Lecture 13

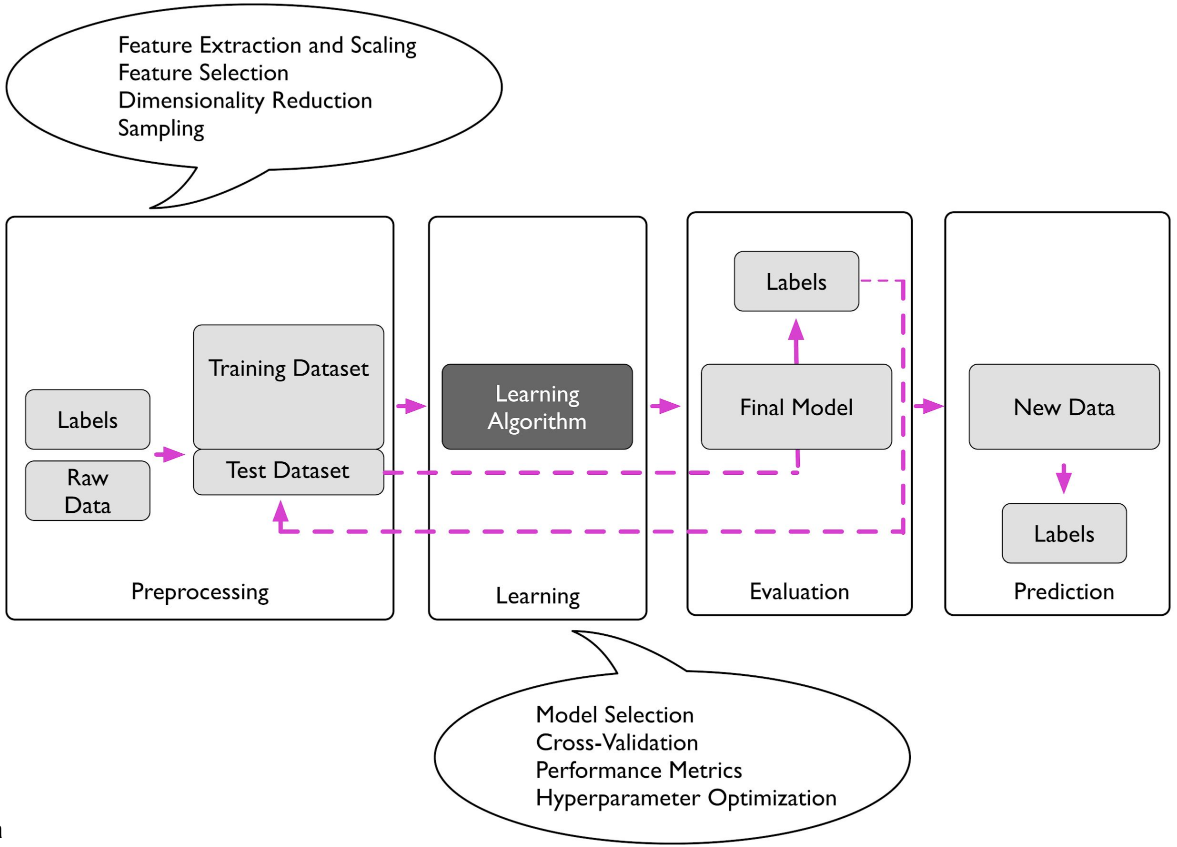
Ensemble methods

https://github.com/dalcimar/MA28CP-Intro-to-Machine-Learning

UTFPR - Federal University of Technology - Paraná

https://www.dalcimar.com/

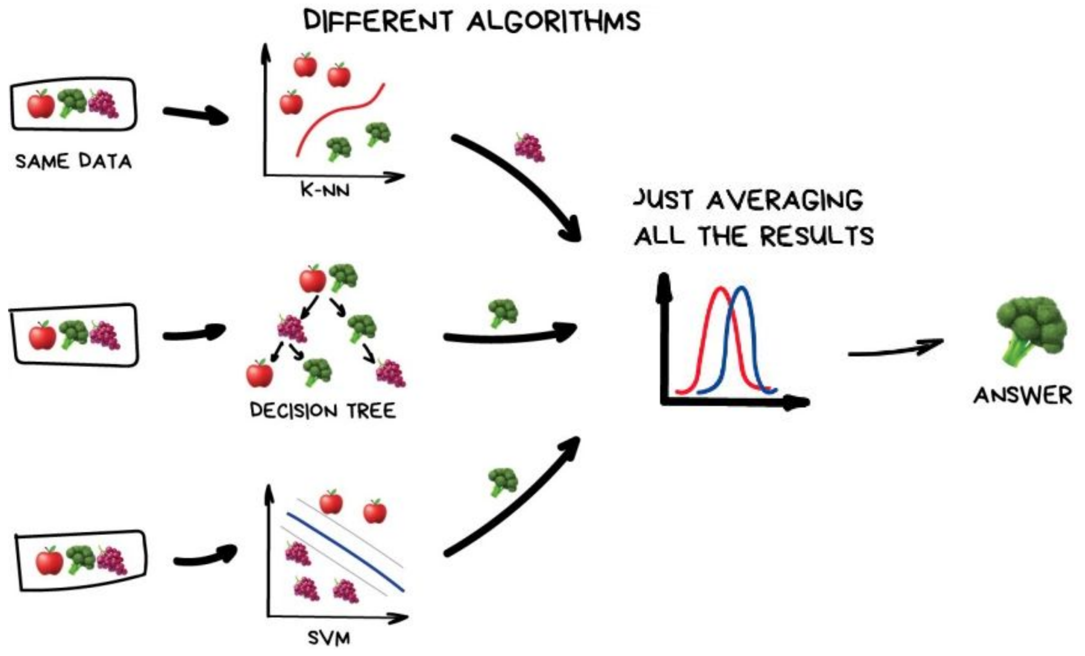
Machine learning pipeline



Lecturer Overview

- **Intro and overview**
- Voting
- Bagging
- Random Forests
- Boosting
- Gradient Boosting
- Stacking

Ensemble


DIFFERENT ALGORITHMS
SAME DATA
K-NN
DECISION TREE
SVM
JUST AVERAGING ALL THE RESULTS
ANSWER

Ensemble

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**. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.
  - Examples: Bagging methods, Forests of randomized trees, …

- By contrast, in boosting methods, base estimators are **built sequentially** and **one tries to reduce the bias of the combined estimator**. The motivation is to combine several weak models to produce a powerful ensemble.
  - Examples: AdaBoost, Gradient Tree Boosting, …

Ensemble

Accuracy and diversity are two vital requirements for constructing classifier ensembles. Many methods for constructing ensembles have been developed. All methods aims construct good individual hypotheses with uncorrelated errors (diversity). General methods:

- Manipulating the hypotheses/classifier
  - Voting and Stacking
- Manipulating the training examples
  - Bagging and Boosting
- Manipulating the input features
  - Feature Subspace
- Manipulating the training examples and features
  - Random Forest
- Manipulating the output targets
- Injecting randomness
  - Neural network ensembles

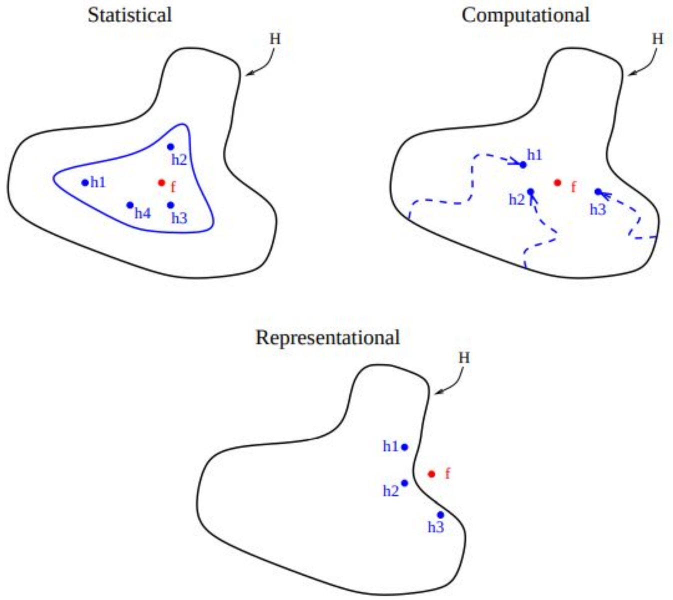


Fig. 2. Three fundamental reasons why an ensemble may work better than a single classifier

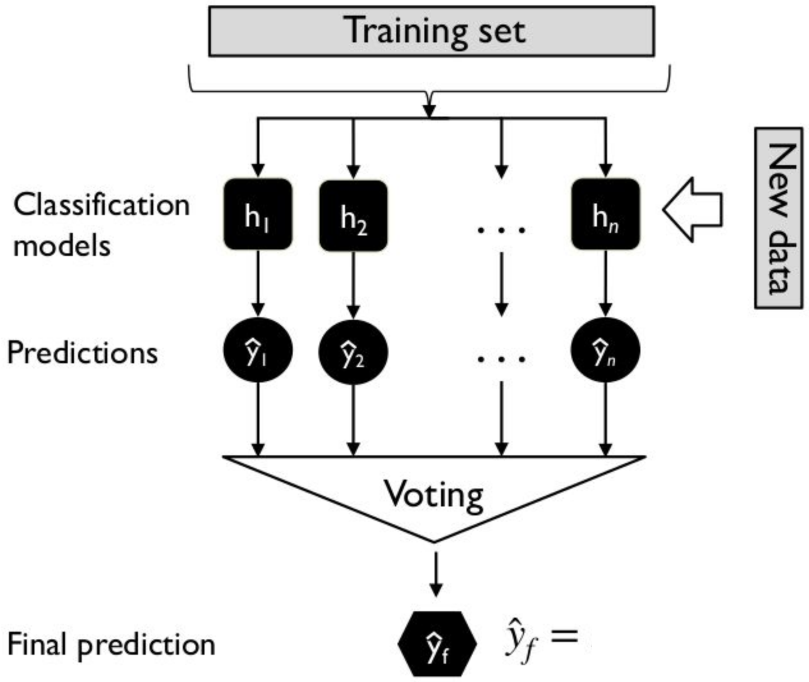
Thomas G. Dietterich. Ensemble Methods in Machine Learning. Multiple Classifier Systems, pp. 1-15, 2000.

Lecturer Overview

- Intro and overview
- **Voting**
- Bagging
- Random Forests
- Boosting
- Gradient Boosting
- Stacking

Voting Classifier

- The idea behind the Voting Classifier is to **combine conceptually different machine learning classifiers** and use a majority vote or the average predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well performing model in order to balance out their individual weaknesses.
  - Majority/Hard voting
  - Soft voting

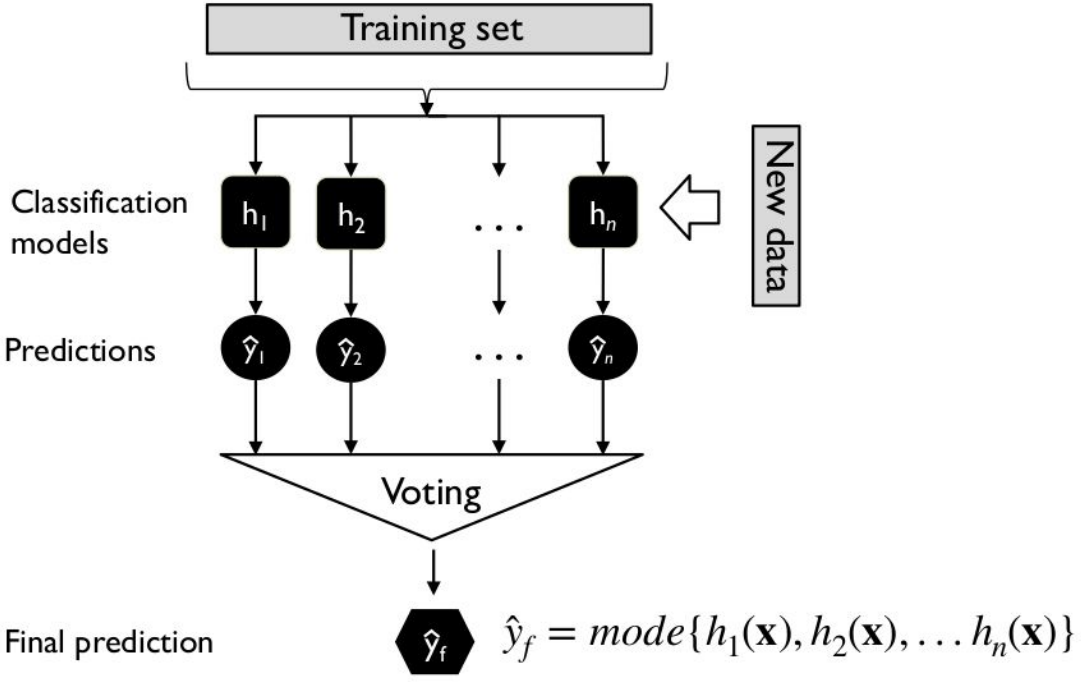


Majority Class Labels (Majority/Hard Voting)

In majority voting, the predicted class label for a particular sample is the class label that represents the majority (mode) of the class labels predicted by each individual classifier.

If the prediction for a given sample is

- classifier 1 -> class 1
- classifier 2 -> class 1
- classifier 3 -> class 2
- the VotingClassifier (with voting='hard') would classify the sample as “class 1” based on the majority class label.



$$\hat{y}_f = mode\{h_1(\mathbf{x}), h_2(\mathbf{x}), \dots h_n(\mathbf{x})\}$$

where $h_i(\mathbf{x}) = \hat{y}_i$

Majority Class Labels (Majority/Hard Voting)

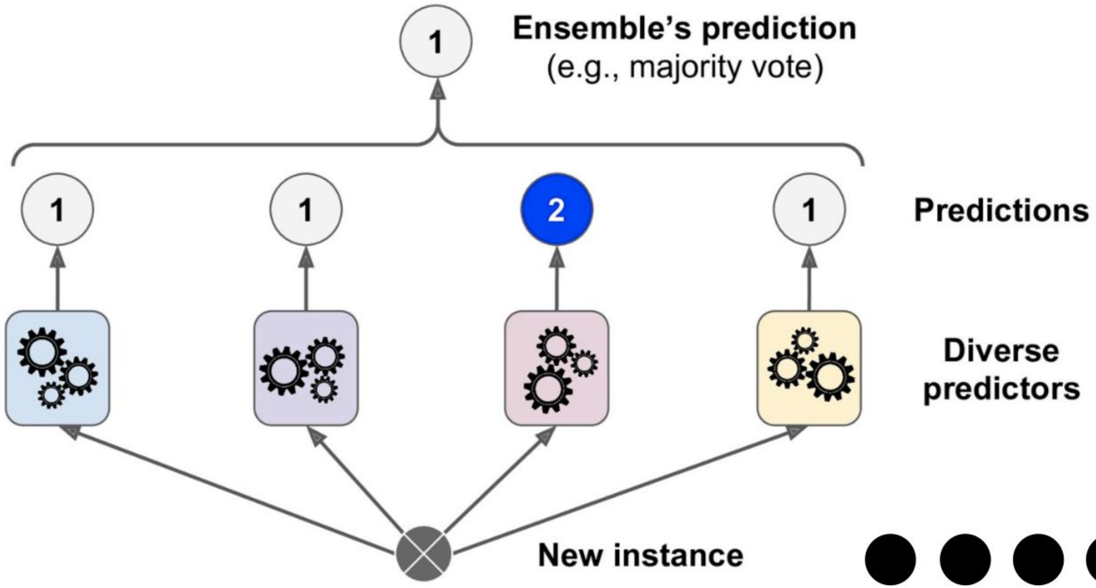
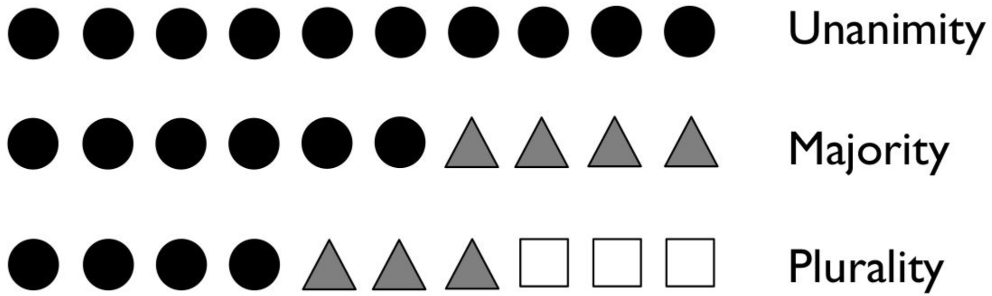


Figure 7-2. Hard voting classifier predictions



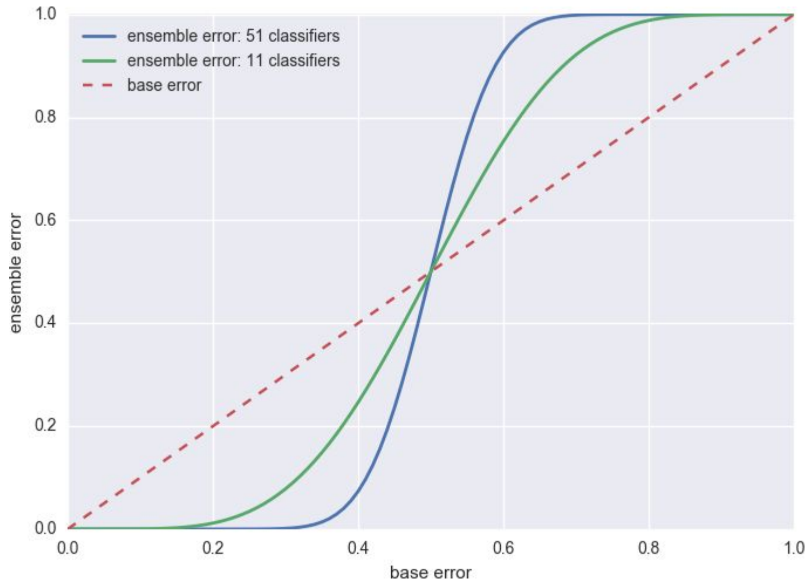
Why ensambles (majority voting) works?

As long as each base classifier does better than random guess (ϵ=0.5), the voting classifier further reduces the error. And when that happens, the more base classifiers the better. In a binary classifier we have:

Ensemble error:

$$\epsilon_{ens} = \sum_{k}^{n} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}$$

$$\epsilon_{ens} = \sum_{k=6}^{11} \binom{11}{k} 0.25^k (1-0.25)^{11-k} = 0.034$$



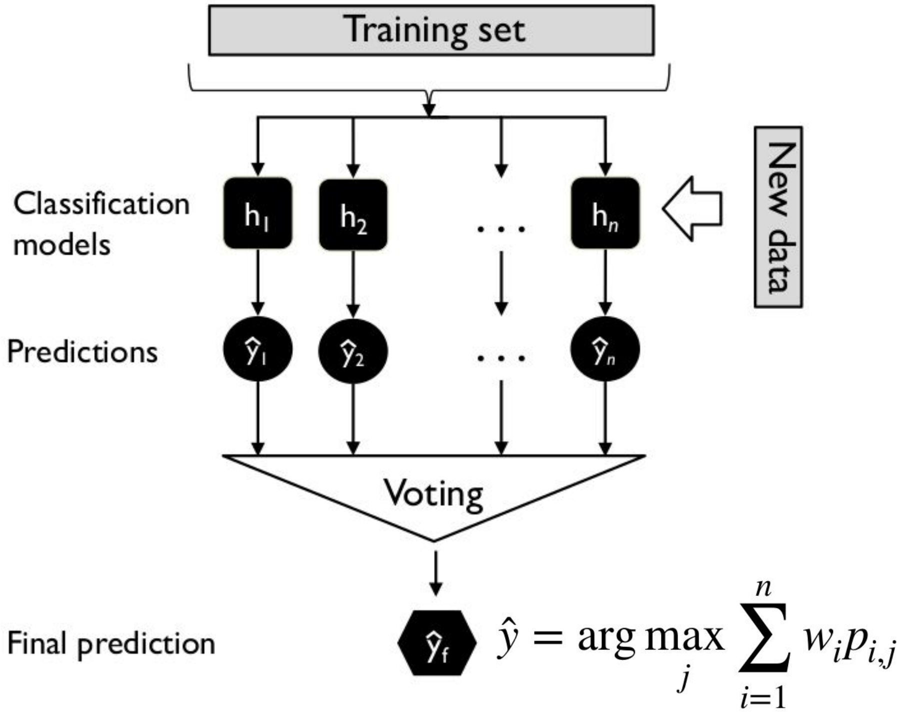
The assumption in this error reduction though:

- the base classifiers are not correlated

Weighted Average Probabilities (Soft Voting)

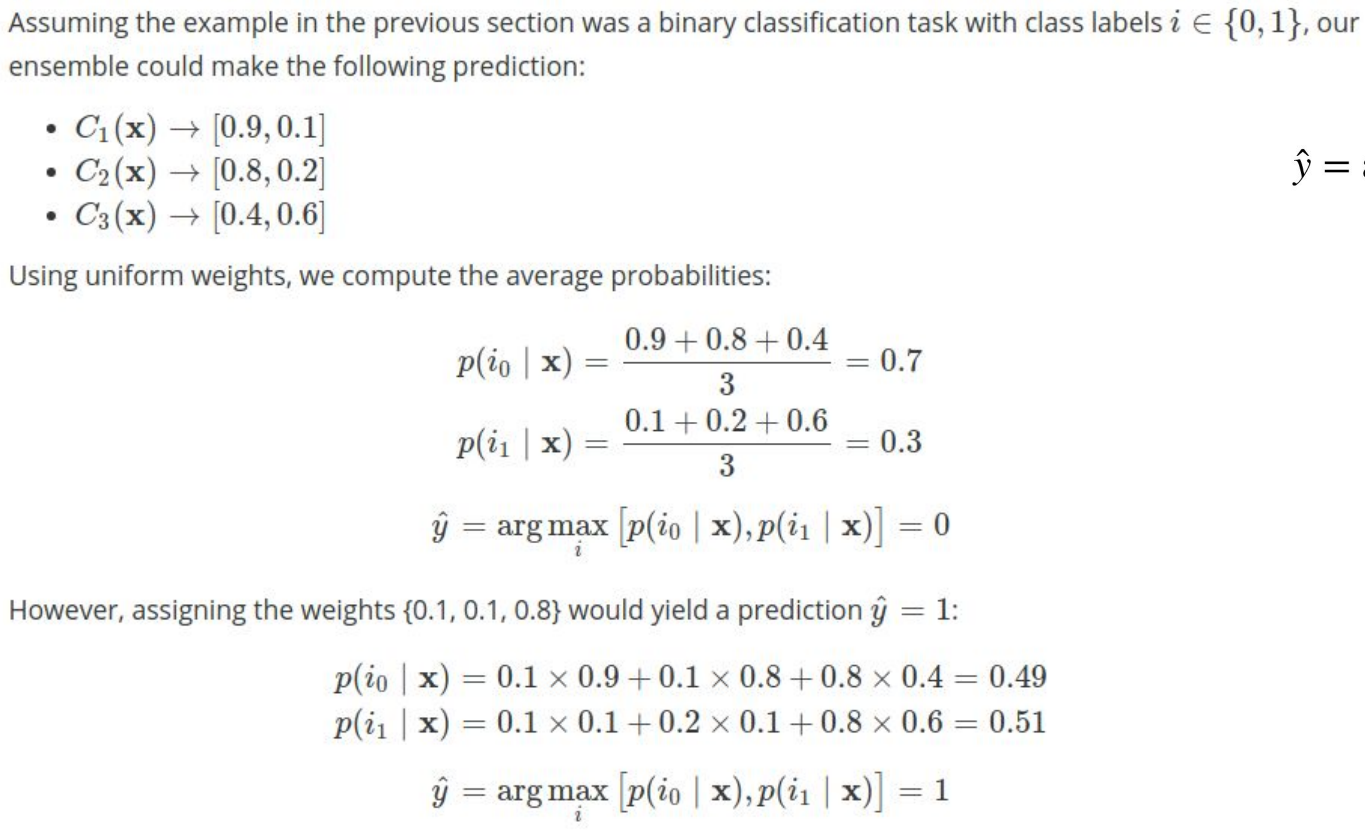
In contrast to majority voting (hard voting), soft voting returns the class label as argmax of the sum of predicted probabilities.

- $p_{i,j}$ predicted class membership probability of the ith classifier for class label j
- wi optional weighting parameter, default $w_i = 1/n$, $\forall w_i \in \{w_1, ..., w_n\}$



$$\hat{y} = \arg\max_j \sum_{i=1}^{n} w_i p_{i,j}$$

Weighted Average Probabilities (Soft Voting)



Assuming the example in the previous section was a binary classification task with class labels $i \in \{0, 1\}$, our ensemble could make the following prediction:

- $C_1(\mathbf{x}) \rightarrow [0.9, 0.1]$
- $C_2(\mathbf{x}) \rightarrow [0.8, 0.2]$
- $C_3(\mathbf{x}) \rightarrow [0.4, 0.6]$

Using uniform weights, we compute the average probabilities:

$$p(i_0 \mid \mathbf{x}) = \frac{0.9 + 0.8 + 0.4}{3} = 0.7$$
$$p(i_1 \mid \mathbf{x}) = \frac{0.1 + 0.2 + 0.6}{3} = 0.3$$

$$\hat{y} = \arg\max_i \left[p(i_0 \mid \mathbf{x}), p(i_1 \mid \mathbf{x})\right] = 0$$

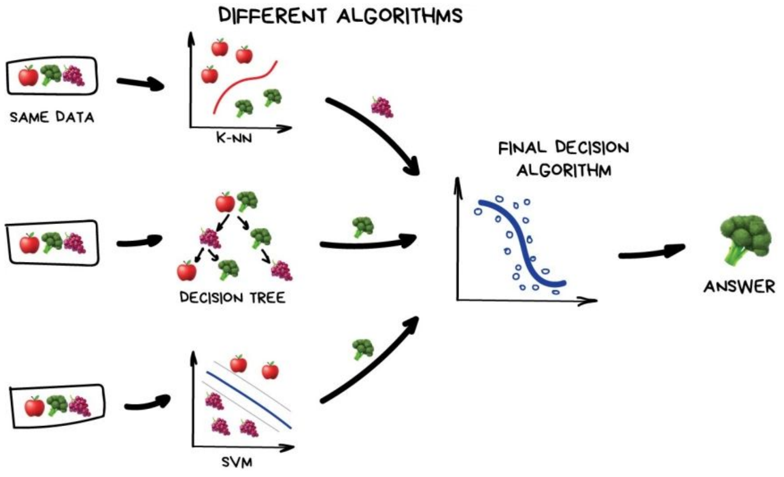
However, assigning the weights {0.1, 0.1, 0.8} would yield a prediction $\hat{y} = 1$:

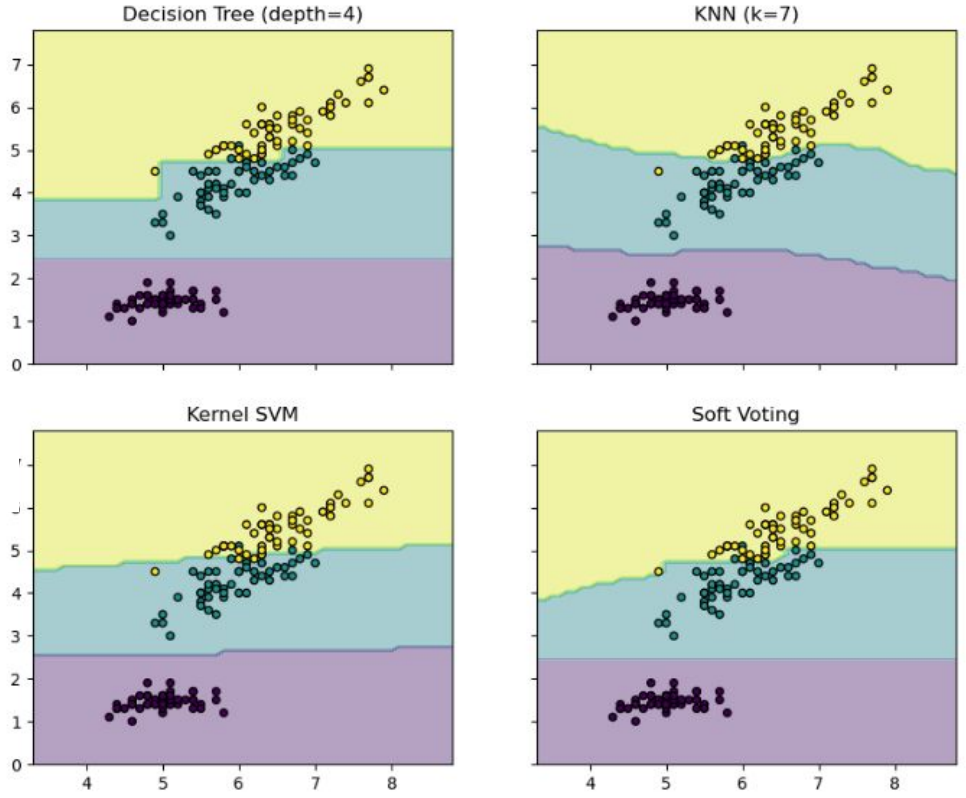
$$p(i_0 \mid \mathbf{x}) = 0.1 \times 0.9 + 0.1 \times 0.8 + 0.8 \times 0.4 = 0.49$$
$$p(i_1 \mid \mathbf{x}) = 0.1 \times 0.1 + 0.2 \times 0.1 + 0.8 \times 0.6 = 0.51$$

$$\hat{y} = \arg\max_i \left[p(i_0 \mid \mathbf{x}), p(i_1 \mid \mathbf{x})\right] = 1$$

$$\hat{y} = \arg\max_j \sum_{i=1}^{n} w_i p_{i,j}$$

Weighted Average Probabilities (Soft Voting)





Lecturer Overview

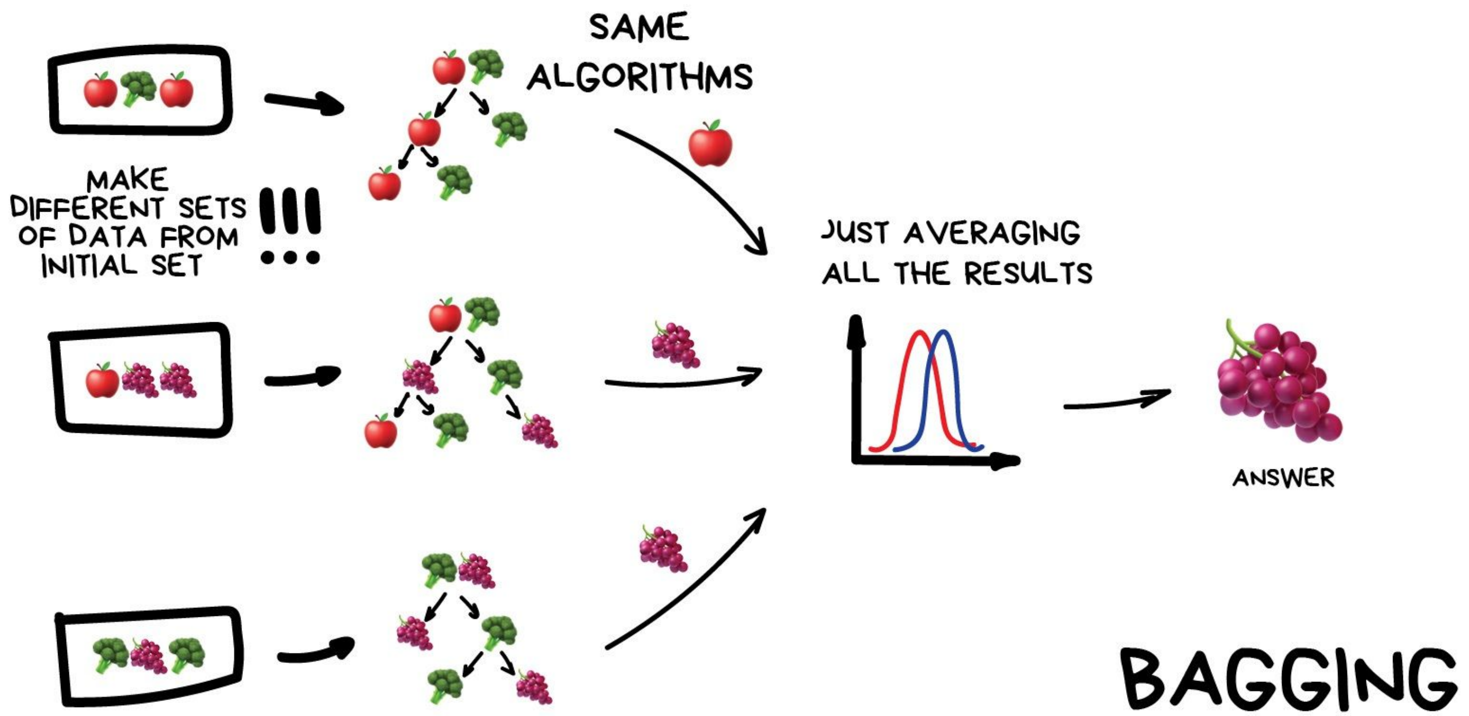
- Intro and overview
- Voting
- **Bagging**
- Random Forests
- Boosting
- Gradient Boosting
- Stacking

Bagging

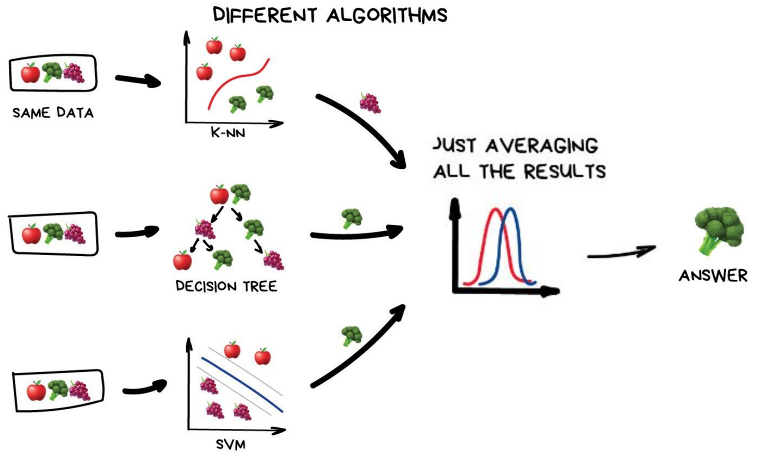
- Bootstrap Aggregating
    - Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

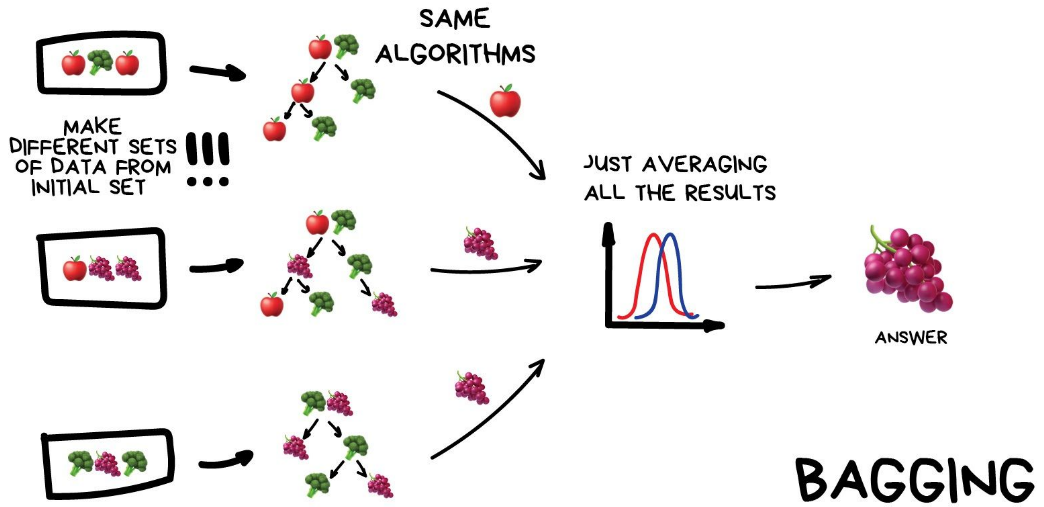
- Ensemble family
    - Average methods (several estimators independently)
- Uncorrelated errors (diversity)
    - manipulating the training examples
    - manipulating the input features

Bagging


SAME ALGORITHMS
MAKE DIFFERENT SETS OF DATA FROM INITIAL SET !!!
JUST AVERAGING ALL THE RESULTS
ANSWER
BAGGING

Voting x Bagging



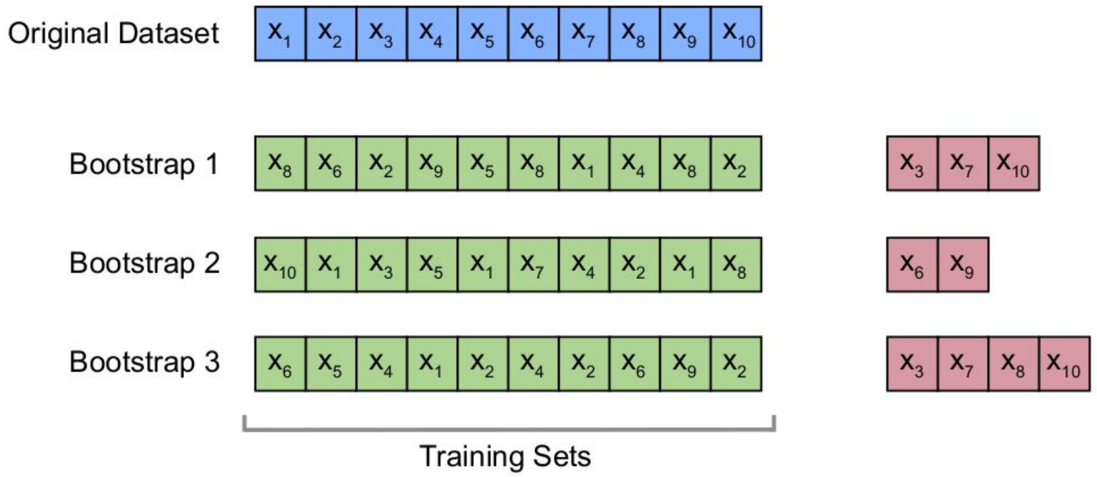

SAME ALGORITHMS
MAKE DIFFERENT SETS OF DATA FROM INITIAL SET !!!
JUST AVERAGING ALL THE RESULTS
ANSWER
BAGGING

Bagging

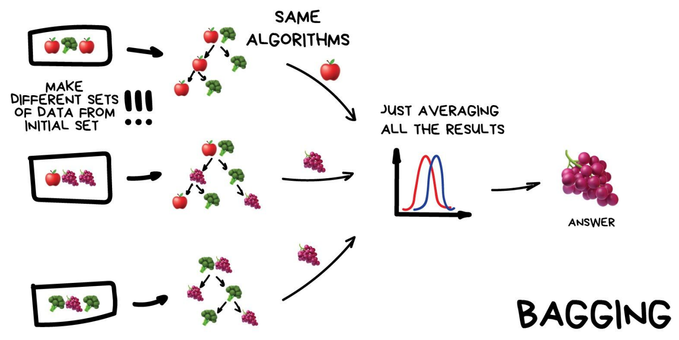
Average methods

Algorithm 1 Bagging

```
1: Let $n$ be the number of bootstrap samples
2:
3: for i=1 to $n$ do
4:     Draw bootstrap sample of size $m$, $\mathcal{D}_i$
5:     Train base classifier $h_i$ on $\mathcal{D}_i$
6: $\hat{y} = mode\{h_1(\mathbf{x}), ..., h_n(\mathbf{x})\}$
```

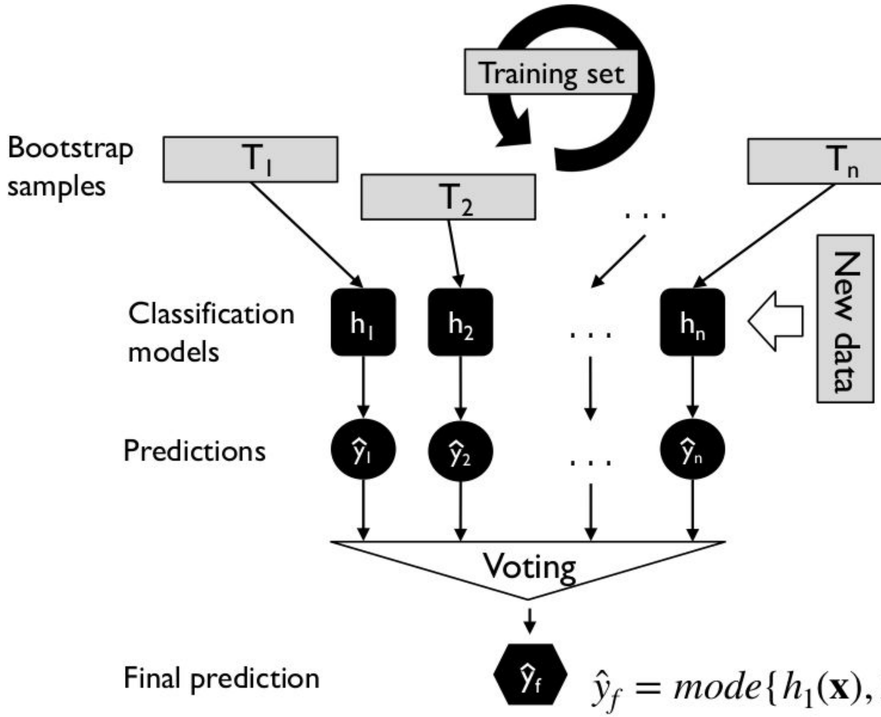


Original Dataset
X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
Bootstrap 1
X8 X6 X2 X9 X5 X8 X1 X4 X8 X2
X3 X7 X10
Bootstrap 2
X10 X1 X3 X5 X1 X7 X4 X2 X1 X8
X6 X9
Bootstrap 3
X6 X5 X4 X1 X2 X4 X2 X6 X9 X2
X3 X7 X8 X10
Training Sets



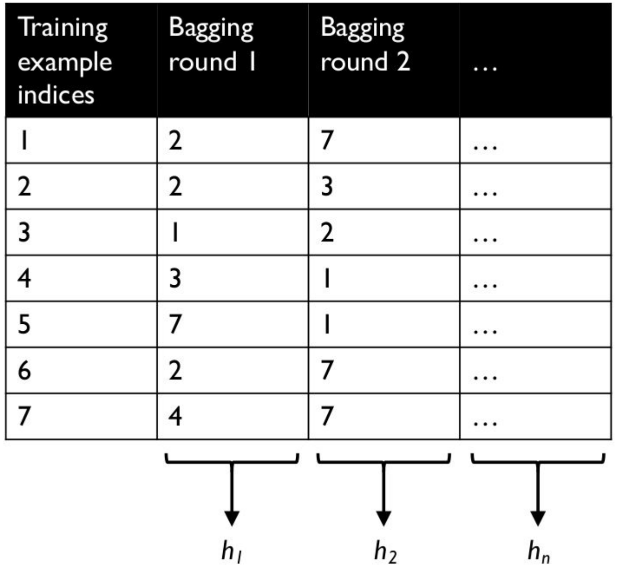
SAME ALGORITHMS
MAKE DIFFERENT SETS OF DATA FROM INITIAL SET !!!...
JUST AVERAGING ALL THE RESULTS
ANSWER
BAGGING

Bagging



$$\hat{y}_f = mode\{h_1(\mathbf{x}), h_2(\mathbf{x}), \dots h_n(\mathbf{x})\}$$

where $h_i(\mathbf{x}) = \hat{y}_i$



Training example indices	Bagging round 1	Bagging round 2	…
1	2	7	…
2	2	3	…
3	1	2	…
4	3	1	…
5	7	1	…
6	2	7	…
7	4	7	…

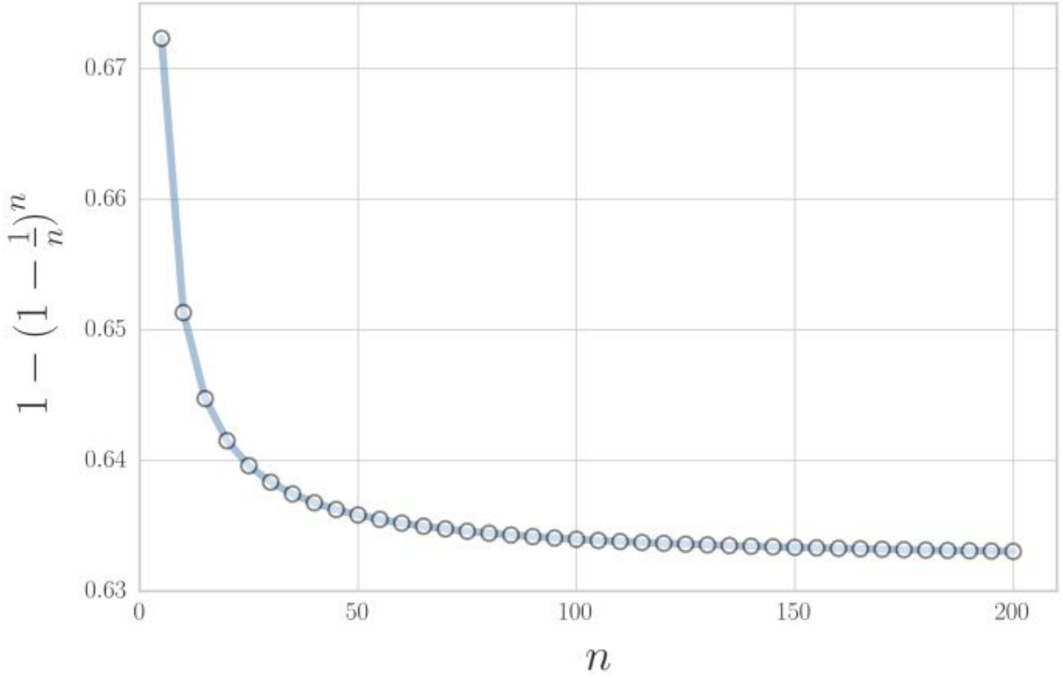
Bagging

Sampling probability

$$P(\textbf{not chosen}) = \left(1 - \frac{1}{n}\right)^n,$$

$$\frac{1}{e} \approx 0.368, \quad n \to \infty\,.$$

$$P(\textbf{chosen}) = 1 - \left(1 - \frac{1}{n}\right)^n \approx 0.632$$



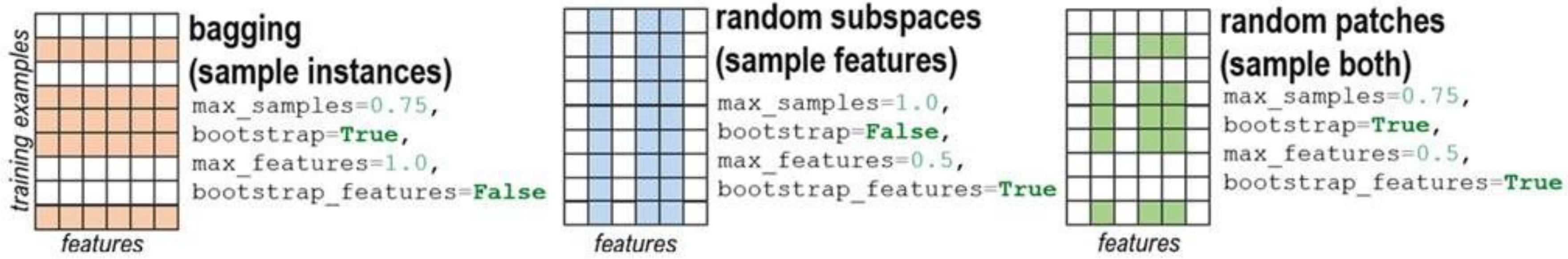
Bagging methods

Bagging methods form a class of algorithms which build several instances of a black-box estimator on random subsets of the original training set and then aggregate their individual predictions to form a final prediction.

Bagging methods come in **many flavours** but mostly **differ** from each other by the **way they draw random subsets** of the training set:

- When random subsets of the dataset are drawn as **random subsets of the samples**, then this algorithm is known as **Pasting** [B1999].
- When **samples are drawn with replacement**, then the method is known as **Bagging** [B1996].
- When random subsets of the dataset are drawn as **random subsets of the features**, then the method is known as **Random Subspaces** [H1998].
- Finally, when base estimators are built on **subsets of both samples and features**, then the method is known as **Random Patches** [LG2012].

Bagging methods



bagging (sample instances)

```
max_samples=0.75,
bootstrap=True,
max_features=1.0,
bootstrap_features=False
```

random subspaces (sample features)

```
max_samples=1.0,
bootstrap=False,
max_features=0.5,
bootstrap_features=True
```

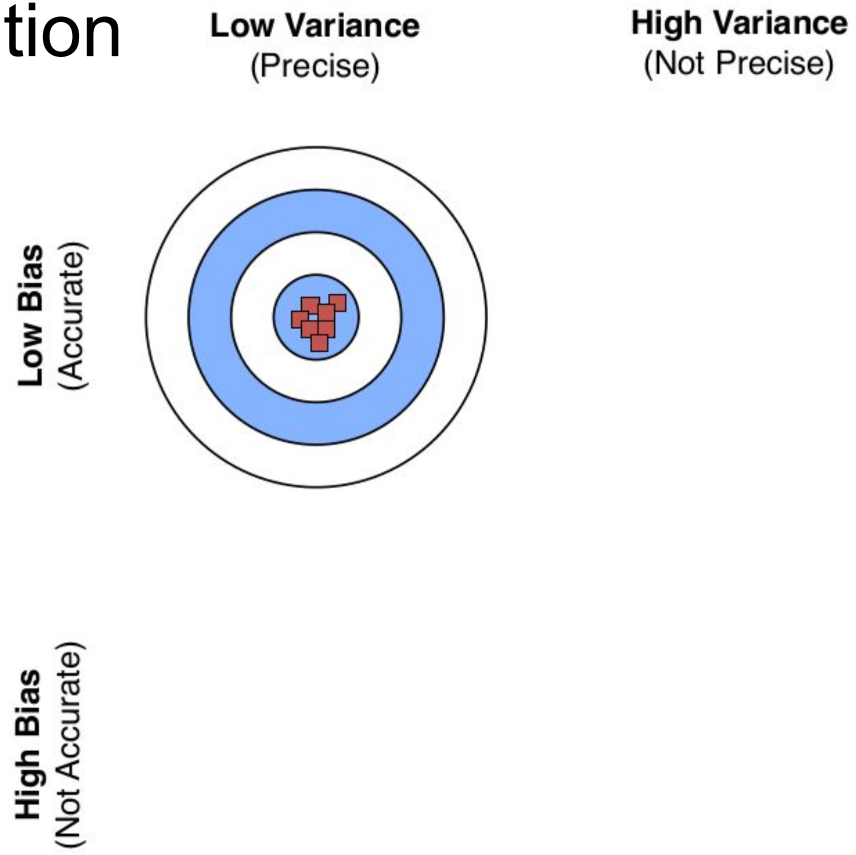
random patches (sample both)

```
max_samples=0.75,
bootstrap=True,
max_features=0.5,
bootstrap_features=True
```

Bias-Variance Decomposition

Loss = Bias + Variance + Noise

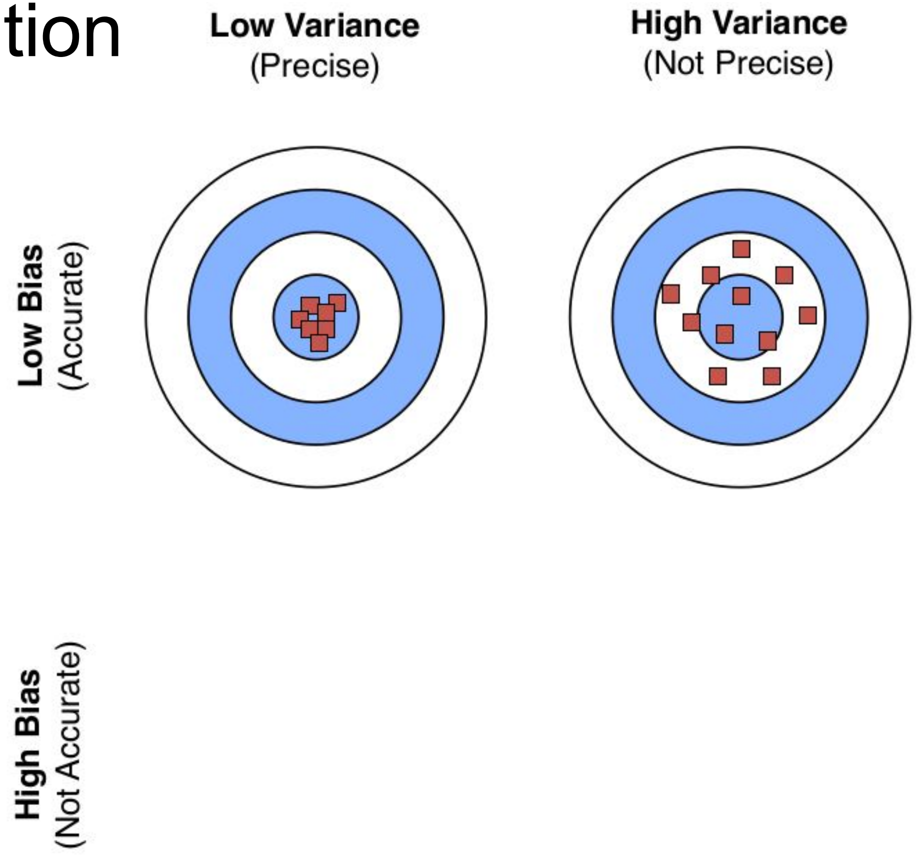
(more technical details in next lecture on model evaluation)



Bias-Variance Decomposition

Loss = Bias + Variance + Noise

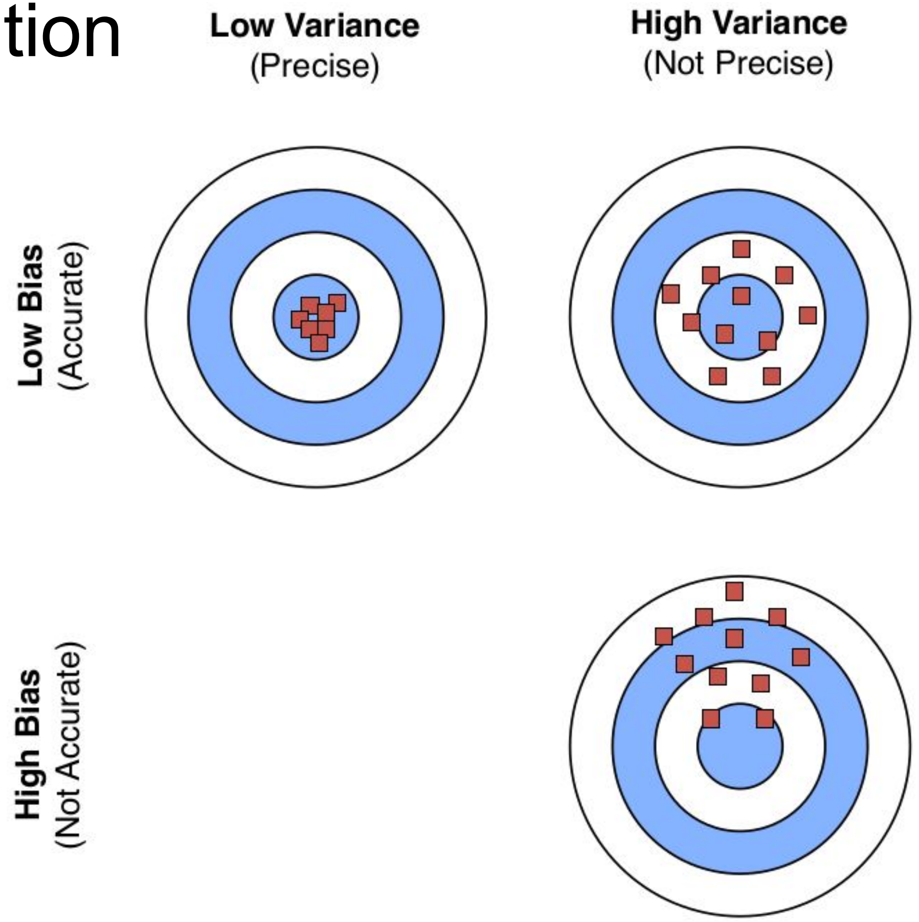
(more technical details in next lecture on model evaluation)



Bias-Variance Decomposition

Loss = Bias + Variance + Noise

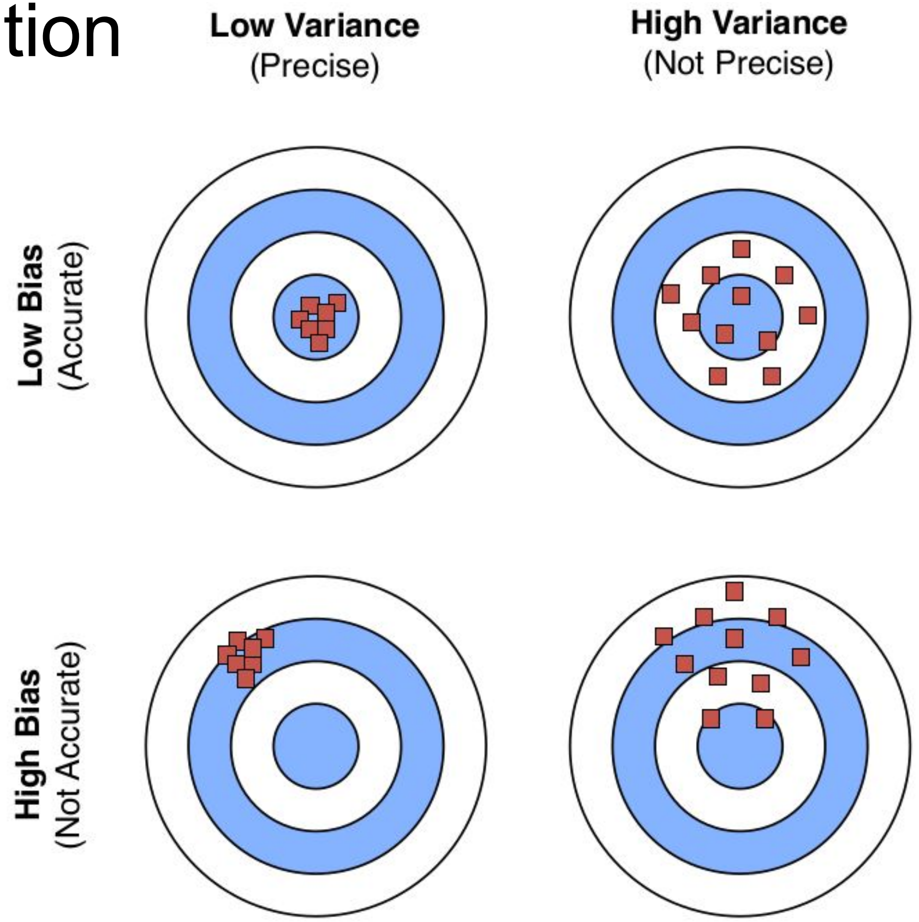
(more technical details in next lecture on model evaluation)



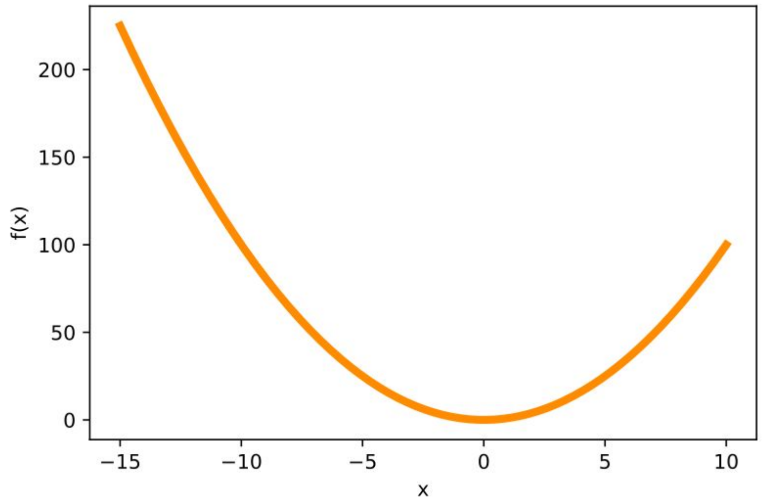
Bias-Variance Decomposition

Loss = Bias + Variance + Noise

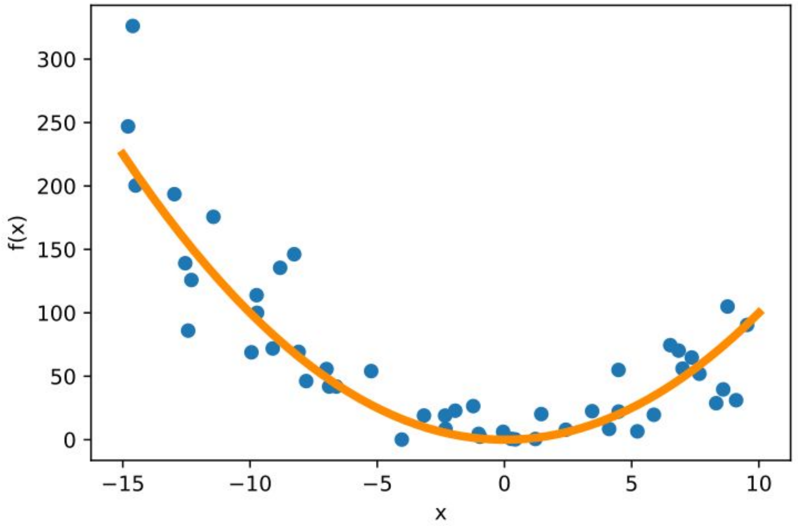
(more technical details in next lecture on model evaluation)



Bias and Variance Example



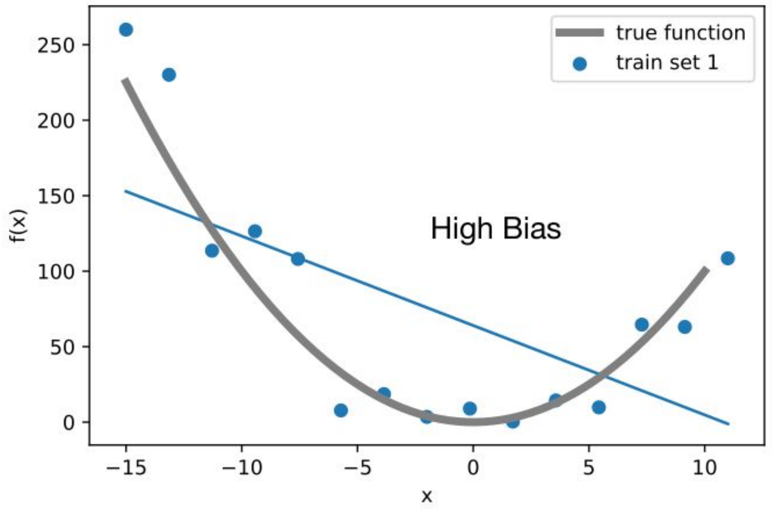
where f(x) is some true (target) function



where f(x) is some true (target) function

the blue dots are a training dataset;
here, I added some random Gaussian noise

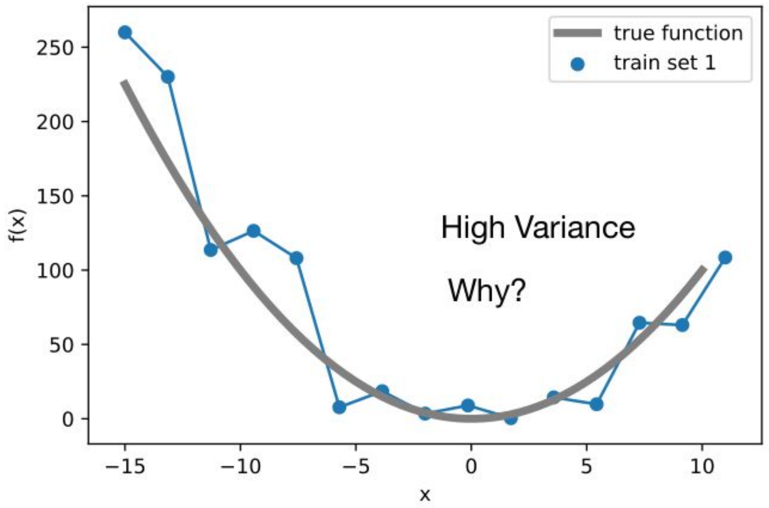
Bias and Variance Example



where f(x) is some true (target) function

the blue dots are a training dataset;
here, I added some random Gaussian noise

here, suppose I fit a simple linear model (linear regression)
or a decision tree stump

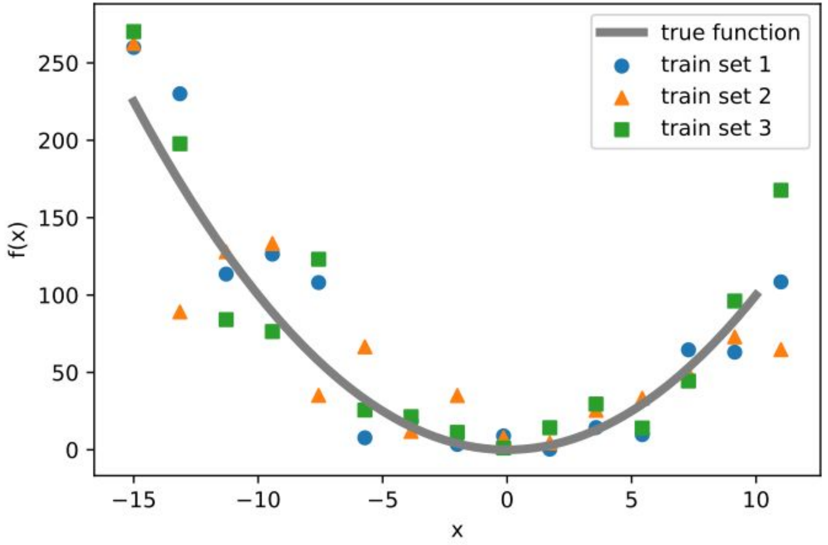


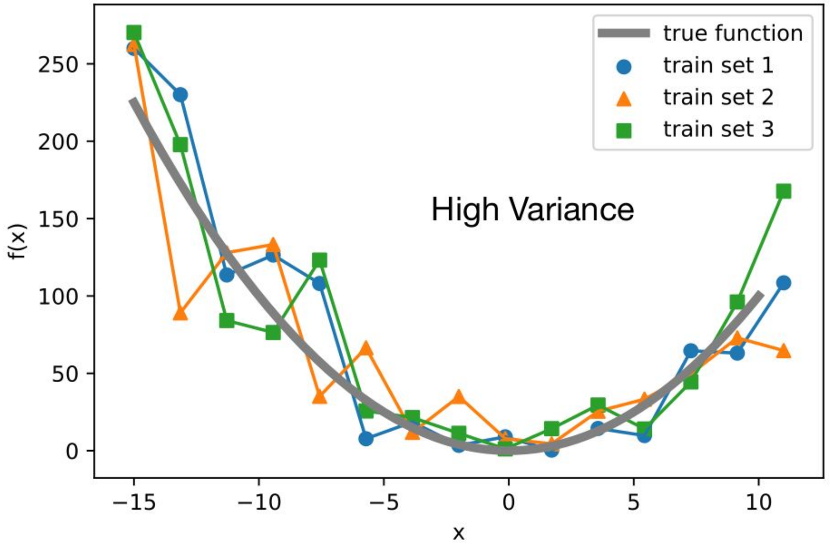
where f(x) is some true (target) function

the blue dots are a training dataset;
here, I added some random Gaussian noise

here, suppose I fit an unpruned decision tree

Bias and Variance Example





where f(x) is some true (target) function

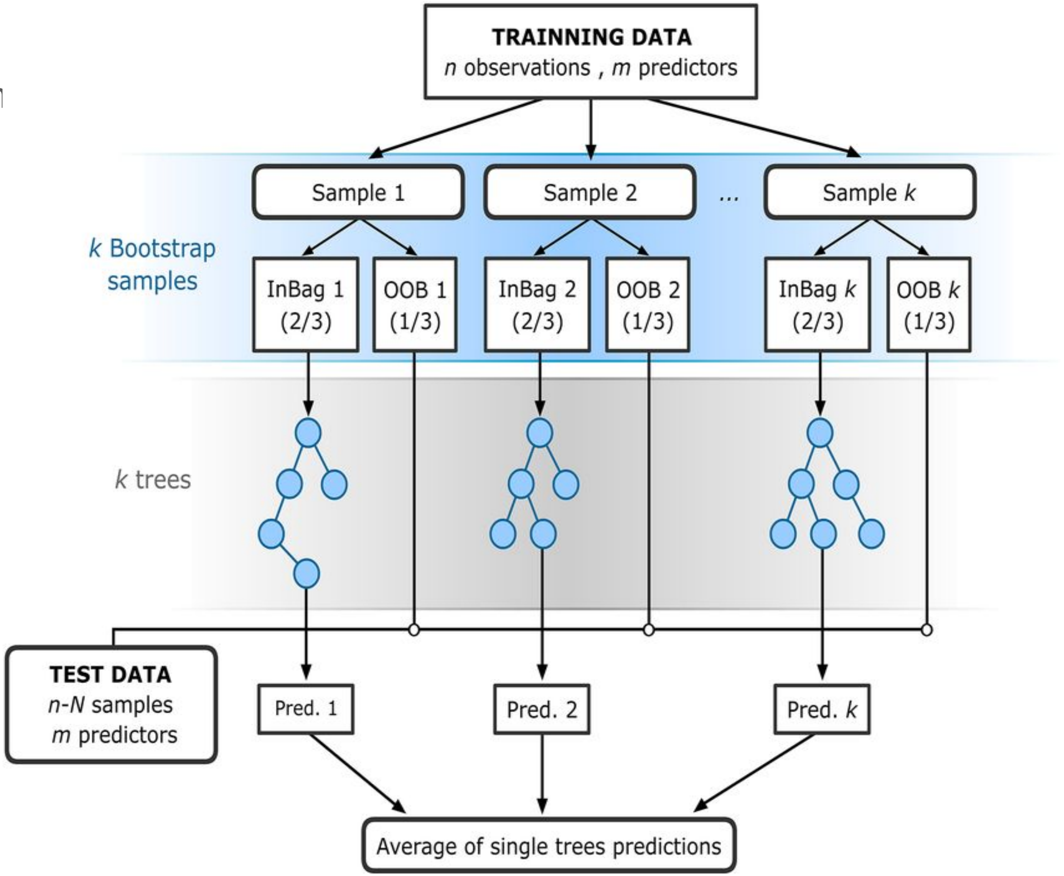
suppose we have multiple training sets

Lecturer Overview

- Intro and overview
- Voting
- Bagging
- **Random Forests**
- Boosting
- Gradient Boosting
- Stacking

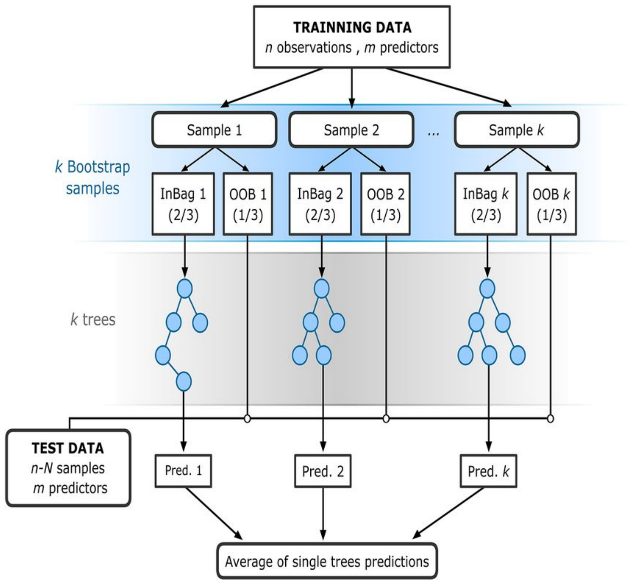
Random Forests

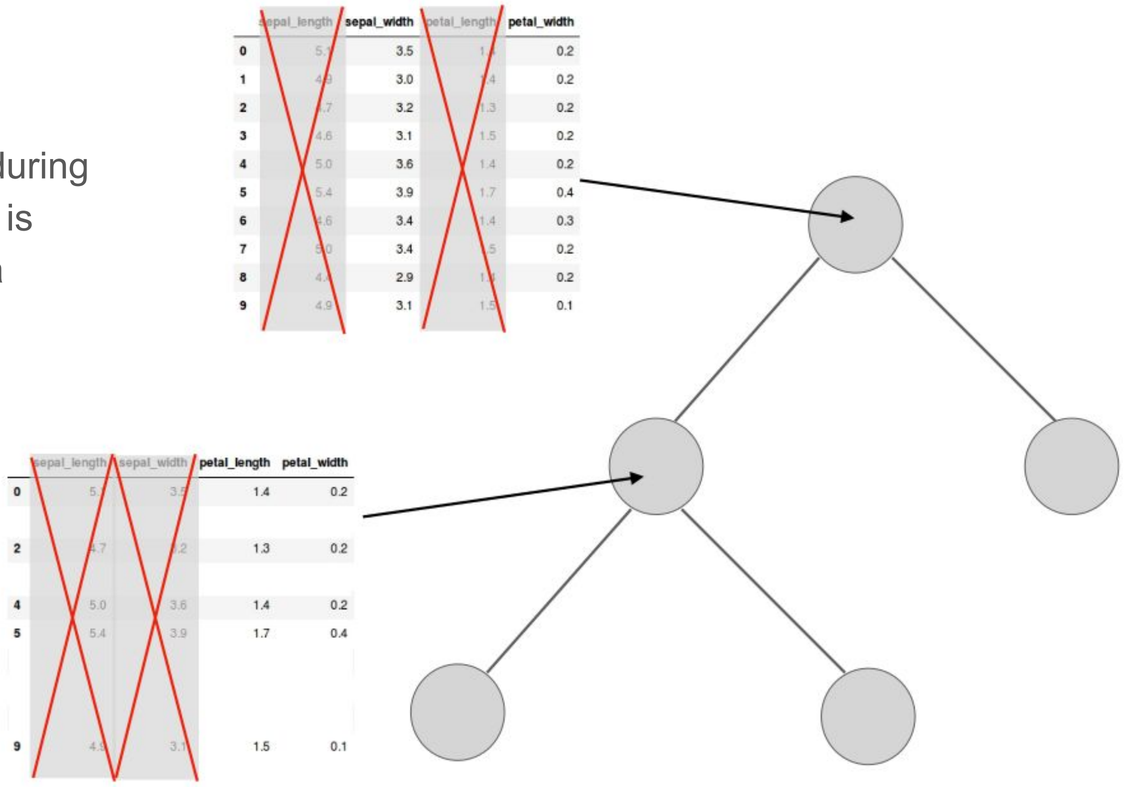
Each tree in the ensemble is built from a san
drawn with replacement (i.e., a bootstrap
sample) from the training set.



Random Forests

Furthermore, when splitting each node during the construction of a tree, the **best split** is **found** either from **all input** features or a **random subset of features**



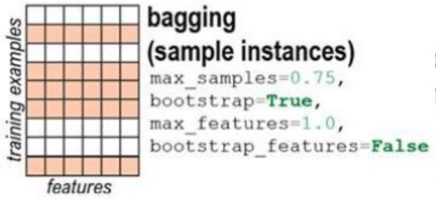


	~~sepal_length~~	sepal_width	~~petal_length~~	petal_width
0	~~5.1~~	3.5	~~1.4~~	0.2
1	~~4.9~~	3.0	~~1.4~~	0.2
2	~~4.7~~	3.2	~~1.3~~	0.2
3	~~4.6~~	3.1	~~1.5~~	0.2
4	~~5.0~~	3.6	~~1.4~~	0.2
5	~~5.4~~	3.9	~~1.7~~	0.4
6	~~4.6~~	3.4	~~1.4~~	0.3
7	~~5.0~~	3.4	~~1.5~~	0.2
8	~~4.4~~	2.9	~~1.4~~	0.2
9	~~4.9~~	3.1	~~1.5~~	0.1

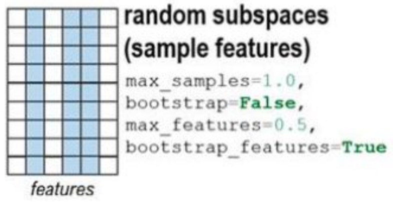
	~~sepal_length~~	~~sepal_width~~	petal_length	petal_width
0	~~5.1~~	~~3.5~~	1.4	0.2
2	~~4.7~~	~~3.2~~	1.3	0.2
4	~~5.0~~	~~3.6~~	1.4	0.2
5	~~5.4~~	~~3.9~~	1.7	0.4
9	~~4.9~~	~~3.1~~	1.5	0.1

Random Forests

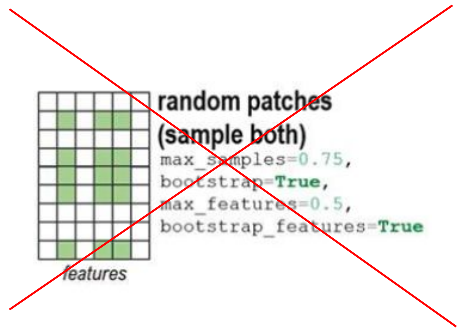
Bagging w. trees + random feature subsets



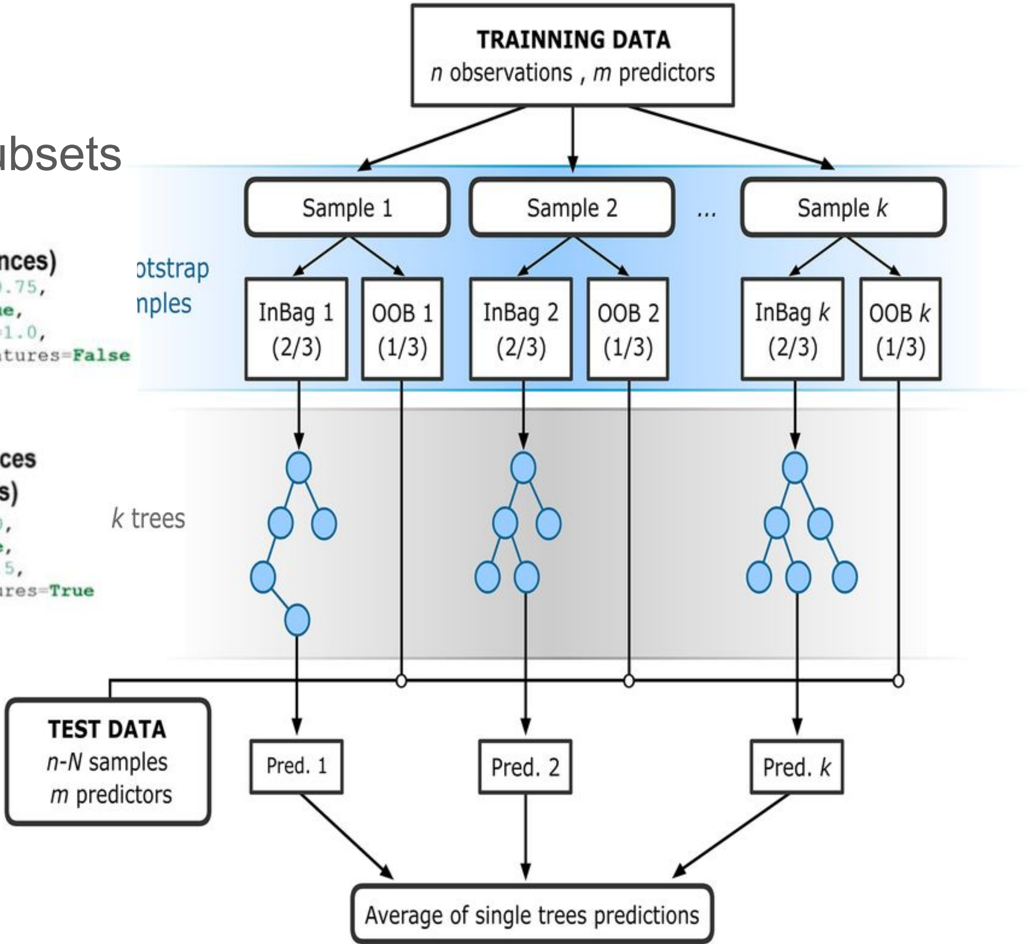
```
max_samples=0.75,
bootstrap=True,
max_features=1.0,
bootstrap_features=False
```



```
max_samples=1.0,
bootstrap=False,
max_features=0.5,
bootstrap_features=True
```



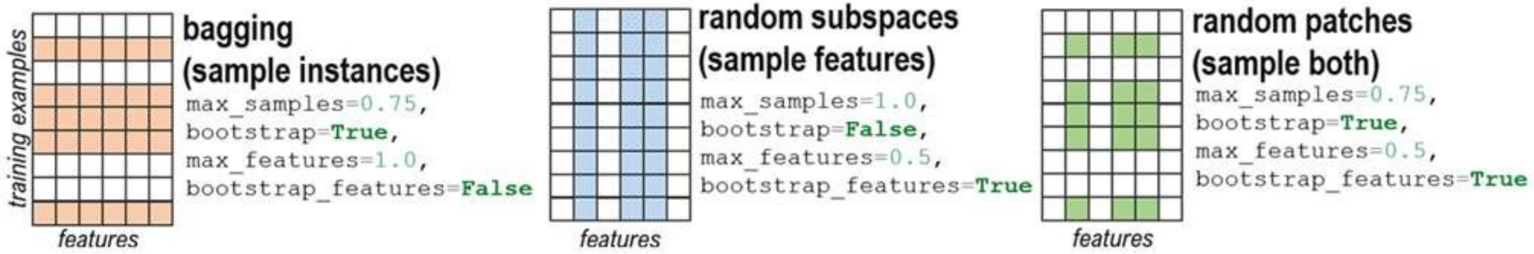
```
max_samples=0.75,
bootstrap=True,
max_features=0.5,
bootstrap_features=True
```



Random Feature Subset for each Tree or Node?

- Ho, Tin Kam. “The **random subspace** method for constructing decision forests.” IEEE transactions on pattern analysis and machine intelligence 20.8 (1998): 832-844.
  - “Our method relies on an autonomous, pseudo-random procedure to select a small number of dimensions from a given feature space ...”
  - Tin Kam Ho used the “random subspace method,” where **each tree got a random subset of features**.

- Breiman, Leo. “**Random Forests**” Machine learning 45.1 (2001): 5-32.
  - “...random forest with random features is formed by selecting at random, at **each node, a small group of input variables** to split on.”



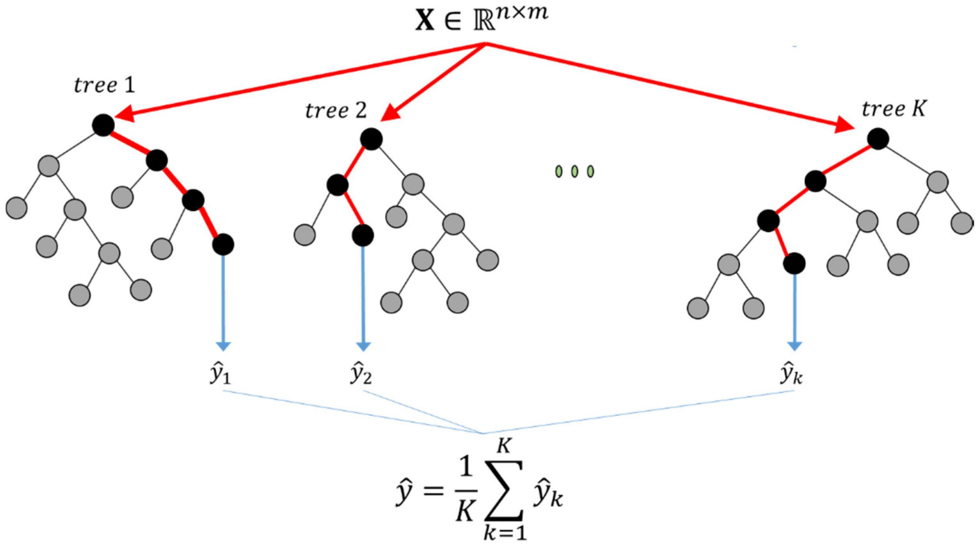
```
max_samples=0.75,
bootstrap=True,
max_features=1.0,
bootstrap_features=False
```

```
max_samples=1.0,
bootstrap=False,
max_features=0.5,
bootstrap_features=True
```

```
max_samples=0.75,
bootstrap=True,
max_features=0.5,
bootstrap_features=True
```

Soft voting x Hard Voting

- Breiman, Leo. “**Random Forests**” Machine learning 45.1 (2001): 5-32
  - In contrast to the original publication the scikit-learn implementation combines classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class.
  - Soft voting x Hard Voting


$\mathbf{X} \in \mathbb{R}^{n\times m}$
$tree\ 1$
$tree\ 2$
$tree\ K$
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_k$
$\hat{y} = \frac{1}{K}\sum_{k=1}^{K} \hat{y}_k$

(Loose) Upper Bound for the Generalization Error

Breiman, Leo. “Random Forests” Machine learning 45.1 (2001): 5-32

$$\text{PE} \leq \frac{\bar{\rho} \cdot (1 - s^2)}{s^2}$$

$\bar{\rho}$	: Average correlation among trees
s	: "Strength" of the ensemble

Extremely Randomized Trees (ExtraTrees)

Random Forest random components:

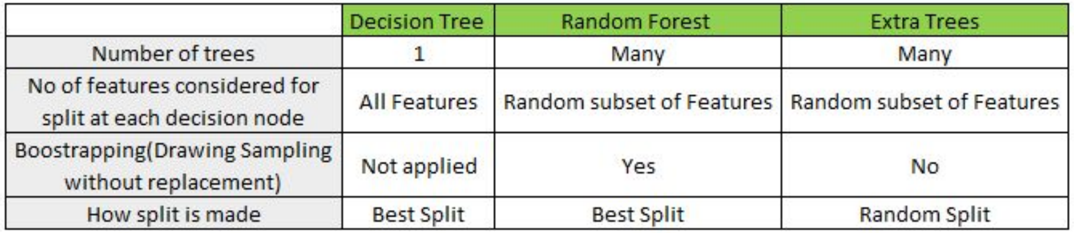
- **each tree** in the ensemble is built from a sample drawn with replacement (i.e., a **bagging**) from the training set
- when splitting each node during the construction of a tree, the **best split** is found on random subset of features (i.e. **random subspaces**)

ExtraTrees algorithm adds one more random component

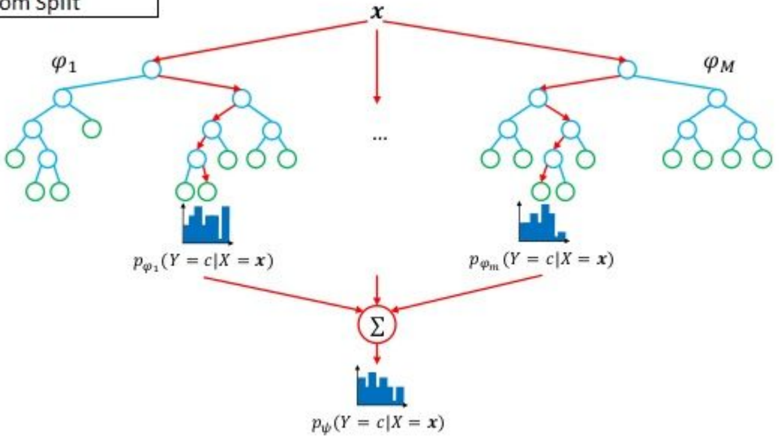
- instead of looking for the most discriminative thresholds, **thresholds are drawn at random** for each candidate feature and t**he best of these randomly-generated** thresholds **is picked as the splitting rule**

This usually allows to **reduce the variance** of the model a bit more, **at the expense of a slightly greater increase in bias**

Extremely Randomized Trees (ExtraTrees)



	Decision Tree	Random Forest	Extra Trees
Number of trees	1	Many	Many
No of features considered for split at each decision node	All Features	Random subset of Features	Random subset of Features
Boostrapping(Drawing Sampling without replacement)	Not applied	Yes	No
How split is made	Best Split	Best Split	Random Split


$\boldsymbol{x}$
φ_1
φ_M
...
$p_{\varphi_1}(Y = c|X = \boldsymbol{x})$
$p_{\varphi_m}(Y = c|X = \boldsymbol{x})$
Σ
$p_{\psi}(Y = c|X = \boldsymbol{x})$

Randomization

- Bootstrap samples
- Random selection of $K \leqslant p$ split variables
- Random selection of the threshold

$\left.\begin{array}{l}\text{Bootstrap samples}\\ \text{Random selection of } K \leqslant p \text{ split variables}\end{array}\right\}$ Random Forests

$\left.\begin{array}{l}\text{Random selection of } K \leqslant p \text{ split variables}\\ \text{Random selection of the threshold}\end{array}\right\}$ Extra-Trees

Lecturer Overview

- Intro and overview
- Voting
- Bagging
- Random Forests
- **Boosting**
- Gradient Boosting
- Stacking