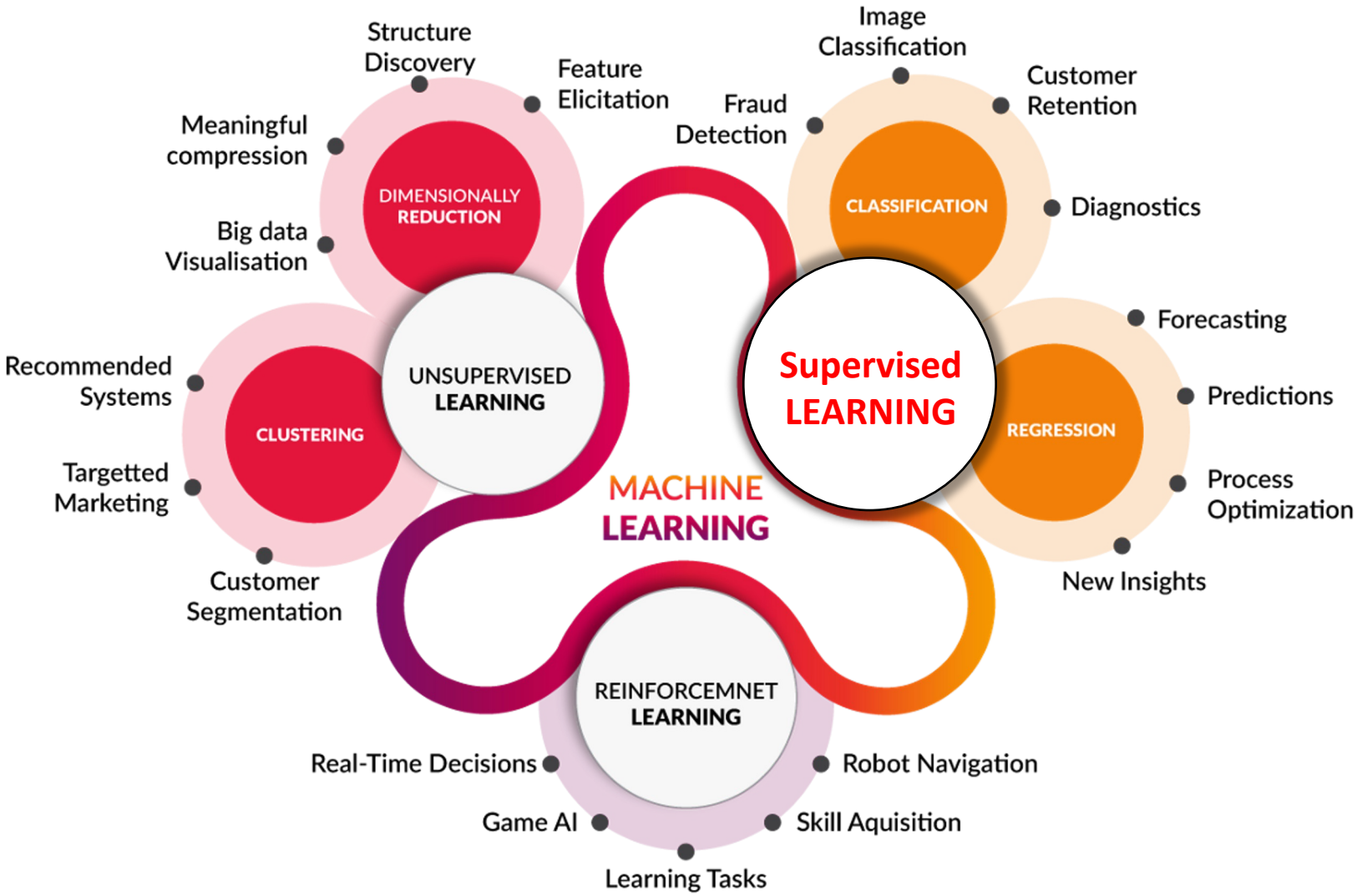
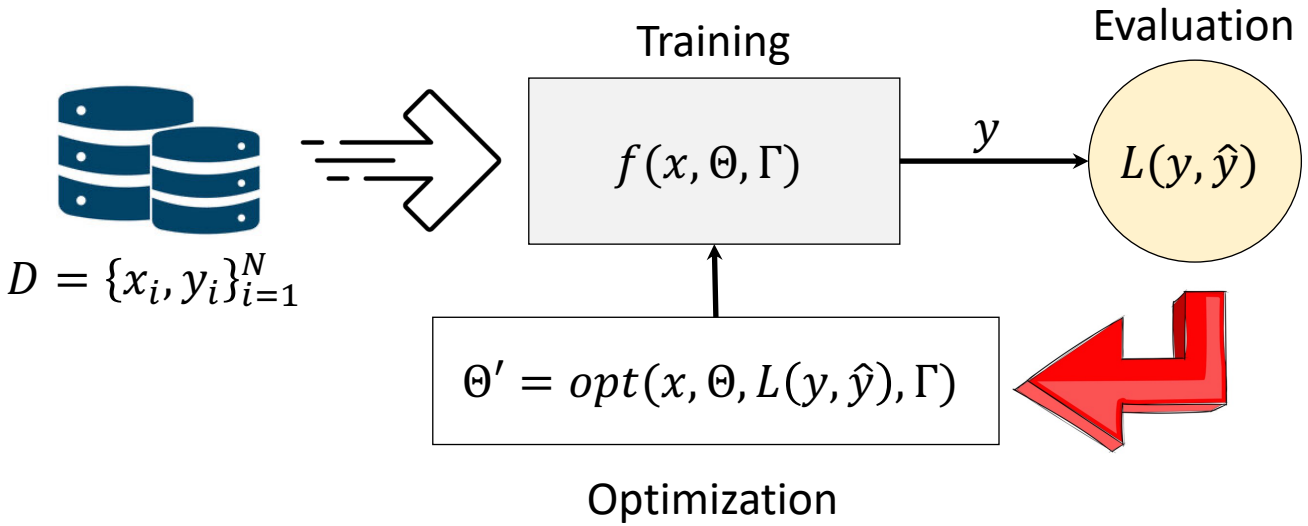

MACHINE LEARNING
UNSUPERVISED LEARNING
DIMENSIONALLY REDUCTION
Structure Discovery
Feature Elicitation
Meaningful compression
Big data Visualisation
CLUSTERING
Recommended Systems
Targetted Marketing
Customer Segmentation
Supervised LEARNING
CLASSIFICATION
Image Classification
Customer Retention
Fraud Detection
Diagnostics
REGRESSION
Forecasting
Predictions
Process Optimization
New Insights
REINFORCEMNET LEARNING
Real-Time Decisions
Robot Navigation
Game AI
Skill Aquisition
Learning Tasks

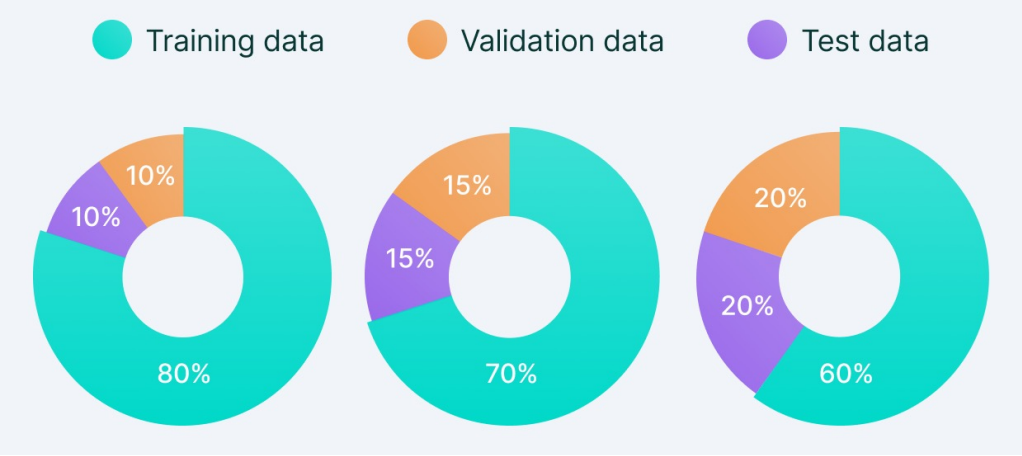
Introduction

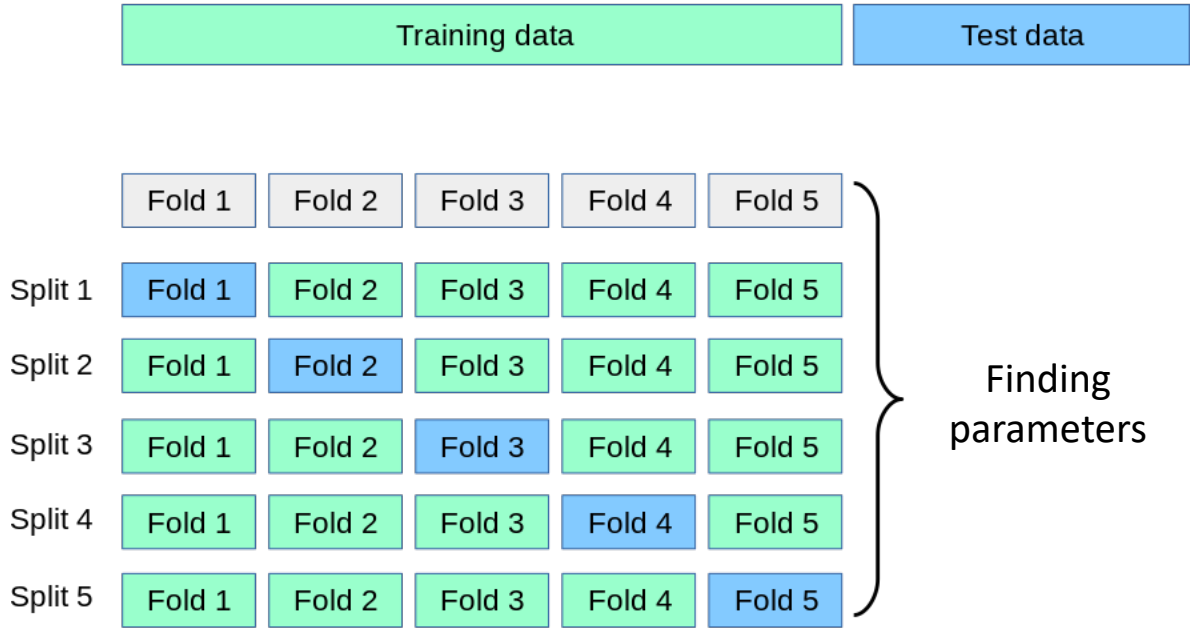
Supervised learning ingredients:

- Dataset where $x_i \in \mathbb{R}^d, y_i \in \mathbb{R}^1$.
- A set of parameters Θ to optimize.
- A set of hyper-parameters Γ.
- A loss function $L(y, \hat{y})$
- An optimizer $opt(\cdot)$



Introduction: splits





Introduction: metrics

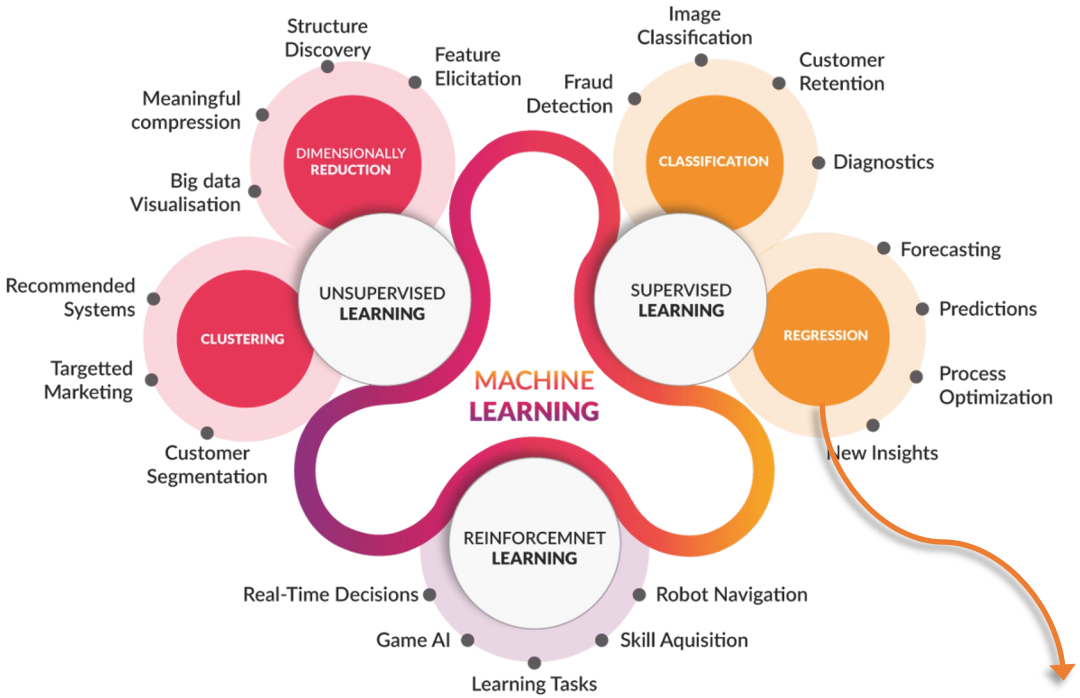
Regression metrics:

- $R^2 = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - y_{avg})^2}$
- $MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$
- $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
- $MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{\max(\epsilon, |y_i|)}$
- $Max = \max_{i=\{1,\dots,N\}}|y_i - \hat{y}_i|$

Regression metrics:

- $ACC = \frac{1}{N}\sum_{i=1}^{N} 1(y_i = \hat{y}_i)$

		Ground truth +	Ground truth -	
Predicted	+	True positive (TP)	False positive (FP)	Precision = TP / (TP + FP)
Predicted	-	False negative (FN)	True negative (TN)	
		Recall = TP / (TP + FN)		



Regression

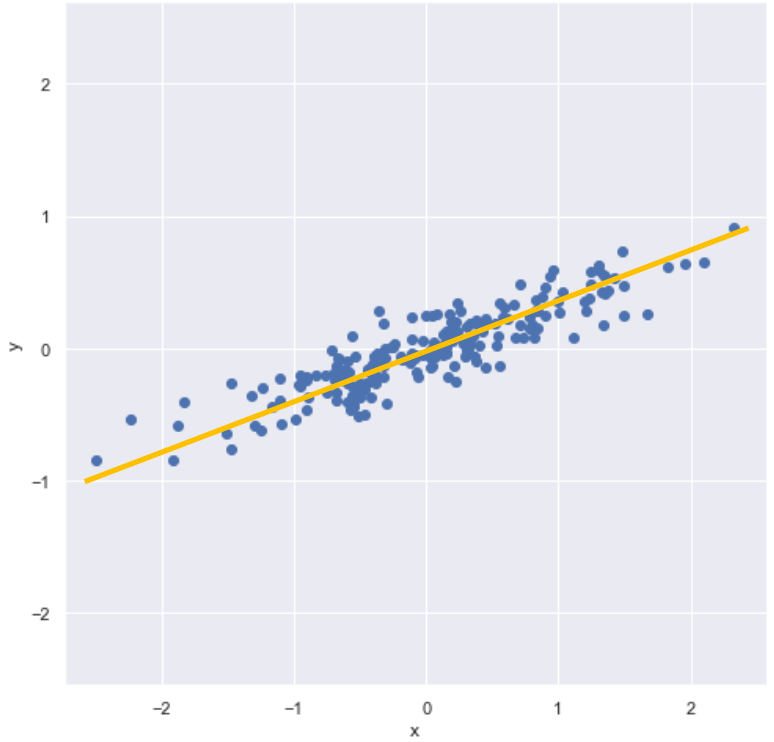
Linear Regression, Decision tree, and Ensembles

Linear Regression

Linear regression is a supervised learning algorithm used to predict a quantitative response. It assumes that there is a **linear relationship** between the feature vector x (explanatory variable) and the response y (dependent variable).
In mathematical notation:

$$y(\omega, x) = \omega_0 + \omega_1 x_1 + \cdots + \omega_d x_d$$

Where the vector ω gives the right weight to each feature of the explanatory variable.



- How could we find the right weights?
  We can learn the weights from a training dataset with n observations $X \in \mathbb{R}^{n \times d}, Y \in \mathbb{R}^{n}$.

Ordinary Least Squares (OLS)

OLS fits a linear model with weights ω to minimize **the residual sum of squares (RSS)** between the observed targets in the dataset, and the targets predicted by the linear approximation. Mathematically, it solves a problem of the form:

$$\min_{\omega} \|X\omega - Y\|_2^2$$

- The weight estimates for OLS rely on the independence of the features. When features are correlated and the columns of the matrix X have an approximately linear dependence, the least-squares estimate becomes highly sensitive to random errors in the observed target, producing a large variance.

Solutions: Ridge and Lasso

- **Ridge regression** imposes a penalty on the size of the coefficients. The ridge objective minimizes a penalized residual sum of squares:

$$\min_{\omega} \|X\omega - Y\|_2^2 + \alpha\|\omega\|_2^2$$

- **Lasso regression** is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer non-zero coefficients**:**

$$\min_{\omega} \frac{1}{2n} \|X\omega - Y\|_2^2 + \alpha\|\omega\|_1$$

The complexity parameter $\alpha \geq 0$ controls the **amount of shrinkage**: the larger the value of α, the greater the amount of shrinkage and thus the coefficients become more robust to collinearity or more sparse.

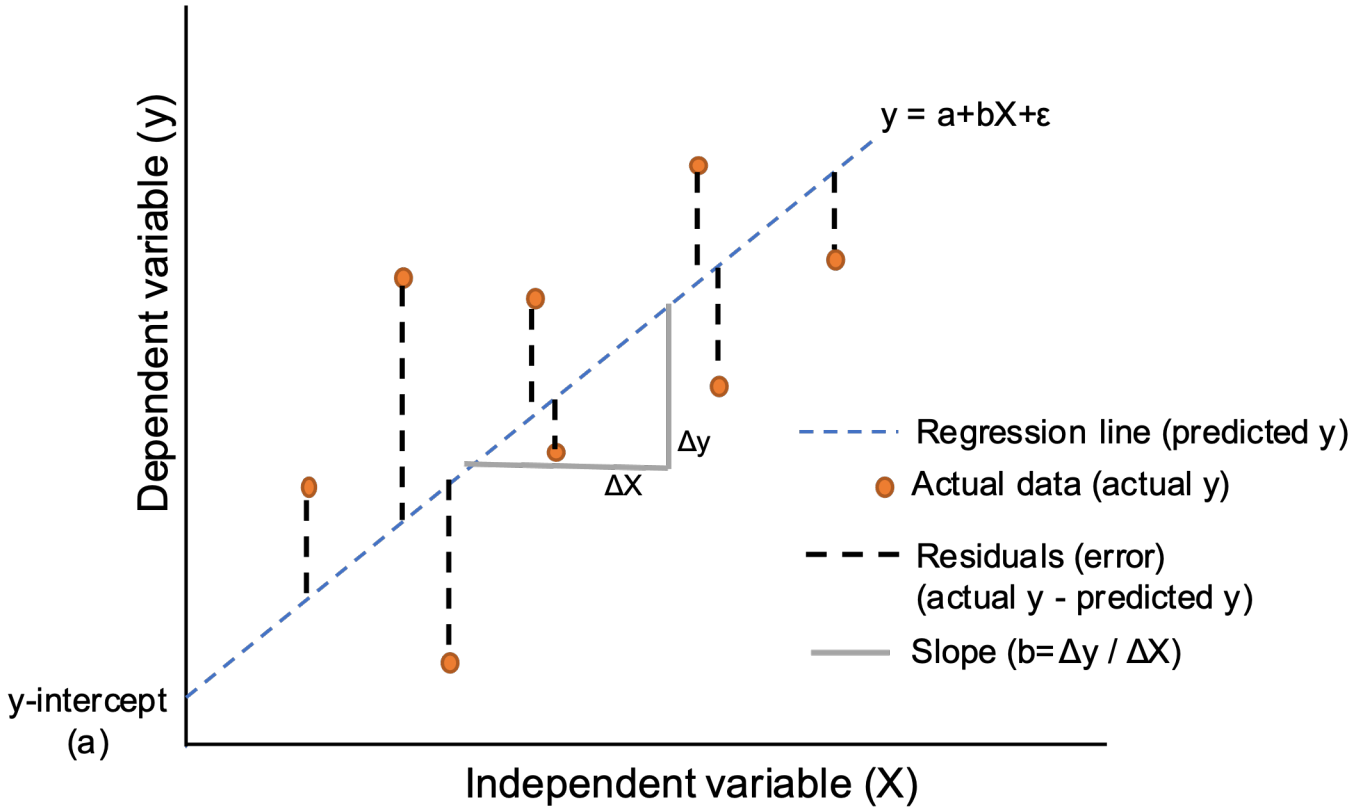
Note: as the Lasso regression yields sparse models, it can be used to perform feature selection (see [Ref. 1]).

Linear Regression: takeaway

- **Linear Assumption** assumes that the relationships between your input and output is linear. When you have a lot of attributes you may need to transform data to make the relationship linear (e.g. log transform for an exponential relationship).
- **Remove Noise**. Linear regression assumes that your input and output variables are not noisy. Consider using data cleaning operations that let you better expose and clarify the signal in your data. Always remember to **remove outliers** in the output variable.
- **Remove Collinearity**. Linear regression will overfit your data when you have highly correlated input variables. Consider calculating pairwise correlations for your input data and removing the most correlated.
- **Rescale Inputs**: Linear regression will often make more reliable predictions if you rescale input variables using standardization or normalization.

- Plot the **residuals** to validate the linear regression assumptions.

Linear Regression: takeaway



Decision tree: how much does a car cost ?

World state observation

Powerful engine?

no

yes

Alloy wheel?

cost = 20000$

no

yes

cost = 10500$

cost = 12000$

Decision tree

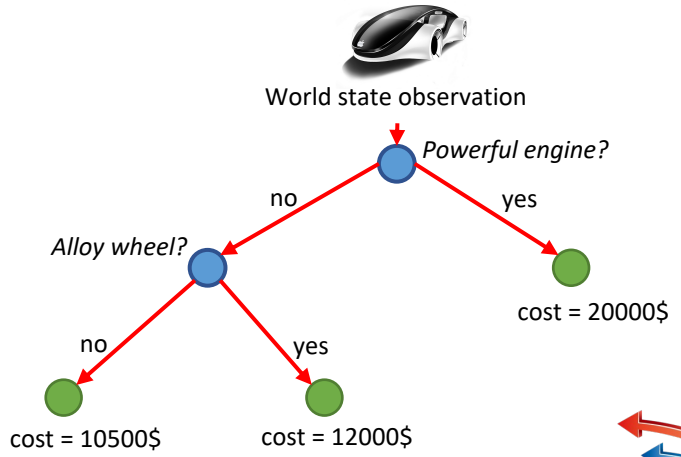
Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression.

Decision Tree models split the data multiple times according to certain cut-off values in the features. Through splitting, different subsets of the dataset are created, with each instance belonging to one subset. This splitting process is repeated on each derived subset in a recursive manner: **recursive partitioning**. The recursion is completed when the subset at a node has all the same values of the target variable, or when splitting no longer adds value to the predictions.

A decision tree is composed by:

- Branch node: represents a choice between a number of observations. How the current set of data is partitioned in two subsets.
- Leaf node: represents a decision. The final set of data.

To predict the outcome in each leaf node, the average outcome of the data is used.

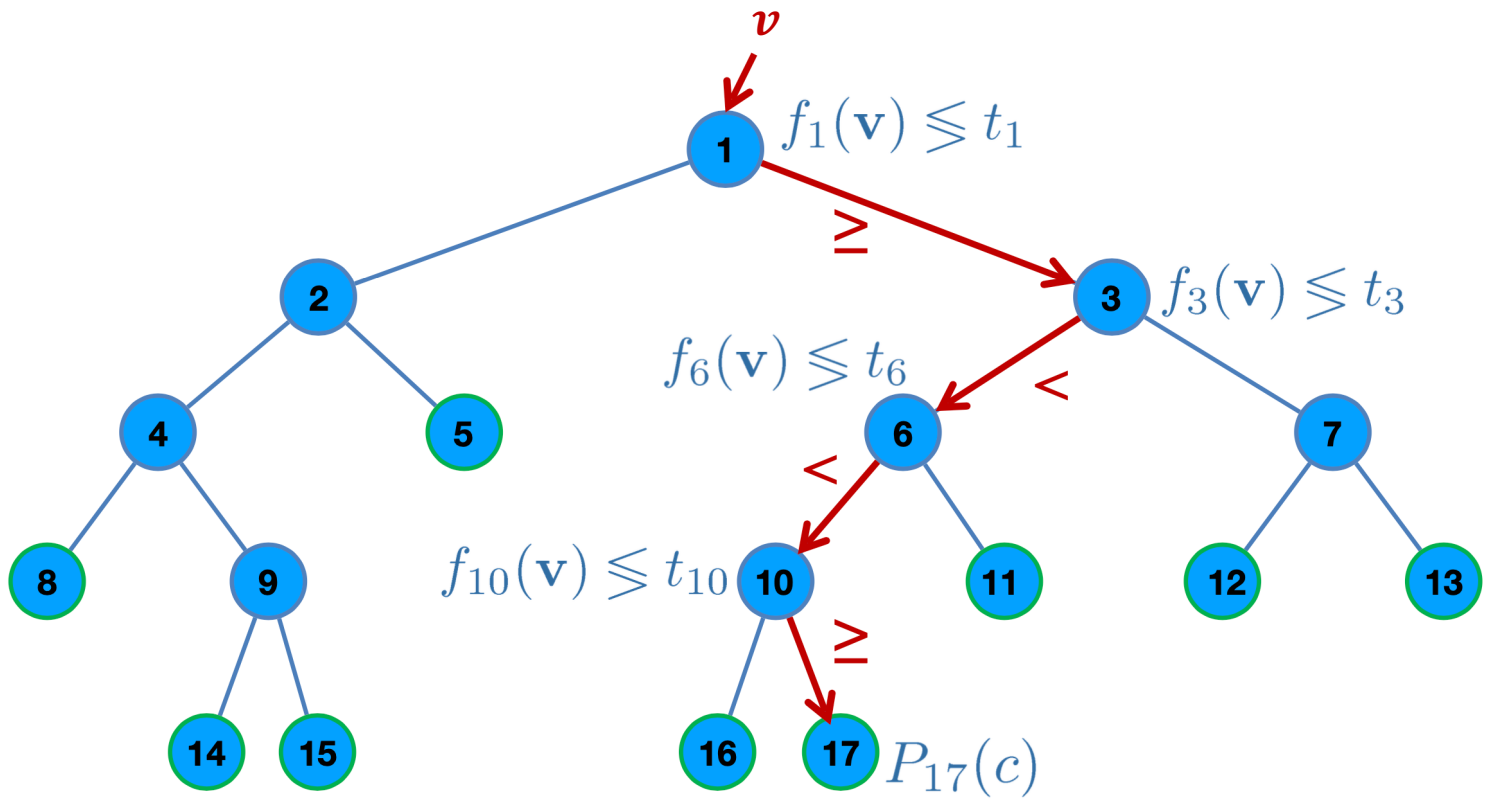


Decision tree

Decision tree nomenclature:

- Feature vector: $v \in \mathbb{R}^N$
- Split function: $f_n(v): \mathbb{R}^N \rightarrow \mathbb{R}$
- Threshold: t_n
- Prediction: P_n

→ How do you choose the right split function and threshold?



Decision tree

1. Choose at random many split functions and thresholds.
   - Normally, one splitting feature is chosen at random from the pool of features.
   - The threshold is chosen at random in the range of variation of the selected feature.
2. Split the data at the current node into two subsets based on the splitting function and threshold:

$$I_l = \{i \in I_n \mid f_n(v_i) < t_n\}$$

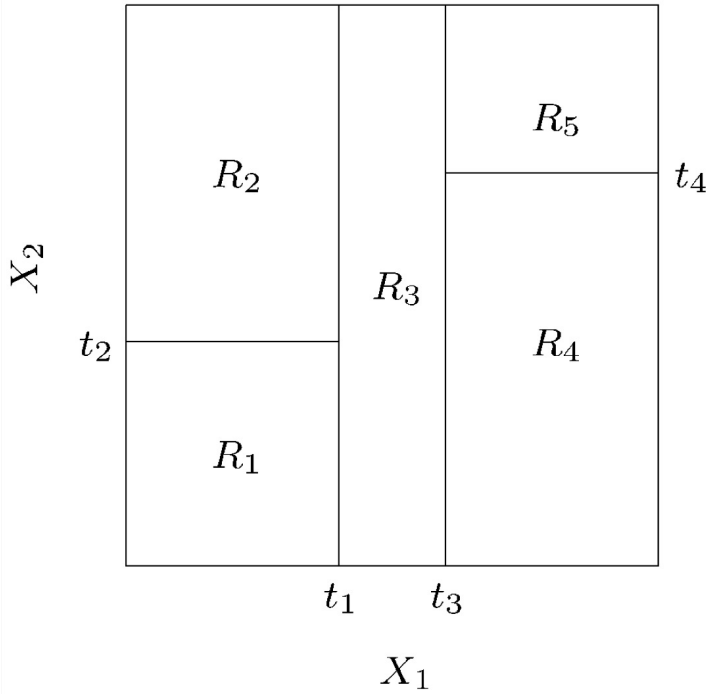
$$I_r = I_n \setminus I_l$$

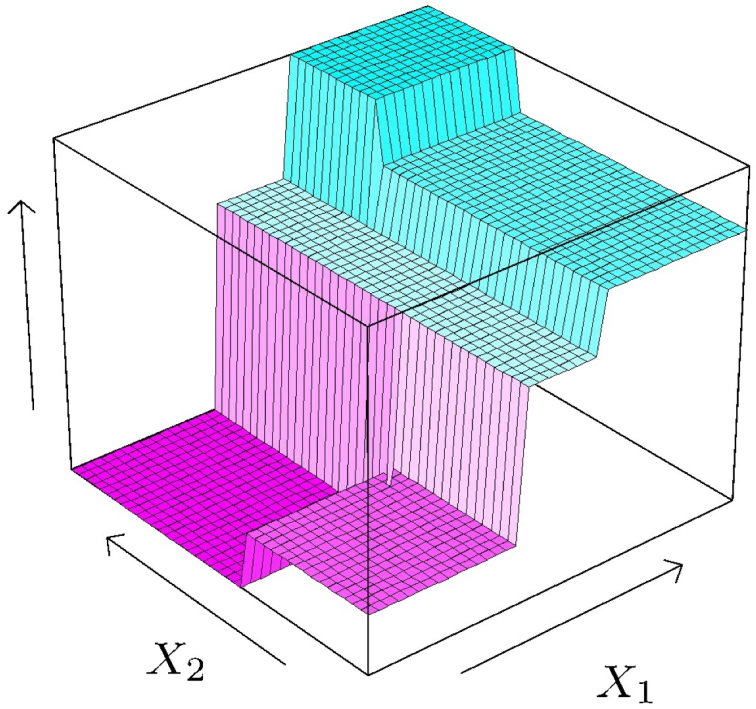
3. Choose f_n and t_n that **maximize the variance reduction**. The variance reduction is defined as the total reduction of the variance of the target variable y due to the split.

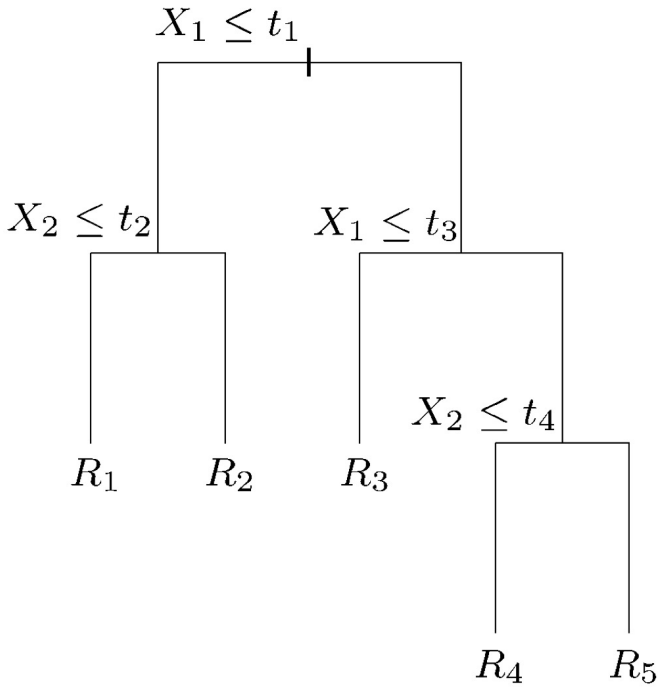
$$VR = \frac{1}{|I_n|^2}\sum_{i\in I_n}\sum_{j\in I_n}\frac{1}{2}(y_i - y_j)^2 - \left(\frac{1}{|I_l|^2}\sum_{i\in I_l}\sum_{j\in I_l}\frac{1}{2}(y_i - y_j)^2 + \frac{1}{|I_r|^2}\sum_{i\in I_r}\sum_{j\in I_r}\frac{1}{2}(y_i - y_j)^2\right)$$

→Each of the above summands are indeed variance estimates written in a form without directly referring to the mean.

Decision tree: 2D example







Decision tree

Advantages:

- **Simple to understand and interpret.** People are able to understand decision tree models after a brief explanation.
- **Able to handle both numerical and categorical data.** Other techniques are usually specialized in analysing datasets that have only one type of variable.
- **Requires little data preparation.** Other techniques often require data normalization.
- **Robust against co-linearity.**

Limitations:

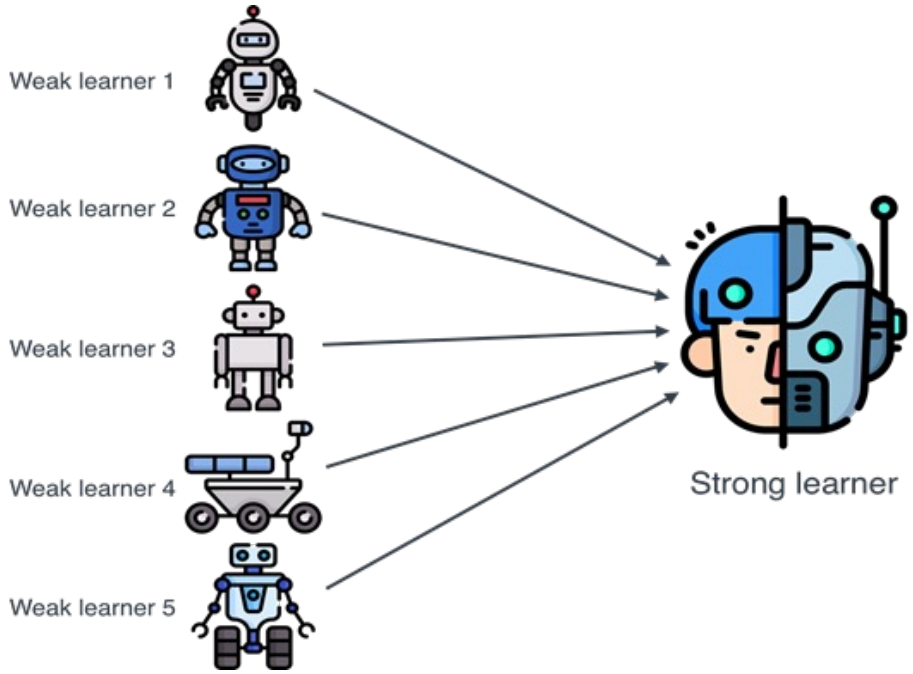
- **Overfitting.** Decision-tree learners can create over-complex trees that do not generalize well from the training data. → Solution: **pruning.**
- Trees can be **very non-robust**. A small change in the training data can result in a large change in the tree and consequently the final predictions.

Ensemble

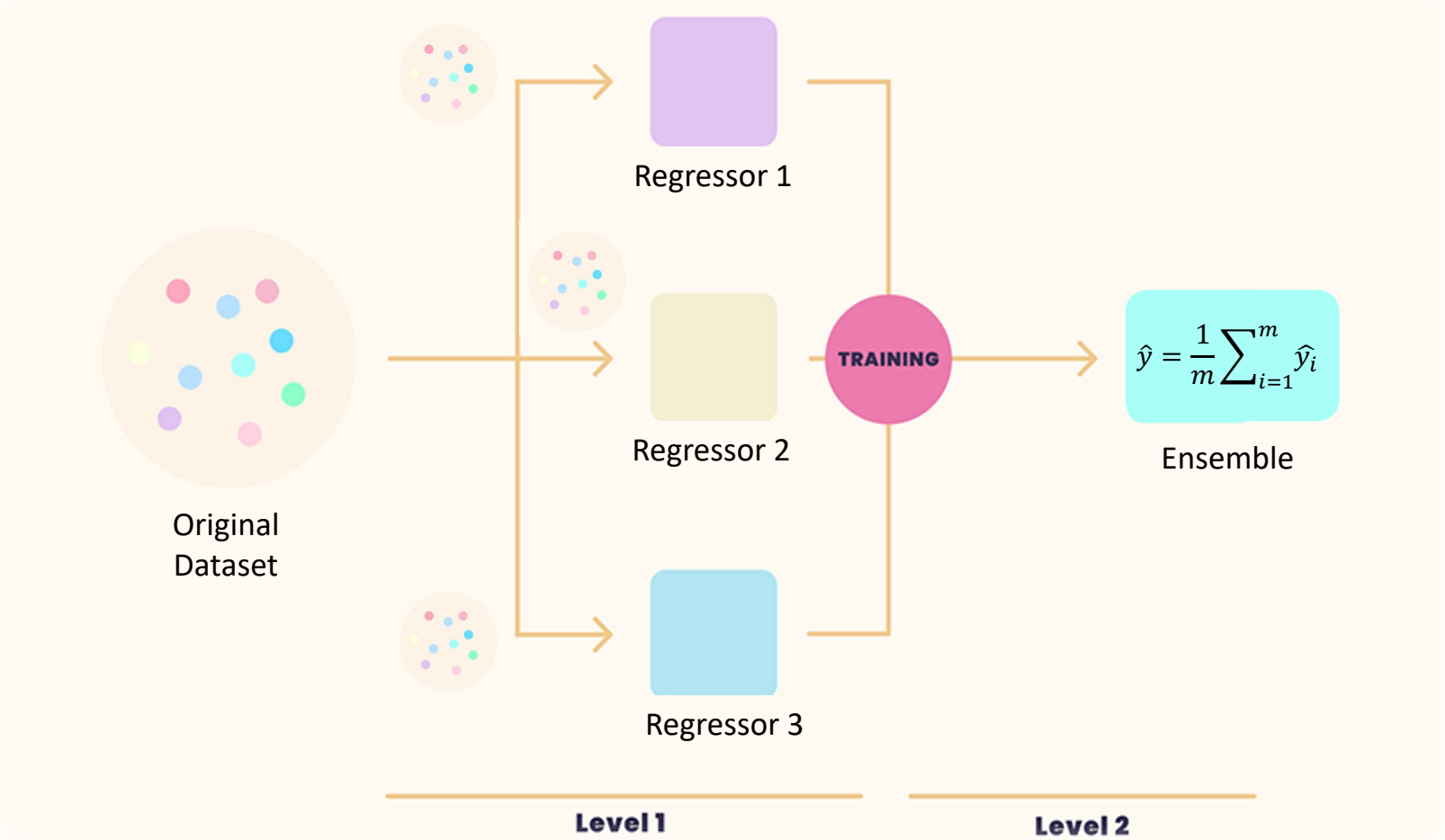
The goal of **ensemble methods** is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Two families of ensemble methods are usually distinguished:

- In **averaging methods**, the driving principle is to build several estimators independently and then to average their predictions.
- By contrast, in **boosting methods**, base estimators are built sequentially, and one tries to reduce the bias of the combined estimator.



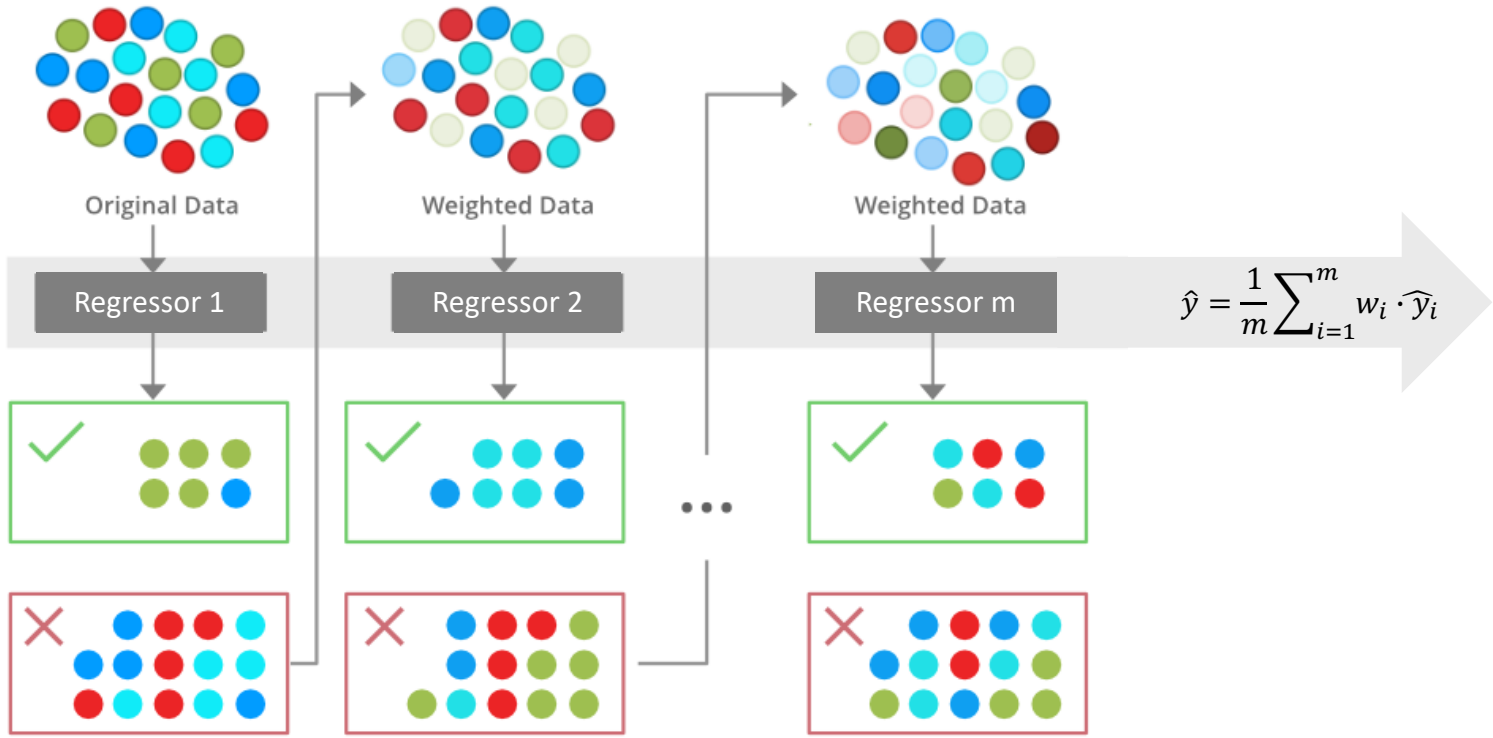
Ensemble: bagging



Examples:

- Random forest [Ref. 2]
- Extremely Randomized Trees [Ref. 3]

Ensemble: boosting



Examples:

- AdaBoost [Ref. 4]
- Gradient Tree Boosting

References

1. *Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. "Feature Selection: A Data Perspective". In ACM Computing Surveys.*
2. Breiman, L. "Random Forests". In *Machine Learning.*
3. Geurts, Pierre, Damien Ernst, and Louis Wehenkel. "*Extremely randomized trees*". In *Machine learning.*
4. *Harris Drucker. "Improving Regressors using Boosting Techniques". In Proceedings of the Fourteenth International Conference on Machine Learning*