Econ 425 Week 8

Ensemble methods: part I

Grigory Franguridi

UCLA Econ

USC CESR

franguri@usc.edu

Motivation for ensemble models

- would you walk up to a car shop and purchase a car based on the advice of the dealer? No.
- instead, browse a few web portals where people have posted their reviews and compare different car models w.r.t. features/prices/customer reviews
- ask your friends and colleagues for opinions
- not reach a direct conclusion, but instead use opinions of others
- **ensemble models**: similar idea; combine the decisions from multiple models to improve the overall performance

Motivation for ensemble models

- example: you made a short movie and want to gather preliminary feedback (ratings) before making it public
- what are the possible ways of gathering this feedback?

$\hat{y} = \hat{f}_j(x)$ $j = 1, \cdots M$ (# of models / weak learner)

为什么?

$y = R$ (regression)

Averaging pred: $\hat{f}(x) = \frac{1}{M}\sum_{j=1}^{M} \hat{f}_j(x)$

Variance: $\mathrm{Var}\hat{f} = \frac{1}{M^2}\sum_{j=1}^{M}\sum_{i=1}^{M} \mathrm{cov}(\hat{f}_j, \hat{f}_i)$

$= \frac{1}{M^2}\left[\sum_{j=1}^{M} \mathrm{var}(\hat{f}_j) + \sum_{j\neq i} \mathrm{cov}(\hat{f}_j, \hat{f}_i)\right]$

if I want to minimize variance: (lowest can be equal to MSE)

为什么? 为了minimize MSE.

for simplicity: assume every model is unbiased

Motivation for ensemble models

- **A**: ask one of your friends to rate the movie
  - severe bias
- **B**: ask 5 colleagues to rate the movie
  - better, but these people may not be experts
- **C**: ask 50 randomly chosen people to rate the movie
  - smallest bias

Motivation for ensemble models

- a diverse group of people are likely to make better decisions as compared to individuals
- similar reasoning for a diverse set of models in comparison to single models (ensemble learning)

Simple ensemble methods

absolute majority : >50%

relative majority : 2. 2. 3. 3. 3. 4. 4

<50%. But majority

- Max Voting
- Averaging
- Weighted Averaging

Max voting

- generally used for classification problems
- prediction by each model is a 'vote'; max vote = the majority prediction
- example: ask 5 colleagues to rate your movie (scale 1-5); 3 colleagues rated as 4, while 2 others rated as 5
  - final rating: 4

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4

Averaging

- used mainly for outcomes in regression problems/probabilities in classification problems
- average of predictions from all the models
- example: $(5 + 4 + 5 + 4 + 4)/5 = 4.4$

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4.4

Weighted averaging

- extension of the averaging method: all models are assigned weights defining the importance of each model for prediction
- example: if 2 of the colleagues are critics, while others have no prior experience in this field, then upweigh the 2 critics, downweigh others
- $[(5*0.23)+(4*0.23)+(5*0.18)+(4*0.18)+(4*0.18)] = 4.41$

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating	
weight	0.23	0.23	0.18	0.18	0.18	
rating	5	4	5	4	4	4.41

- more in **AdaBoost**

Model averaging widely used

- useful when there is **uncertainty** about the best model to use, when different models capture **different patterns** in the data, or when the goal is to **reduce the variance** of the predictions
- common technique in predictive modeling competitions, where combining the strengths of diverse models can lead to better overall performance

Model averaging

- **train multiple models**: develop several candidate models using different modeling techniques, algorithms, or parameter settings. These models can be of the same type (e.g., all regression models) or different types (e.g., a mix of regression, decision trees, and neural networks)
- **make predictions**: use each model to make predictions on a hold-out set, validation set, or through cross-validation to ensure unbiasedness
- **combine predictions**: aggregate predictions of the individual models to create a final prediction
  - max voting, (weighted) average, etc.
- **evaluate performance** on a test set or using other validation techniques. The model averaging process often results in improved predictive accuracy and reduced overfitting compared to using a single model

Advanced ensemble techniques

- Stacking
- Blending
- Bagging
- Boosting

Stacking

- use predictions from multiple models (e.g., decision tree, KNN, or SVM) to build a new model
- reminder about ML pipeline: training, testing, validation

Stacking

is like 1-layer FNN

① Train each individual model on training data $j = 1, \ldots M$
⇒ pred_m1, pred_m2 .. pred_mM (new features)
② Train a meta-model (ensemble model) using only new features
③ Validate

Step 1: use, say, 4-fold cross-validation to train base models

train_data

	x_0	x_1	x_2	x_3	y
fold_1	0.94	0.27	0.80	0.34	1
	0.02	0.22	0.17	0.84	0
fold_2	0.83	0.11	0.23	0.42	1
	0.74	0.26	0.03	0.41	0
fold_3	0.08	0.29	0.76	0.37	0
	0.71	0.76	0.43	0.95	1
fold_4	0.08	0.71	0.97	0.04	0
	0.84	0.97	0.89	0.05	1

test_data

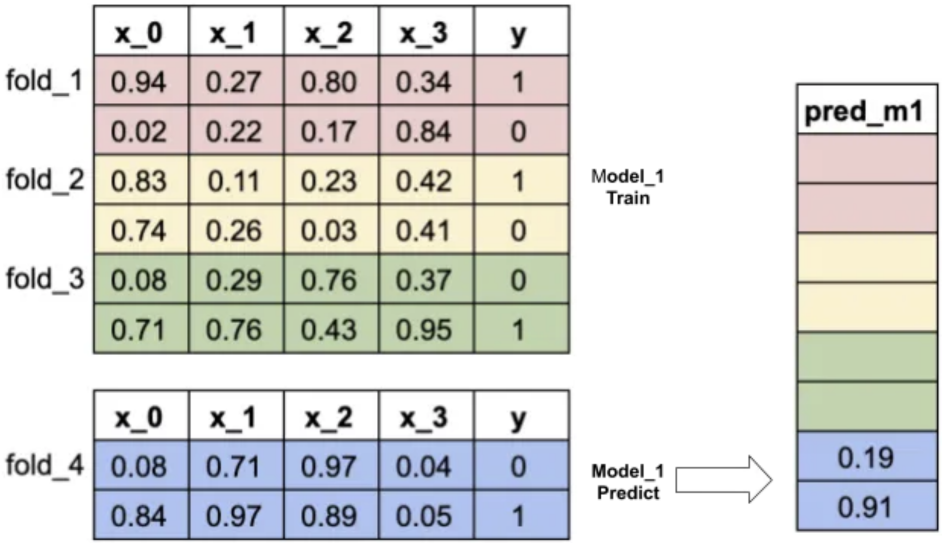
x_0	x_1	x_2	x_3	y
0.74	0.17	0.820	0.31	1
0.04	0.27	0.13	0.80	0
0.87	0.10	0.24	0.39	1

new features.
used on second step → predict

Training data (4-fold) and Testing data

Stacking

Step 2: base model 1 (say, a decision tree) is fitted on 3 folds, and predictions are made for the 4th fold; repeat for each fold



Model 1 training and prediction using 4-fold cross validation

Stacking

Step 2: now all training instances have a prediction, call it **pred m1**

train_data

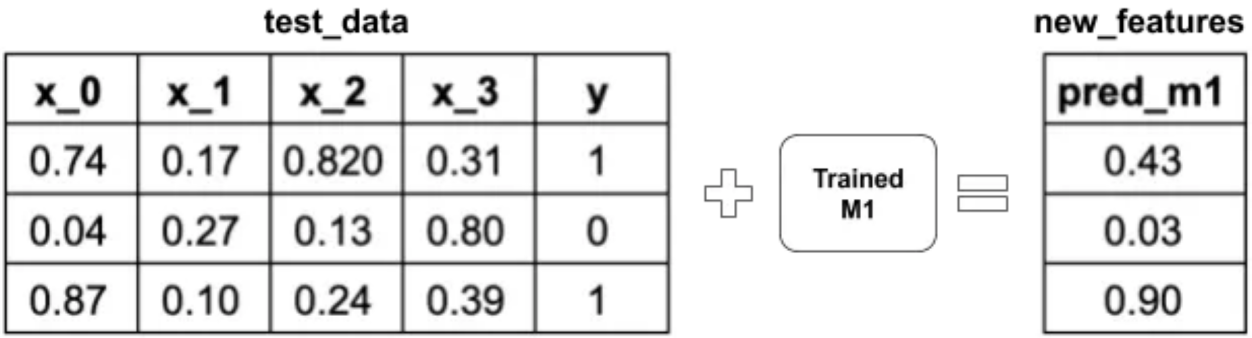
	x_0	x_1	x_2	x_3	y
fold_1	0.94	0.27	0.80	0.34	1
	0.02	0.22	0.17	0.84	0
fold_2	0.83	0.11	0.23	0.42	1
	0.74	0.26	0.03	0.41	0
fold_3	0.08	0.29	0.76	0.37	0
	0.71	0.76	0.43	0.95	1
fold_4	0.08	0.71	0.97	0.04	0
	0.84	0.97	0.89	0.05	1

new_features

pred_m1
0.96
0.03
0.90
0.12
0.03
0.77
0.19
0.91

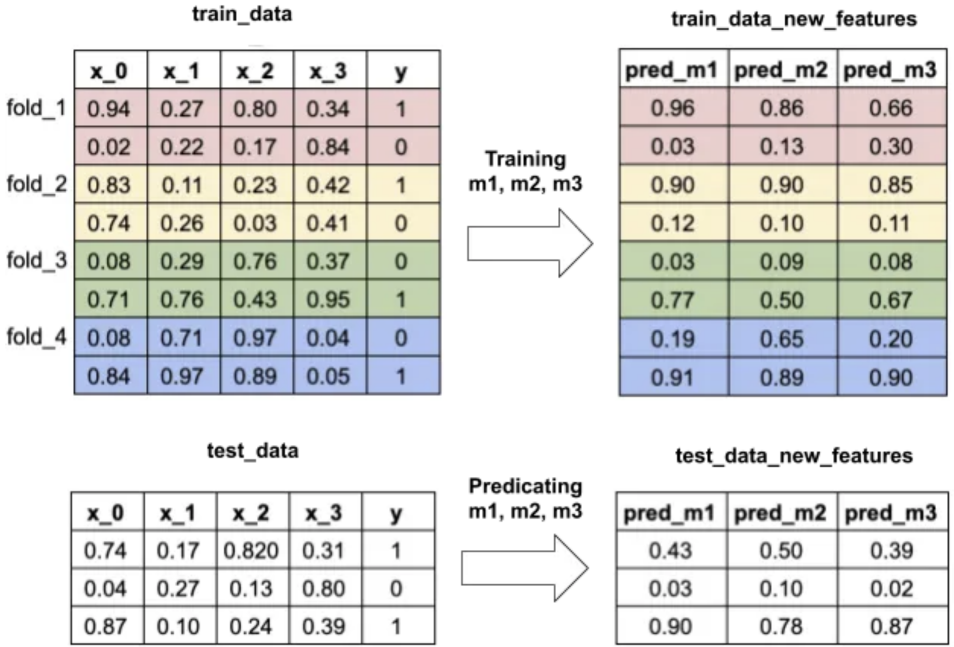
Stacking

Step 3. M1 (decision tree) generates predictions on test data (no folding)



Stacking

Step 4: repeat steps 2-3 for M2 (say, KNN) M3 (say, SVM), obtain pred m2 and pred m3

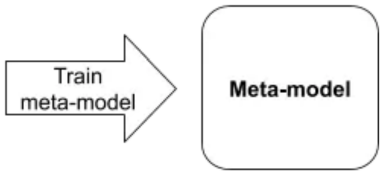


Stacking

Step 5: to train the meta-model (say, logistic regression), use **only the newly added features** [pred m1, pred m2, pred m3]

New train_data for meta-model

pred_m1	pred_m2	pred_m3	y
0.96	0.86	0.66	1
0.03	0.13	0.30	0
0.90	0.90	0.85	1
0.12	0.10	0.11	0
0.03	0.09	0.08	0
0.77	0.50	0.67	1
0.19	0.65	0.20	0
0.91	0.89	0.90	1



Step 6: final prediction is given by the trained meta-model

Blending

- similar to stacking: also uses base models to provide base predictions as new features and a meta-model is trained on the new features
- only difference is that training of the meta-model is performed on a holdout set (say, 10% of training data) instead of the folded training set

Holdout set和folded training set都是交叉验证（cross-validation）中常用的技术，用于评估模型性能和调整参数。它们的主要区别在于数据集的划分方式和使用方式：

1. **Holdout Set**：
   - Holdout set是将原始数据集分成两个部分：一个用于训练模型，另一个用于评估模型性能。
   - 通常，大部分数据用于训练模型，而一小部分数据被保留为holdout set来评估模型的性能。
   - 一般情况下，将数据集划分为训练集和测试集（holdout set）的比例为70%-90%的数据用于训练，10%-30%的数据用于测试。

2. **Folded Training Set**：
   - Folded training set是将原始数据集分成多个（通常是k个）相似大小的子集，其中一个子集被保留作为验证集，其余子集用于训练模型。
   - 模型在每个fold上训练k次，每次使用不同的fold作为验证集，其余的fold作为训练集。
   - 最终的性能评估是基于所有fold上的性能的平均值或其他汇总统计量。

因此，主要区别在于数据集的划分方式和使用方式。Holdout set适用于简单的模型评估和参数调整，而folded training set更适用于更准确的模型评估和参数选择。Folded training set通常比单个holdout set提供更可靠的性能评估，因为它充分利用了所有可用数据，并且对模型的性能具有更好的统计稳定性。

Blending

Step 1: split train data into base train data and holdout set

train_data

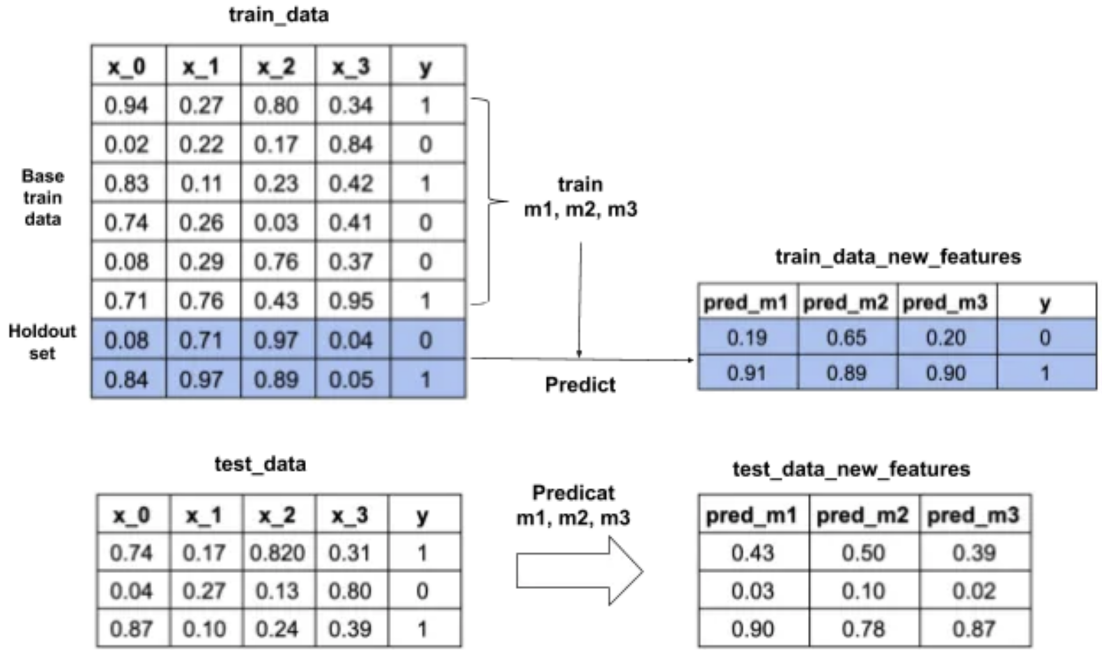
	x_0	x_1	x_2	x_3	y
	0.94	0.27	0.80	0.34	1
	0.02	0.22	0.17	0.84	0
base_train_data	0.83	0.11	0.23	0.42	1
	0.74	0.26	0.03	0.41	0
	0.08	0.29	0.76	0.37	0
	0.71	0.76	0.43	0.95	1
Holdout set	0.08	0.71	0.97	0.04	0
	0.84	0.97	0.89	0.05	1

test_data

x_0	x_1	x_2	x_3	y
0.74	0.17	0.820	0.31	1
0.04	0.27	0.13	0.80	0
0.87	0.10	0.24	0.39	1

Blending

Step 2: fit base models on base train data and make predictions on holdout set and test data; this creates new prediction features



Blending

Step 3: fit a meta-model on holdout set with new prediction features; use both original and meta features from holdout set

Step 4: use the meta-model to make final predictions on the test data using both original and new meta-features

Bagging

- if all the models are trained on the same data, will the combined model be useful?
- high chance that these models will give the same result since they are getting the same input
- **bootstrapping** (Efron, 1979): create subsets of observations from the dataset with replacement. The size of the subsets may or may not be the same as the size of the original set
  - allows for the estimation of the distribution of almost any statistic, including means, medians, variances, and regression coefficients, without making strong assumptions about the distribution of the data
- **bagging** (bootstrap aggregating) uses these subsets (bags) to estimate the distribution of the data

Bootstrap

- **Original data** X with n observations: $X = \{x_1, x_2, \cdots, x_n\}$
- **Resampling**: create a large number of bootstrap samples $X_1^*, \ldots, X_B^*$ by randomly drawing n observations with replacement from the original dataset. Each bootstrap sample is the same size as the original dataset but may contain duplicates due to the sampling with replacement
- **Compute statistic**: For each bootstrap sample X_b^*, compute the statistic of interest, e.g. mean $\bar{X}_b^* = \frac{1}{n}\sum_{i=1}^{n} x_{bi}^*$
- **Estimate distribution**: use the collection of bootstrap statistics to estimate the sampling distribution of the statistic. This can be used to compute confidence intervals, standard errors, and other measures of statistical uncertainty

Bagging

- draw bootstrap samples
- train base model (weak learner) on each bootstrap sample (run in parallel)
- ensemble prediction

Goal: appox the std err of $\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} X_i$ population level formula

Solution 1: $se = \sqrt{Var\,\hat{\theta}} = \sqrt{\frac{1}{n} Var X_i} = \frac{\sigma_{X_i}}{\sqrt{n}}$

$\hat{se} = \frac{\hat{\sigma}_X}{\sqrt{n}} \Rightarrow CI = \hat{\theta} \pm q^{1-\frac{\alpha}{2}}_{N(0,1)} \cdot \hat{se}$

Solution 2: Bootstrap

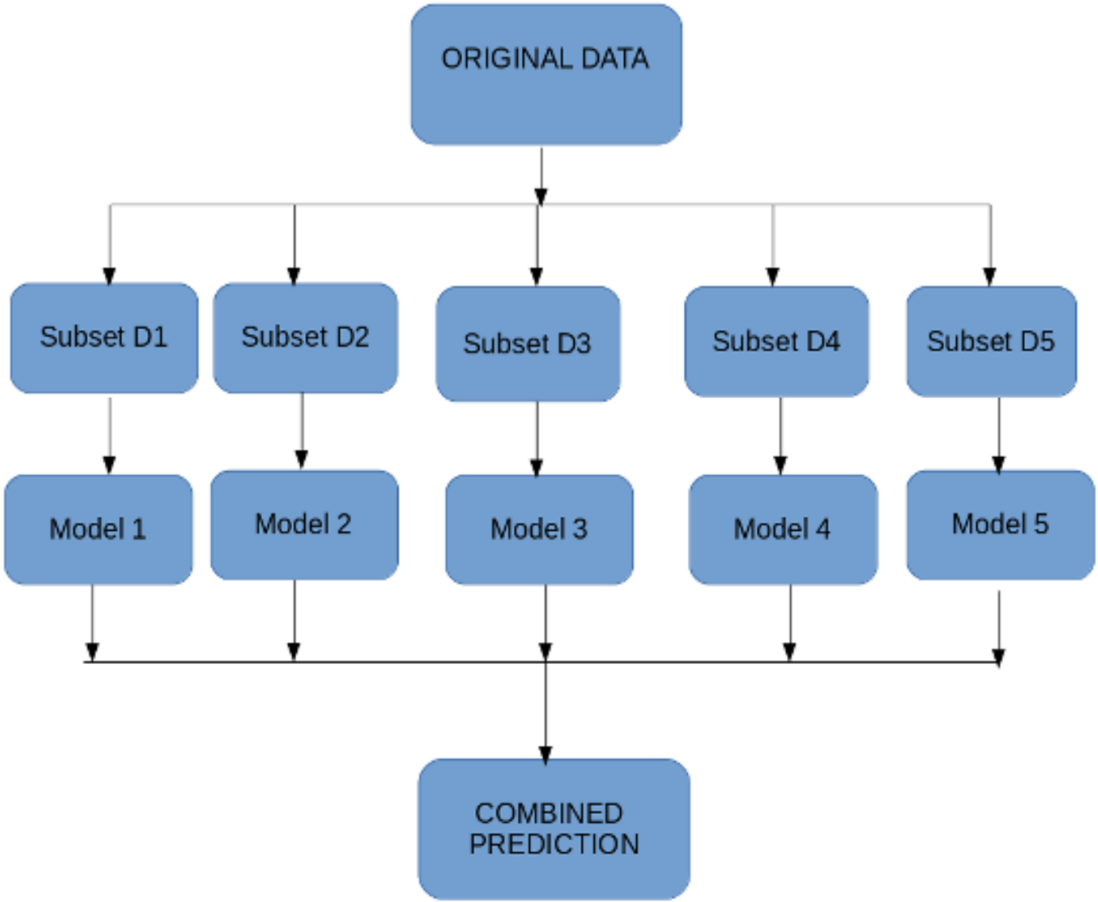
① Draw Boot samples: $X_1^* \cdots X_B^*$ from X (original dataset)

eg. $X_1^* = \{\underbrace{x_1, x_3, x_{10}, x_6 \cdots}_{\#=n}\}$

② $\hat{\theta}_B = \frac{1}{n}\sum_{i=1}^{n} x^*_{Bi} \quad B = 1 \cdots B$

③ $\hat{se} = Sd\{\hat{\theta}_1^* \cdots \hat{\theta}_B^*\}$

Bagging

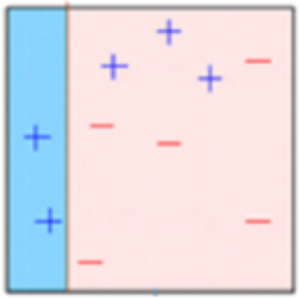

ORIGINAL DATA
Subset D1
Subset D2
Subset D3
Subset D4
Subset D5
Model 1
Model 2
Model 3
Model 4
Model 5
COMBINED PREDICTION

Boosting

- if a data point is incorrectly predicted by the first model, and then the next (probably all models), will combining the predictions provide better results?
- **boosting** is a *sequential process*, where each subsequent model attempts to correct the errors of the previous model. The succeeding models are dependent on the previous model

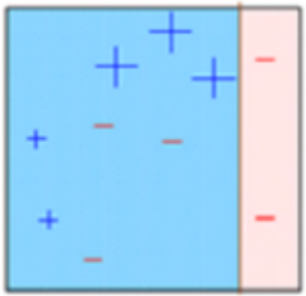
Boosting

1. A subset is created from the original dataset
2. Initially, all data points are given equal weights
3. A base model is created on this subset
4. This model is used to make predictions on the whole dataset



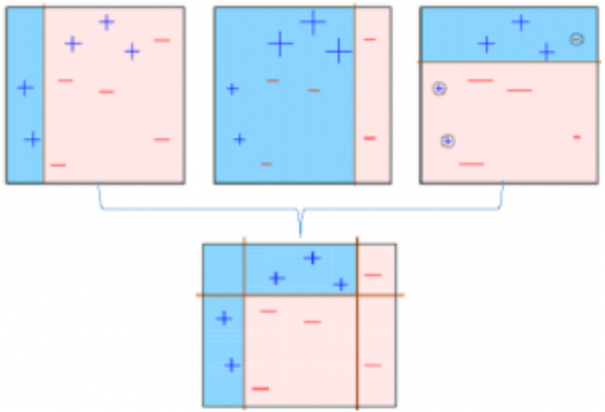
Boosting

5. Errors are calculated using the actual values and predicted values
6. Incorrectly predicted observations (here, the three misclassified blue-plus points) are given larger weights
7. Another model is trained and predictions are made on the dataset (this model tries to correct the errors from the previous model)



Boosting

8. Similarly, multiple models are trained, each correcting the errors of the previous model
9. The final model (*strong learner*) is the weighted mean of all the models (*weak learners*)



AdaBoost

- AdaBoost (Adaptive Boosting) is an ensemble method for improving the performance of weak classifiers
- introduced by Yoav Freund and Robert Schapire in 1995
- main idea: introduce *adaptive weights* for combining multiple weak classifiers (models that perform slightly better than random guessing) into a single strong classifier

AdaBoost

- **Initialization**: assign equal weights w_i to all training examples (if N training examples, each example is assigned a weight of $1/N$)
- **Iterative training**: For each iteration $t = 1, \ldots, T$:
  1. Train a weak classifier h_t on the weighted training examples
  2. Calculate the error ε_t of h_t as the weighted sum of misclassified examples

$$\varepsilon_t = \frac{\sum_{i=1}^{N} w_i I(y_i \neq h_t(x_i))}{\sum_{i=1}^{N} w_i}$$

AdaBoost

- **Iterative training**: For each iteration $t = 1, \ldots, T$
  - Compute the classifier's weight

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$$

    The smaller the error ε_t, the larger weight α_t
  - Update the weights of the training examples:

$$w_i \leftarrow w_i \exp(-\alpha_t y_i h_t(x_i))$$

    and then normalize the weights so that they sum up to 1

AdaBoost

- **Final model**: weighted average of the weak classifiers

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

where sign$(\cdot)$ is the sign function that returns $+1$ for positive input and -1 for negative input

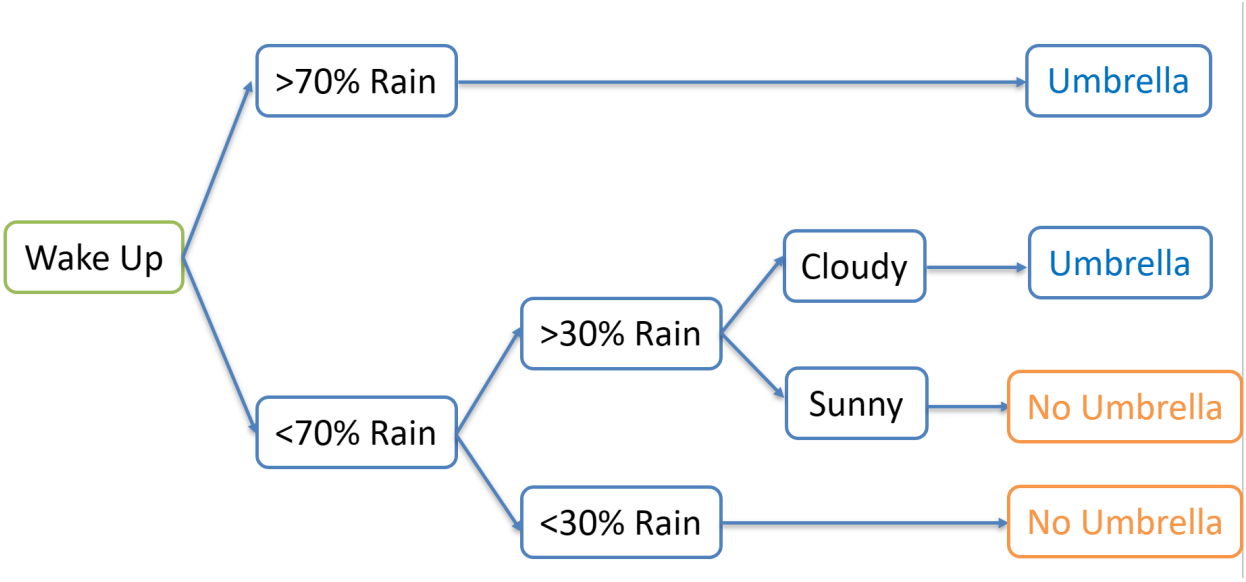
Random forest

- combines the opinions of many "trees" (individual models) to make better predictions, creating a more robust and accurate overall model
- introduced by Leo Breiman (2001)

Random forest

- combines the opinions of many "trees" (individual models) to make better predictions, creating a more robust and accurate overall model
- introduced by Leo Breiman (2001)

Random forest: decision trees

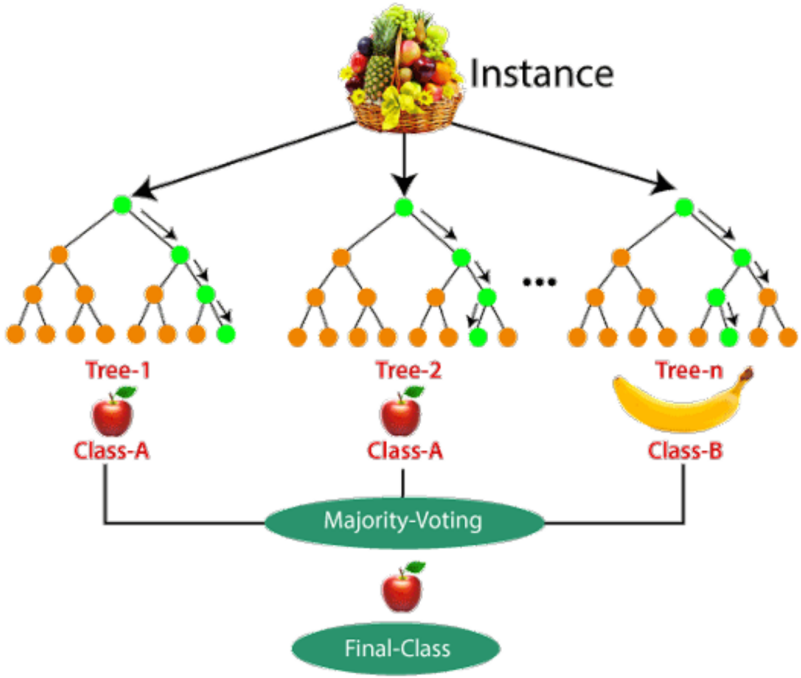


- uses a sequence of inquiries to reach a conclusion

Random forest: algorithm

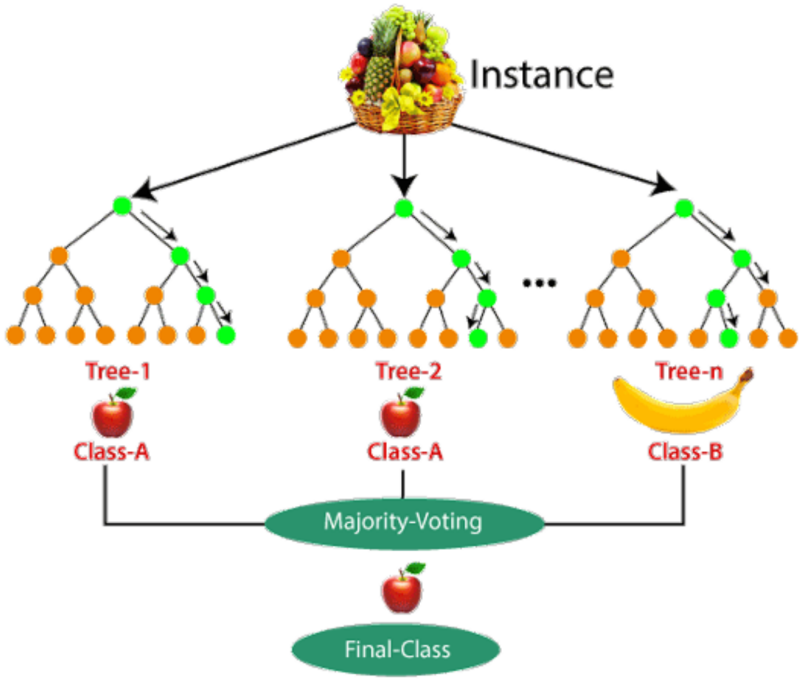
- **Step 1**: a subset of data and a subset of features are selected for constructing each decision tree
- **Step 2**: individual decision trees are constructed for each sample
- **Step 3**: each decision tree generates an output
- **Step 4**: final prediction is by majority voting/averaging for classification/regression, resp.

Random forest: example



- sample of n fruits is drawn with replacement from the fruit basket, and an individual decision tree is trained on each such sample

Random forest: example



- each decision tree generates an output; final prediction is by majority voting

Exercise: random forest by hand

- Given a small dataset, manually construct a simple random forest model with two decision trees and make a prediction for a new observation
- Data:

Observation	Feature 1 (X1)	Feature 2 (X2)	Target (Y)
1	1	5	0
2	2	6	0
3	3	7	1
4	4	8	1

Exercise: random forest by hand

- **Bootstrap sampling**:
  - Create two bootstrap samples from the original dataset. Each sample should have the same number of observations as the original dataset, but some observations may be repeated due to sampling with replacement
  - Sample 1: [Fill in your sample]
  - Sample 2: [Fill in your sample]
- **Build DTs**:
  - for each bootstrap sample, build a simple decision tree with just one split. Choose the split based on a simple criterion (e.g., the median of a feature)
  - Tree 1 (from Sample 1): [Describe the split and the resulting leaf nodes]
  - Tree 2 (from Sample 2): [Describe the split and the resulting leaf nodes]

Exercise: random forest by hand

- **Make a Prediction**:
  - Use RF to make a prediction for a new observation with $X_1 = 2$ and $X_2 = 7$
  - Tree 1 Prediction: [Your prediction]
  - Tree 2 Prediction: [Your prediction]
  - RF prediction (majority vote): [Your final prediction]
- **Task**: fill in the blanks for each step based on the instructions provided

Exercise: random forest by hand

- **Example solution**:
- **Bootstrap sample 1**

Observation	Feature 1 (X1)	Feature 2 (X2)	Target (Y)
1	1	5	0
2	2	6	0
3	3	7	1
4	2	6	0

- **Bootstrap sample 2**

Observation	Feature 1 (X1)	Feature 2 (X2)	Target (Y)
1	4	8	1
2	3	7	1
3	1	5	0
4	4	8	1

Exercise: random forest by hand

- **Step 2: grow DTs**
- For simplicity, use only one split based on the feature that provides the best separation (e.g., using Gini impurity/information gain)
- Tree 1 (on bootstrap sample 1):
  - Split on $X_1 \leq 2.5$
  - Left node: Majority class $= 0$
  - Right node: Majority class $= 1$
- Tree 2 (on bootstrap sample 2):
  - Split on $X_1 \leq 3.5$
  - Left node: Majority class $= 0$
  - Right node: Majority class $= 1$

Exercise: random forest by hand

- **Step 4: final prediction**
- Predict Y for a new observation with $X_1 = 2$ and $X_2 = 7$
  - Tree 1 prediction: $X_1 \leq 2.5 \rightarrow Y = 0$
  - Tree 2 prediction: $X_1 \leq 3.5 \rightarrow Y = 0$
  - Final prediction (majority vote): $Y = 0$