Text Classification III

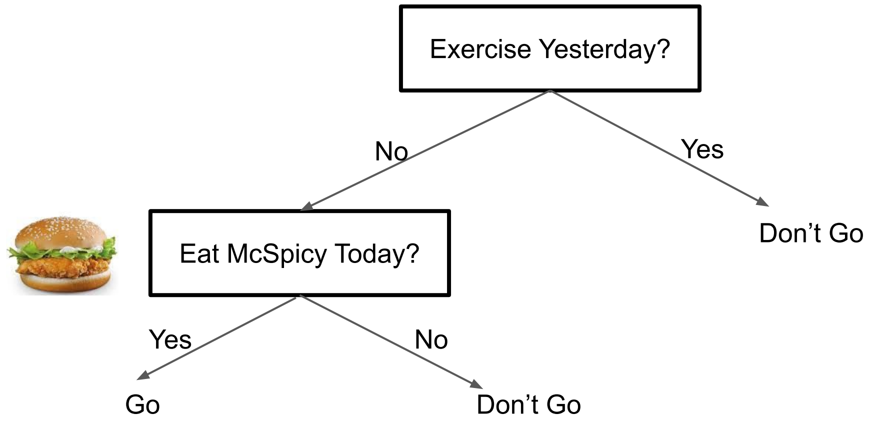
Decision Tree and Ensemble Methods

Decision Tree

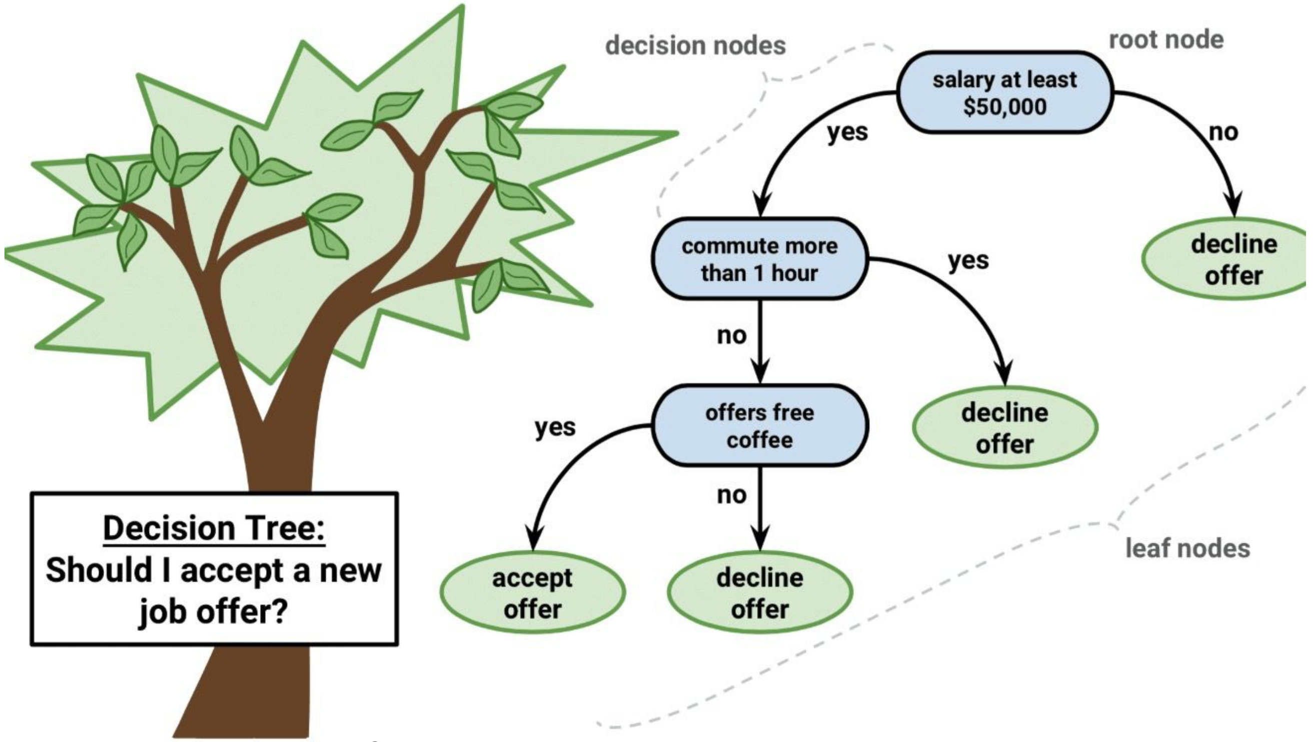
What is Decision Tree

- Decision Tree: a decision support tool that uses a tree-like model/graph of decisions and their possible consequences.

Should I do exercise today?



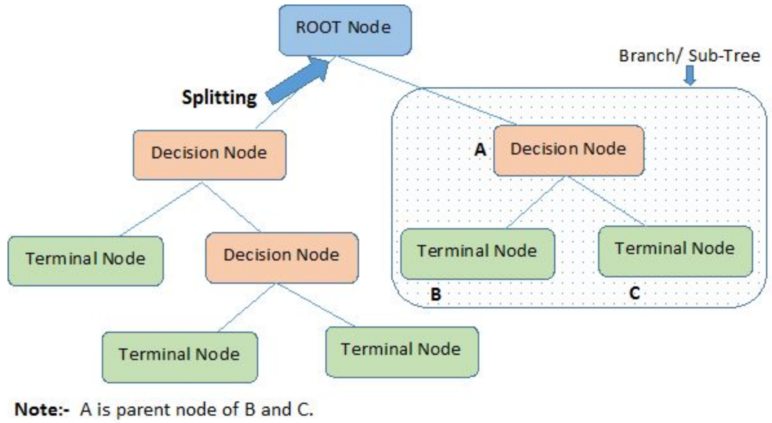
1. Top-down approach
2. Divide and Conquer



Source: https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb

Terminology

- **Root Node**: represent entire population or sample, which will be further divided into multiple subsets.
- **Splitting**: a process of dividing a node into two or more sub-nodes.
- **Decision Node**: a sub-node that can be split into further sub-nodes.
- **Leaf/Terminal Node**: nodes do not split.
- **Branch/Sub-Tree**: a sub section of entire tree.
- **Pruning**: remove nodes (opposite of Splitting)
- **Parent and child Node**: a node which is divided into sub-nodes is called parent node of sub-nodes where sub-nodes are the child of parent node.

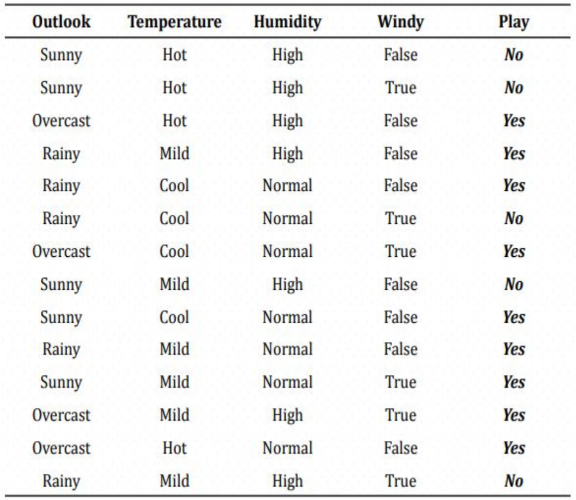


Source:
https://medium.com/@rishabhjain_22692/decision-trees-it-begins-here-93ff54ef134

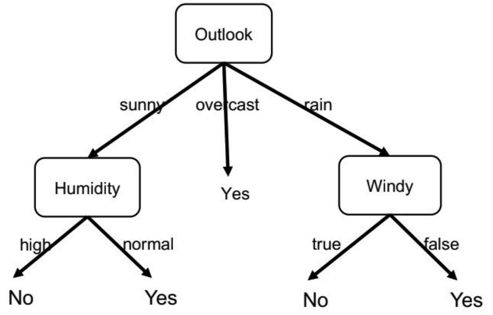
Function Approximation

Function Approximation-Decision Tree

- Problem Setting:
  - Set of possible instances $\mathbb{X}$
    - Each instance x in X is a feature vector. E.g., <humidity: low, wind: weak, outlook:rain, temp: hot>
  - Unknown target function $f : \mathbb{X} \rightarrow \mathbb{Y}$
    - A decision tree map **x** to y
  - Set of function hypotheses $H = \{h | h : \mathbb{X} \rightarrow \mathbb{Y}\}$
    - Each hypothesis is a decision tree, which sorts x to leaf and assign its corresponding y
- Input
  - Training examples $\{< \mathbf{x}^i, y^i >\}$ of unknown target function f
- Output
  - Hypothesis $h \in H$ that best approximates target function f
    - The decision tree fit the data best



Outlook	**Temperature**	**Humidity**	**Windy**	**Play**
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***



Quick Questions

- Suppose X = [x1, x2, x3, x4] is a boolean valued vector
  - How would you represent x1 AND x2 AND x4 using decision tree?
  - How would you represent x1 OR x3 using decision tree?
  - Is decision tree able to represent every boolean function over any number of boolean variables?

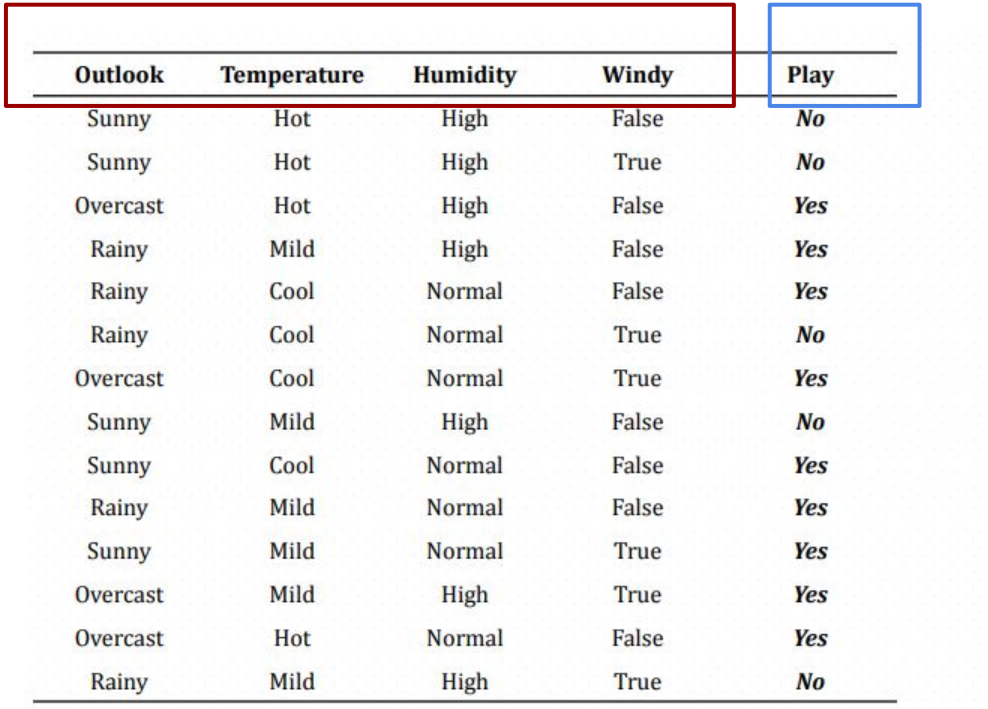
Decision Tree: How to Build and Use

Can a “Good Tree” be automatically built?

- We can always come up with some decision tree for a dataset
  - Pick any feature not used above, brach on its values, recursively.
  - However starting with a random feature may lead to a large and complex tree.

- In general, we prefer short trees over larger and complex ones because a simple consistent hypothesis is more likely to be true

Toy Example: Play Tennis

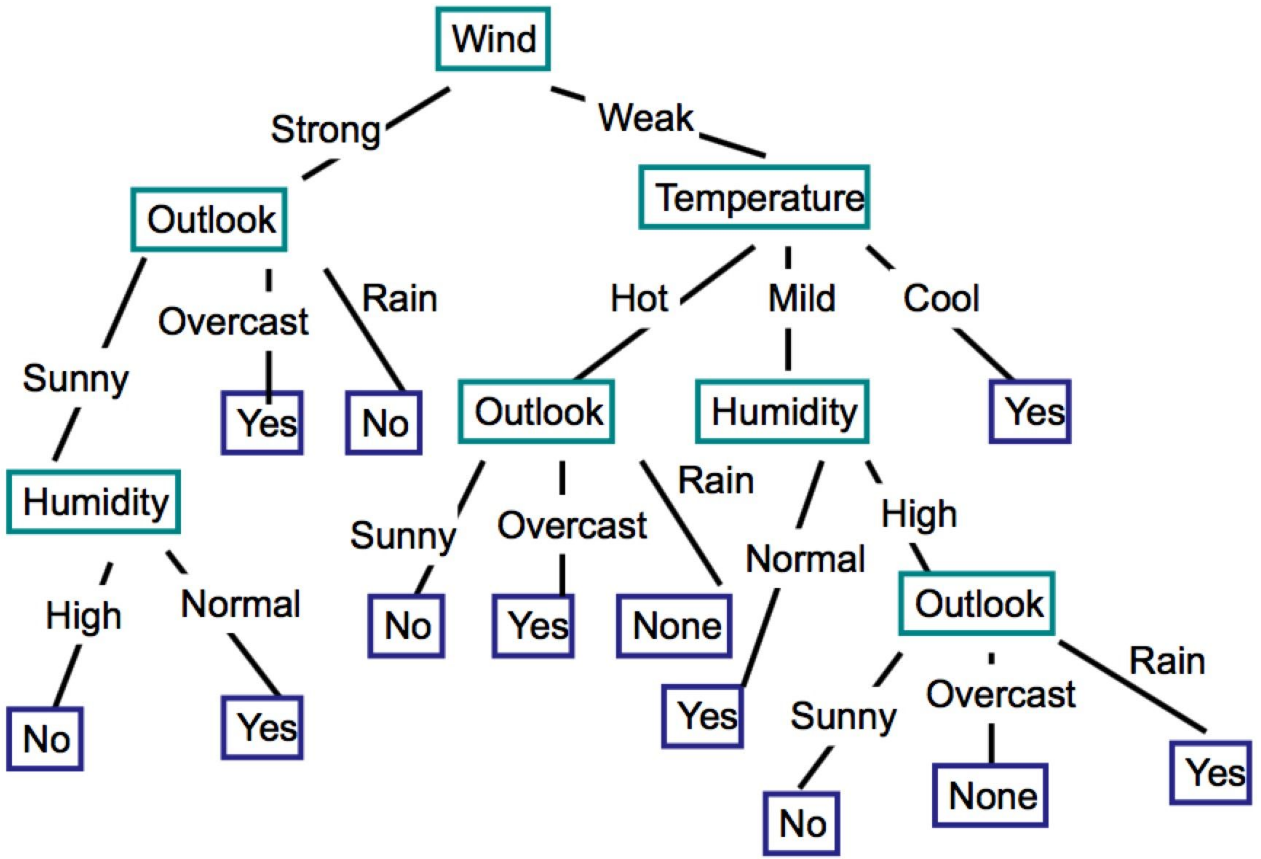


Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***

Features/ Attributes

Label

Poor Decision Tree



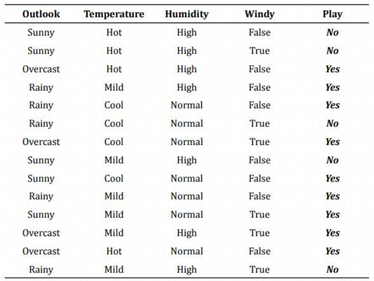
Algorithm for Building Decision Tree

node=root

Main Loop:

1. Decide the “best” attribute or feature (Say **A**) for next node;
2. For each value of **A**, create a new descendant of node;
3. Partition training examples to leaf nodes;
4. **IF** training examples are perfectly classified, **THEN STOP**; **ELSE** iterate over new leaf nodes

How to Select “Optimal” Attribute ?



Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***

Learning Process

- Finding the smallest decision tree turns out to be intractable.
- However, there is a simple heuristics algorithm that does a good job of finding small trees.
- Inductive decision tree algorithm 3 (ID3)

ID3

- ID3 is a well-known decision tree algorithms that uses a top-down greedy search through the hypothesis space.

- ID3 was designed to handle large training sets with many attributes.

- ID3 tries to generate fairly simple trees, but is not guaranteed to produce the best one.

Search Heuristic in ID3

- Central choice in ID3: Which attribute to test at each node in the tree?
    - The attribute that is most useful for classifying examples.

- Define a statistical property, called information gain, measuring how well a given attribute separates the training examples according to their target classification.

Information Gain Recap

- Entropy:

$$E(X) = -\sum_{i=1}^{K} p(X = X_i) log_2 p(X = X_i)$$

  - **Intuition**: uncertainty of X, information contained in X, expected information bits required to represent X.
- Conditional Entropy

$$E(X|Y) = \sum_{i=1} p(Y = Y_i) E(X|Y = Y_i)$$

  - **Intuition**: given y, how much uncertainty remains in X
- Mutual Information (Information Gain)

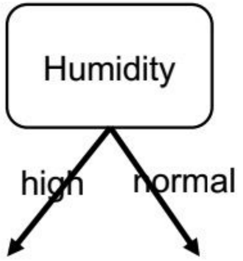
$$I(X, Y) = E(X) - E(X|Y) = E(Y) - E(Y|X)$$

  - **Intuition**: how much knowing Y reduces uncertainty about X, and vice versa.

Information Gain Splitting

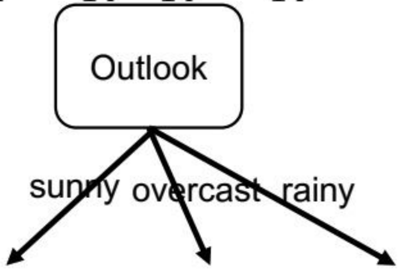
E(X)

S=[9+, 5-]
E=0.940



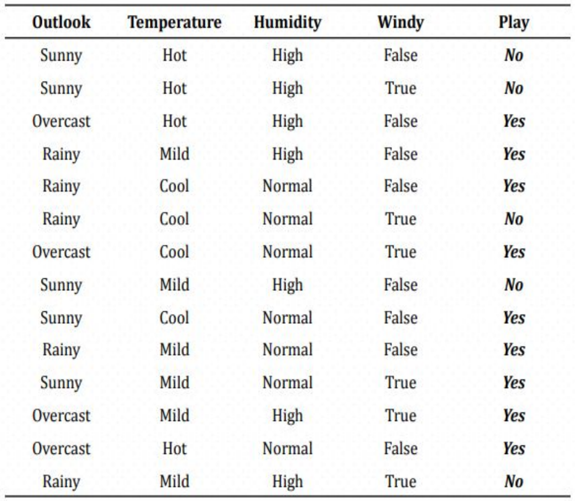
S=[9+, 5-]

$$E = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$$



E(XIY)

I(X, Y)



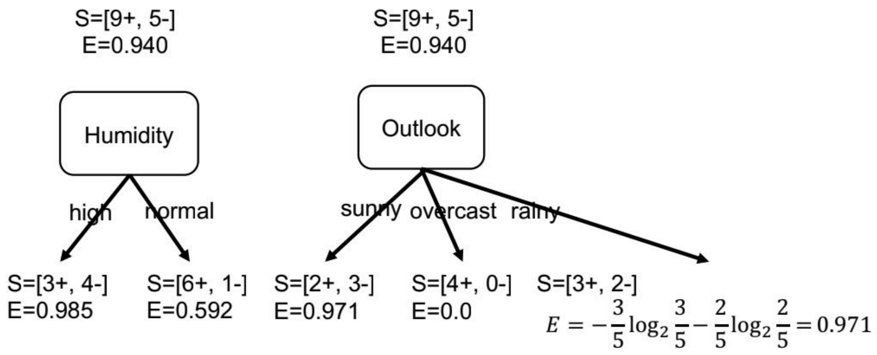
Outlook	**Temperature**	**Humidity**	**Windy**	**Play**
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***

Information Gain Splitting

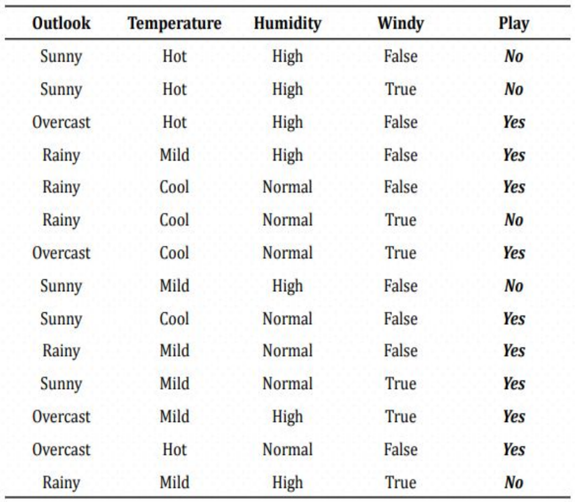
E(X)

E(X|Y)

I(X, Y)



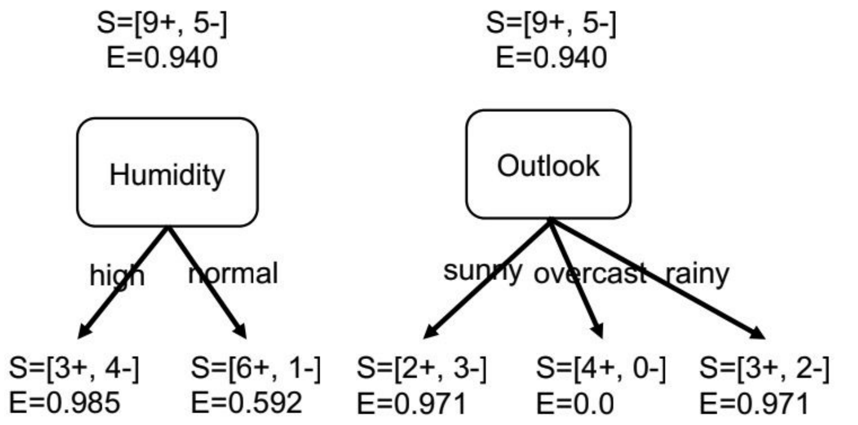
$$E = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$$



Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***

Information Gain Splitting

E(X)



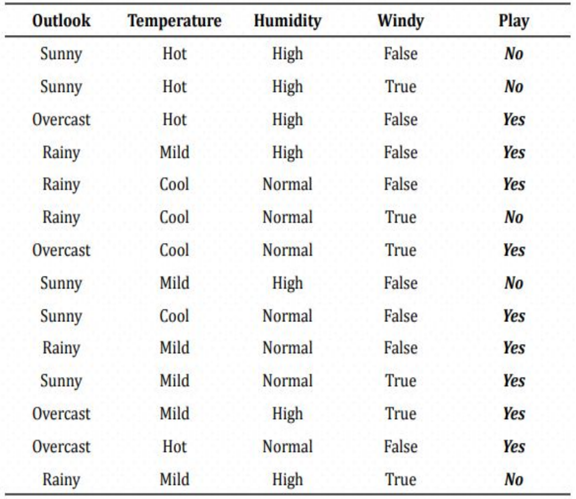
E(X|Y)

I(X, Y)

I=0.940-0.5*0.985-0.5*0.592
=0.151

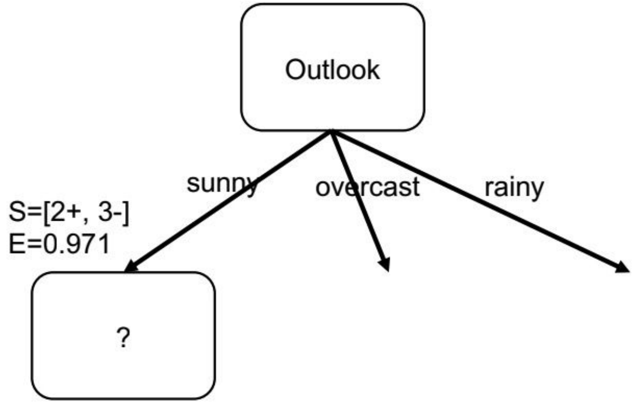
I=0.940-0.357*0.971*2
=0.247

Outlook wins over Humidity

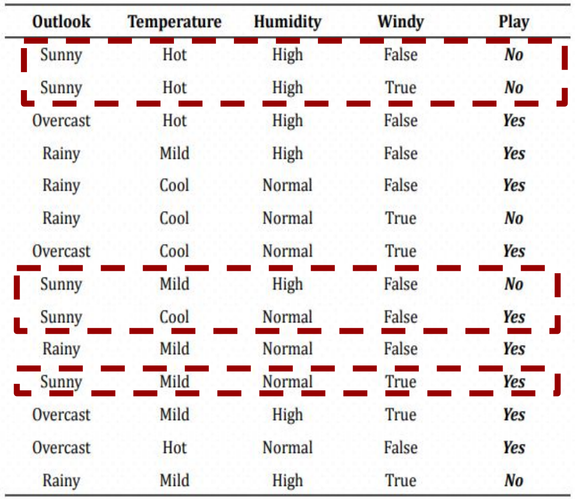


Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	*No*
Sunny	Hot	High	True	*No*
Overcast	Hot	High	False	*Yes*
Rainy	Mild	High	False	*Yes*
Rainy	Cool	Normal	False	*Yes*
Rainy	Cool	Normal	True	*No*
Overcast	Cool	Normal	True	*Yes*
Sunny	Mild	High	False	*No*
Sunny	Cool	Normal	False	*Yes*
Rainy	Mild	Normal	False	*Yes*
Sunny	Mild	Normal	True	*Yes*
Overcast	Mild	High	True	*Yes*
Overcast	Hot	Normal	False	*Yes*
Rainy	Mild	High	True	*No*

Next Step



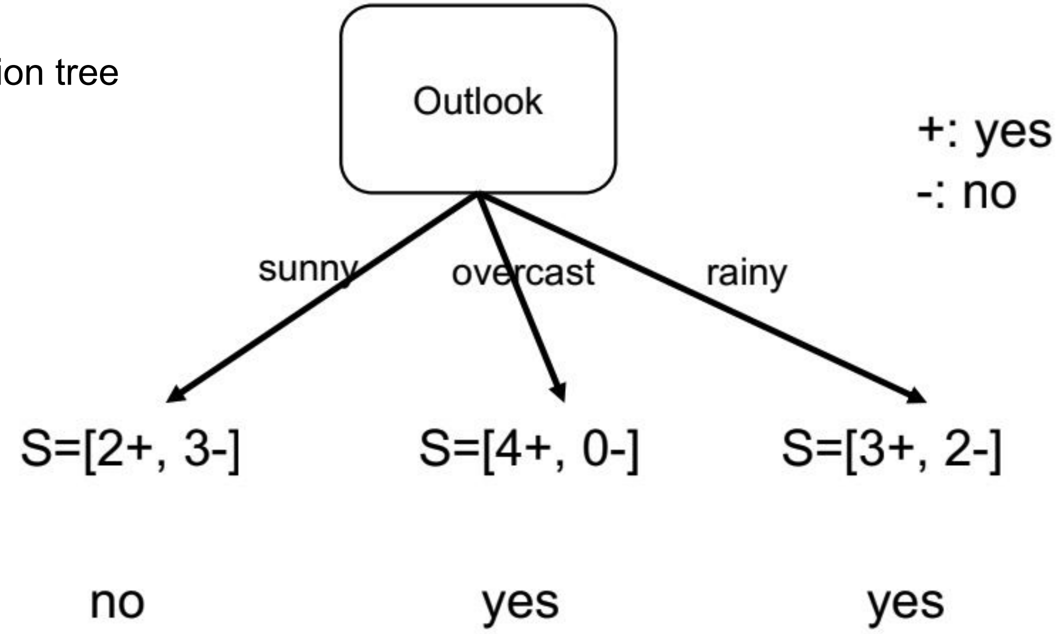
Next attributes to split ?



Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	*No*
Sunny	Hot	High	True	*No*
Overcast	Hot	High	False	*Yes*
Rainy	Mild	High	False	*Yes*
Rainy	Cool	Normal	False	*Yes*
Rainy	Cool	Normal	True	*No*
Overcast	Cool	Normal	True	*Yes*
Sunny	Mild	High	False	*No*
Sunny	Cool	Normal	False	*Yes*
Rainy	Mild	Normal	False	*Yes*
Sunny	Mild	Normal	True	*Yes*
Overcast	Mild	High	True	*Yes*
Overcast	Hot	Normal	False	*Yes*
Rainy	Mild	High	True	*No*

Prediction with Decision Tree

Use outlook only to build the decision tree



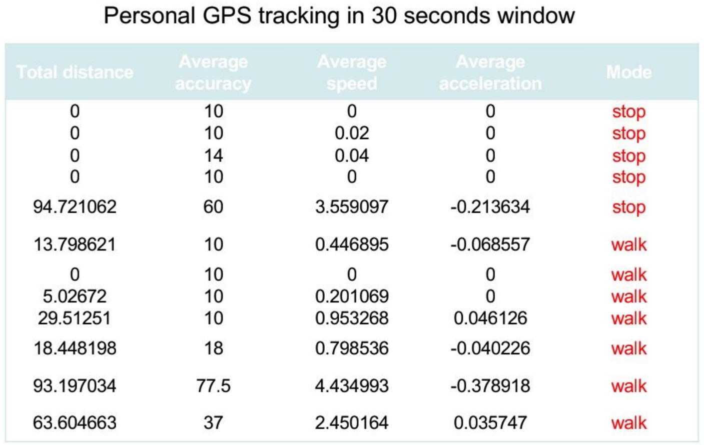
Classify be majority vote:

How about Continuous Attributes?

- Real-valued attributes can, in advance, be discretized into ranges, such as big, medium, small.

- Alternatively, for continuous attribute A, one can develop splitting nodes based on thresholds of the form $A<c$ that partition the examples into those with $A<c$ and those with $A>=c$.
  - The information gain of such splits are easily computed and compared to splits on discrete features in order to select the best split.

Continuous Value Attribute

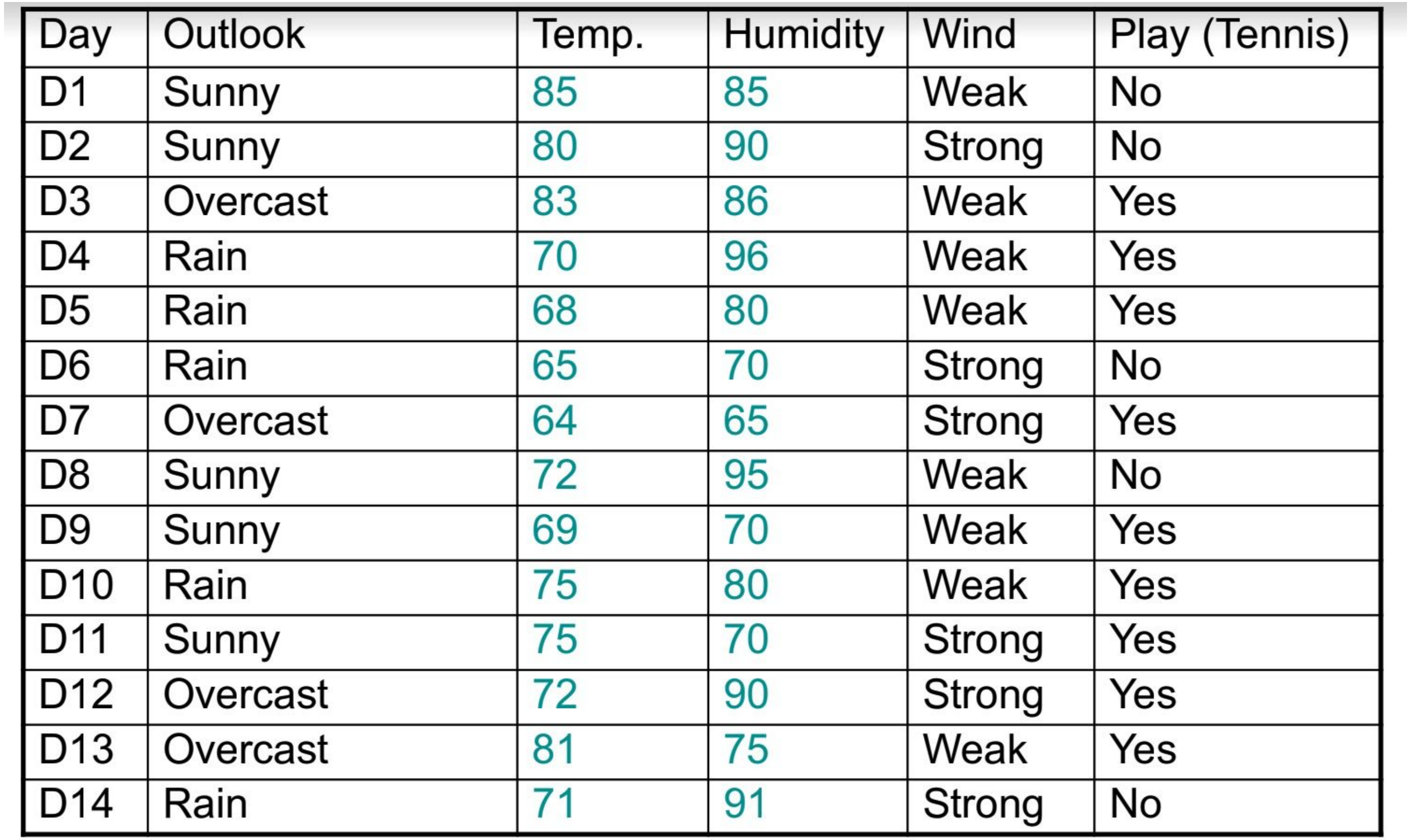


Personal GPS tracking in 30 seconds window

Total distance	Average accuracy	Average speed	Average acceleration	Mode
0	10	0	0	stop
0	10	0.02	0	stop
0	14	0.04	0	stop
0	10	0	0	stop
94.721062	60	3.559097	-0.213634	stop
13.798621	10	0.446895	-0.068557	walk
0	10	0	0	walk
5.02672	10	0.201069	0	walk
29.51251	10	0.953268	0.046126	walk
18.448198	18	0.798536	-0.040226	walk
93.197034	77.5	4.434993	-0.378918	walk
63.604663	37	2.450164	0.035747	walk

- Defining a discrete attribute that partitions the continuous attribute value into a discrete set of intervals
  - Speeds in the sample: [3,0.5, 10, 4,0.3,0.1,1]
  - Speed < 3?:
    - [0.5, 0.3, 0, 2,1]
    - [3, 10, 4]

Training Examples

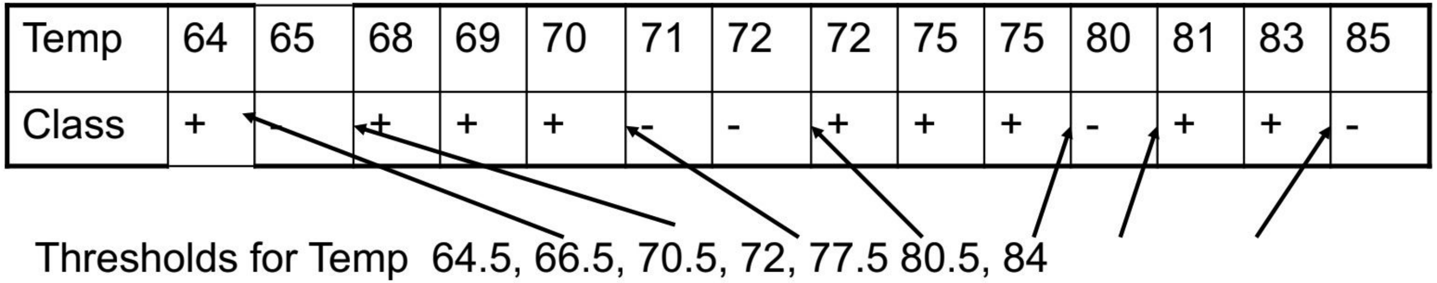


Day	Outlook	Temp.	Humidity	Wind	Play (Tennis)
D1	Sunny	85	85	Weak	No
D2	Sunny	80	90	Strong	No
D3	Overcast	83	86	Weak	Yes
D4	Rain	70	96	Weak	Yes
D5	Rain	68	80	Weak	Yes
D6	Rain	65	70	Strong	No
D7	Overcast	64	65	Strong	Yes
D8	Sunny	72	95	Weak	No
D9	Sunny	69	70	Weak	Yes
D10	Rain	75	80	Weak	Yes
D11	Sunny	75	70	Strong	Yes
D12	Overcast	72	90	Strong	Yes
D13	Overcast	81	75	Weak	Yes
D14	Rain	71	91	Strong	No

For Continuous Attributes

- The single threshold with the highest gain for a set of data can be found by examining the following choices.

```
for (each continuous feature A){
   Sort the examples according to their value for A;
   for (each ordered pair, $X_i$, $X_{i+1}$, in the sorted list)
      if (the category of $X_i$ and $X_{i+1}$ are different)
          find the midpoint of $X_i$ and $X_{i+1}$ denoted as $c_i$  to
          define threshold $A < c_i$
}
```



Temp	64	65	68	69	70	71	72	72	75	75	80	81	83	85
Class	+	-	+	+	+	-	-	+	+	+	-	+	+	-

Thresholds for Temp 64.5, 66.5, 70.5, 72, 77.5 80.5, 84

Attributes with Many Values

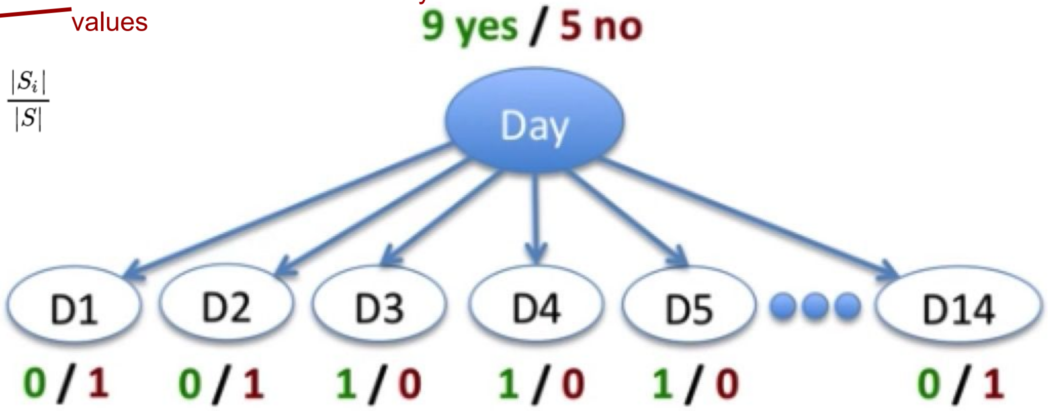
- Information Gain will bias toward the attribute with many values
- For example, using day as the attribute, the data will be perfectly splitted into subsets of size 1.
- However, it won't work for new data.
- Use GainRatio instead of information gain as criteria:

$$GainRatio(S, A) = \frac{Gain(S,A)}{SplitInformation(S,A)}$$

Penalizes attributes with many values

$$SplitInformation(S, A) = -\sum_{i \in Values(A)} \frac{|S_i|}{|S|} log_2 \frac{|S_i|}{|S|}$$

A	candidate attribute
i	possible values of A
S	set of examples {X}
S_i	subset where $X_A = i$

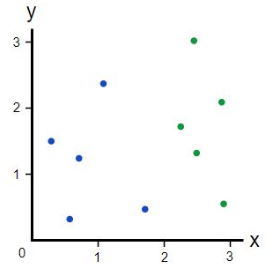


Other Criterions

- Gini Impurity (corresponding to entropy)
  - Suppose we
    - Randomly pick a datapoint in our dataset, then
    - **Randomly classify it according to the class distribution in the dataset.**
    - The probability we classify the data point incorrectly is Gini Impurity
- The formula:

$$G = \sum_{i=1}^{K} p(i)(1 - p(i))$$

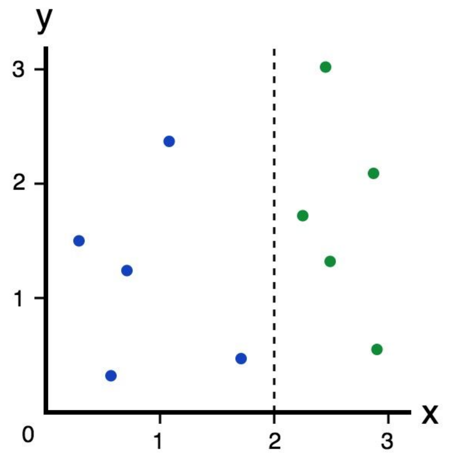
K	total classes
p(i)	the probability of picking a sample with class i

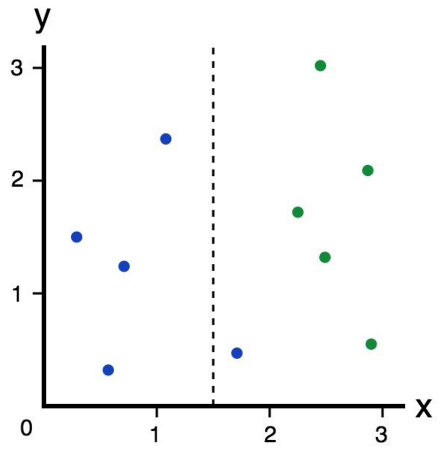


The Dataset

$$\begin{aligned} G &= p(1) * (1 - p(1)) + p(2) * (1 - p(2)) \\ &= 0.5 * (1 - 0.5) + 0.5 * (1 - 0.5) \\ &= \boxed{0.5} \end{aligned}$$

Gin Impurity





$$GI(X) = \sum_{i=1}^{2} P(X=i) * (1 - P(X=i))$$

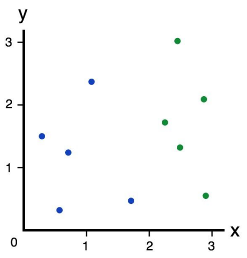
$$GI_{Left}(X) = \sum_{i=1}^{2} P(X=i) * (1 - P(X=i))$$?

$$GI_{Right}(X) = \sum_{i=1}^{2} P(X=i) * (1 - P(X=i))$$?

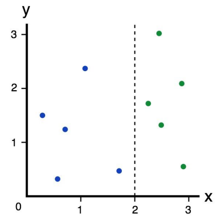
$$GI(X) = 0.4 * GI_{Left}(X) + 0.6 * GI_{Right}(X)$$

Weighting the impurity of each branch by how many elements it has

Gin Gain

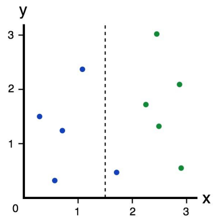


$G_{original}$=0.5



G=0.5 * 0 + 0.5 * 0 = 0

Gain is 0.5 - 0 = 0.5



G=0.4 * 0 + 0.6 * 278 = 0.167

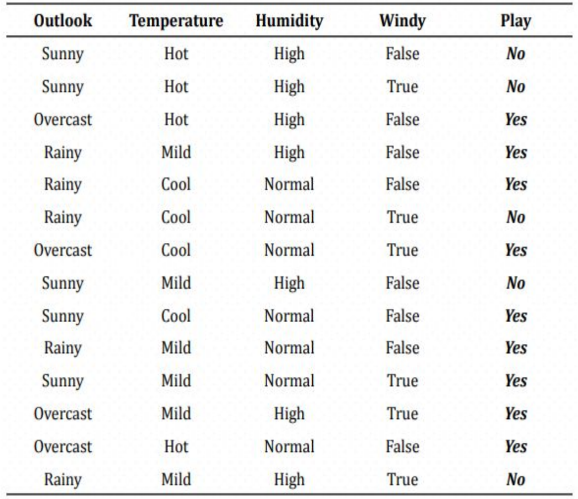
Gain is 0.5 - 0.167 = 0.333

$$\begin{aligned} G_{right} &= \frac{1}{6} * (1 - \frac{1}{6}) + \frac{5}{6} * (1 - \frac{5}{6}) \\ &= \frac{5}{18} \\ &= \boxed{0.278} \end{aligned}$$

When training a decision tree, the best split is chosen by **maximizing the Gini Gain**, which is calculated by subtracting the weighted impurities of the branches from the original impurity.

Exercise

Calculate Gini Gain for Outlook



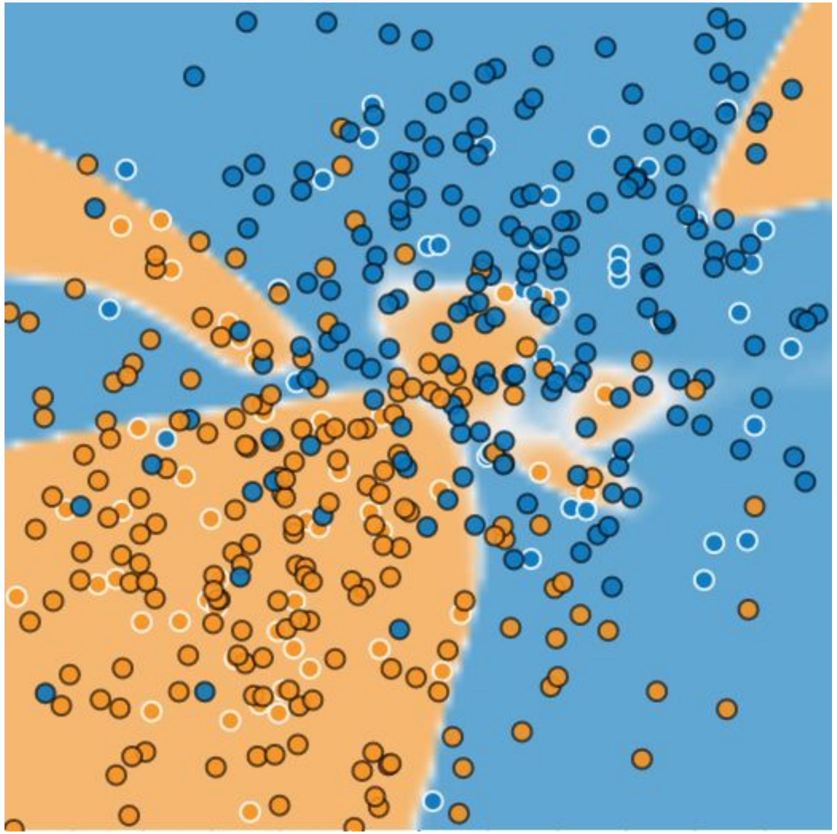
Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***

Decision Tree Learning

- Pros
  - Easy to understand: decision tree output is very easy to understand
  - Data exploration: feature selection
  - Less data cleaning required: not influenced by scale and missing values to a fair degree
  - Data type is not a constraint: handle both numerical and categorical variables
  - Non parametric method: no assumptions about the space distribution and the classifier structure
  - In nature, can handle multiclass directly
- Cons
  - Overfitting: overfitting is one of the most practical difficulty for decision tree models
  - Not ideal for continuous variables: while working with continuous numerical variables, decision tree lost information when it categorizes variables in different categories.

Decision Tree: Overfitting

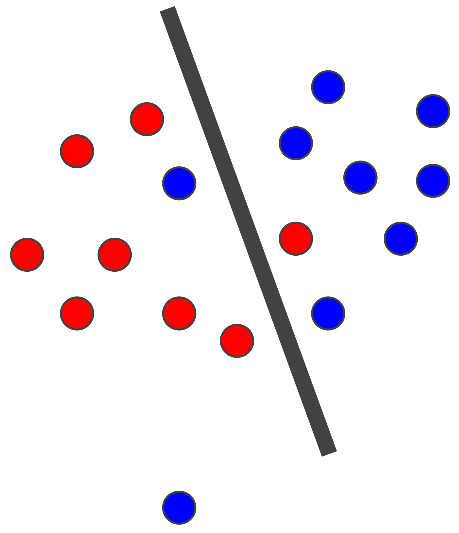
Recap on Overfitting

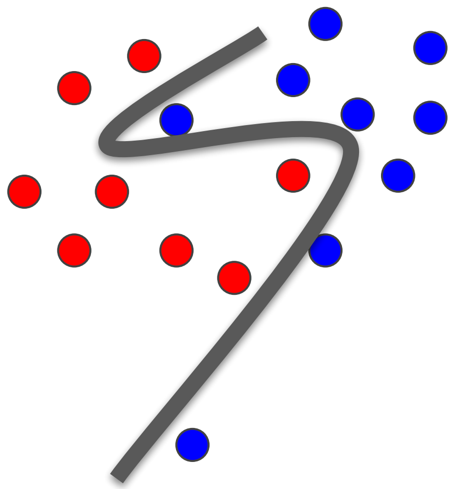


Generalization

- In ML, a model is used to fit the data
- Once trained, the model is applied upon new data
- Generalization is the prediction capability of the model on live/new data

Which model is better?





SPAM VS Not SPAM

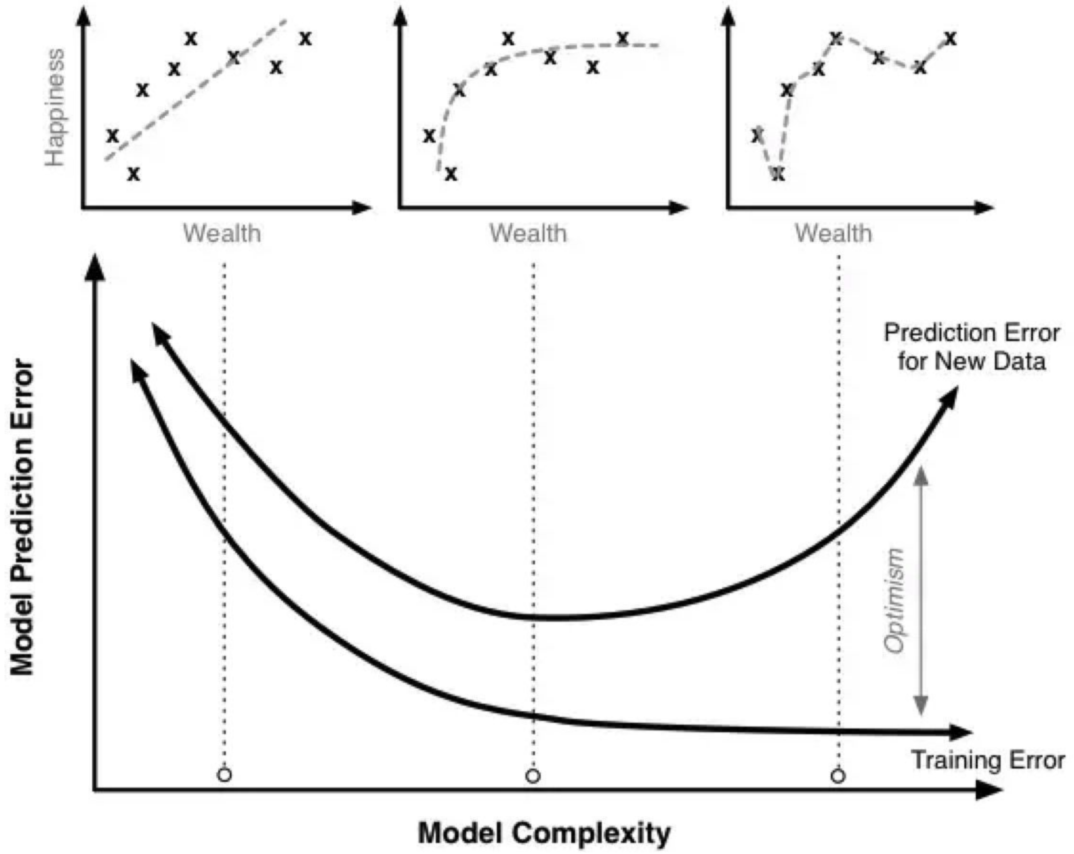
Model Complexity

- Complex model easily overfits the training data

- Then, the trained model is unable to generalize on testing data

- overfitting vs underitting
  - overfitting: small training error but large testing error
  - underfitting: large training and testing errors

Model Complexity



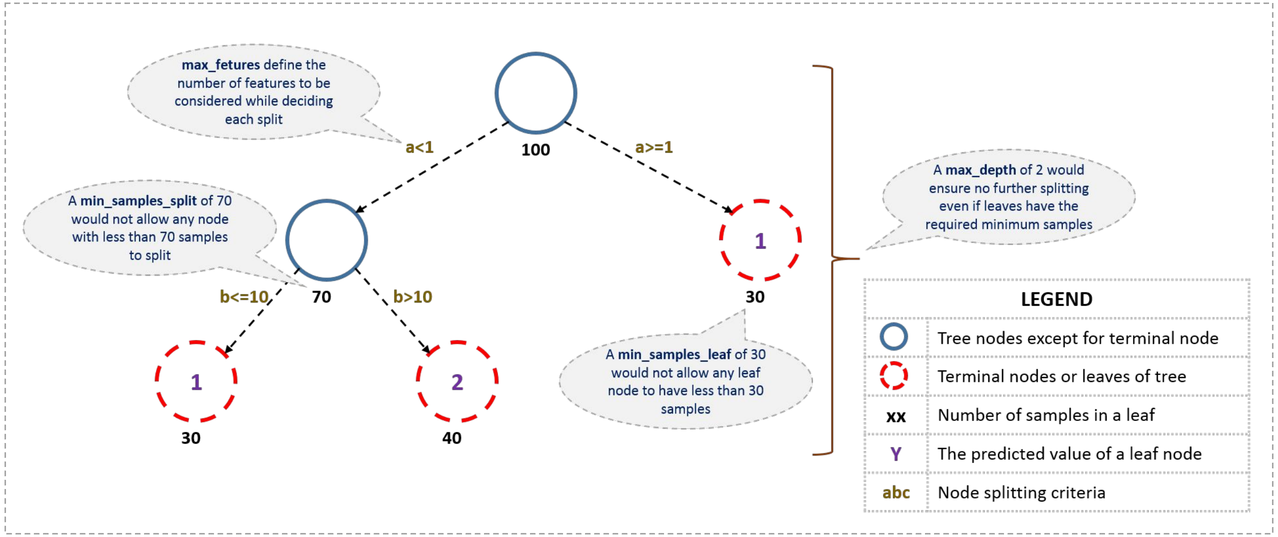
source: stackoverflow

Overfitting for Decision Tree

- Is it a good idea to grow the full tree all the time?
    - Suppose we add one noisy training sample: [sunny, hot, normal, true, no]
    - What effect on earlier tree?
- There may exist multiple trees that perform exactly the same, then which one should we select?
    - General Principle: **prefer the simplest hypothesis that fits the data**

How to Avoid Overfitting I

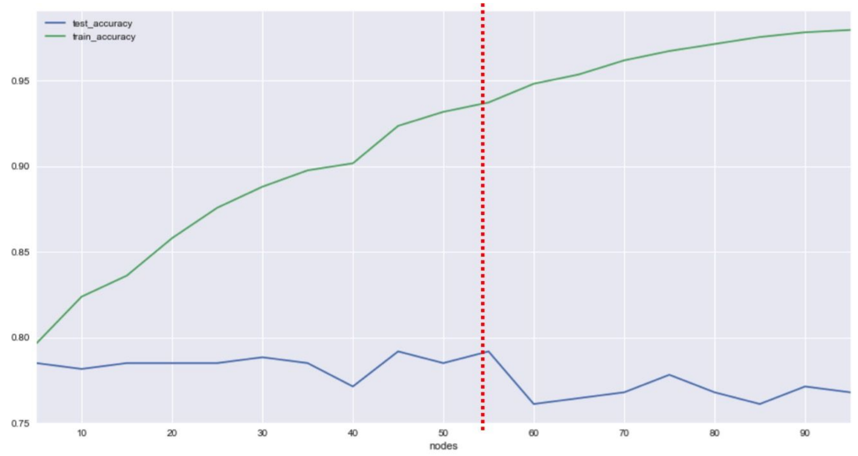
- Stop growing the tree given stopping criterions
  - Minimum samples for a node split
  - Minimum samples for a terminal node (leaf)
  - Maximum depth of tree (vertical depth)
  - ….


max_fetures define the number of features to be considered while deciding each split
a<1
100
a>=1
A min_samples_split of 70 would not allow any node with less than 70 samples to split
70
b<=10
b>10
1
30
2
40
1
30
A min_samples_leaf of 30 would not allow any leaf node to have less than 30 samples
A max_depth of 2 would ensure no further splitting even if leaves have the required minimum samples
LEGEND
Tree nodes except for terminal node
Terminal nodes or leaves of tree
xx Number of samples in a leaf
Y The predicted value of a leaf node
abc Node splitting criteria

source:https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/

How to Avoid Overfitting II

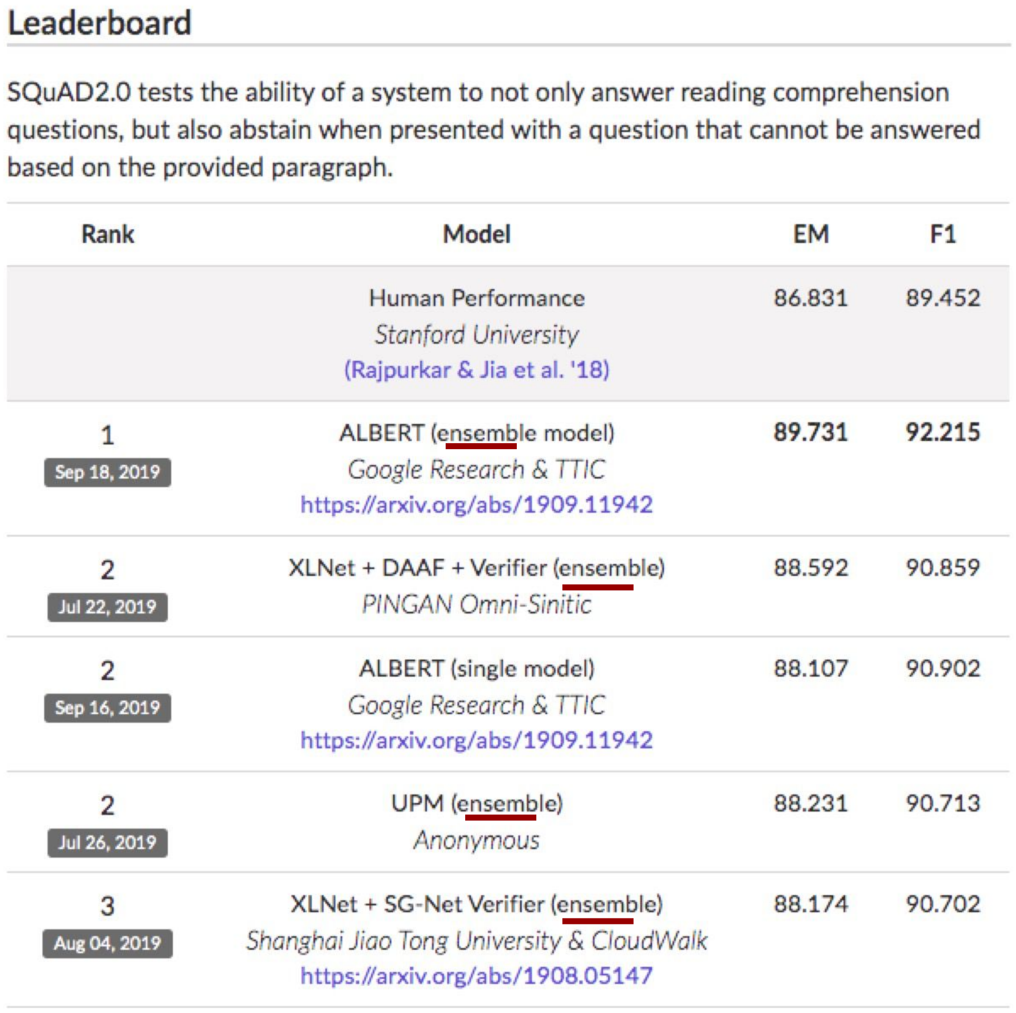
- Grow a “full” tree, and then to perform post-pruning
    - Split data into training and validation set
    - Build a full tree that classify training data
    - Do until further pruning is harmful, greedily remove the one that most improves validation set accuracy.



What is Ensemble Learning?

Intuition of Ensemble Learning

Three kids can defeat a master



Leaderboard

SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph.

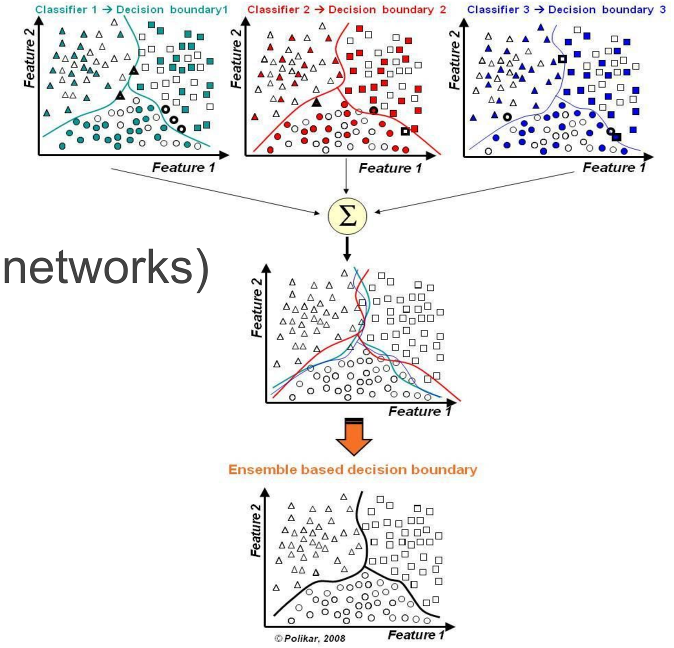
Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Sep 18, 2019	ALBERT (ensemble model) *Google Research & TTIC* https://arxiv.org/abs/1909.11942	**89.731**	**92.215**
2 Jul 22, 2019	XLNet + DAAF + Verifier (ensemble) *PINGAN Omni-Sinitic*	88.592	90.859
2 Sep 16, 2019	ALBERT (single model) *Google Research & TTIC* https://arxiv.org/abs/1909.11942	88.107	90.902
2 Jul 26, 2019	UPM (ensemble) *Anonymous*	88.231	90.713
3 Aug 04, 2019	XLNet + SG-Net Verifier (ensemble) *Shanghai Jiao Tong University & CloudWalk* https://arxiv.org/abs/1908.05147	88.174	90.702

Ensemble Learning

- Techniques that generate a group of base learner when combined have higher accuracy

- Strong v.s. Weak learner

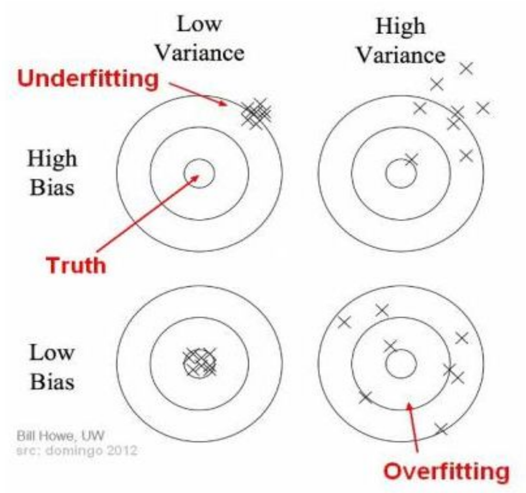
- Stable (kNN) v.s. Unstable (decision trees, neural networks) machine learning algorithms


Classifier 1 → Decision boundary1
Classifier 2 → Decision boundary 2
Classifier 3 → Decision boundary 3
Feature 2
Feature 1
Σ
Ensemble based decision boundary
© Polikar, 2008

Bias-Variance

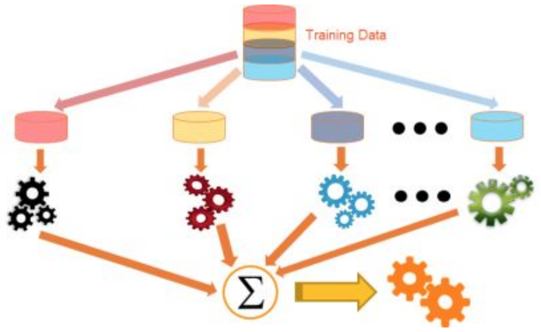
- **Bias**: the difference between the average prediction of our model and the correct value which we are trying to predict

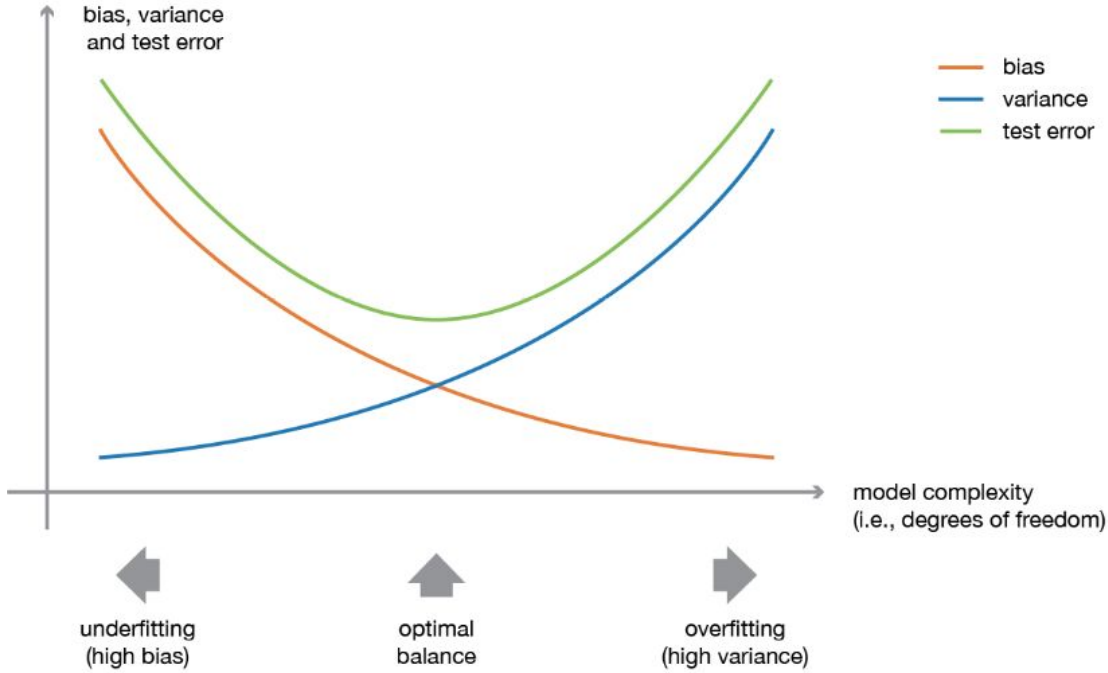
- **Variance**: the variability of model prediction for a given data point or a value which tells us spread of our data



Why Ensemble?

- Reduce Bias
- Reduce Variance
- Prediction Error:
  = Bias ^2
  + Variance
  + Irreducible Error





Common Ensemble Techniques

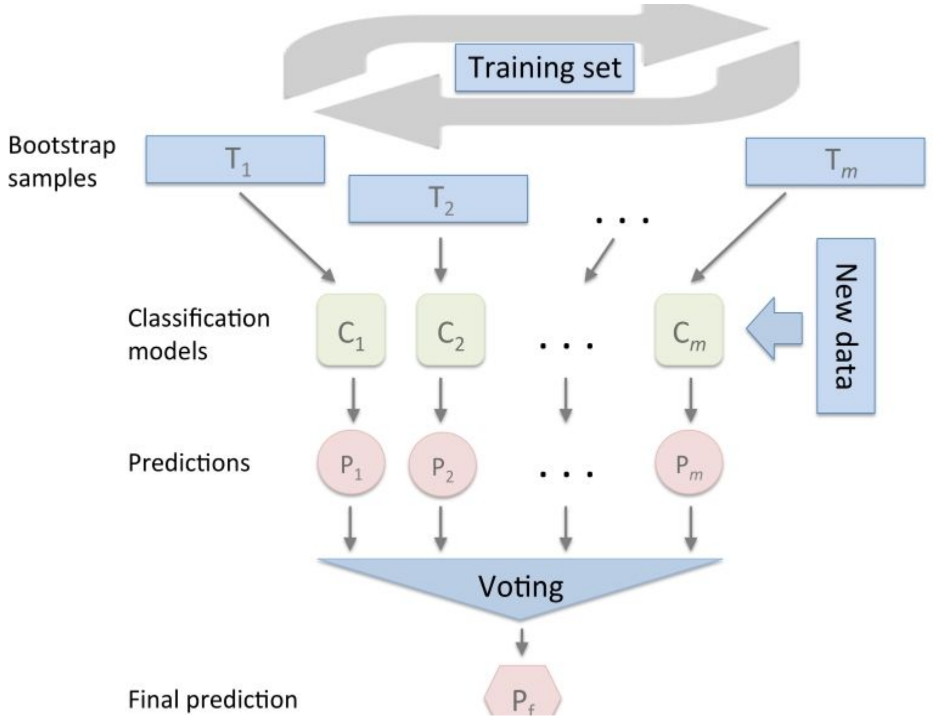
Ensemble Learning

- Bagging: reduce the variance in a model
  - Random Forest
- Boosting: reduce the bias in a model
  - Ada-Boost, XGBoost, Gradient Boosted Decision Trees
- Stacking: increase the prediction accuracy of a model
  - Mlxtend library

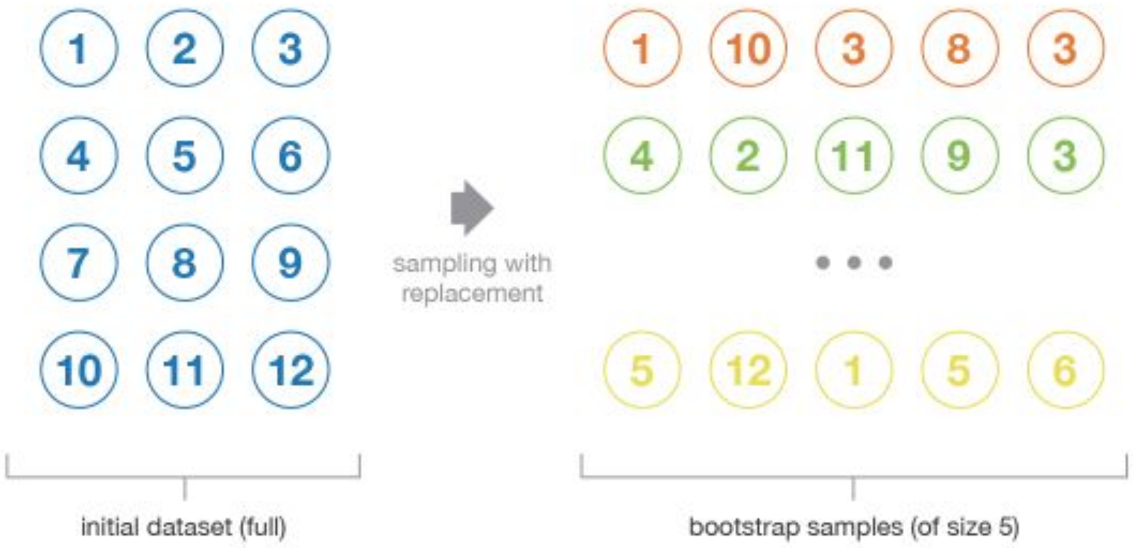
Bagging

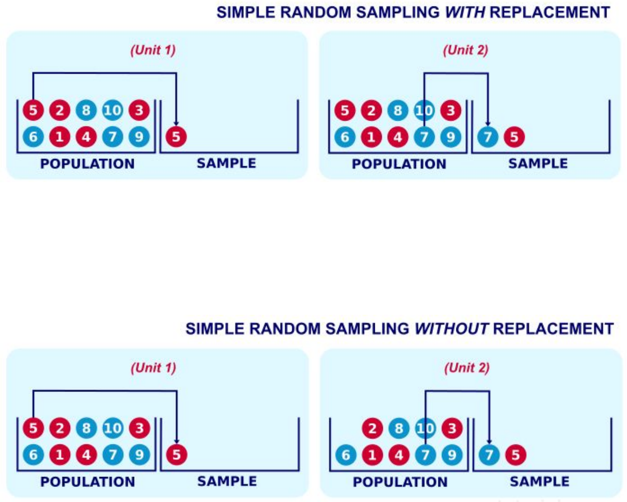
Bagging

- Bootstrap aggregation
- Train *m* classifier from *m* bootstrap replication
- Combine outputs by voting
- Decreases error by decreasing the variance
- Random Forest (Randomly select features)
- ExtraTrees (Randomized top-down split)



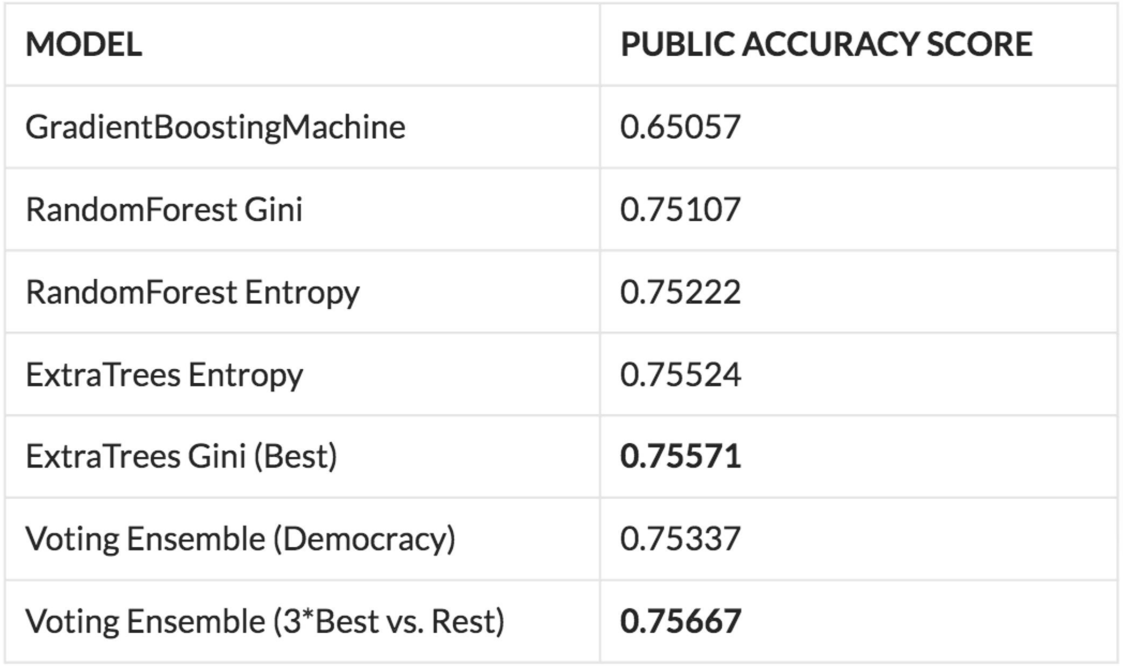
Bootstrapping





Majority Voting

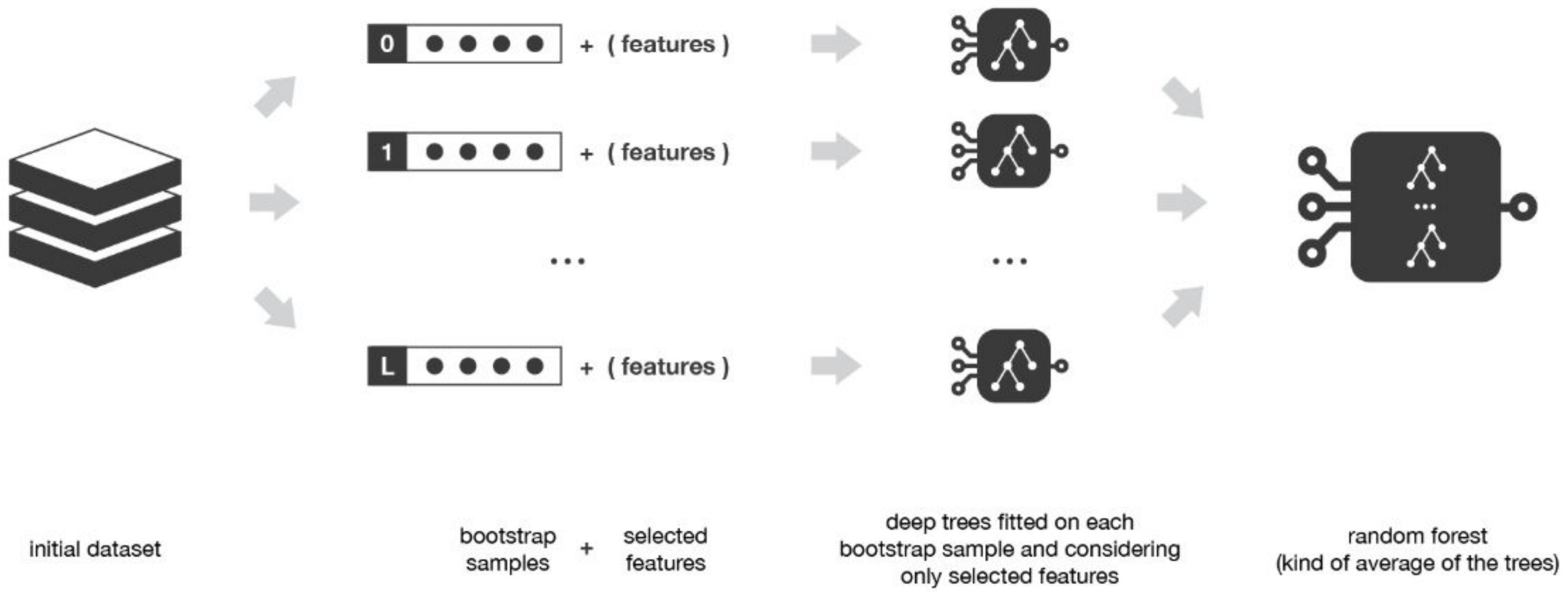
- **Equal**: same weight for all
- **Weighted**: best model get more weight in a vote



MODEL	PUBLIC ACCURACY SCORE
GradientBoostingMachine	0.65057
RandomForest Gini	0.75107
RandomForest Entropy	0.75222
ExtraTrees Entropy	0.75524
ExtraTrees Gini (Best)	**0.75571**
Voting Ensemble (Democracy)	0.75337
Voting Ensemble (3*Best vs. Rest)	**0.75667**

Random Forests

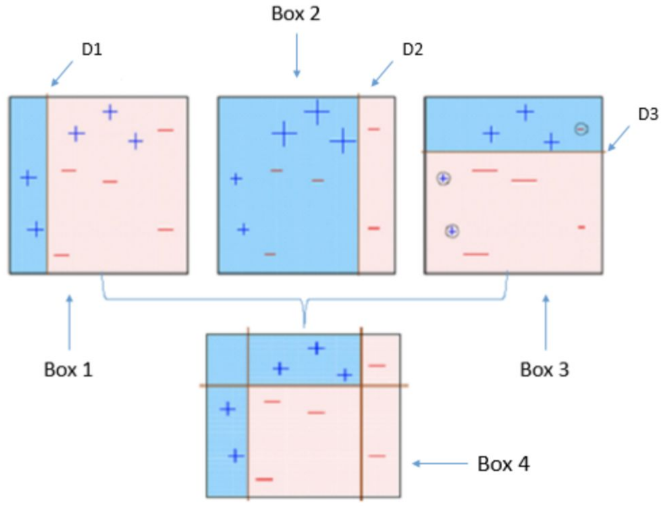
Source: Towards Data Science


0 ● ● ● ● + (features)
1 ● ● ● ● + (features)
...
L ● ● ● ● + (features)
...
initial dataset
bootstrap samples + selected features
deep trees fitted on each bootstrap sample and considering only selected features
random forest (kind of average of the trees)

Boosting

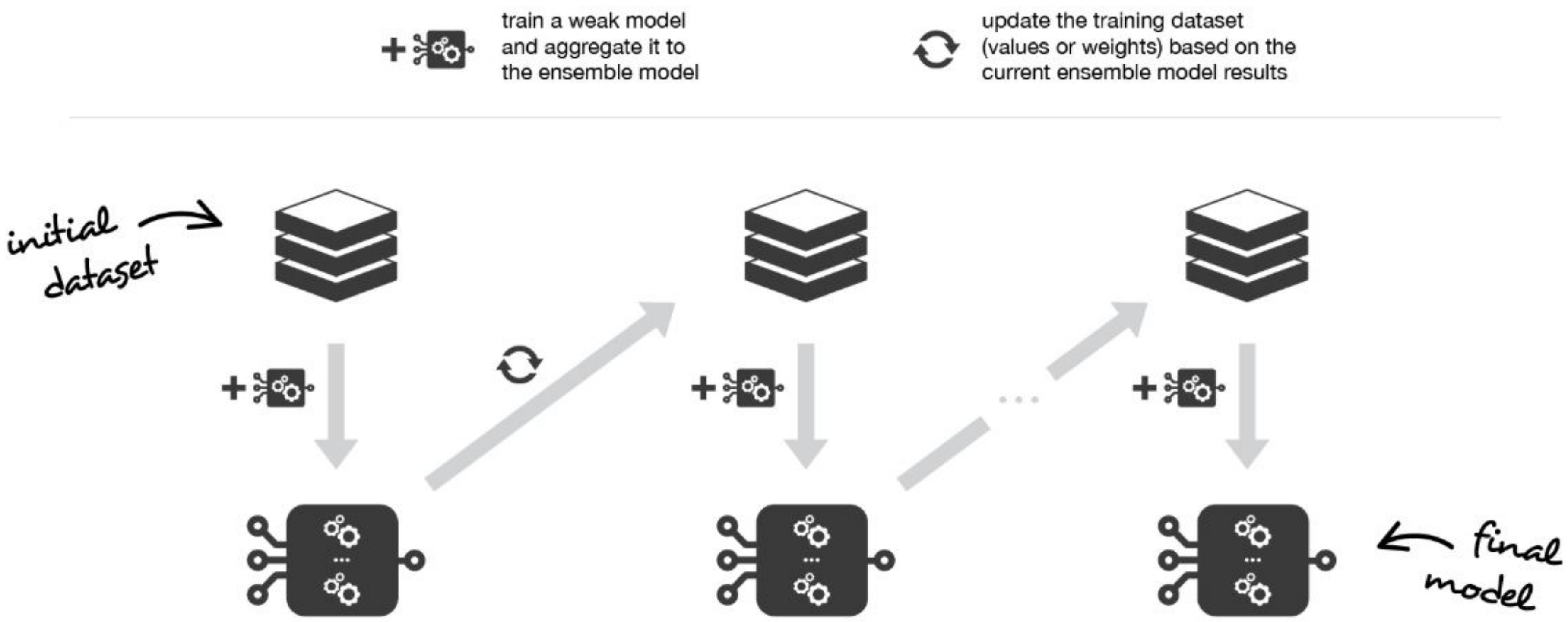
Boosting

- Training samples are given weights (initially same weight)
- At each iteration, a new hypothesis is learned.
- Training samples are reweighted to focus the model on samples that the most recently learned classifier got wrong.
- Combine output by voting
- Gradient Boosting, Adaboost, XGBoost, LightGBM



Boosting

Source: Towards Data Science

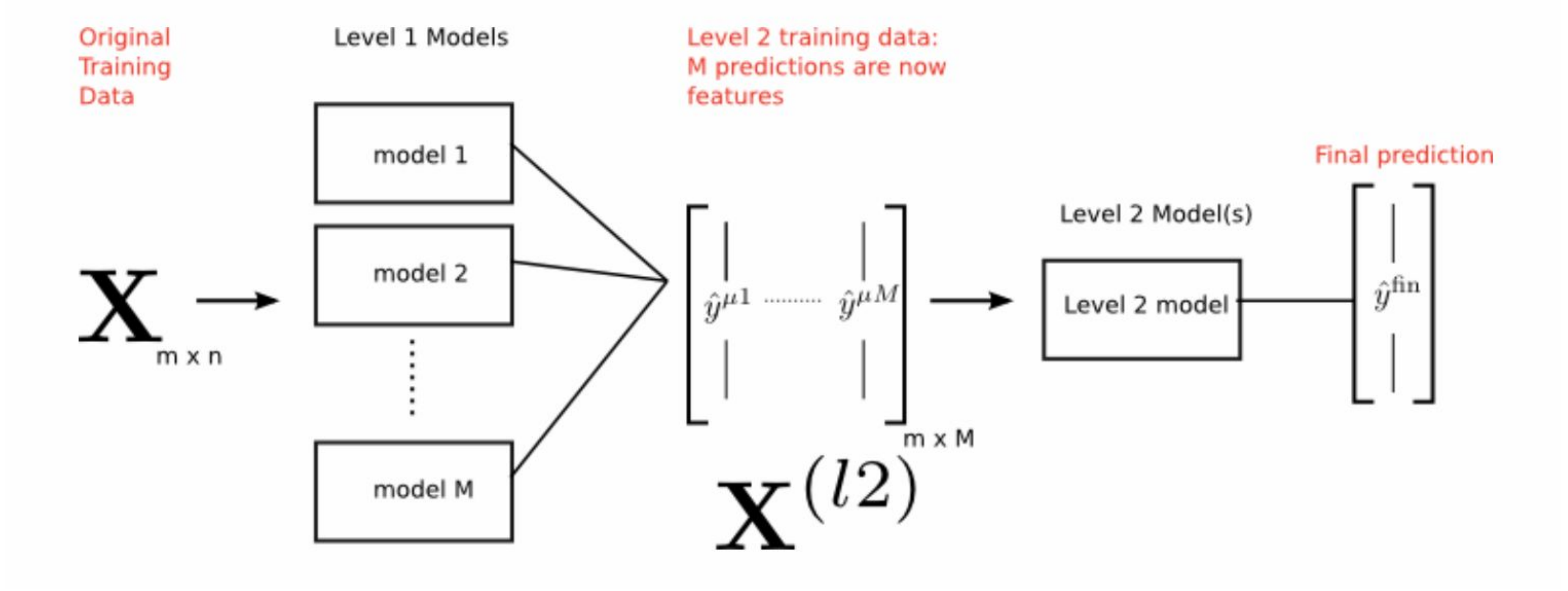


train a weak model and aggregate it to the ensemble model
update the training dataset (values or weights) based on the current ensemble model results
initial dataset
final model

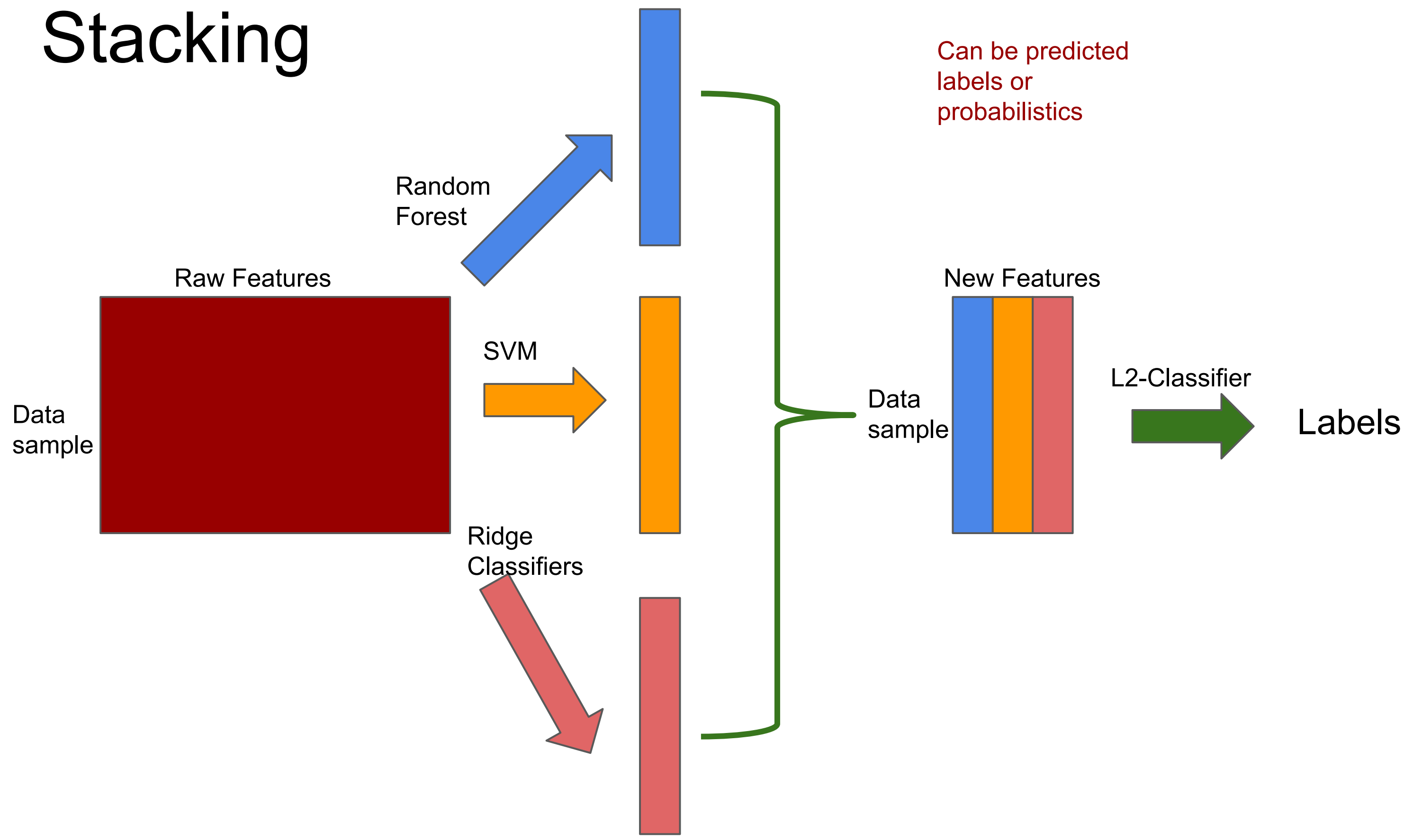
Stacking

Stacking

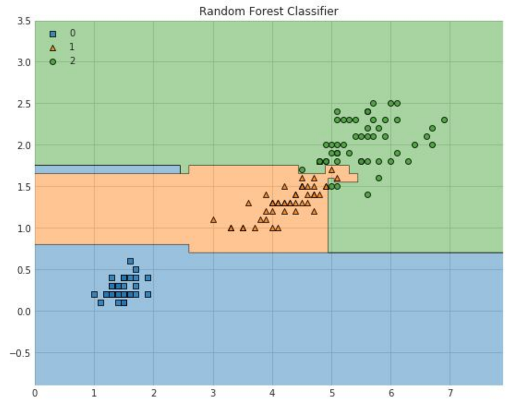
- Core idea: use a pool of base classifiers, then using another classifier (stacker) to combine their prediction for the final decision

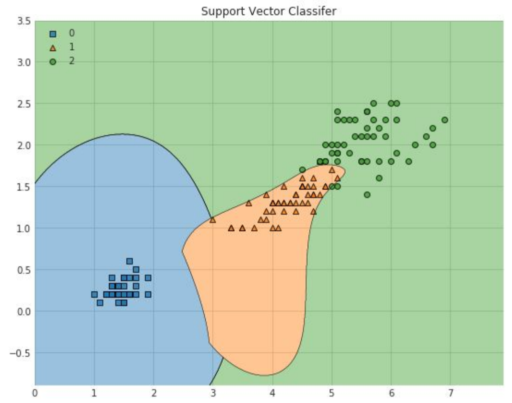


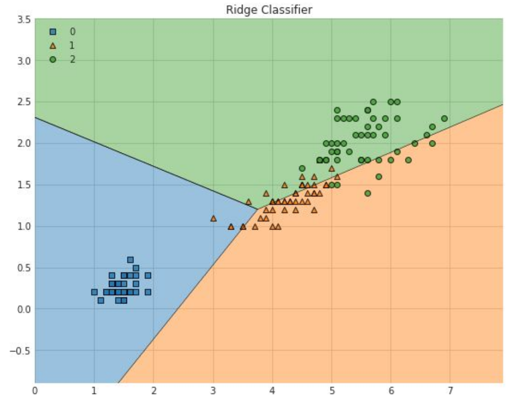
Stacking

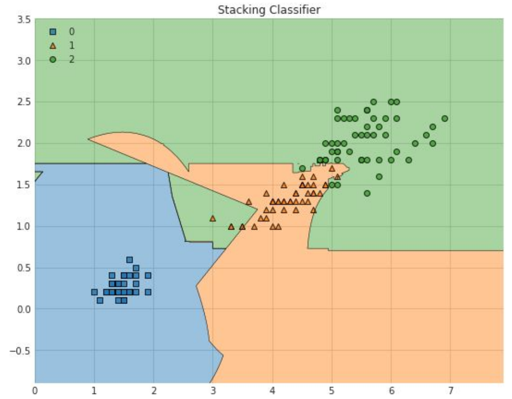

Random Forest
Raw Features
SVM
Data sample
Ridge Classifiers
Can be predicted labels or probabilistics
New Features
Data sample
L2-Classifier
Labels

Decision Regions: Demo Case



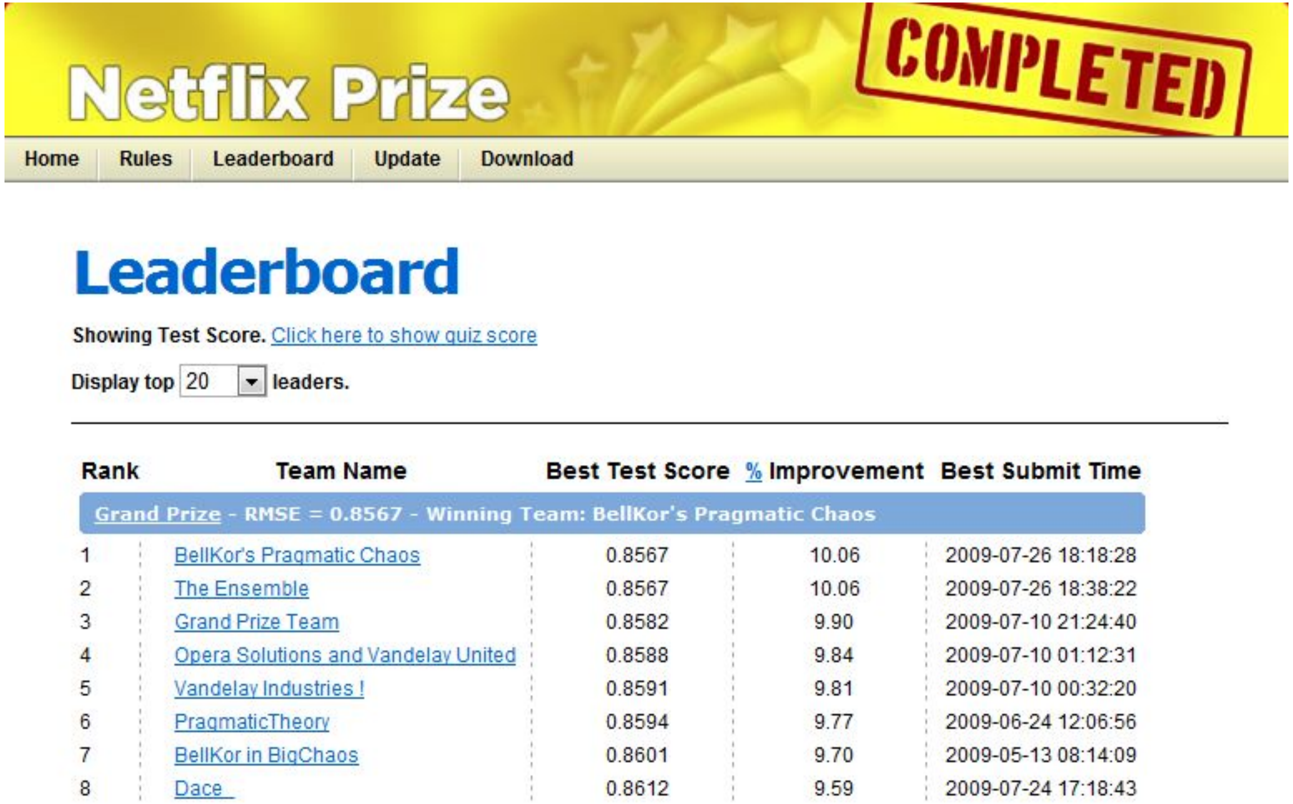






From Competition to Industry

Netfilx Competition



Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43

1 The winning solution is a final combination of **107** algorithms;

2 Are not fully implemented.

Some possible pitfalls

- Exponentially increasing training times and computational requirements
- Increase demand on infrastructure to maintain and update these models.

In a nutshell

- **No Free Lunch Theorem**: There is no one algorithm that is always the most accurate.
- Our efforts should focus on obtaining base models which make different kinds of errors, rather than obtaining highly accurate base models
- What we need to do is to build weak learners that are at least more accurate than random guessing
- Keep trying (experimenting, tuning, etc.) !