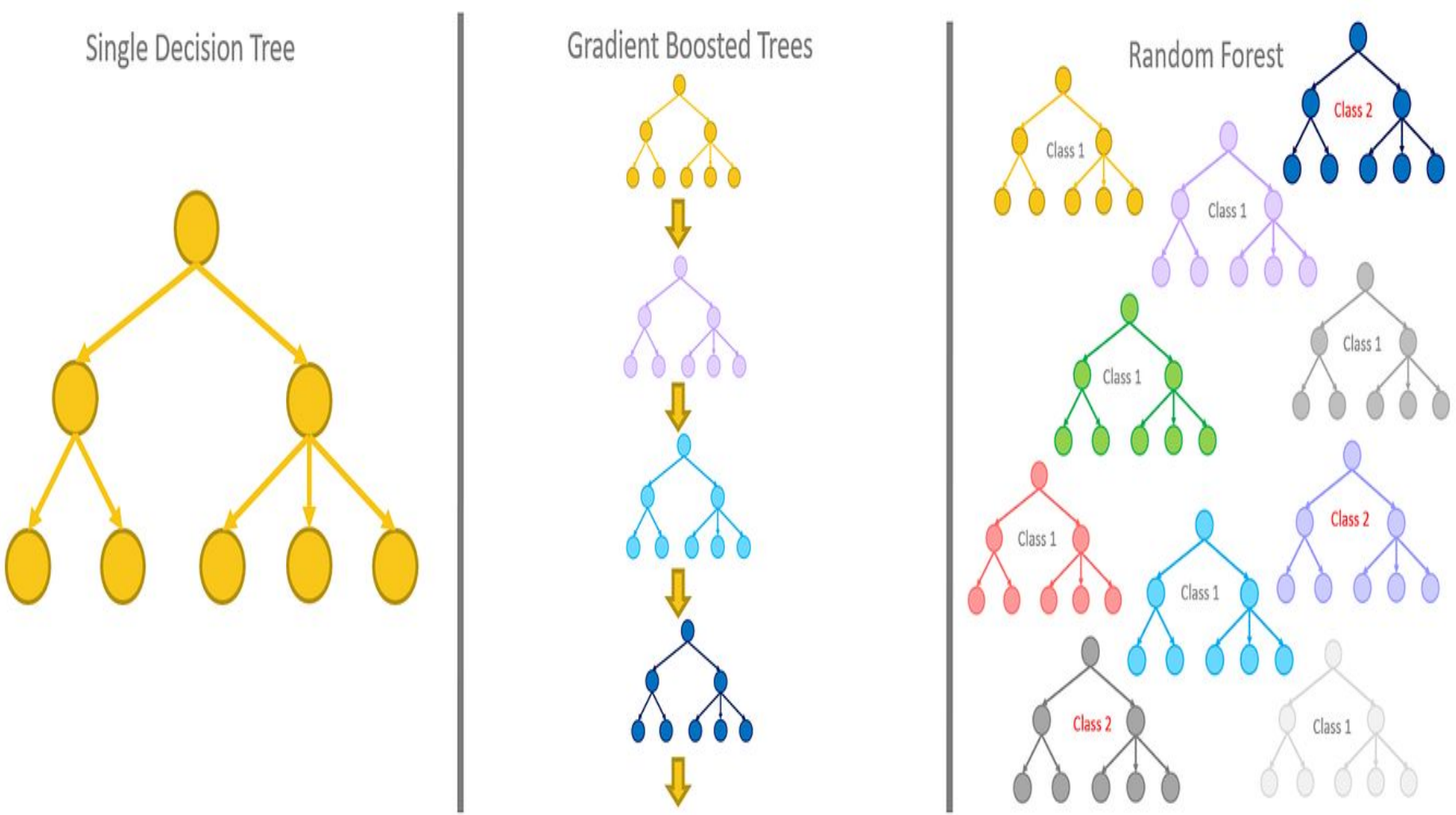


Dr. Nilina Bera | CSE(DS) – DSC3101 | HITK

Ensemble Methods: Bagging, Boosting, Random Forests, FP-growth algorithm, Sub-graph mining

Course Name: Introduction to Data Mining					
Course Code: DSC3101					
Contact Hours per week:	L	T	P	Total	Credit points
	3	0	0	3	3

Module 3 [9L]

Ensemble Methods: Bagging, Boosting, Random Forests, FP-growth algorithm, Sub-graph mining Association Rule Mining: Introduction, rules and item-set generation, Frequent item-set generation, (Apriori principle, candidate generation and pruning), Compact representation of frequent item sets, correlation analysis,

What is an ensemble method?

Ensemble models in data mining & KDD combine the decisions from multiple models to improve the overall performance.

- The main causes of error in learning models are due to noise, bias and variance. Ensemble methods help to minimize these factors.
- These methods are designed to improve the stability and the accuracy of Machine Learning algorithms.

noise, bias and variance ???

A supervised data mining model aims to train itself on the input variables(X) in such a way that the predicted values(Y) are as close to the actual values as possible. This difference between the actual values and predicted values is the error and it is used to evaluate the model. The error for any supervised data mining algorithm comprises of 3 parts:

- Bias error
- Variance error
- The noise

While the noise is the irreducible error that we cannot eliminate, the other two i.e. Bias and Variance are reducible errors that we can attempt to minimize as much as possible.

What is Ensemble Learning? Why use ensemble models? When and where to use ensemble models?

Ensemble Learning

Definition

Ensemble Learning is a machine learning technique where **multiple models (called base learners or weak learners) are combined to form a stronger, more accurate predictive model.**

- Each individual model may make errors.
- By aggregating them (through voting, averaging, or stacking), ensemble methods improve **accuracy, stability, and generalization.**

It is often described as the "**wisdom of the crowd**" in machine learning.

Why use Ensemble Models?

1. **Higher Accuracy** – Combines strengths of many models → better than a single one.
2. **Reduced Variance** – Averaging over many models (like bagging) smooths out noise and overfitting.
3. **Reduced Bias** – Boosting methods correct errors step by step, lowering bias.
4. **Robustness** – Works well even when individual models are unstable or weak learners.
5. **Handles Complex Problems** – Useful for high-dimensional, nonlinear, or noisy data.

When and Where to Use Ensemble Models?

Use Ensembles when:

- You need **high prediction accuracy** (e.g., Kaggle competitions, critical ML applications).
- Base models are **weak/unstable** (like decision trees, which change a lot with small data changes).
- Dataset is **complex, noisy, or imbalanced**, where single models fail.
- You want a **robust, generalizable** model.

Avoid Ensembles when:

- **Interpretability** is more important than accuracy (ensembles are often "black boxes").
- **Computation/Latency** matters (ensembles can be slow and memory-heavy).
- Small datasets where a **simple model already performs well**.

Examples of Where They're Used

- **Finance:** Fraud detection, credit scoring.
- **Healthcare:** Disease prediction, medical imaging classification.
- **E-commerce:** Recommendation systems, customer churn prediction.
- **Competitions:** Kaggle winners almost always use ensembles (like XGBoost + stacking).

Introduction to Ensemble Methods

- - Combine multiple models
- - Reduce variance or bias
- - Types: Bagging, Boosting, Random Forests

Bagging (Bootstrap Aggregating)

- - Uses bootstrapped data subsets
- - Trains base learners independently
- - Final output by majority vote/average
- - Reduces variance

Boosting

- - Sequential model training
- - Focuses on previous errors
- - Examples: AdaBoost, Gradient Boosting
- - Reduces bias

Random Forests

- - Ensemble of decision trees
- - Uses bagging + random feature selection
- - Final prediction by majority vote

Introduction to Association Rule Mining

- - Discover relationships in large datasets
- - Format: {A} $\Rightarrow$ {B}
- - Applications: Market Basket Analysis

Rules and Item-set Generation

- - Itemset: Items appearing together
- - Rule: If {A, B} then {C}
- - Measures: Support, Confidence, Lift

Frequent Item-set Generation

- - Items that occur frequently
- - Must meet minimum support threshold

Apriori Principle

- - If an itemset is frequent, all its subsets are also frequent
- - Helps prune the search space

Candidate Generation & Pruning

- - Generate k-itemsets from (k−1)-itemsets
- - *Prune candidates with infrequent subsets*
- "Prune candidates with infrequent subsets" is a key step in the Apriori algorithm for association rule mining, which uses the Apriori property: if an itemset has an infrequent subset, then that itemset must also be infrequent. Candidates are generated by combining frequent itemsets from the previous iteration, and then this pruning step efficiently removes any candidate itemsets that contain an infrequent subset, significantly reducing the number of candidates that need their support counted and speeding up the overall process.

Compact Representation of Frequent Itemsets

- - Use closed/maximal itemsets
- - Reduces memory and computational cost

Correlation Analysis in ARM

- - Measures: Lift, Conviction, Leverage
- - Identifies interesting rules
- - Filters weak correlations

FP-Growth Algorithm

- - Avoids candidate generation
- - Uses FP-Tree
- - Applies conditional pattern bases and recursive mining

Sub-graph Mining

- - Finds frequent subgraphs in graphs
- - Applications: Bioinformatics, Social Networks
- - Algorithms: gSpan, FSG

Ensemble Method

- The classification techniques (eg., decision trees, rule based, naive-based & SVM classifiers) predict the class labels of unknown examples (i.e., test data) using a single classifier induced from training data (predict Y, the class value, when X, attribute values are known).
- We need to apply techniques for improving classification accuracy by aggregating the predictions of multiple classifiers.
- These techniques are known as the ensemble or classifier combination methods.
- *An ensemble method constructs a set of base classifiers (eg., decision trees, rule based, naive-based & SVM classifiers) from training data and performs classification by taking a vote on the predictions made by each base classifier*.

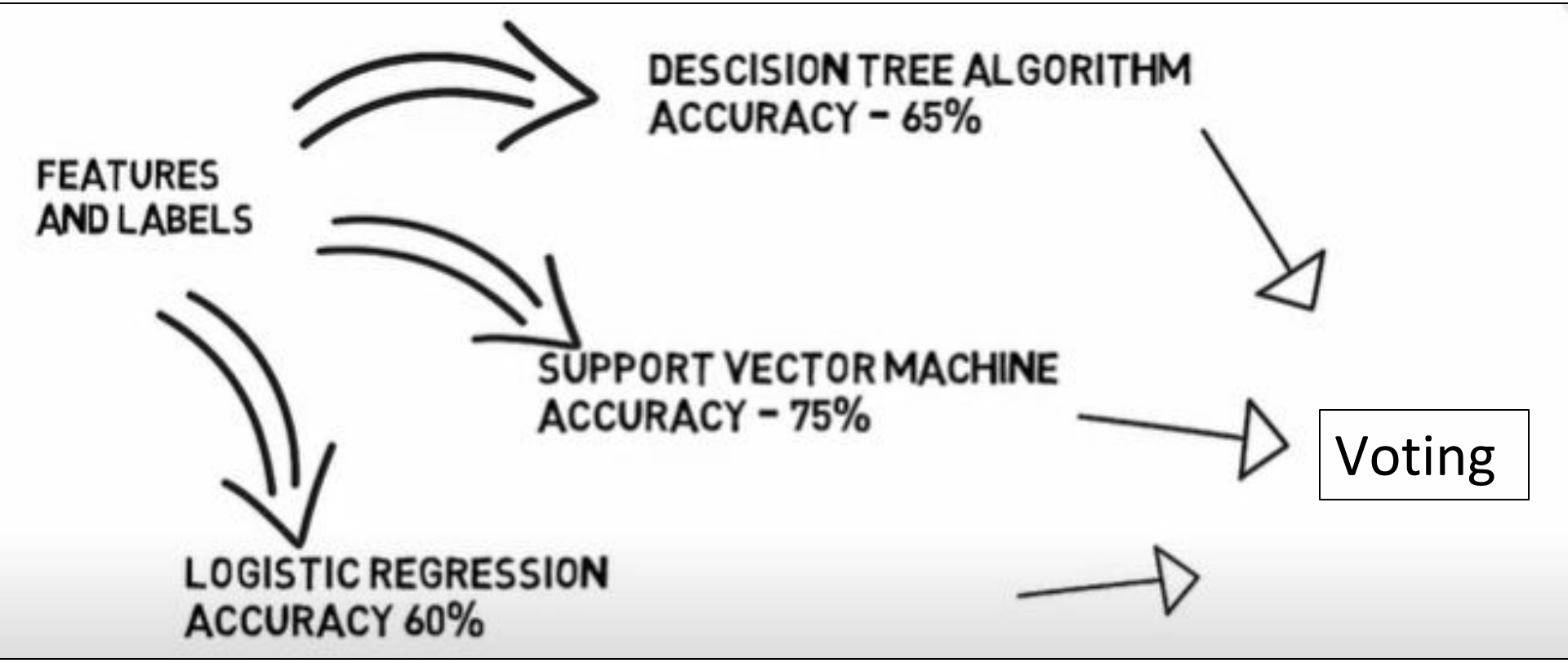
What is Ensemble Learning? Why use ensemble models? When and where to use ensemble models?

Ensemble learning uses multiple learning algorithms at the same time to obtain predictions for the same task to get better predictions than the individual model.

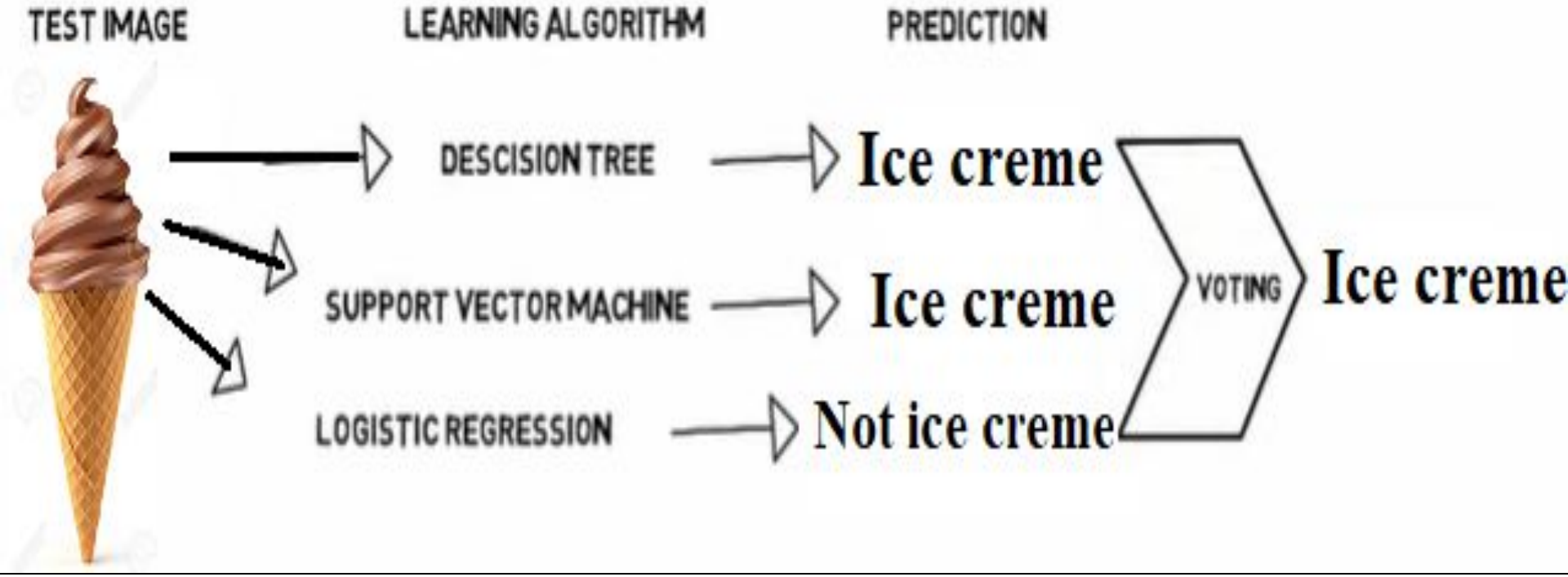
Suppose we want to classify delicacies as Ice crèmes (YES) or not (NO)

When and where to use ensemble models?

You label the individual data points as (1, -1) 1 is for ice crème, -1 is not and feed it to classification algorithm as explained in the figure for accuracy estimation:



We are trying three different algorithms, what if we combine these three models. Then, we train the three models with input and when output is generated, in the case of classification, we perform a Voting over the output and majority wins…This can be done by regression by taking the mean of the output



Random forest algorithm is an ensemble of decision tree. The accuracy of the ensemble model is better than the individual learners.

WHY USE ENSEMBLE MODELS ?

- BETTER ACCURACY (LOW ERROR)
- HIGHER CONSISTENCY (AVOIDS OVERFITTING)
- REDUCES BIAS AND VARIANCE ERRORS

When and where to use ensemble models?

- When a single model overfits like a decision tree that generally overfits we can use random forests, i.e, we can use multiple similar models to give us a better fit.
- When the difference of accuracy between the ensemble model and individual models is better with extra training then we can consider ensemble learning.
- Ensemble model can be used for classification & regression
- Bagging (bootstrap aggregating), Boosting & Random Forest

Rationale for Ensemble Method

- Construct a set of base classifiers from the training data and predict class label of test records by combining the predictions made by multiple classifiers.. We will take this definition and try to understand how ensemble method can improve a classifier's performance….
- Example: Consider an ensemble of *twenty-five* binary classifiers, each of which has an error rate of $\varepsilon = 0.35$. The ensemble classifier predicts the *class label of a test example by taking a majority vote on the predictions made by the base classifiers (say, decision tree).*
- If the base classifiers are identical, then the ensemble will misclassify the same examples predicted incorrectly by the base classifiers. Thus, the error rate of the ensemble remains 0.35.
- On the other hand, if the base classifiers are independent—i.e., their errors are uncorrelated—then the *ensemble makes a wrong prediction only if more than half of the base classifiers predict incorrectly.*

Rationale for Ensemble Method

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

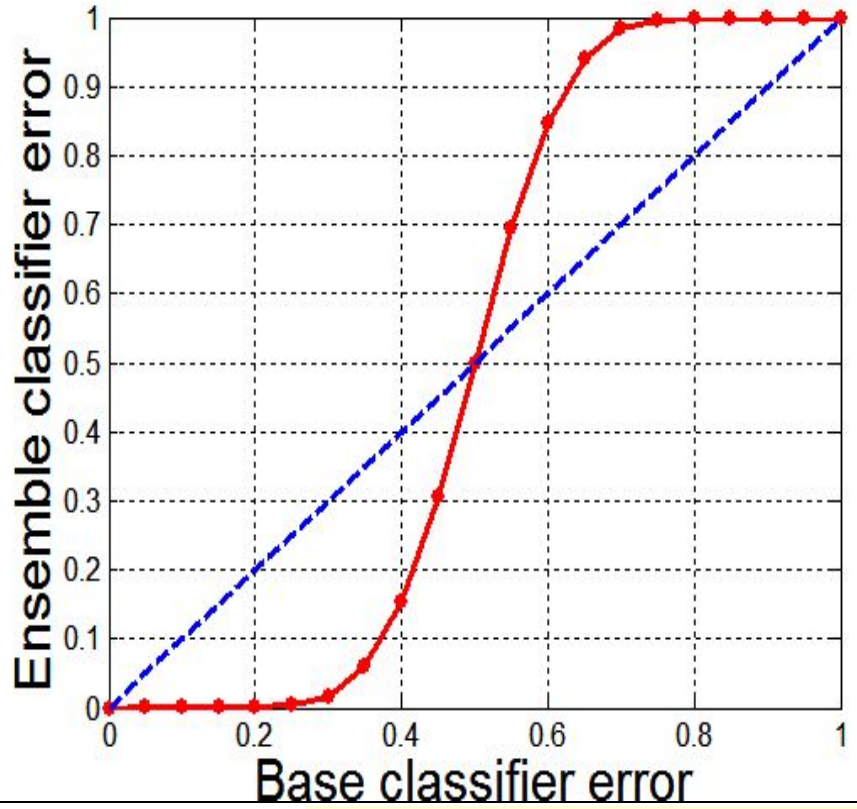


Figure 1. Comparison between errors of base classifiers and errors of the ensemble classifier.

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

ε' = 0.06 which is considerably lower than the error rate (ε = 0.35) of the base classifiers.

- Figure 1 shows the error rate of an ensemble of twenty-five binary classifiers ($e_{ensemble}$) for different base classifier error rates (ε). The diagonal line represents the case in which the base classifiers are identical, while the solid line represents the case in which the base classifiers are independent. Observe that the ensemble classifier performs worse than the base classifiers when ε is larger than 0.5.
- The preceding example illustrates two necessary conditions for an ensemble classifier to perform better than a single classifier:

  (1) the base classifiers should be independent of each other, and

  (2) the base classifiers should do better than a classifier that performs random guessing.
- In practice, it is difficult to ensure total independence among the base classifiers. Nevertheless, improvements in classification accuracies have been observed in ensemble methods in which the *base classifiers are slightly correlated*.

Methods for Constructing an Ensemble Classifier

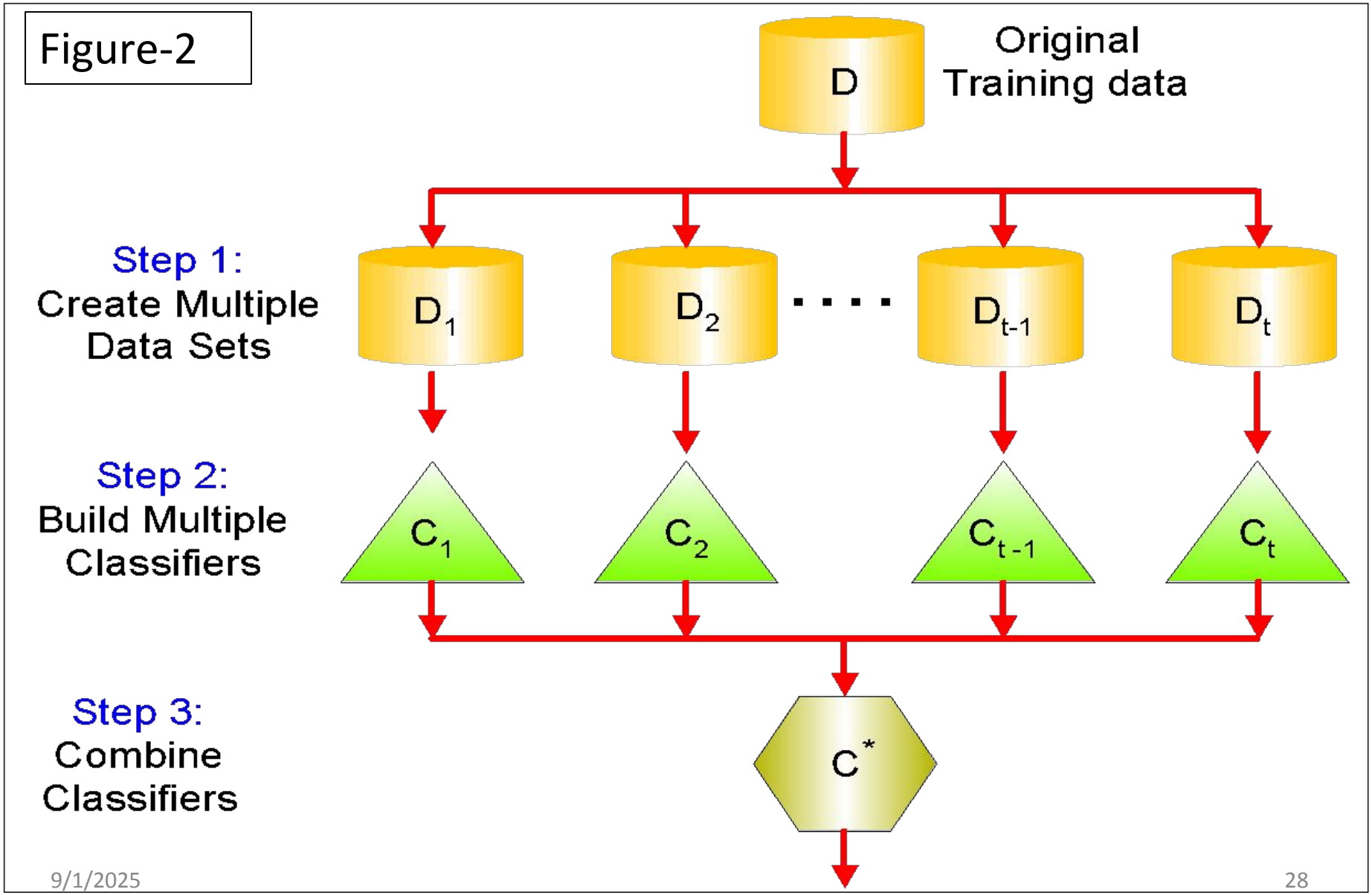
A logical view of the ensemble method is presented in Figure 2. The basic idea is to construct multiple classifiers from the original data and then aggregate their predictions when classifying unknown examples.

The ensemble of classifiers can be constructed in many ways:

1. By manipulating the training set.
2. By manipulating the input features.
3. By manipulating the class labels.
4. By manipulating the learning algorithm.

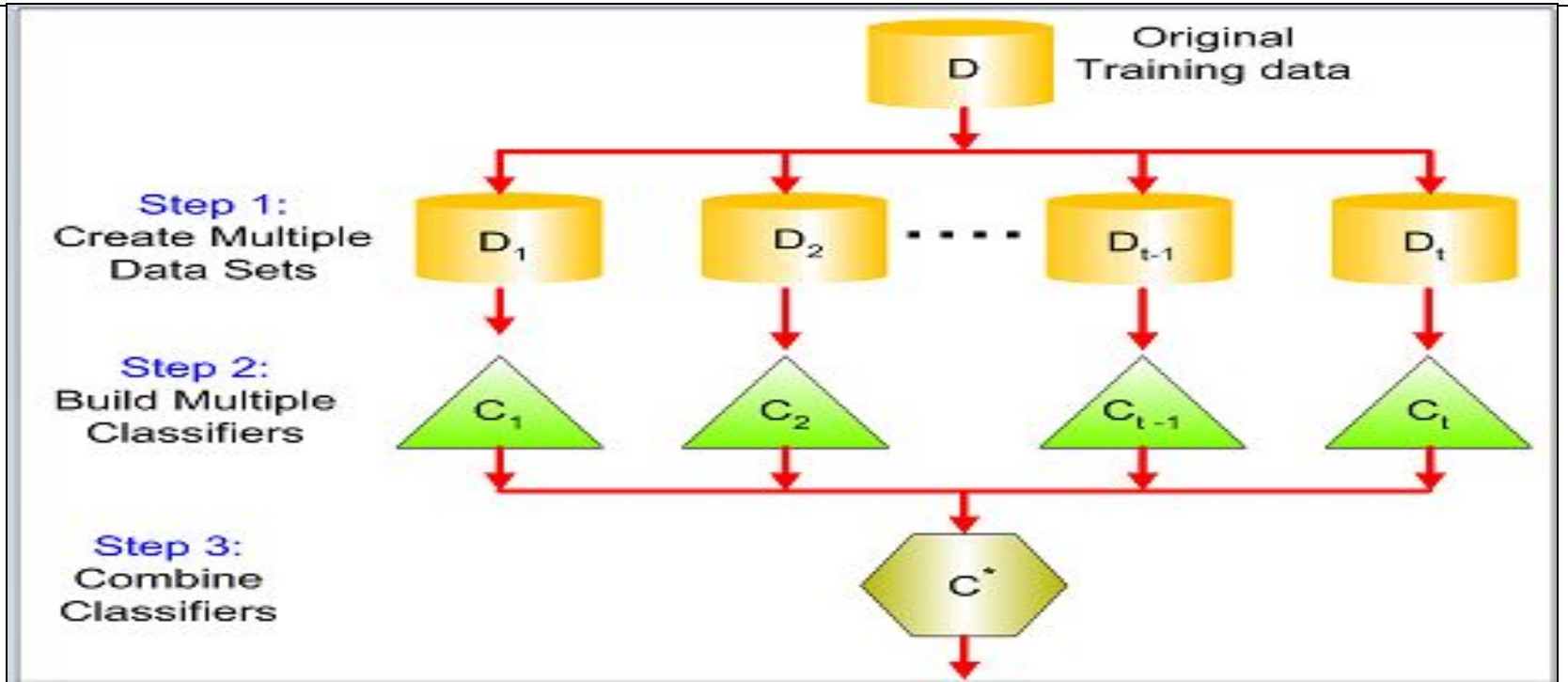
A logical view of the ensemble learning method.

Figure-2



Algorithm 1 General procedure for ensemble method.

1: Let D denote the original training data, k denote the number of base classifiers, and T be the test data.
2: **for** $i = 1$ to k **do**
3: Create training set, D_i from D.
4: Build a base classifier C_i from D_i.
5: **end for**
6: **for** each test record $x \in T$ **do**
7: $C^*(x) = Vote(C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_k(\mathbf{x}))$
8: **end for**



- Algorithm 1 shows the steps needed to build an ensemble classifier in a sequential manner.
- The first step is to create a training set from the original data D.
- Depending on the type of ensemble method used, the training sets are either identical to or slight modifications of D. The size of the training set is often kept the same as the original data, but the distribution of examples may not be identical; i.e., some examples may appear multiple times in the training set, while others may not appear even once.
- A base classifier C_i is then constructed from each training set D_i. Ensemble methods work better with unstable classifiers, i.e., base classifiers that are sensitive to minor perturbations in the training set.
- Examples of unstable classifiers include decision trees, rule-based classifiers, and artificial neural networks. By aggregating the base classifiers built from different training sets, this may help to reduce such types of errors.
- Finally, a test example x is classified by combining the predictions made by the base classifiers $C_i(x)$:

$$C^*(\mathbf{x}) = Vote(C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_k(\mathbf{x})).$$

The class can be obtained by taking a majority vote on the individual predictions or by weighting each prediction with the accuracy of the base classifier.

Types of Ensemble Methods

- Manipulate data distribution or training set
  - Example: bagging, boosting
- Manipulate input features
  - Example: random forests
- Manipulate class labels
  - Example: error-correcting output coding

Bagging

- Bagging, which is also known as bootstrap aggregating, is a technique that repeatedly samples (with replacement) from a data set according to a uniform probability distribution.
- Each bootstrap sample has the same size as the original data. Because the sampling is done with replacement, some instances may appear several times in the same training set, while others may be omitted from the training set.
- On average, a bootstrap sample D_i contains approximately 63% of the original training data because each sample has a probability $1 - (1 - 1/N)^N$ of being selected in each D_i.
- If N (training set size) is sufficiently large, this probability converges to $1 - 1/e \simeq 0.632$.
- The basic procedure for bagging is summarized in Algorithm X.
- After training the k classifiers, a test instance is assigned to the class that receives the highest number of votes.

Bagging Algorithm

Algorithm X Bagging Algorithm

1: Let k be the number of bootstrap samples.
2: for $i = 1$ to k do
3: Create a bootstrap sample of size n, D_i.
4: Train a base classifier C_i on the bootstrap sample D_i.
5: end for
6: $C^*(x) = \arg\max_y \sum_i \delta\big(C_i(x) = y\big)$, $\{\delta(\cdot) = 1$ if its argument is true, and 0 otherwise.$\}$

Bagging

- Sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample

- Each data instance has probability $1- (1 – 1/n)^n$ of being selected as part of the bootstrap sample

- To illustrate how bagging works, consider the data set shown in Table Y. Let x denote a one-dimensional attribute and y denote the class label. Suppose we apply a classifier that induces only one-level binary decision trees, with a test condition $x \leq k$, where k is a split point chosen to minimize the entropy of the leaf nodes. Such a tree is also known as a decision stump.
- Without bagging, the best decision stump we can produce splits the records at either $x \leq 0.35$ or $x \leq 0.75$. Either way, the accuracy of the tree is at most 70%. Suppose we apply the bagging procedure on the data set using ten bootstrap samples. The examples chosen for training in each bagging round are shown in upcoming Figures. On the right-hand side of each table, we also illustrate the decision boundary produced by the classifier.
- We classify the entire data set given in Table Y by taking a majority vote among the predictions made by each base classifier. The results of the predictions are shown in Figures (shown below). Since the class labels are either −1 or +1, taking the majority vote is equivalent to summing up the predicted values of y and examining the sign of the resulting sum (refer to the second to last row in Figures below). Notice that the ensemble classifier perfectly classifies all ten examples in the original data.
- Finally, since every sample has an equal probability of being selected, bagging does not focus on any particular instance of the training data. It is therefore less susceptible to model overfitting when applied to noisy data.

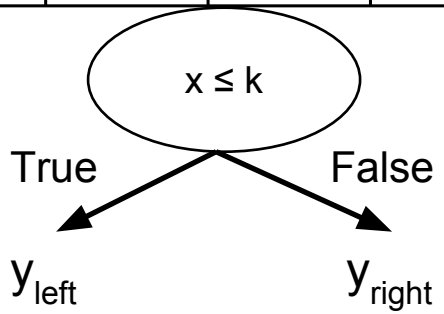
Bagging Example

- Consider 1-dimensional data set in Table Y:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump
  - Decision rule: $x \leq k$ versus $x > k$
  - Split point k is chosen based on entropy



Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

x <= 0.7 ➔ y = 1
x > 0.7 ➔ y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ➔ y = 1
x > 0.3 ➔ y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ➔ y = 1
x > 0.05 ➔ y = 1

Bagging Example

- Summary of Training sets:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Bagging Example

Original Data: **Test set is taken as same as the training data**

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Predicted Class Sign	1	1	1	-1	-1	-1	-1	1	1	1

- The above example determines class (Y values) same as original data set. We would be trying a different training set for eight bagging rounds and check if the final class values are computed same or different values are obtained.

Original Data

x	1	2	3	4	5	6	7	8	9	10
Class Y	-1	1	-1	1	-1	1	-1	1	-1	1

Round-1

Split

x	1	3	3	3	5	5	6	5	7	9
y	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
y'	1	1	1	1	-1	-1	-1	-1	-1	-1

Rule

x <= 3.5 □ y = 1

x > 3.5 □ y = -1

Round-2

Split

x	1	2	4	5	4	6	7	8	9	10
y	-1	1	1	-1	1	1	-1	1	-1	1
y'	-1	-1	-1	-1	-1	-1	-1	1	1	1

Rule

x <= 7.5 □ y = -1

x > 7.5 □ y = 1

Round-3

Split

x	2	3	4	4	5	6	6	7	8	9
y	1	-1	1	1	-1	1	1	-1	1	-1
y'	1	1	-1	-1	-1	-1	-1	-1	-1	-1

Rule

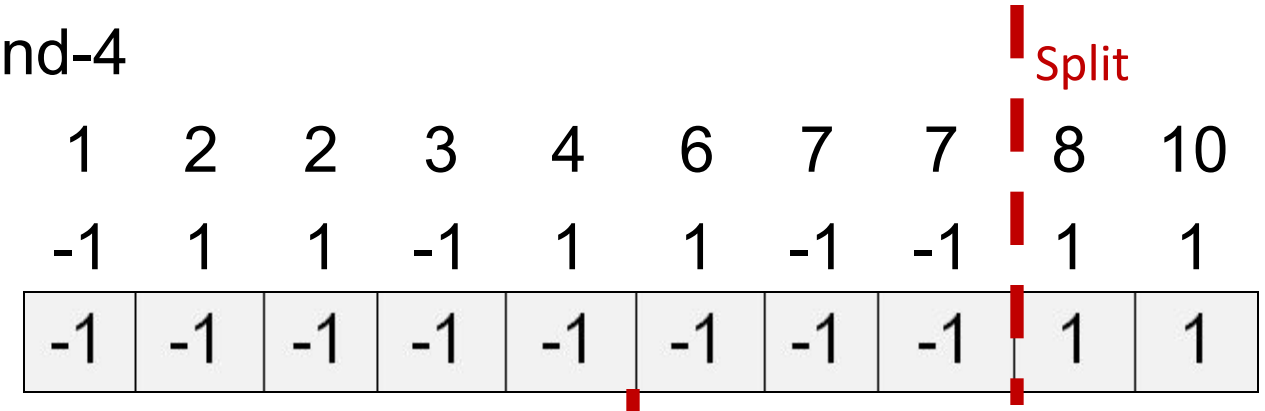
x <= 3.0 □ y = 1

X > 3.0 □ y = -1

Original Data

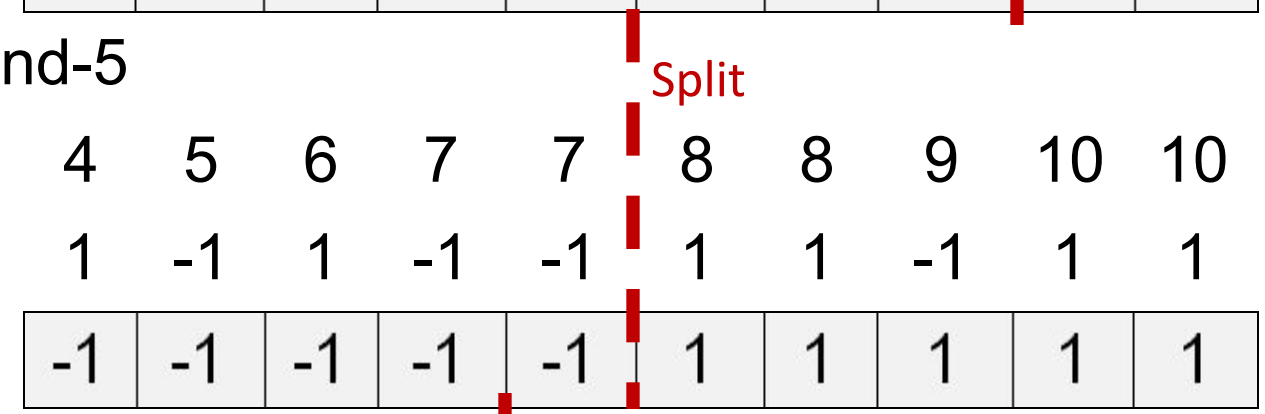
x	1	2	3	4	5	6	7	8	9	10
Class Y	-1	1	-1	1	-1	1	-1	1	-1	1

Round-4



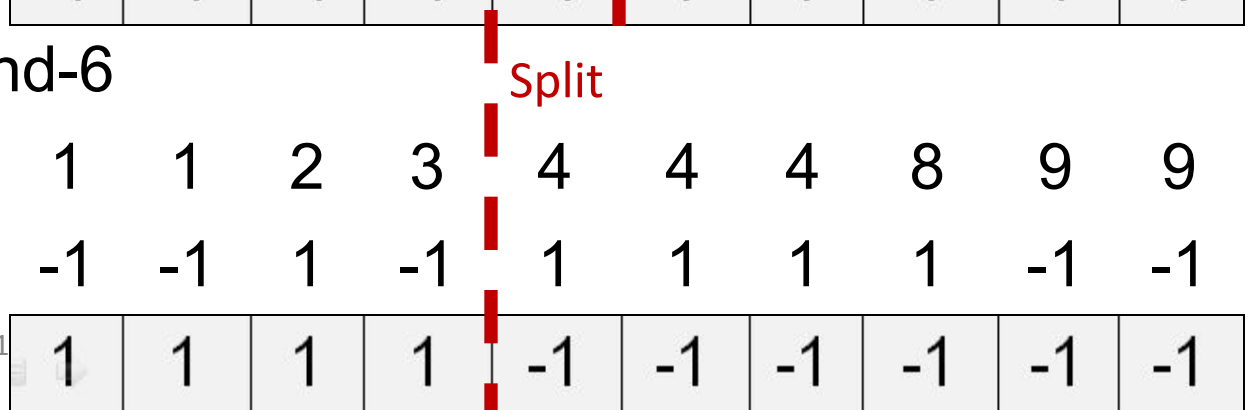
Rule
x <= 7.0 □ y = -1
x > 7.0 □ y = 1

Round-5



Rule
x <= 7.5 □ y = -1
x > 7.5 □ y = 1

Round-6



Rule
x <= 3.5 □ y = 1
x > 3.5 □ y = -1

Original Data

x	1	2	3	4	5	6	7	8	9	10
Class Y	-1	1	-1	1	-1	1	-1	1	-1	1

Round-7

Split

x	2	3	4	5	6	6	7	8	8	9
y	1	-1	1	-1	1	1	-1	1	1	-1
y'	-1	-1	-1	-1	-1	-1	-1	1	1	1

Rule

x <= 7.0 □ y = -1

x > 7.0 □ y = 1

Round-8

Split

x	1	2	2	3	3	4	5	6	7	7
y	-1	1	1	-1	-1	1	-1	1	-1	-1
y'	1	1	1	1	1	-1	-1	-1	-1	-1

Rule

x <= 3.5 □ y = 1

x > 3.5 □ y = -1

x	1	2	3	4	5	6	7	8	9	10
Round1 (y_1')	1	1	1	1	-1	-1	-1	1	-1	1
Round2 (y_2')	-1	-1	1	-1	-1	-1	-1	1	1	1
Round3 (y_3')	-1	1	1	-1	-1	-1	-1	-1	-1	1
Round4 (y_4')	-1	-1	-1	-1	-1	-1	-1	1	-1	1
Round5 (y_5')	-1	-1	-1	-1	-1	-1	-1	1	1	1
Round6 (y_6')	1	1	1	-1	-1	-1	-1	-1	-1	1
Round7 (y_7')	1	-1	-1	-1	-1	-1	-1	1	1	1
Round8 (y_8')	1	1	1	-1	-1	-1	-1	1	1	1
Sum	-4+4 = 0	0	5-3 = 2	1-7 = -6	-8	-8	-8	8	4-4 = 0	8
Sign	1	1	1	-1	-1	-1	-1	1	1	1

{0, 2, 8}, {-6, -8} ➔ {+1, -1}

x	1	2	3	4	5	6	7	8	9	10
y (final)	1	1	1	-1	-1	-1	-1	1	1	1

Boosting

- Boosting is an iterative procedure used to *adaptively change the distribution of training examples* so that the *base classifiers (decision tree, rule based, Naive Based, KNN, SVM) will focus on examples that are hard to classify*.
- Unlike bagging, *boosting assigns a weight to each training example and may adaptively change the weight at the end of each boosting round*. The **weights** assigned to the training examples can be used in the following ways:

1. Weights can be used as a sampling distribution to draw a set of bootstrap samples from the original data.
2. Weights can be used by the base classifier to learn a model that is biased toward higher-weight examples.

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

Weights

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

Boosting

- Initially, the training examples are assigned equal weights, 1/N (where N is the size of entire training set), so that they are equally likely to be chosen for training.
- *A sample is drawn according to the sampling distribution of the training examples to obtain a new training set.*
- *Next, a classifier is induced from the training set and used to classify all the examples in the original data.*
- The weights of the training examples are updated at the end of each boosting round.
- Examples that are classified incorrectly will have their weights increased, while those that are classified correctly will have their weights decreased.
- This forces the classifier to focus on examples that are difficult to classify in subsequent iterations.

A sample is drawn according to the sampling distribution of the training examples to obtain a new training set.

Next, a classifier is induced from the training set and used to classify all the examples in the original data.

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

(a)

P(Home Owner=Yes|No) = 3/7
P(Home Owner=No|No) = 4/7
P(Home Owner=Yes|Yes) = 0
P(Home Owner=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No) = 1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/3
P(Marital Status=Divorced|Yes) = 1/3
P(Marital Status=Married|Yes) = 0

For Annual Income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

(b)

Figure Y The naïve Bayes classifier for the loan classification problem.

The following table shows the examples chosen during each boosting round.

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Initially, all the examples are assigned the same weights (1/10, where N = 10). However, some examples may be chosen more than once, e.g., examples *3* and *7*, because the sampling is done with replacement.
- A classifier built from the data is then used to classify all the examples. Suppose example *4* is difficult to classify. The weight for this example will be increased in future iterations as it gets misclassified repeatedly.
- Meanwhile, examples that were not chosen in the previous round, e.g., examples *1* and *5*, also have a better chance of being selected in the next round since their predictions in the previous round were likely to be wrong.
- As the boosting rounds proceed, examples that are the hardest to classify tend to become even more prevalent.
- The final ensemble is obtained by aggregating the base classifiers obtained from each boosting round.

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

Table - A

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

AdaBoost

Let $\{(\mathbf{x}_j, y_j) \mid j = 1, 2, \ldots, N\}$ denote a set of N training examples. In the AdaBoost algorithm, the importance of a base classifier C_i depends on its error rate, which is defined as

$$\epsilon_i = \frac{1}{N}\left[\sum_{j=1}^{N} w_j \, I\Big(C_i(\mathbf{x}_j) \neq y_j\Big)\right], \quad 1$$

where $I(p) = 1$ if the predicate p is true, and 0 otherwise. The importance of a classifier C_i is given by the following parameter,

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right). \quad 2$$

Note that α_i has a large positive value if the error rate is close to 0 and a large negative value if the error rate is close to 1, as shown in Figure Z

The α_i parameter is also used to update the weight of the training examples. To illustrate, let $w_i^{(j)}$ denote the weight assigned to example $(\mathbf{x}_i, y_i)$ during the j^{th} boosting round. The weight update mechanism for AdaBoost is given by the equation:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \times \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(\mathbf{x_i}) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(\mathbf{x_i}) \neq y_i \end{cases}, \quad 3$$

where Z_j is the normalization factor used to ensure that $\sum_i w_i^{(j+1)} = 1$. The weight update formula given in Equation 3 increases the weights of incorrectly classified examples and decreases the weights of those classified correctly.

AdaBoost

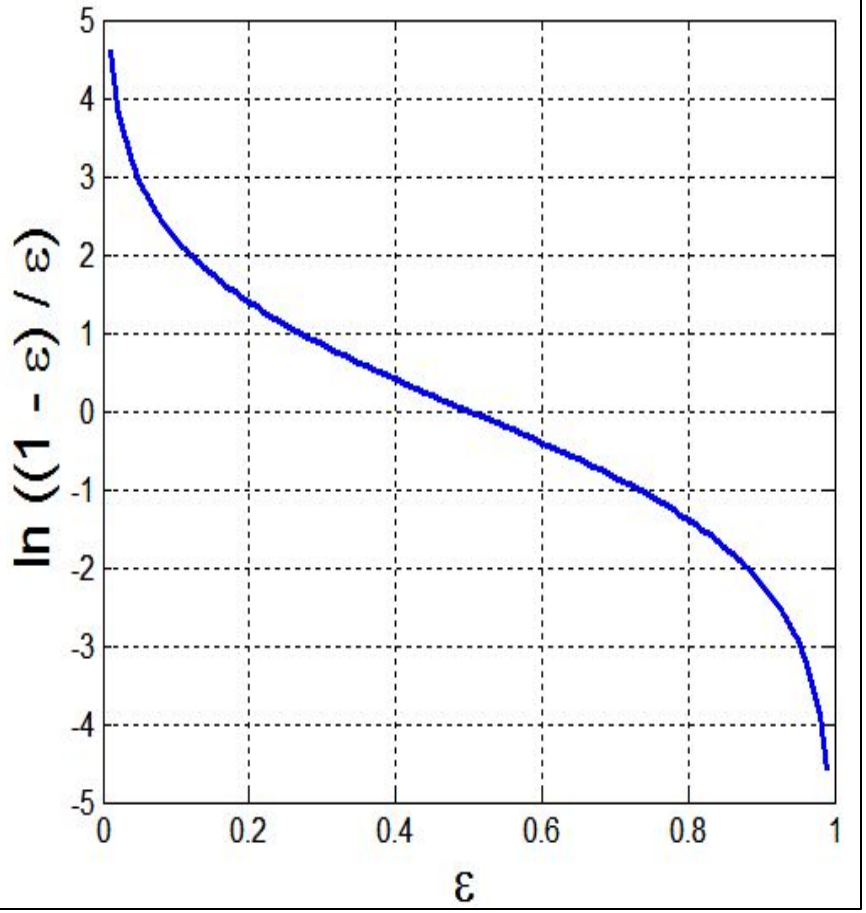
- Base classifiers: C_1, C_2, …, C_T
- Error rate:

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

Fig. Z



AdaBoost Algorithm

- Weight update:

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \begin{cases} \exp^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ \exp^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

where Z_i is the normalization factor

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated
- Classification:

$$C^*(x) = \underset{y}{\operatorname{argmax}} \sum_{i=1}^{T} \alpha_i \delta(C_i(x) = y)$$

AdaBoost Algorithm

Algorithm 5.7 AdaBoost Algorithm

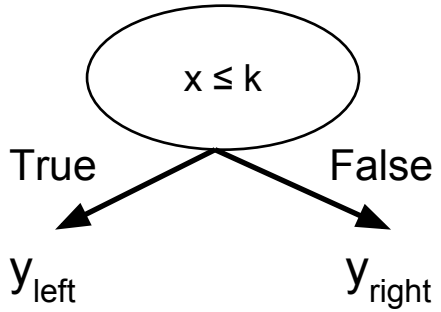
1: $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \cdots, n\}$. {Initialize the weights for all n instances.}
2: Let k be the number of boosting rounds.
3: **for** $i = 1$ to k **do**
4: Create training set D_i by sampling (with replacement) from D according to $\mathbf{w}$.
5: Train a base classifier C_i on D_i.
6: Apply C_i to all instances in the original training set, D.
7: $\epsilon_i = \frac{1}{n}\left[\sum_j w_j \, \delta\left(C_i(x_j) \neq y_j\right)\right]$ {Calculate the weighted error}
8: **if** $\epsilon_i > 0.5$ **then**
9: $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \cdots, n\}$. {Reset the weights for all n instances.}
10: Go back to Step 4.
11: **end if**
12: $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
13: Update the weight of each instance according to equation (5.88).
14: **end for**
15: $C^*(\mathbf{x}) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y))$.

AdaBoost Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1



- Classifier is a decision stump
  - Decision rule: $x \leq k$ versus $x > k$
  - Split point k is chosen based on entropy

AdaBoost Example

- Training sets for the first 3 boosting rounds:

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

- Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \begin{cases} \exp^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ \exp^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

where Z_i is the normalization factor

- Weights

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

- Classification

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

- **Σ** (alpha * predicted-class-values in each round) gives the sum, **i,e, 1.738*-1 + 2.7784*1 + 4.1195 * 1 = 5.16.**

Random Forests

- Random forest is a class of ensemble methods specifically designed for decision tree classifiers.
- It combines the predictions made by multiple decision trees, where each tree is generated based on the values of an independent set of random vectors, as shown in Figure Z.
- The random vectors are generated from a fixed probability distribution, unlike the adaptive approach used in AdaBoost, where the probability distribution is varied to focus on examples that are hard to classify.
- Bagging using decision trees is a special case of random forests, where randomness is injected into the model-building process by randomly choosing N samples, with replacement, from the original training set.
- Bagging also uses the same uniform probability distribution to generate its bootstrapped samples throughout the entire model-building process.

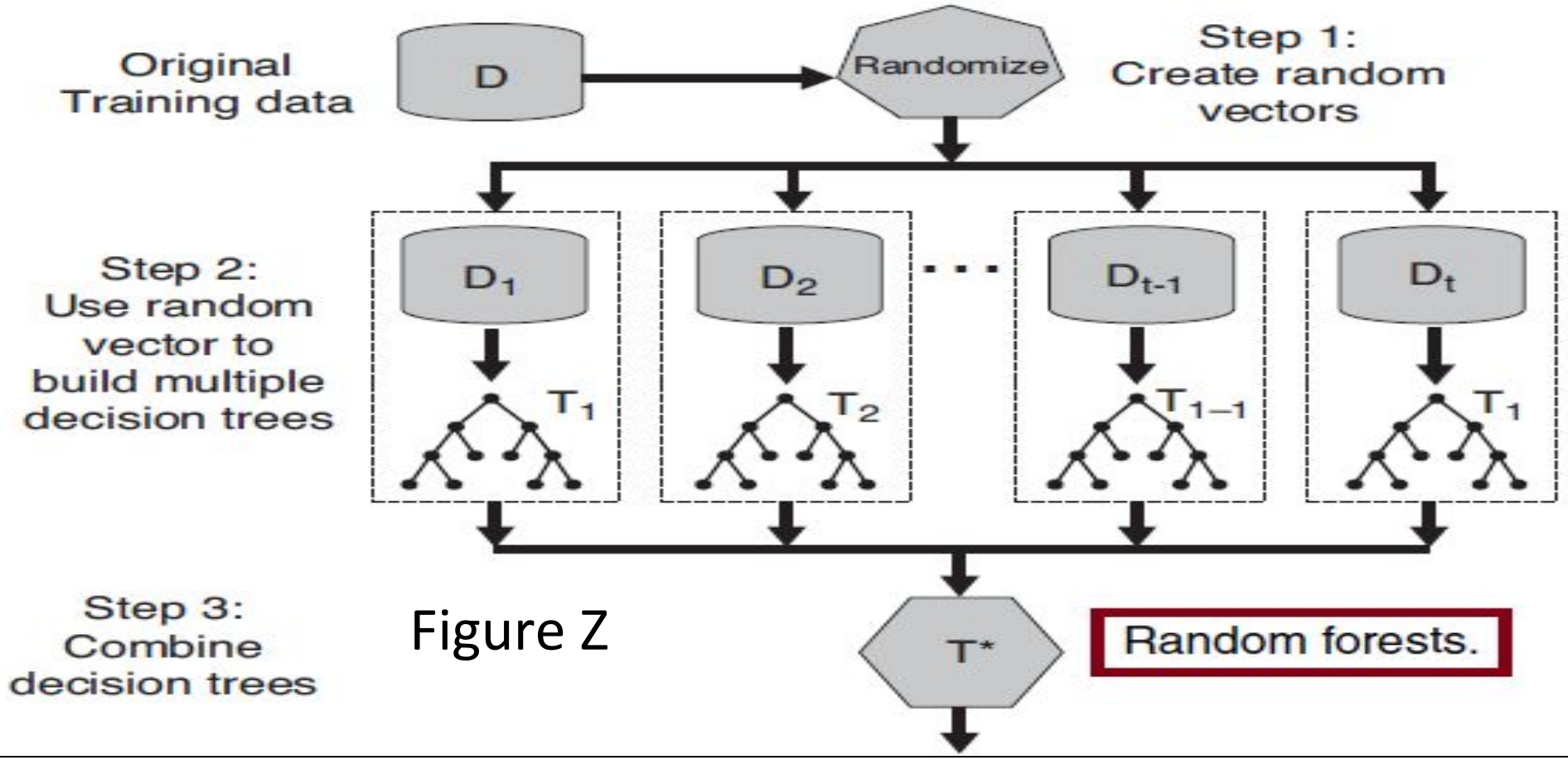
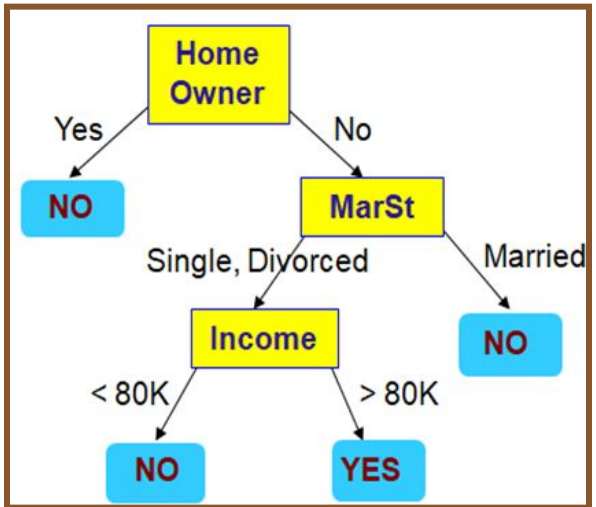
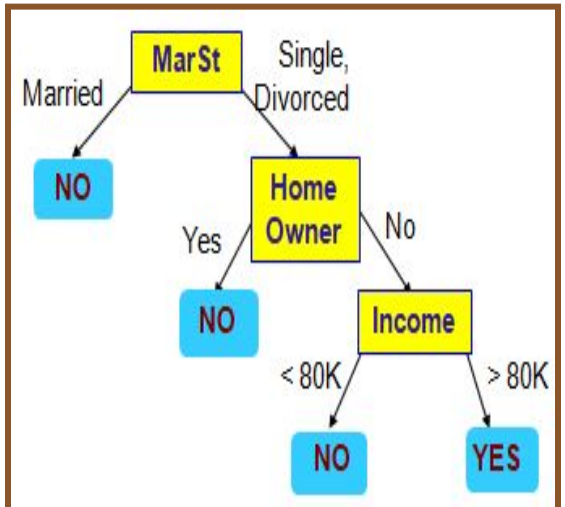
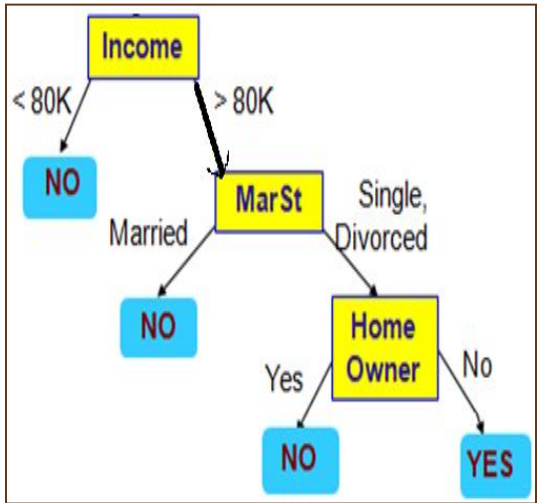

Original Training data
D
Randomize
Step 1: Create random vectors
Step 2: Use random vector to build multiple decision trees
D1
D2
Dt-1
Dt
T1
T2
T1–1
T1
Step 3: Combine decision trees
T*
Random forests.

Figure Z

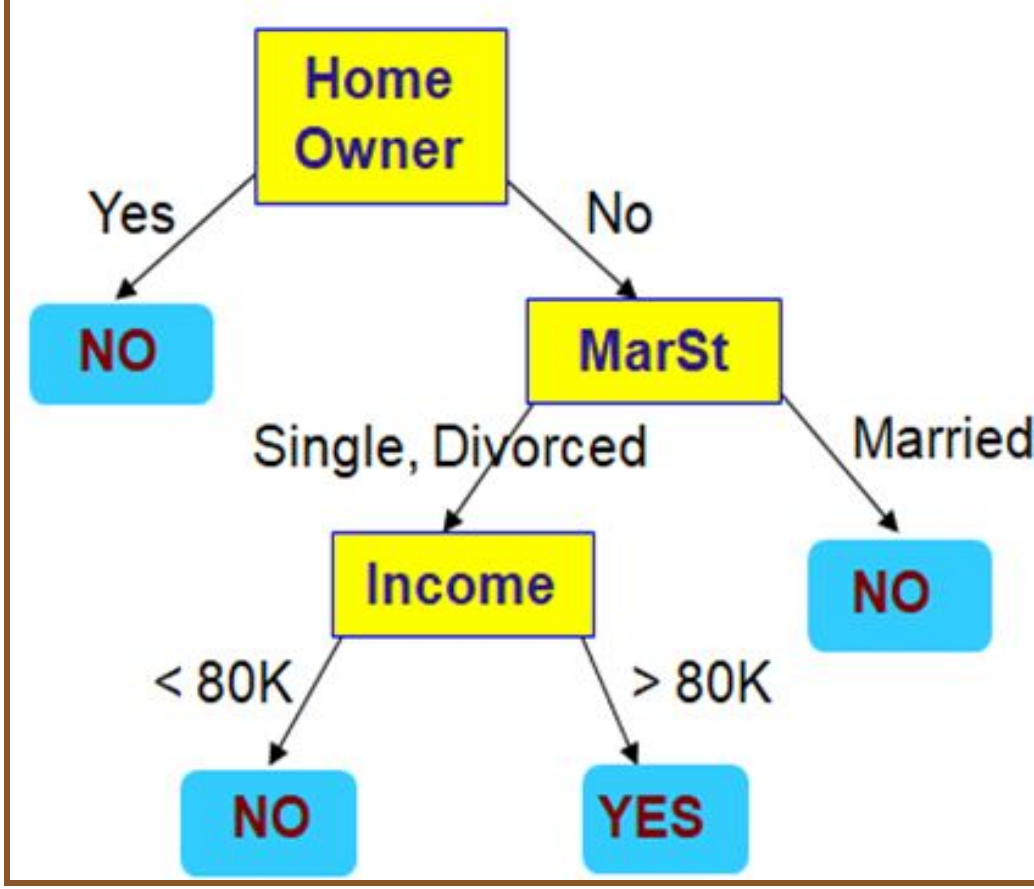

Home Owner
Yes
No
NO
MarSt
Single, Divorced
Married
Income
NO
< 80K
> 80K
NO
YES


MarSt
Married
Single, Divorced
NO
Home Owner
Yes
No
NO
Income
< 80K
> 80K
NO
YES


Income
< 80K
> 80K
NO
MarSt
Married
Single, Divorced
NO
Home Owner
Yes
No
NO
YES

Training Set

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Test Set

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
11	No	Married	55K	?
12	Yes	Divorced	80K	?
13	Yes	Single	110K	?
14	No	Single	95K	?
15	No	Married	67K	?

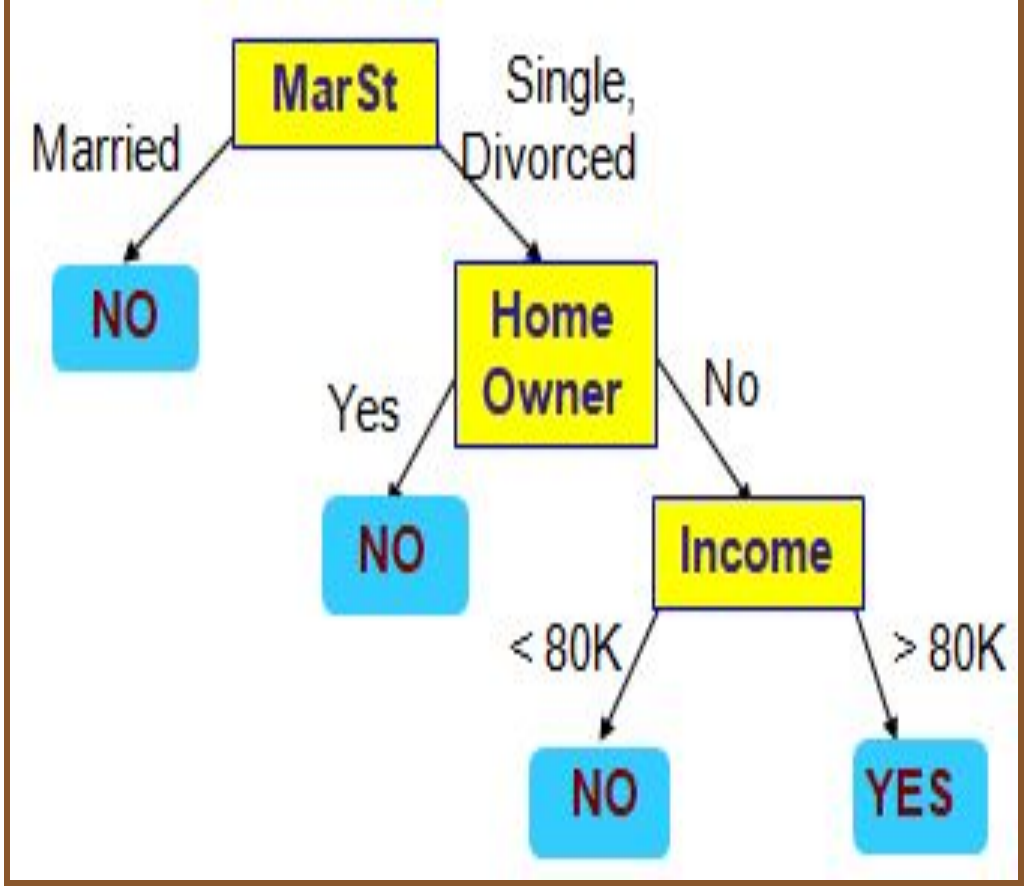
Test Set

Results after classification

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
11	No	Married	55K	No
12	Yes	Divorced	80K	No
13	Yes	Single	110K	No
14	No	Single	95K	Yes
15	No	Married	67K	No

Training Set

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Test Set

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
11	No	Married	55K	?
12	Yes	Divorced	80K	?
13	Yes	Single	110K	?
14	No	Single	95K	?
15	No	Married	67K	?

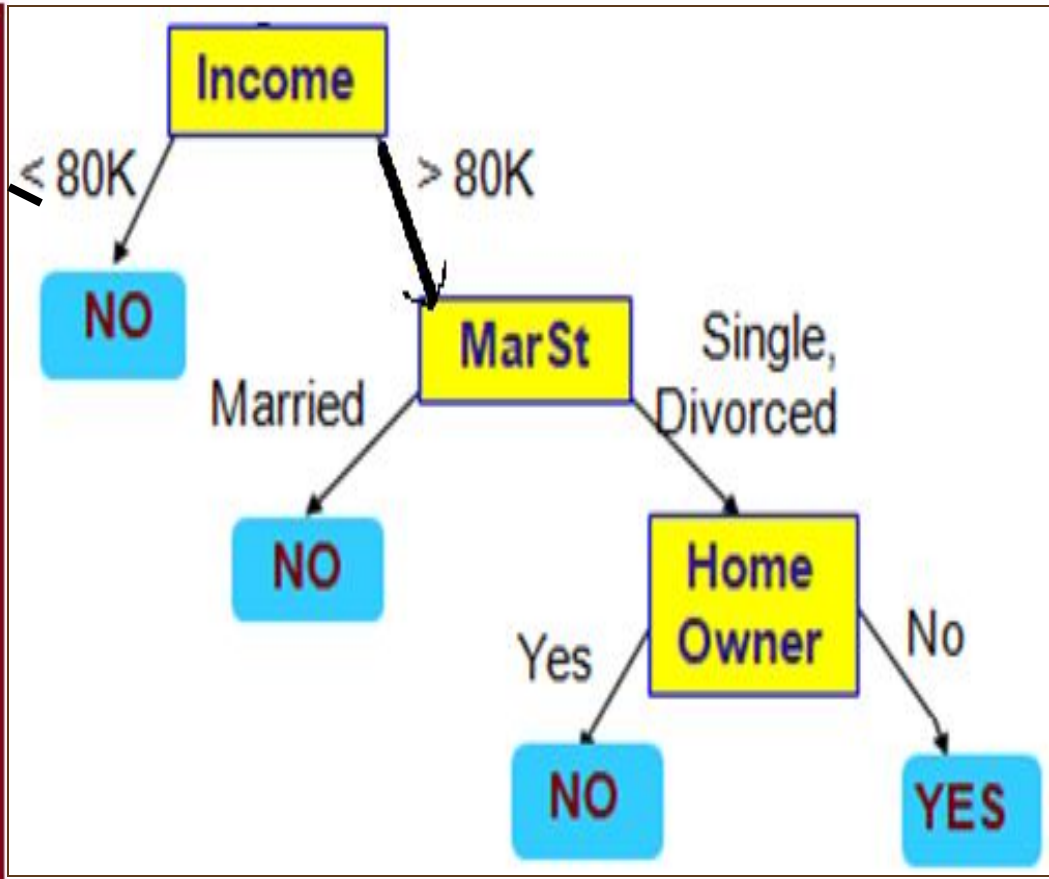
Test Set

Results after classification

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
11	No	Married	55K	No
12	Yes	Divorced	80K	No
13	Yes	Single	110K	No
14	No	Single	95K	Yes
15	No	Married	67K	No

Training Set

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Test Set

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
11	No	Married	55K	?
12	Yes	Divorced	80K	?
13	Yes	Single	110K	?
14	No	Single	95K	?
15	No	Married	67K	?

Test Set

Results after classification

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
11	No	Married	55K	Yes
12	Yes	Divorced	80K	No
13	Yes	Single	110K	No
14	No	Single	95K	Yes
15	No	Married	67K	Yes

Random Forests

It was theoretically proven that the upper bound for generalization error of random forests converges to the following expression, when the number of trees is sufficiently large.

$$\text{Generalization error} \leq \frac{\overline{\rho}(1-s^2)}{s^2}, \quad \mathbf{1}$$

where $\overline{\rho}$ is the average correlation among the trees and s is a quantity that measures the "strength" of the tree classifiers. The strength of a set of classifiers refers to the average performance of the classifiers, where performance is measured probabilistically in terms of the classifier's margin:

$$\text{margin}, M(\mathbf{X}, Y) = P(\hat{Y}_\theta = Y) - \max_{Z \neq Y} P(\hat{Y}_\theta = Z), \quad \mathbf{2}$$

where $\hat{Y}_\theta$ is the predicted class of $\mathbf{X}$ according to a classifier built from some random vector θ. The higher the margin is, the more likely it is that the

classifier correctly predicts a given example X.

Random Forests

Equation 1 shows that as the trees become more correlated or the strength of the ensemble decreases, the generalization error bound tends to increase. Randomization helps to reduce the correlation among decision trees so that the generalization error of the ensemble can be improved.

Questions

Q1. Which of the following algorithm is not an example of an ensemble method?

A. Extra Tree Regressor

B. Random Forest

C. Gradient Boosting

D. Decision Tree

Solution: (D)

Option D is correct. In case of decision tree, we build a single tree and no ensembling is required.

Q2. What is true about an ensembled classifier?

1. Classifiers that are more “sure” can vote with more conviction
2. Classifiers can be more “sure” about a particular part of the space
3. Most of the times, it performs better than a single classifier

A. 1 and 2

B. 1 and 3

C. 2 and 3

D. All of the above

Solution: (D)

In an ensemble model, we give higher weights to classifiers which have higher accuracies. In other words, these classifiers are voting with higher conviction.

On the other hand, weak learners are sure about specific areas of the problem. By ensembling these weak learners, we can aggregate the results of their sure parts of each of them.

The final result would have better results than the individual weak models.

Questions

Q3. Which of the following option is / are correct regarding benefits of ensemble model?

1. Better performance
2. Generalized models
3. Better interpretability

A. 1 and 3
B. 2 and 3
C. 1 and 2
D. 1, 2 and 3

Solution: (D)
1 and 2 are the benefits of ensemble modeling.
Option 3 is incorrect because when we ensemble multiple models, we lose interpretability of the models.

Q4) Which of the following can be true for selecting base learners for an ensemble?

1. Different learners can come from same algorithm with different hyper parameters
2. Different learners can come from different algorithms
3. Different learners can come from different training spaces

A. 1
B. 2
C. 1 and 3
D. 1, 2 and 3

Solution: (D)
We can create an ensemble by following any / all of the options mentioned above. So option D is correct.

Questions

Q5. True or False: Ensemble learning can only be applied to supervised learning methods.

A. True

B. False

Solution: (B)

Generally, we use ensemble technique for supervised learning algorithms. But, you can use an ensemble for unsupervised learning algorithms also. (https://en.wikipedia.org/wiki/Consensus_clustering)

Q6. True or False: Ensembles will yield bad results when there is significant diversity among the models.

Note: All individual models have meaningful and good predictions.

A. True

B. False

Solution: (B)

An ensemble is an art of combining a diverse set of learners (individual models) together to improvise on the stability and predictive power of the model. So, creating an ensemble of diverse models is a very important factor to achieve better results.

Q7.True or False: Ensemble of classifiers may or may not be more accurate than any of its individual model.

A. True

B. False

Solution: (A)

Usually, ensemble would improve the model, but it is not necessary. Hence, option A is correct.

Questions

Q8. Generally, an ensemble method works better, if the individual base models have _____?

Note: Suppose each individual base models have accuracy greater than 50%.

A. Less correlation among predictions

B. High correlation among predictions

C. Correlation does not have any impact on ensemble output

D. None of the above

Solution: (A)

A lower correlation among ensemble model members will increase the error-correcting capability of the model. So it is preferred to use models with low correlations when creating ensembles.

In an election, N candidates are competing against each other and people are voting for either of the candidates. Voters don't communicate with each other while casting their votes.

Q.9 Which of the following ensemble method works similar to above-discussed election procedure?

Hint: Persons are like base models of ensemble method.

A. Bagging

B. Boosting

C. A Or B

D. None of these

Solution: (A)

In bagged ensemble, the predictions of the individual models won't depend on each other. So option A is correct.