# Ensemble methods

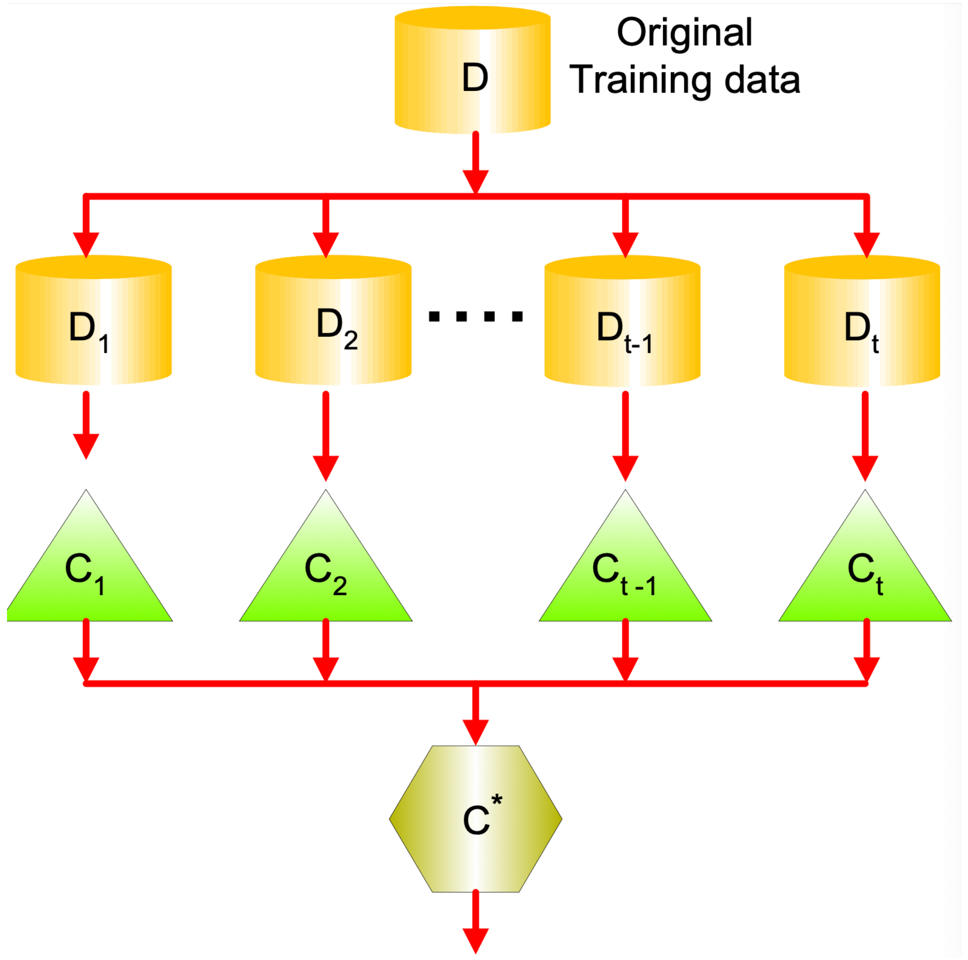
## Ensemble Methods

- Predict by combining the predictions made by multiple predictors

**Step1:** build different dataset

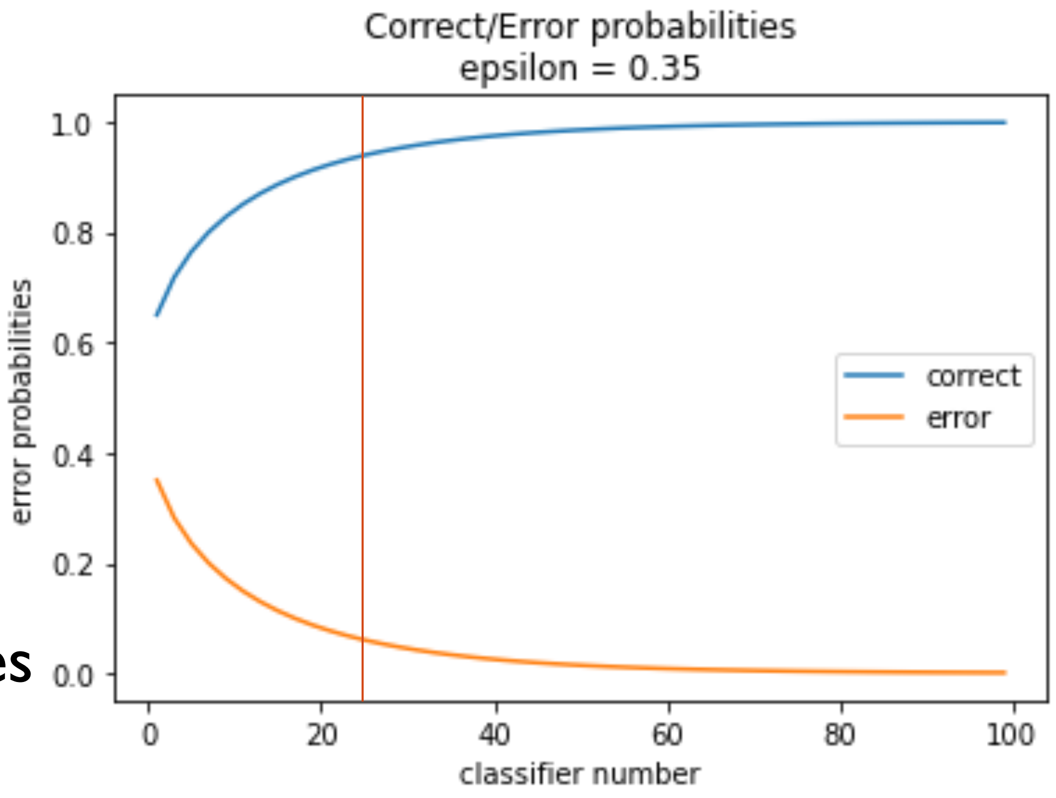
**Step2:** build different predictors

**Step3:** combine all predictions



## Why Ensemble Methods work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35 (<0.5)
  - Assume errors made by classifiers are uncorrelated
  - Vote for the result
- Probability that the ensemble classifier makes a wrong prediction:



$$P(X \geq 13) = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Or 94 % gives a correct prediction with a correct rate of 0.65

## Ensemble methods

- Useful for classification or regression
  - For classification, aggregate predictions by voting.
  - For regression, aggregate predictions by averaging.
- Model types can be:
  - Heterogeneous
    - Example: neural net combined with SVM combined with decision tree combined with …
  - Homogeneous – most common in practice
    - Individual models referred to as base classifiers (or regressors)
    - **Example:** ensemble of 1000 decision trees
- Base classifiers: important properties
  - Computationally fast: usually need to compute large numbers of classifiers
  - Accuracy: error rate of each base classifier better than random
  - Diversity: lack of correlation
    - independent model → they are not wrong on the same examples.

## Base classifiers: important properties

- Diversity
  - Predictions vary significantly between classifiers
  - Usually attained by using unstable classifier
    - small change in training data (or initial model weights) produces large change in model structure

  - **Examples of unstable classifiers:**
    - decision trees (very sensible of dataset)
    - neural nets (very sensible of weights initialization)
    - rule-based (very sensible of dataset)
    - https://machinelearningmastery.com/different-results-each-time-in-machine-learning/

  - **Examples of stable classifiers:**
    - Linear models: linear regression or logistic Regression

## How to create diverse base classifiers

- Use random projection of the dataset on a lower-dimentional space

- Resample / subsample training data
  - Sample instances
    - Select disjoint partitions
      - Reduce the number of training examples
    - Select sample randomly without replacement
      - Reduce the number of training examples
    - Select sample randomly with replacement
      - Constant number of training examples
  - Sample features (random subspace approach)
    - Select randomly the features used for training

## Elementary ensemble algorithm

- S : learning data set of m elements

1. Learn the classifier h1 on subset S1 ⊂ S. Test of h1 on S\S1.
2. Learn the classifier h2 on subset S2 ⊂ S \ S1 with half of elements of S2 wrongly classified by h1.
3. Repeat operation until no more remaining element
4. H : Majority vote between answers of h1, h2 and h3.
    - Notebook: ensemble voting from scratch.ipynb

3. Alternative for 3.
   Learn classifier h3 on subset S3 ⊂ S \ S1 \ S2 : contains elements for which rules h1 and h2 answer differently.
4. H : Majority vote between answers of h1, h2 and h3.

## How to use ensemble in sklearn

- sklearn.ensemble.VotingRegressor
- sklearn.ensemble.VotingClassifier
  - **Estimators:** ***list of (str, estimator)***
  - **Voting:** ***{'hard', 'soft'} # Only for classifier***
  - **Weights:** ***array-like of shape (n_classifiers,)***

- ***Usual use***
  - ***clf.fit***
  - ***clf.predict***

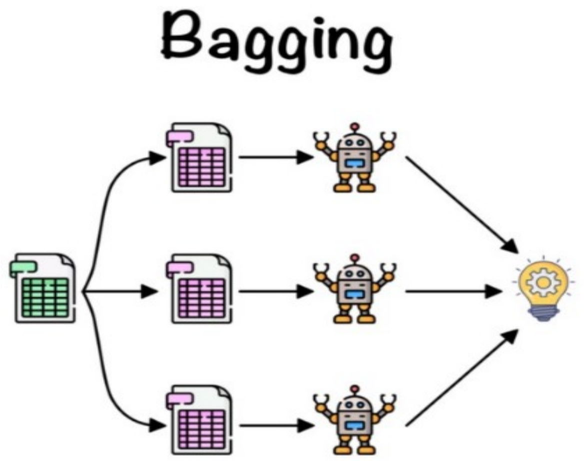
- Use with fitted estimator

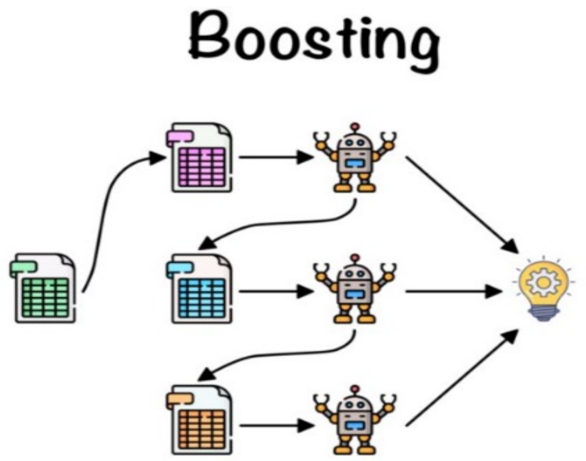
```
clf_list = [clf1, clf2, clf3]
eclf = VotingClassifier(estimators = [('1' ,clf1), ('2', clf2), ('3', clf3)], voting='soft')
eclf.estimators_ = clf_list
eclf.le_ = LabelEncoder().fit(y)
eclf.classes_ = seclf.le_.classes_
# Now it will work without calling fit
eclf.predict(X,y)
```

## Main approach of Ensemble Methods

- 1. Bagging
  - Create different training subset from sample training data with replacement
  - final output is based on majority voting.
- For example:
  - Random Forest

- 2. Boosting
  - Combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy.
- For example:
  - ADA BOOST: any classifier
  - XG BOOST: with tree





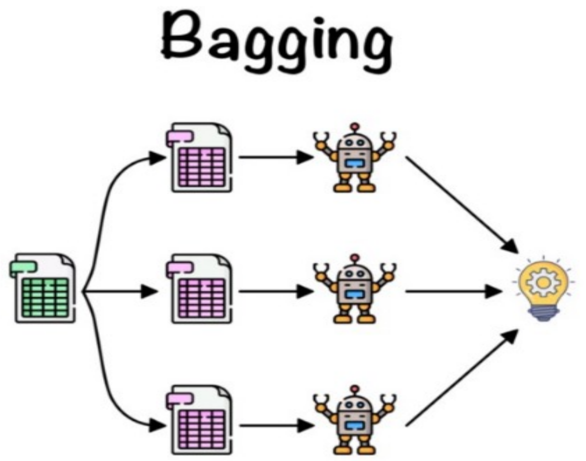
# Bagging versus Boosting

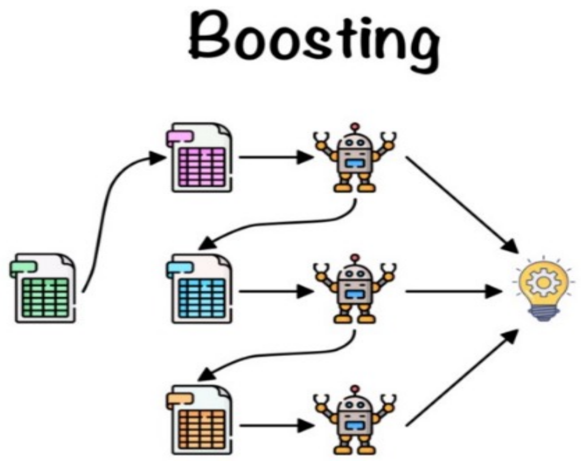
Bagging

- Weight of each classifier is same
- Only reduces variance
- Robust to noise and outliers
- Easily parallelized

Boosting

- Weight of a classifier based on its performance
- Reduces both bias and variance
- Sensible to noise and outliers
- Sequential





**Today: with RandomForest**

## *Random Forest*
## *an example of bagging method*

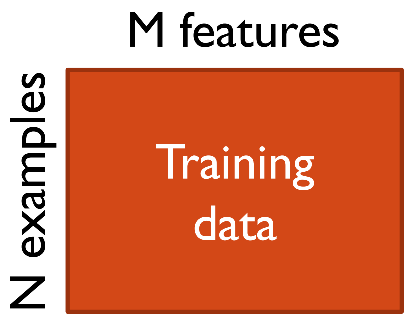
Original presentation from Jeff Howbert

## From decision tree to random forest

- Decision trees are greedy
  - They choose which variable to split on using a greedy algorithm that minimizes error
  - Sensitive to small changes in the data set
  - Overfitting, it's easy

- Combining predictions from multiple trees should work better.

- The random forest
  - Use a subset of data
  - Built different models
- In order to reduce the correlation between the prediction made by each model

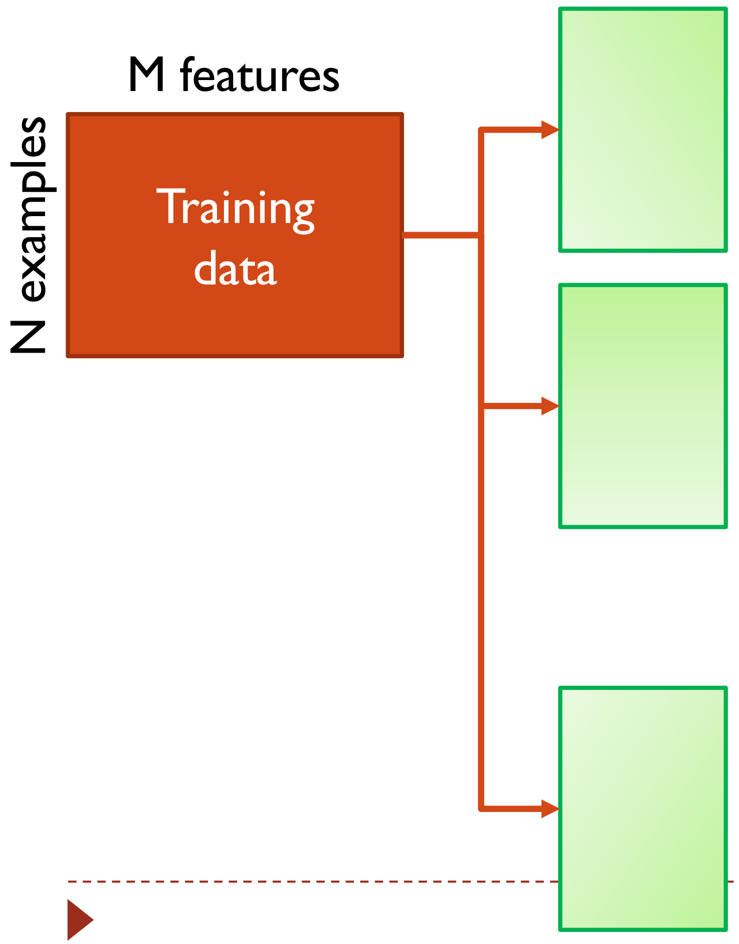

M features
N examples
Training data

Create many (100) samples from the training data

- with a same number of observations N identical of the original data

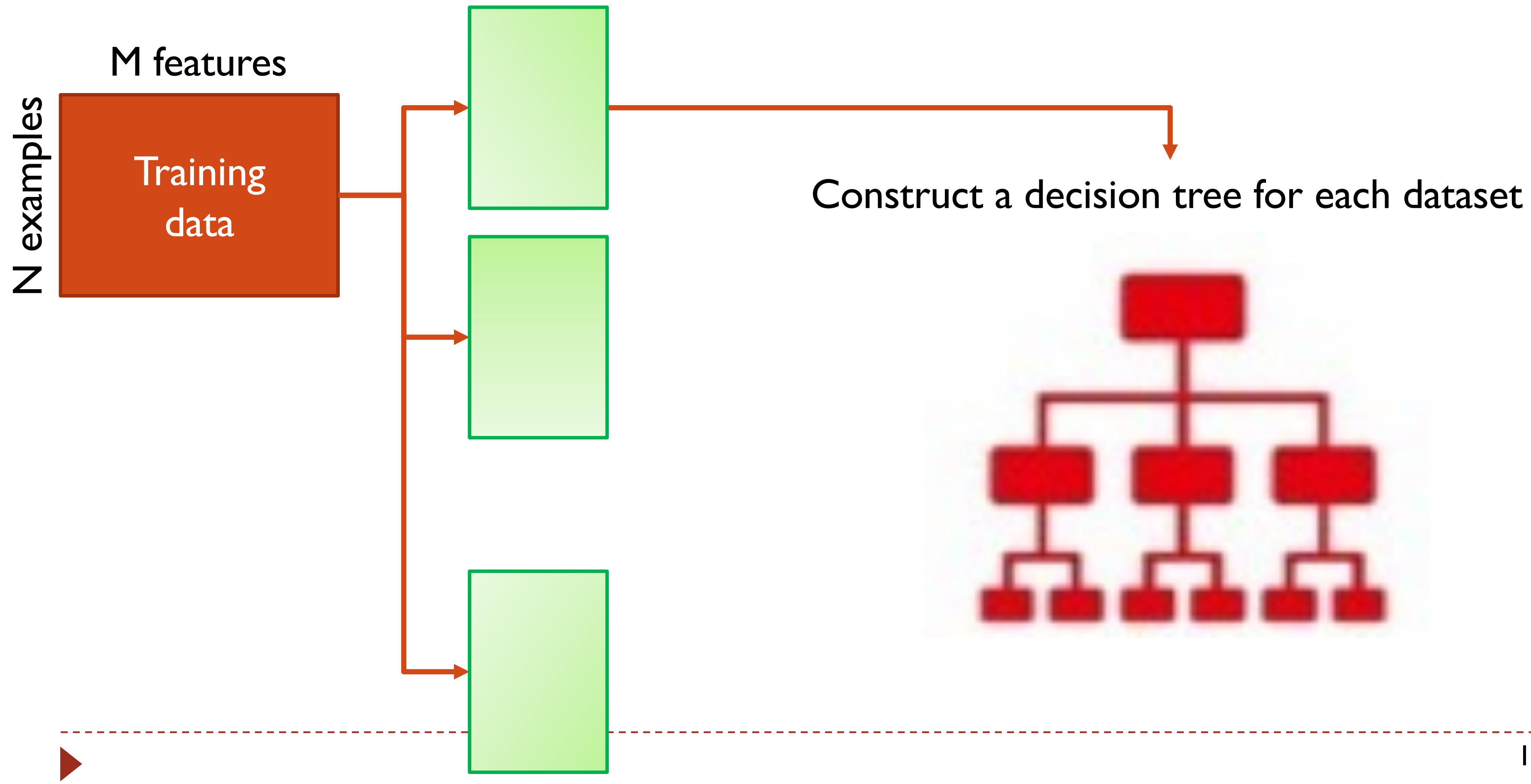
  (remove some samples and duplicate some others – bootstrap)

- with m random features (generally $m < \sqrt{M}$)

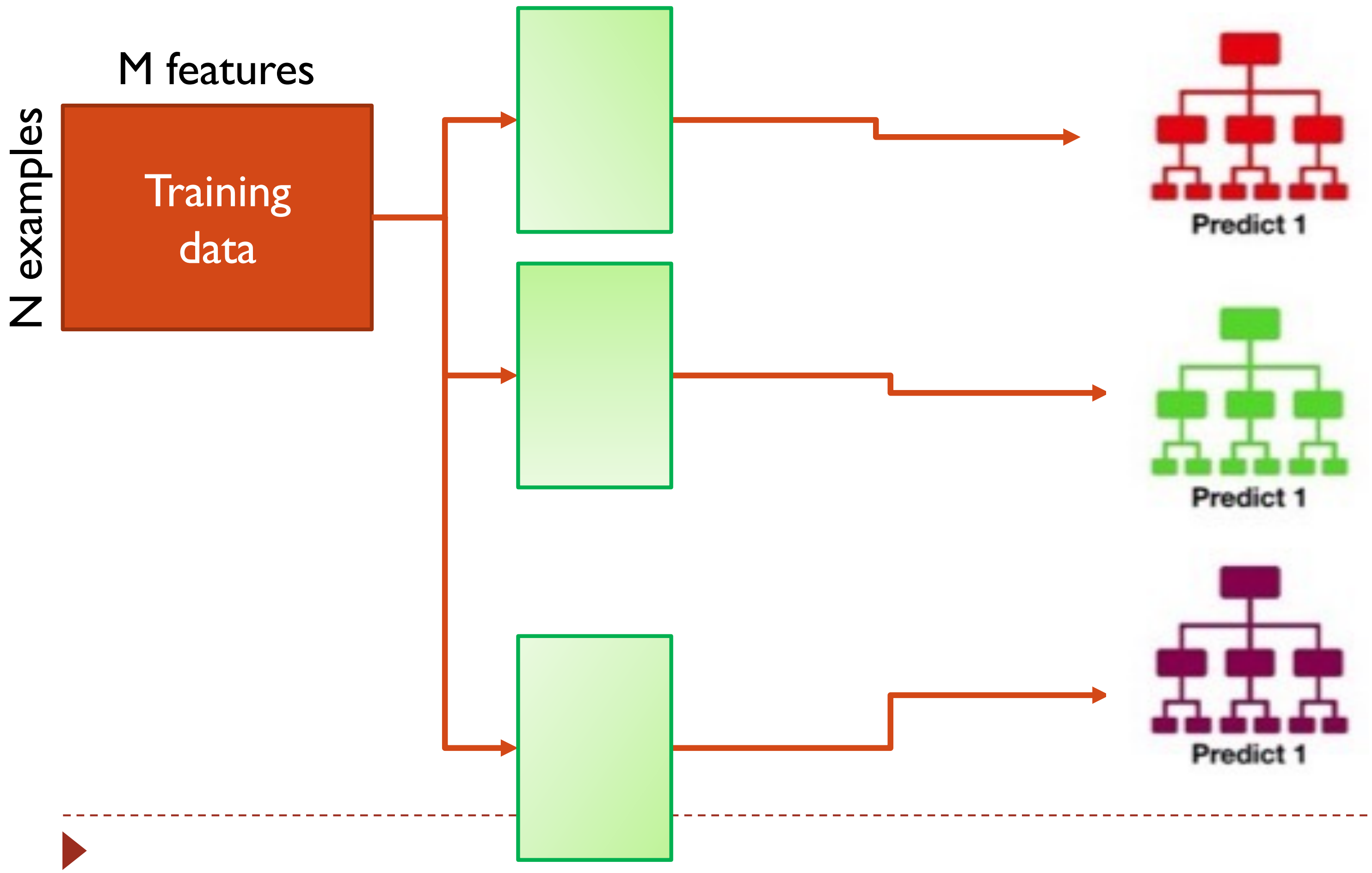


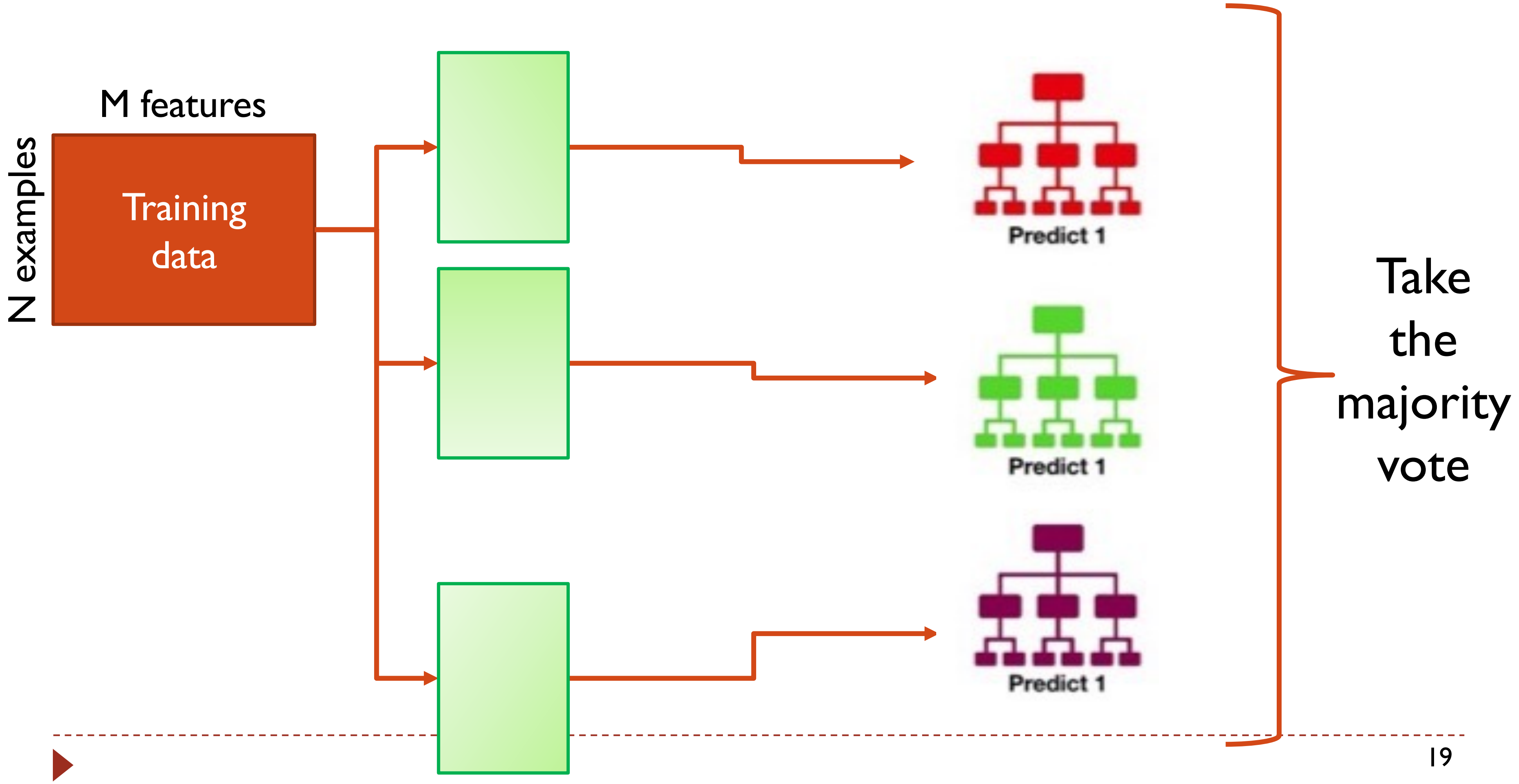
Create many (100) samples from the training data

- with a same number of observations N identical of the original data (remove some samples and duplicate some others – bootstrap)
- with m random features (generally $m < \sqrt{M}$)



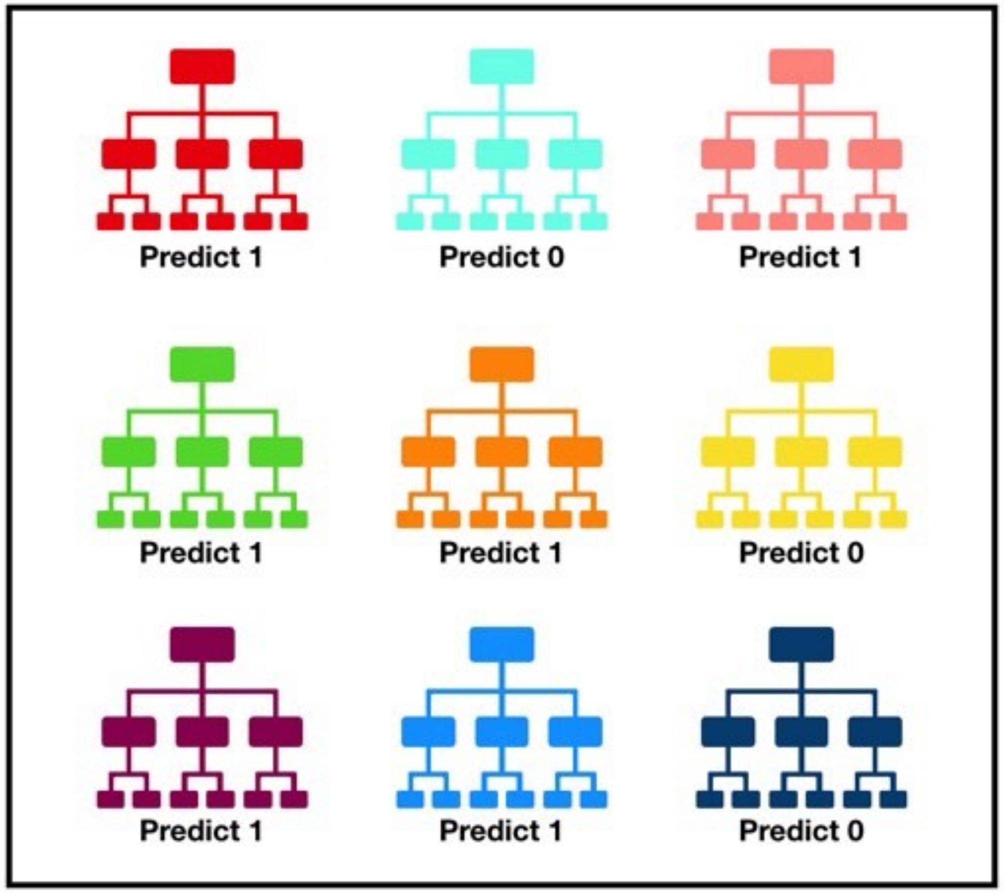
Create decision tree from each bootstrap sample




M features
N examples
Training data
Predict 1
Predict 1
Predict 1
Take the majority vote

## Why many trees give a more valuable result than one tree

- The reason for this wonderful effect is that the trees protect each other from their individual mistakes.
- While some trees may be wrong, many others will be right.
- Remember
  - With 25 classifier
  - Error rate = 0.35
  - 94 % of correct prediction



Tally: Six 1s and Three 0s
**Prediction: 1**

## Random Forest with sklearn

- Use RandomForestClassifier or RandomForestRegressor

- Main parameters for random forest
  - **n_estimators** : *integer, optional (default=10)*
    - The number of trees in the forest.

## Random Forest with sklearn

- Parameters of the base model (Decision Tree)
  - **max_depth** : *integer or None, optional (default=None)*
    - The maximum depth of the tree.
    - If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
  - **min_samples_split** : *int, float, optional (default=2)*
    - The minimum number of samples required to split an internal node
  - **min_samples_leaf** : *int, float, optional (default=1)*
    - The minimum number of samples required to be at a leaf node.
    - A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches.
  - **max_features** : *int, float, string or None, optional (default="auto")*
    - The number of features to consider when looking for the best split
  - **min_impurity_decrease** : *float, optional (default=0.)*
    - A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
  - **min_impurity_split** : *float, (default=1e-7)*
    - Threshold for early stopping in tree growth.
    - A node will split if its impurity is above the threshold, otherwise it is a leaf.

## PROs and CONs of Random Forest Algorithm

- PROs

1. It solves the problem of overfitting as output is based on majority voting or averaging.
2. It performs well even if the data contains null/missing values.
3. It is highly stable as the average answers given by a large number of trees are taken.
4. It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature space is reduced.

- CONs

1. It is becoming difficult to explain decisions
2. long training time if no parallelism

## The lab of today

- integrate the use of random forests with a search for hyper-parameters into the project.

## Some reading

## Some complements to help you revise