, Bagging and Stacking

Linear models

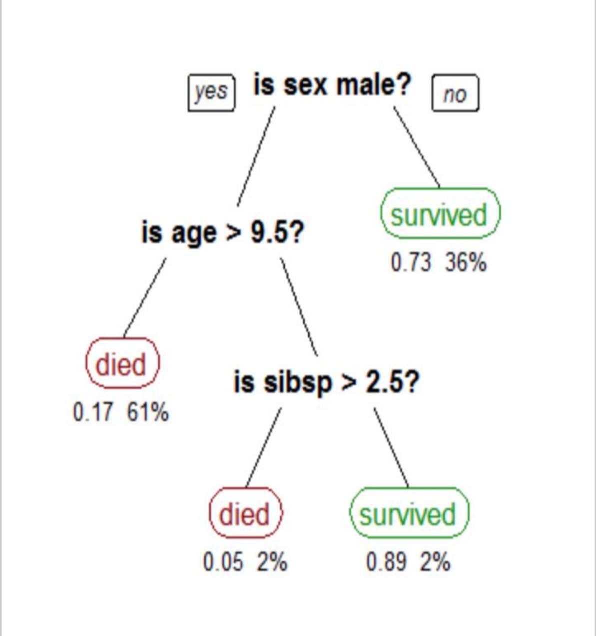
- Output is a linear combinations of the features

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- Non linear effect might be added using interactions and polynomial features

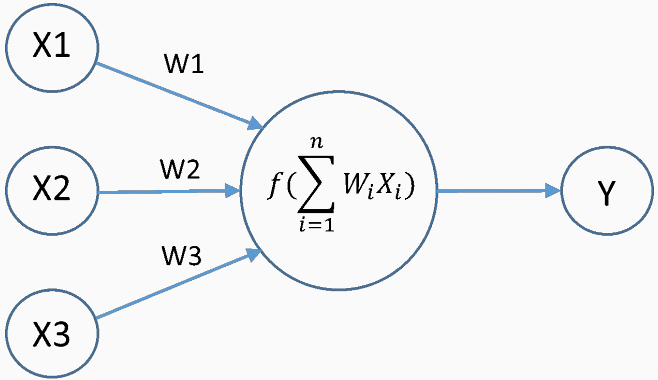
Tree based models

- Model complex and non-linear interactions
- Recursively find the best split
- The leafs give the results



Neural network models

- Model complex interaction
- Architecture inspired on human brain neural networks
- Can be very complex



Blend of models

- Bagging: averaging results of multiples models randomly different (ex: random forest)

For a new input x:

$tree_1(x)=y_1$

……

$tree_n(x)=y_n$

$$RandomForest(x)=\sum_{i=1..n} y_i \,/\, n$$

Blend of models

- Boosting: Succession of weak learner to build a strong one (ex: Gradient Boosting Machine)

$Model_1(x)=y + error_1$
$Model_2(x)=error_1 + error_2$
…
$Model_n(x)=error_{n-1}+ error_n$

$Model(x)=\sum_{i=1..n}Model_i(x)$

Blend of models

- Stacking: use several layers of models. The features of the layer n being the outputs of the layer n-1

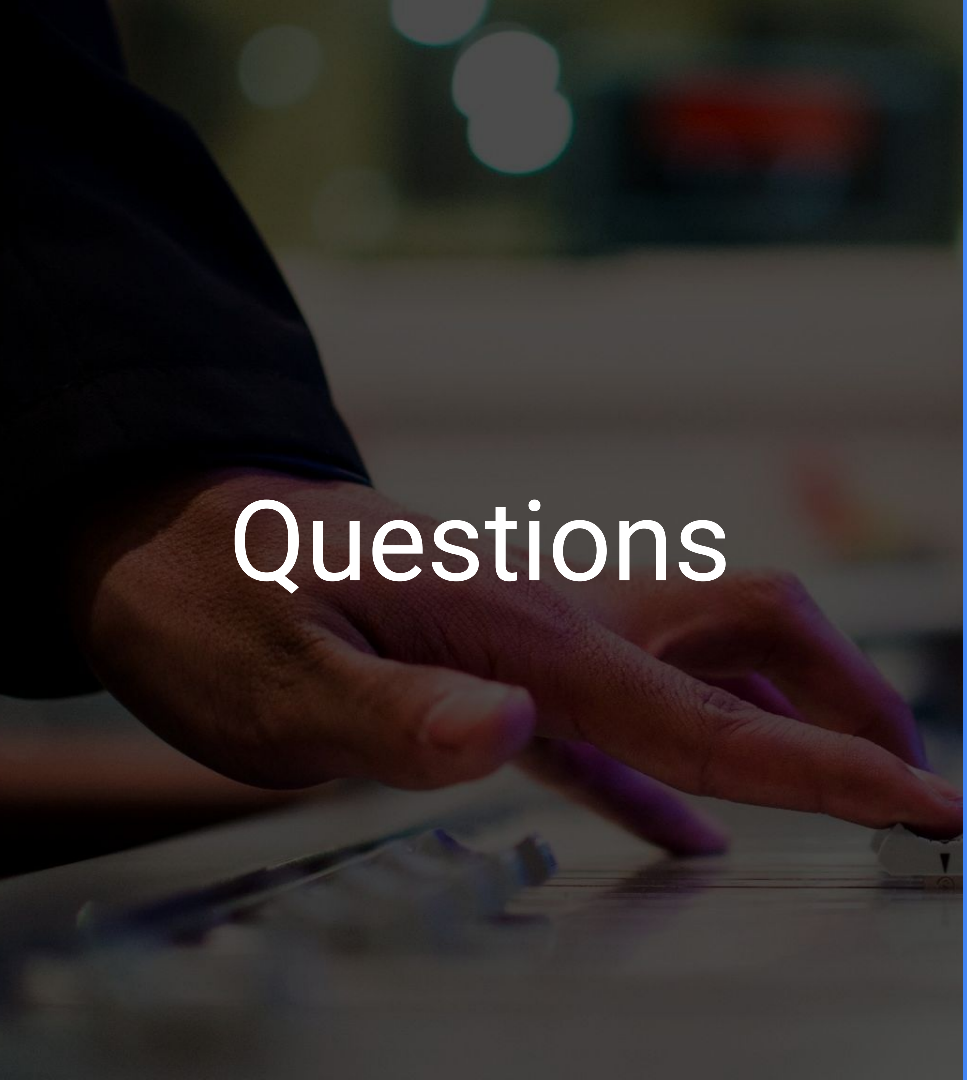
Layer 1:
- Regression logistique (RL)
- Random Forest (RF)
- Gradient Boosting Machine (GBM)

$$Y = NN(RL(x), RF(x), GBM(x))$$

Layer 2: Neural Network (NN)

Ressources

- MOOC: *Machine Learning* by Andrew NG on Coursera
- Kaggle: *Getting Started* and *Playground* competitions and Forums
- *Elements of Statistical Learning* by *Hastie, Tibshirani and Friedman*
- Scikit-learn.org



Questions