Ensemble Learning (一)

Substract:
- Voting, Bagging and Pasting --> Decision Tree, Random Forest
- Boosting(Adaboost, GBDT), Stacking --> Xgboost, LightGBM

Wisdom of the crowd (aggregated > individual)

Ensemble: a group of predictors (weak estimators --> strong estimators)

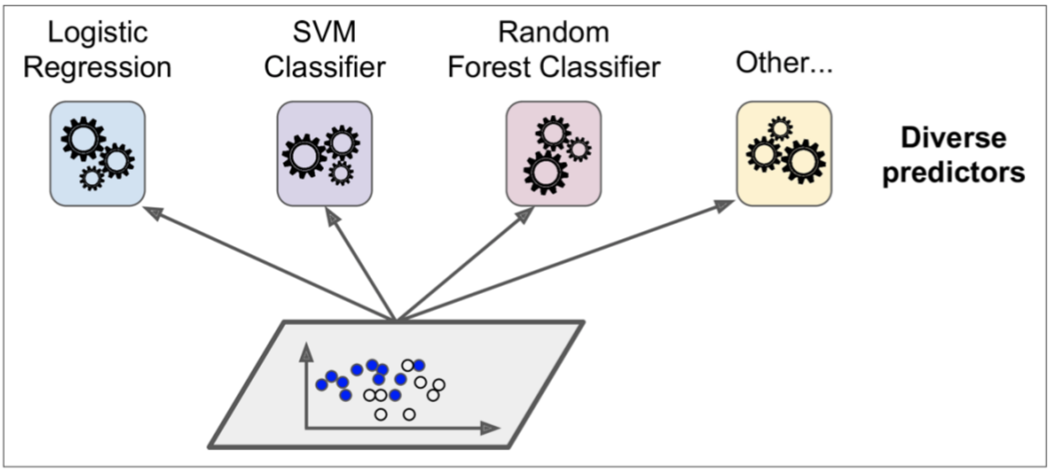
Insights: Ensemble methods -- an aggregated predictor

Popular ensemble methods:
- Voting
- Bagging and Pasting (e.g. random forest)
- Boosting (e.g. xgboost)
- Stacking

Best Case: all classifiers are perfectly independent, making uncorrelated errors

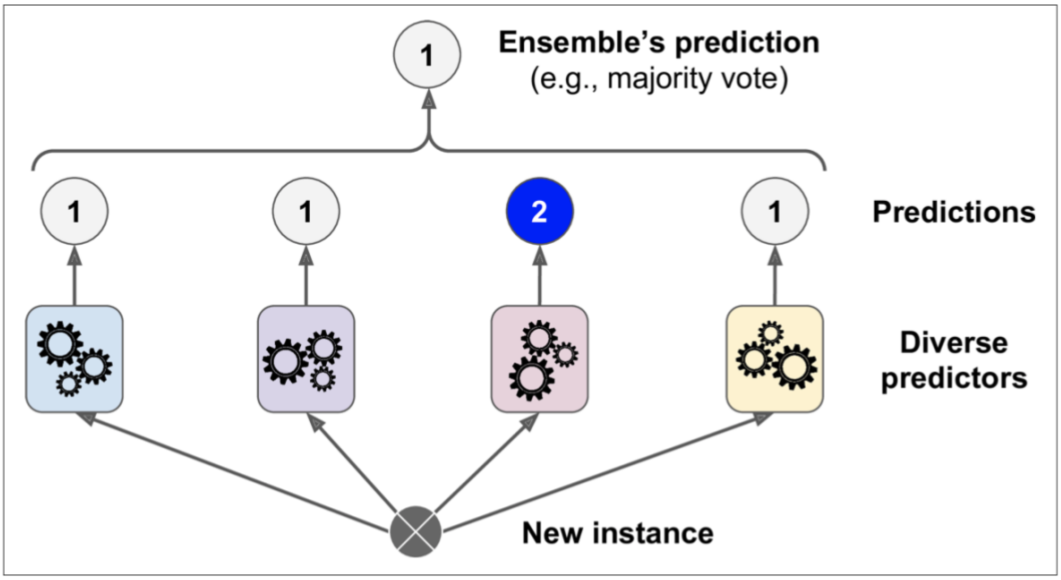
Status Quo: classifiers trained on the same data –> the same types of errors

Improvement : diverse classifiers --> various errors --> accuracy



一、Voting
- aggregate the predictions of each classifier and predict the class that gets the most votes
- two types:
  - hard voting classifier: majority-vote
    - the majority of the results == the final result
  - soft voting classifier
    - base: all classifier capable to estimate class probabilities
    - predict the class with the highest class probability

- notice: SVM , by default, incapable to do --> set **probability = True** --> use cross validation under the hood, slow down training --> add a **predict_proba()** method



二、Bagging and Pasting

Improvement: diverse classifiers

- different training algorithms
- train on different random subsets of the training set

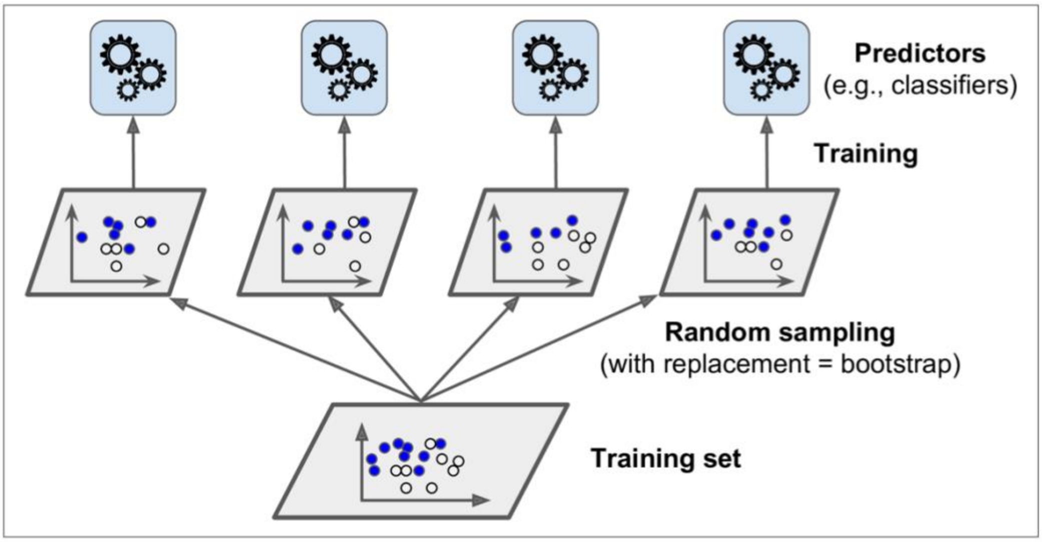
Bagging (bootstrap aggregating) : sampling is performed with replacement
Pasting: sampling is performed without replacement

How to aggregate?

- Classification: statistical mode, i.e., the most frequent prediction
- Regression: the average prediction

Benefit:

- reduce both bias and variance
- trained in parallel via different CPU cores or even different servers



Out-of-Bag Evaluation:

- The ramining instances that are not sampled are called out-of-bag(oob) instances --> seen as validation set
- Different predictors have different oob instances
- evaluate the ensemble itself by averaging out the oob evaluations of each predictor

Hyperparameters:

- max_samples & bootstrap
- max_features & bootstrap_features

Random Patches Method: Sampling both training instances and features

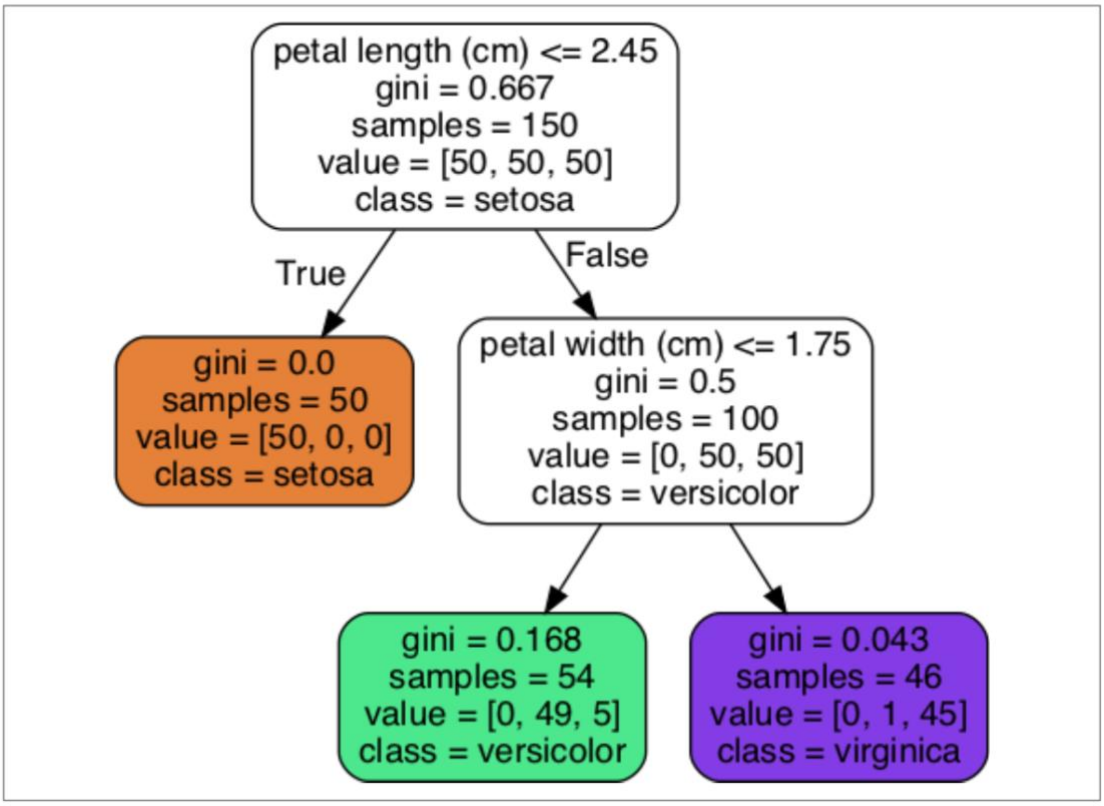
Random Subspace Method:

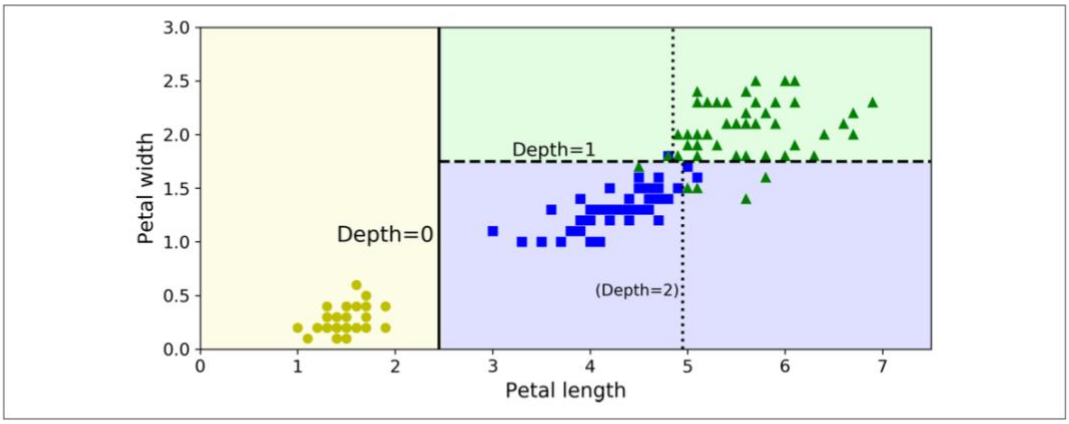
- Keeping all training instances (i.e., bootstrap=False and max_samples=1.0)
- but sampling features (i.e., bootstrap_features=True and/or max_fea tures smaller than 1.0)

Decision Tree : fundamental components of Random Forests

- Data preparation: don't require feature scaling or centering
- Node: Internal node (feature) and leaf node (class)
- 3 steps:
  - Feature selection
  - Tree grow
  - Tree prune (regularization)

Iris Decision Tree





- Procedure(classification with CART traning Algorithm):
  - Find the best feature to classify the data set
  - Gini impurity or Entropy?
    - Gini impurity

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

$p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node.

  - Use Feature k and a threshold t_k (e.g., "petal length ≤ 2.45 cm") to split the the training set (weighted by their size)
    - CRAT cost function for classification

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

$$\text{where } \begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$$

- Repeat the process at each level
- Regularization
  - Nonparametric model vs parametric model (e.g. linear regression)
    - Predetermined parameters --> degree of freedom
  - Hyperparameters
    - Max_depth
    - Max_leaf_nodes
    - Max_features
    - Min_samples_split
    - Min_samples_leaf

- **Benefit**
  - Whilt box model
    - Intuitive and interpretation
- **Instability**
  - Orthogonal decision boundaries --> sensitve to training set rotation / small variation
  - Overfitting

Random Forests :

- an ensemble of Decision Trees via **bagging** method generally (sometimes **pasting**), typically with **max_samples** set to the size of the training set
- **RandomForestClassifier** ~= **DecisionTreeClassifier** + **BaggingClassifier**
- Notice: search for the best feature among a random subset of features instead of searching for the very best feature when splitting a node

Extra-Tree (Extremely Randomized Trees)

- using random thresholds for each feature rather than searching for the best possible thresholds (like regular Decision Trees do) when splitting a node --> more random / more faster

Feature importance:

- Feature selection (vs PCA for feature extraction)
- how much the tree nodes use that feature to reduce impurity on average across all trees in the forest
- a weighted average, where where each node’s weight is equal to the number of training samples that are associated with it