Supervised Machine Learning Algorithms

Naive Bayes, KNN, Decision Trees, Random Forests

Naïve Bayes

- Bayes Theorem

Generative Model (Likelihood)

Probability of Disease (Prior)

Marginal Probability of Features (Normalizing Constant)

Posterior Probability

$$P(D|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|D)P(D)}{P(\boldsymbol{X})}$$

$$= \frac{P(X_1 \cap X_2 \cap X_3 \cap \cdots \cap X_m|\mathrm{D})\,\mathrm{P(D)}}{P(\boldsymbol{X})}$$

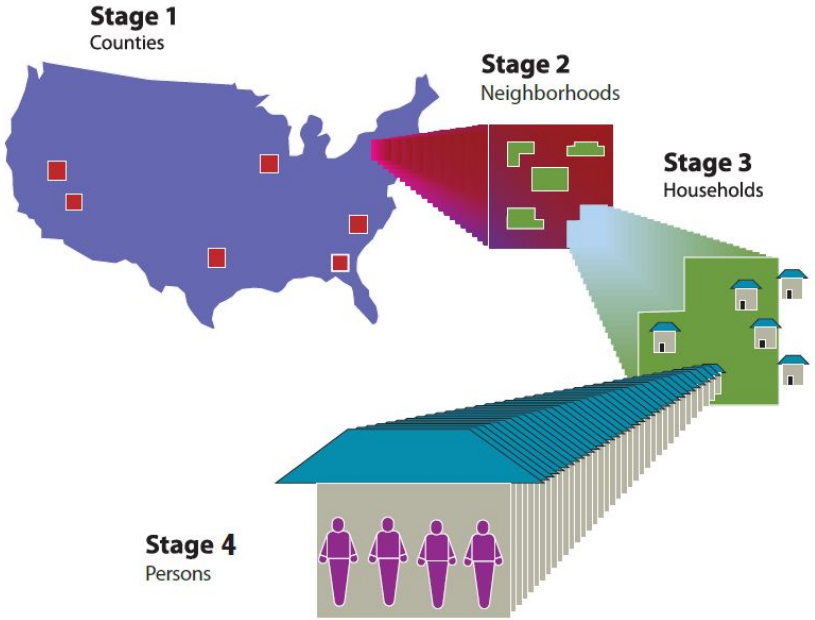
$$= \frac{P(X_1|X_2 \cap \cdots \cap X_m \cap D)P(X_2|X_3 \cap \cdots \cap X_m \cap D)P(X_3|X_4 \cap \cdots \cap X_m \cap D)\cdots P(D)}{P(\boldsymbol{X})}$$

$$\propto P(X_1|D)P(X_2|D)P(X_3|D)\ldots P(X_m|D)P(D)$$

"Naïve" Assumption of Conditional Independence

Example Data

- Survey Years: 2009-2010 & 2011-2012
- Sample Size: 2,876
- Outcome: General Health
  - Excellent/Very Good (42.53%)
  - Good (39.26%)
  - Fair/Poor (18.20 %)
- Features
  - Age
  - Poverty
  - BMI
  - Sex
  - Race/Ethnicity
  - Diabetes
  - Education
  - Marijuana Use



Naïve Bayes – Multinomial

- Using categorical (multinomial)/ continuous(gaussian) features
- Train
  - Construct frequency table to define the probability of each feature given the outcome status
  - Calculate marginal probabilities for each feature
  - Calculate marginal probability of the outcome
- Test
  - Calculate the posterior probability for each outcome with a specific set of features to define predicted class

Example: NHANES data

Education	Excellent/ Very Good	Good	Fair/Poor	
8th Grade	22	52	73	147
9-11th Grade	105	156	120	381
High School	213	268	139	620
Some College	375	401	144	920
College Grad	521	264	53	838
	1236	1141	529	

Marijuana Use	Excellent/ Very Good	Good	Fair/Poor	
No	521	464	252	1237
Yes	715	647	277	1639
	1236	1111	529	

Naïve Bayes – Multinomial/Gaussian

- Using categorical (multinomial)/ continuous(gaussian) features
- Train
  - Construct frequency table to define the probability of each feature given the outcome status
  - Calculate marginal probabilities for each feature
  - Calculate marginal probability of the outcome
- Test
  - Calculate the posterior probability for each outcome with a specific set of features to define predicted class

Example: NHANES data

P(X|D)

Education	Excellent/ Very Good	Good	Fair/Poor	P(X)
8th Grade	0.02	0.05	0.14	0.05
9-11th Grade	0.08	0.14	0.23	0.13
High School	0.17	0.23	0.26	0.21
Some College	0.30	0.35	0.27	0.32
College Grad	0.42	0.23	0.10	0.29
	0.43	0.39	0.18	

P(D)

Marijuana Use	Excellent/ Very Good	Good	Fair/Poor	
No	0.42	0.43	0.48	0.43
Yes	0.58	0.57	0.52	0.57
	0.43	0.39	0.18	

$$P(Excellent/VeryGood \mid 8th\ Grade\ \&\ No\ Marijuana\ Use) \propto (0.02 * 0.42) * 0.43 = 0.0036$$

$$P(Good \mid 8th\ Grade\ \&\ No\ Marijuana\ Use) \propto (0.05 * 0.43) * 0.39 = 0.0084$$

$$P(Fair/Poor \mid 8th\ Grade\ \&\ No\ Marijuana\ Use) \propto (0.14 * 0.48) * 0.18 = 0.01210$$

What if there's a zero in the frequency table?

$$P(Excellent/VeryGood \mid 8th\ Grade\ \&\ No\ Marijuana\ Use) \propto (0 * 0.42) * 0.43 = 0$$

$$P(Good \mid 8th\ Grade\ \&\ No\ Marijuana\ Use) \propto (0.05 * 0.43) * 0.39 = 0.0084$$

$$P(Never \mid 8th\ Grade\ \&\ No\ Marijuana\ Use) \propto (0.14 * 0.48) * 0.18 = 0.01214$$

Laplace Smoothing

Add α to every category in the frequency table (hyperparameter tuning)

Example:	α = 1	α = 100	α = 10000
P(8th grade | Very Good)	0.0185	0.0703	0.1956
P(8th grade | Good)	0.0462	0.0926	0.1966
P(8th grade | Fair)	0.1386	0.1681	0.1994

$$P(X|D) = \frac{\sum 1(X \cap D) + \alpha}{\sum D + \alpha * k}$$

where k = # of categories of X

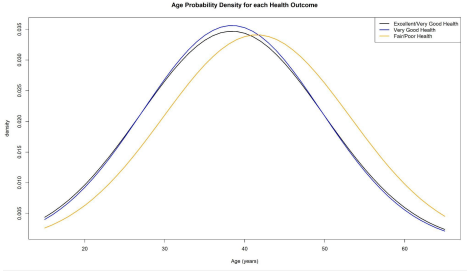
As α increases, it introduces bias and leads to uniform conditional probabilities for each outcome

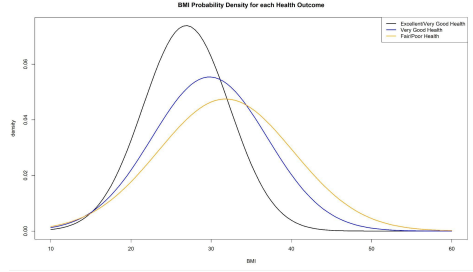
Gaussian distribution: $f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$

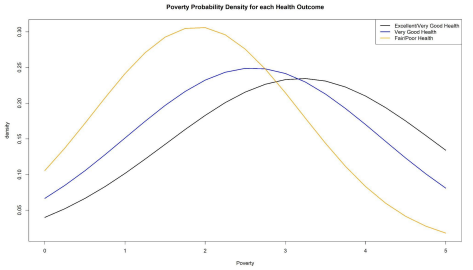
Naïve Bayes – Gaussian

- Using continuous(gaussian) features
- Train
  - Calculate the mean and standard deviation (sd) of the continuous feature for each outcome status
- Test
  - Calculate the likelihood for each outcome status [P(**X**|D)] based on the product of the conditional likelihoods and probability of the outcome status or the sum of the log-likelihoods

	Excellent/ Very Good	Good	Fair/Poor
Age	μ:38.4 σ:11.5	μ:38.4 σ:11.2	μ:41.5 σ:11.7
BMI	μ:26.9 σ:5.4	μ:29.8 σ:7.2	μ:31.8 σ:8.4
Poverty	μ:3.2 σ:1.7	μ:2.6 σ:1.6	μ:1.9 σ:1.3







$$P\left(Excellent/VeryGood \mid Age = 50, BMI = 30, Poverty = 3\right) \propto (0.021 * 0.063 * 0.233) * 0.43 = 0.00013$$

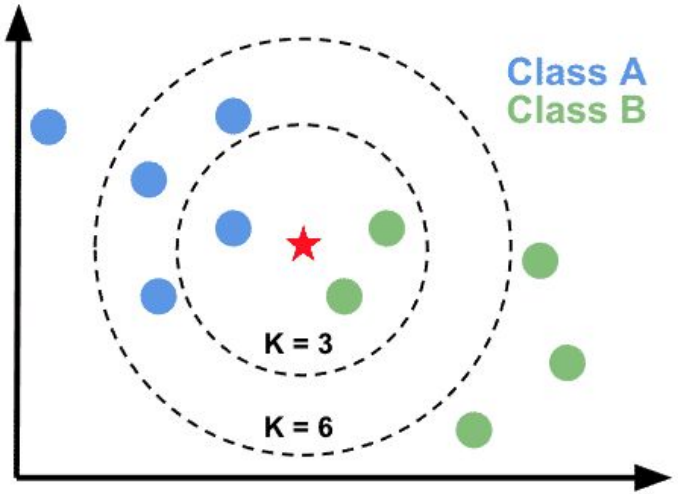
$$P\left(Good \mid Age = 50, BMI = 30, Poverty = 3\right) \propto (0.021 * 0.055 * 0.242) * 0.39 = 0.00011$$

$$P\left(Fair/Poor \mid Age = 50, BMI = 30, Poverty = 3\right) \propto (0.026 * 0.046 * 0.215) * 0.18 = 4.69\ x\ 10^{-5}$$

K-Nearest Neighbors (k-NN)

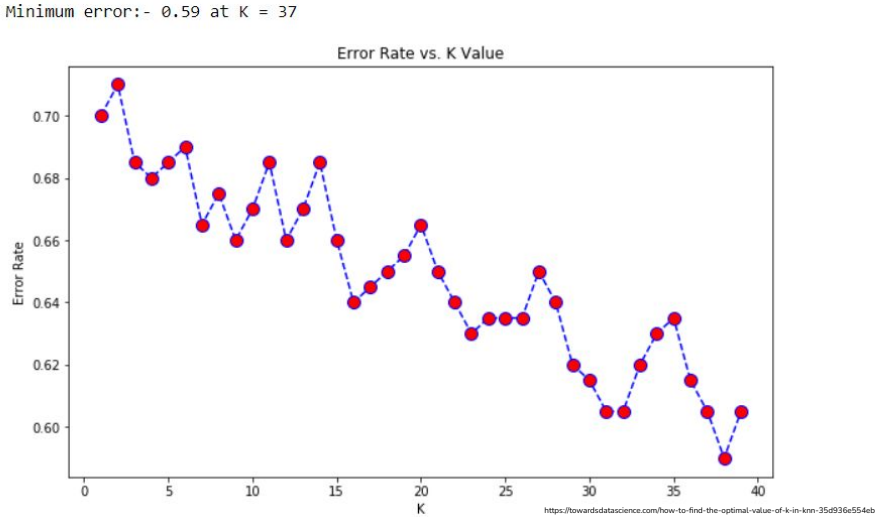
General premise:

1. “Birds of a feather flock together”
2. Assumption: similar data points are proximal
3. Can be used for classification or regression
4. Vote for the most frequent label (classification)
5. Use mean of the labels (regression)
6. Choice of K matters



Considerations: Choosing K

- There are no generalizable statistical methods for determining the optimal K-value
- In general, want to avoid very small K-values to prevent unstable decision boundaries
- Increasing value of K too high can lead to increased error rate
- Ultimately, the algorithm will need to be run several times



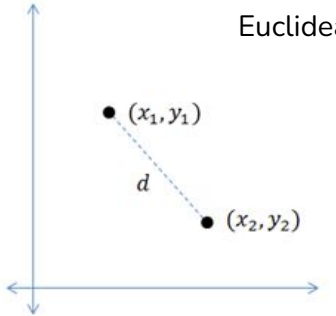
Considerations: Distance Metrics

- **Euclidean Distance** (most popular and familiar)
  - Default in SKlearn
  - Easy to conceptualize
  - Suffers in high-dimensional problems– distances become too uniform

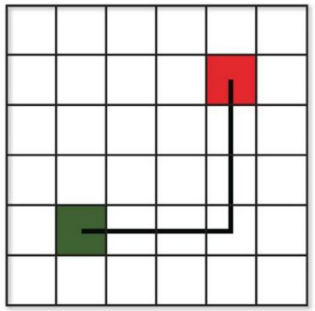
- **Manhattan Distance**
  - “Taxicab distance”
  - Better performance in high-dimensional problems

- Other distance metrics could be appropriate: problem-specific

Euclidean Distance



$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



Manhattan Distance

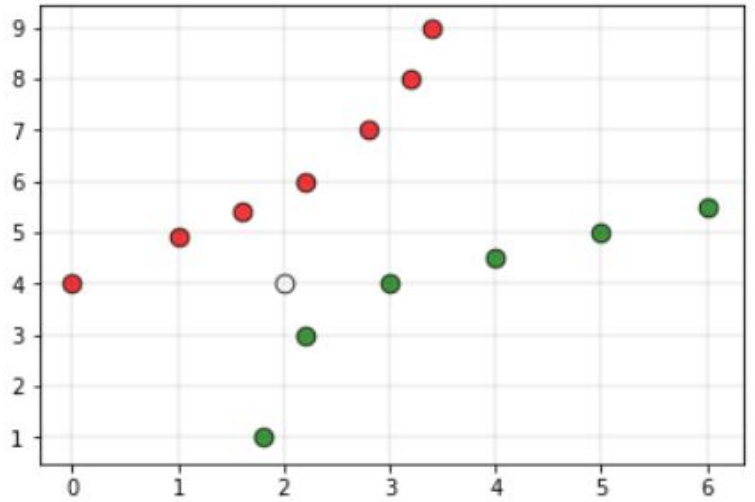
$$d = \sum_{i=1}^{n} |x_i - y_i|$$

https://www.kdnuggets.com/2020/11/most-popular-distance-metrics-knn.html

Considerations: Distance Weighting

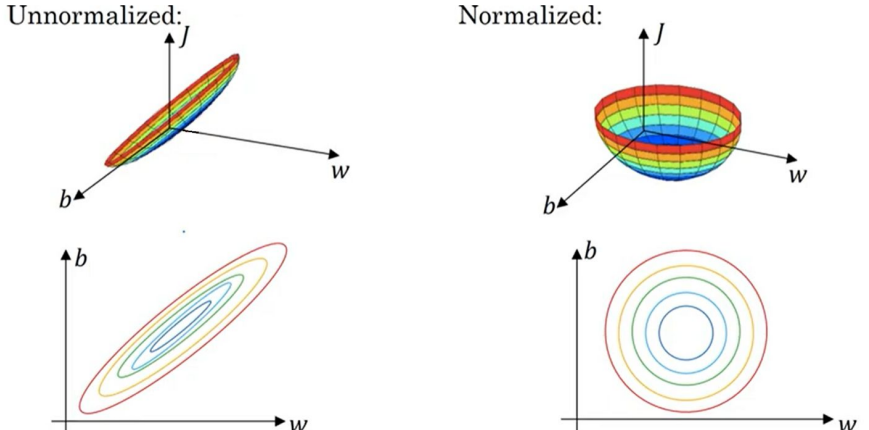
- Rather than taking votes directly from the k nearest neighbors, it can be desired to weight these votes based on their distance from the new data point
- Benefits: reduce chance of ties, can improve accuracy if closer neighbors more reliably indicate the class
- Different ways to weight:

a. 1-distance
b. 1/distance
c. 1/distance^2



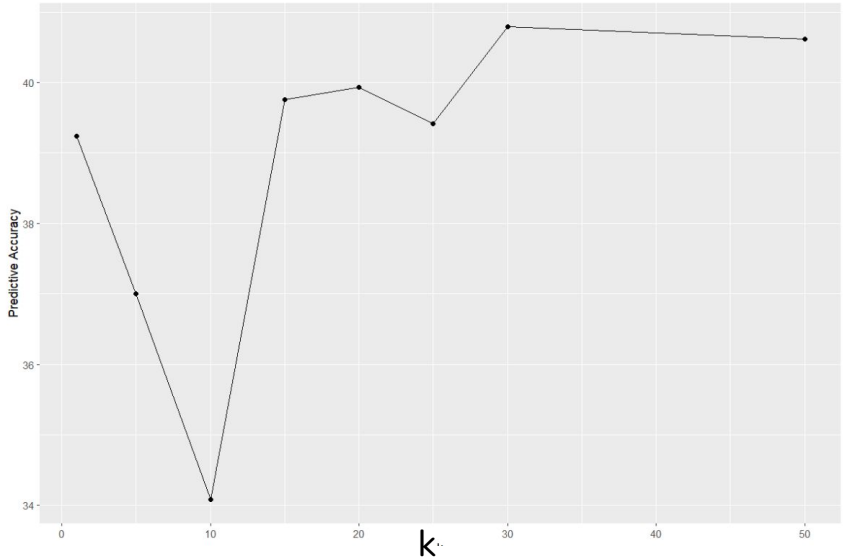
Additional Considerations

- The balance of the classes in the data should be evaluated prior to the selection of the k value
- Highly imbalanced classes could hinder classification performance (k nearest neighbors more likely to belong to the class with more examples). A small value of k may perform better in this situation.
- Features should be normalized prior to calculation of distances to prevent uneven contributions



An Example: nhanes

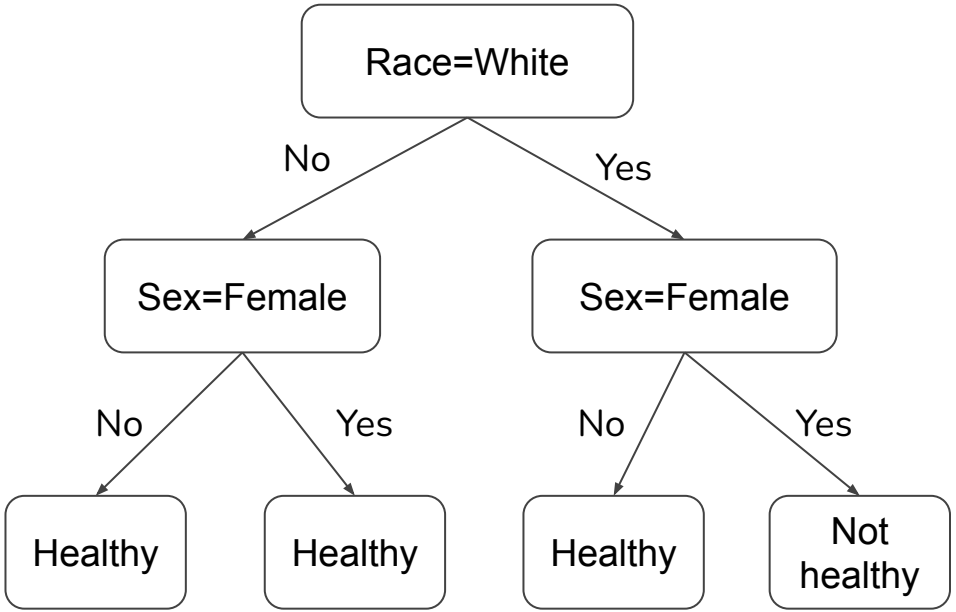
- Test accuracy maximized around k=30, no improvement for larger k
- “Weak learner”

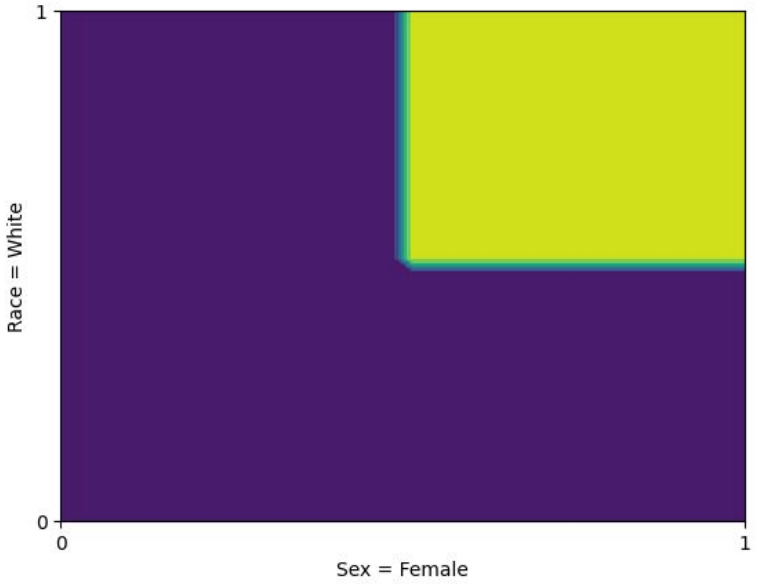


Decision Tree

- Non-parametric, supervised learning method for classification/regression
- Can be expressed as a tree-like graph
- Nodes represents decision rules, edges represent T/F of the decision rule
- A piecewise constant approximation

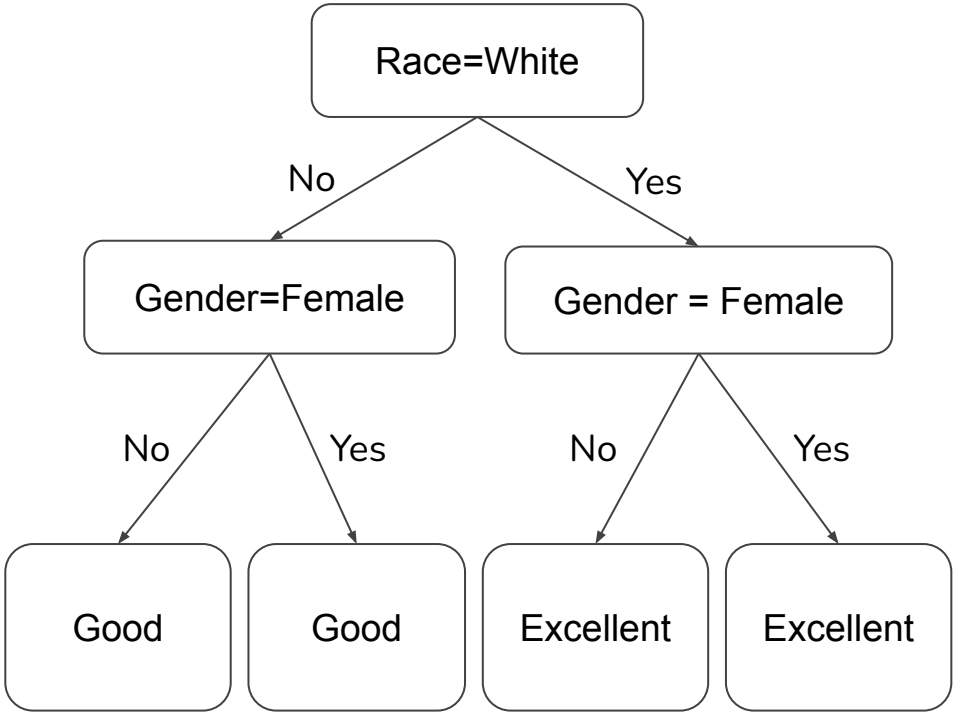
Example: Predict (binary) general health status with sex and race

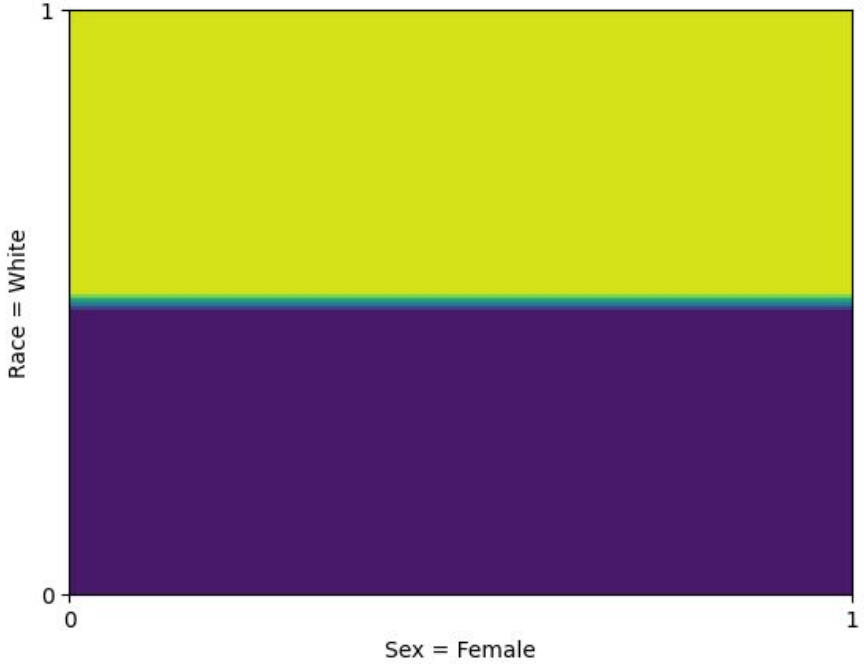




Decision Tree: Multi-level outcome

General health status: Poor, Good and Excellent





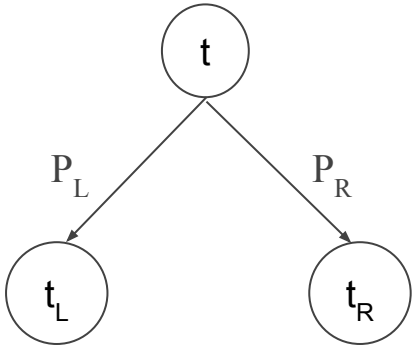
Build a decision tree：when and where to split

- **Gini impurity**
- In a J-class classification problem, draw a random sample X at a node t

$$g(t) = \sum_{j=1}^{J} P(X=j|t)P(X\neq j|t) = 1-\sum_{j=1}^{J} P^2(X=j|t)$$

- The probability of a random sample being misclassified
- **Gini gain**: decrease of Gini index after splitting

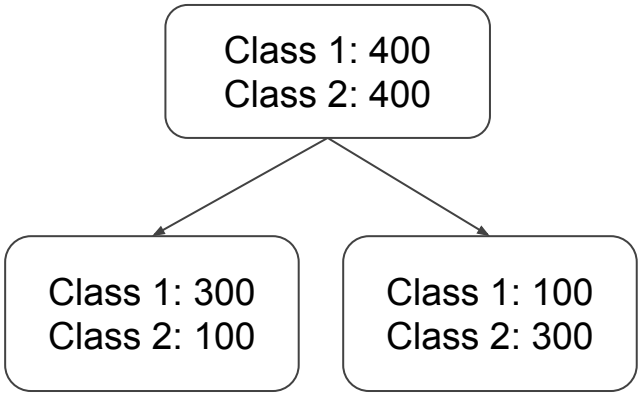
$$\Delta g = g(t)-P_L g(t_L)-P_R g(t_R)$$

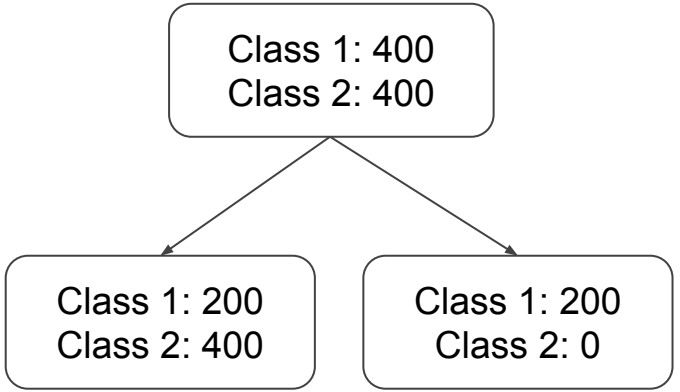


- **Algorithm: CART**
- Iterate through all discrete threshold of features,
- Find the split at t that maximizes Gini gain
- Why not use misclassification rate?

Build a decision tree：when and where to split

- The short answer: misclassification rate does not reward favorable splits properly
- Consider the following example: binary classification



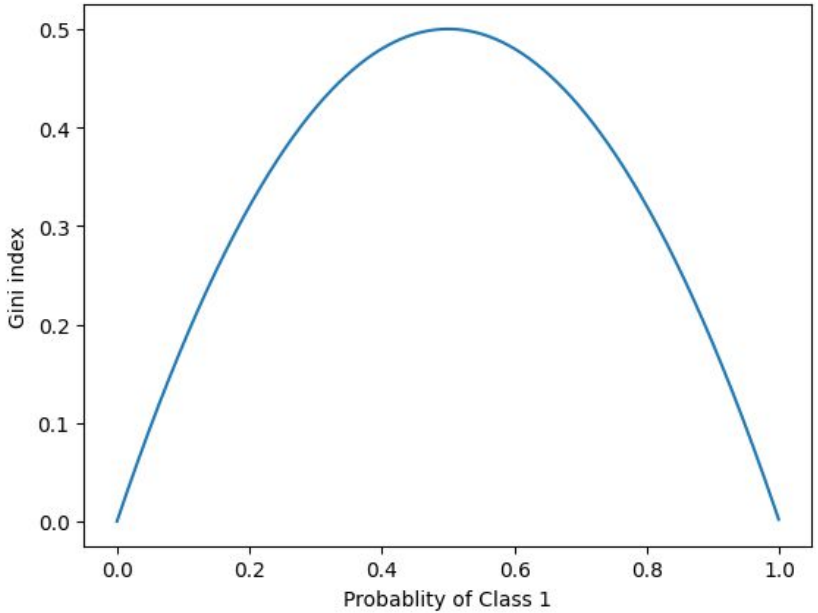


Misclassification rate	25%	vs	25%
Gini impurity	37.5%	vs	33.3%

Build a decision tree：when and where to split

- Desired properties of Gini index and Gini gain
  - Interpretation of Gini index attached to misclassification rate
  - Gini index is minimized when all subjects are classified to the same category
  - Gini gain is strictly concave

- Not desired properties
  - Gini gain is non-negative for all splits
  - Not as bad, but still prone to overfit

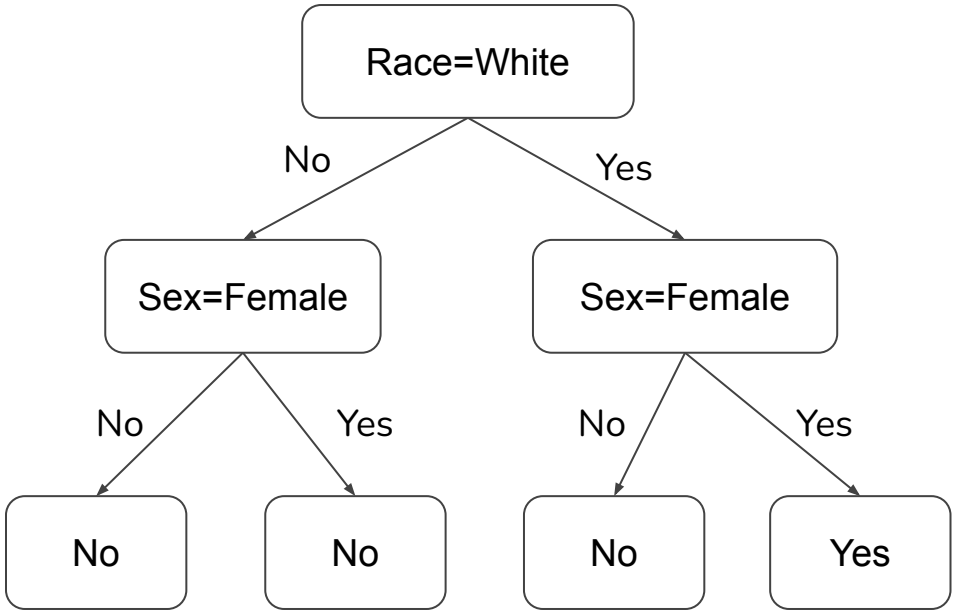


Build a Decision Tree: When to stop?

- Goal: to find the smallest tree with accurate estimation
- Threshold for impurity gain: stop when $\Delta g(\lambda, t) < \beta$
  - Difficult to choose the threshold value
  - Potential splits with large impurity gain following the current split
- Pruning
  - Grow a large tree, then prune it upwards for an "**optimal subtree**"
  - The number of subtrees is usually very large, leading to "**selective pruning**"
- Minimal Cost-Complexity pruning (CCP)
  - Penalized Loss: $R_\alpha(T)=R(T)+\alpha|T_{leaf}|$
  - $|T_{leaf}|$ is the number of "leaf" or "terminal" nodes of a subtree, a measure of tree complexity
  - α is the "complexity parameter", typically determined by cross-validation
  - Solves mathematical problems associated with selective pruning
- Tree depth
  - The maximum number of splits before a prediction (length of the "longest path")
  - Another measure of tree complexity
  - Typically determined by cross validation

Weak vs Strong learner

- **Weak learner:** performs slightly better than a random guess
- **Strong learner:** performs much better than a random guess
- **Ensemble models:** construct a strong learner by pooling many weak learners

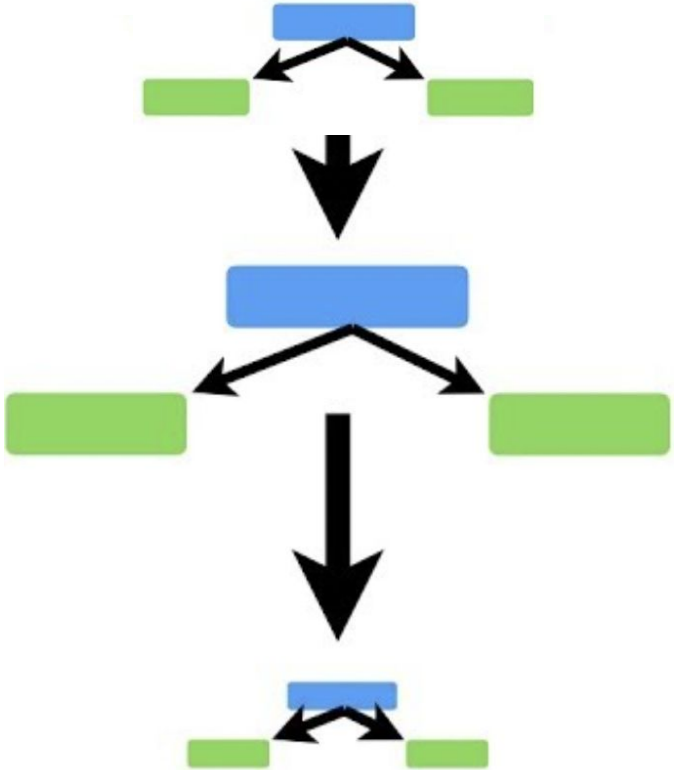


Classification accuracy is 58.12% - weak learner for sure!

AdaBoost

AdaBoost with decision trees

- Trees made with AdaBoost are just a node and two leaves.
- AdaBoost combines a lot of “weak learners” to make classifications.
- Some trees get more say (aka importance/influence) in the classification than others.
- Each tree is made by taking the previous tree’s error into account.



AdaBoost

- At start, all samples are equally important.
- Find the feature that does the best job classifying the samples.
- The total error for a tree is the sum of the weights associated with the incorrectly classified samples.
- Amount of Say = ½ log [(1-Total Error)/Total Error].

BMI	College Graduate	Diabetes	Health Outcome	Weight
28	Yes	No	Good	1/3
30	No	Yes	Poor	1/3
35	Yes	No	Excellent	1/3

AdaBoost

- Modify the weights so that the next tree will take the errors that the current tree made into account.
- Increase the sample weight for the incorrectly classified sample. New sample weight=sample weight * exp(amount of say). Large amount of say increase the sample weight more.
- Decrease the sample weight for the correctly classified sample. New sample weight=sample weight * exp(-amount of say). Large amount of say decrease the sample weight more.

BMI	College Graduate	Diabetes	Health Outcome	Weight	New Weight
28	Yes	No	Good	1/3	1/6
30	No	Yes	Poor	1/3	2/3
35	Yes	No	Excellent	1/3	1/6

AdaBoost

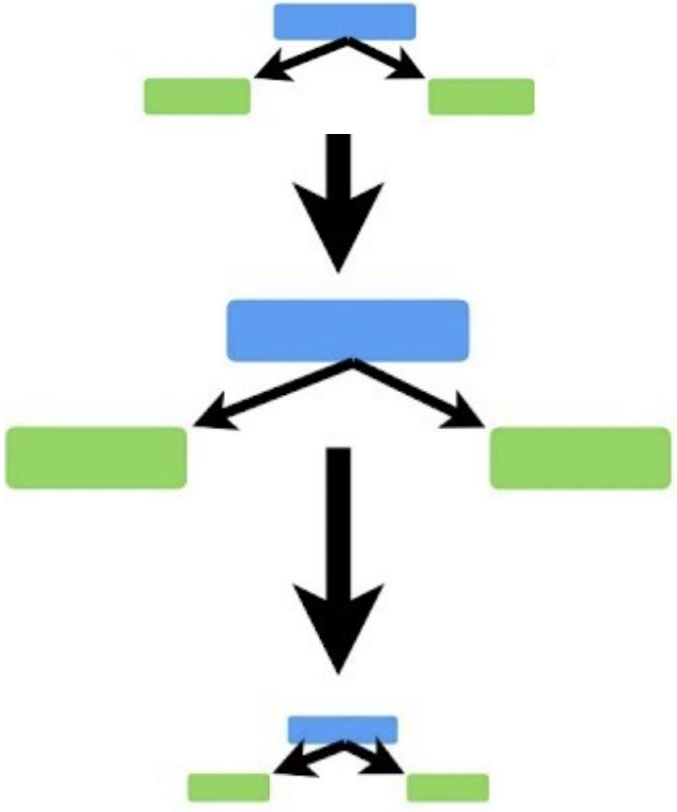
- Using weighted gini indexes to determine which variable should split the next tree.
- The weighted gini index would put more emphasis on the sample with the large weight.
- Find the tree that does the best job classifying the collection of samples with new weight.

BMI	College Graduate	Diabetes	Health Outcome	Weight	New Weight
28	Yes	No	Good	1/3	1/6
30	No	Yes	Poor	1/3	2/3
35	Yes	No	Excellent	1/3	1/6

AdaBoost

How to make classification

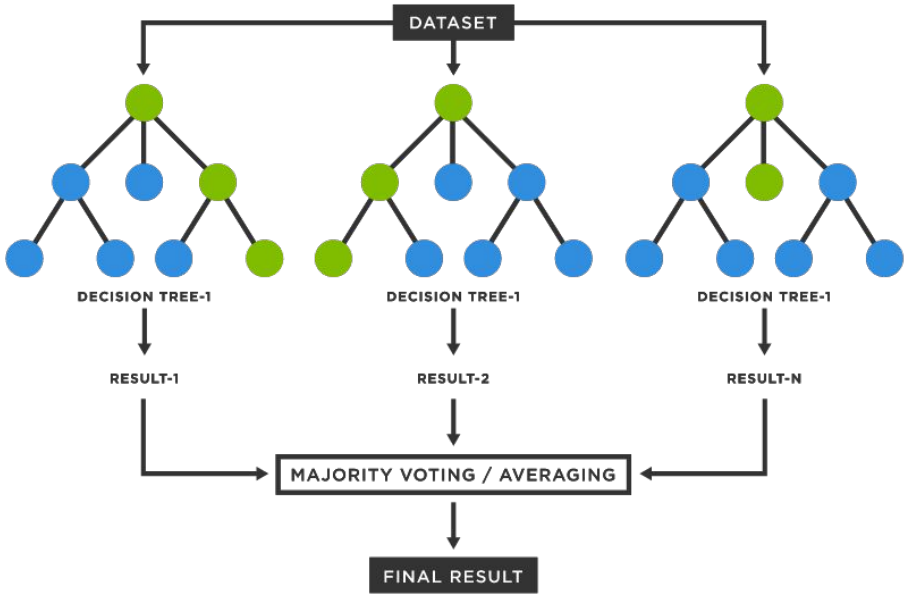
- First batch of trees classified the new sample as “Good” for health outcome, and second batch of trees classified the new sample as “Poor” for health outcome.
- Add amount of say for two batch of trees respectively, and we classify the new sample based on the larger sum.



Random Forests

Motivation

- Decision trees are highly sensitive to the training data, which could result in high variance.
- The inaccuracy is the major issue of decision trees for predictive learning. Trees work great with the training data, but they are not flexible when it comes to classifying new samples.



Random Forests

Creation of decision tree bags

1. Create a bootstrapped dataset that is the same size as the original.
2. Create a decision tree using the bootstrapped dataset, but only use a random subset of variables (features) at each step.
3. Repeat the 1 and 2 to build wide variety of trees.

BMI	College Graduate	Diabetes	Outcome
28	Yes	No	Good
30	No	Yes	Poor
35	Yes	No	Excellent

Original Dataset

BMI	College Graduate	Diabetes	Outcome
28	Yes	No	Good
28	Yes	No	Good
35	Yes	No	Excellent

Bootstrapped Dataset

Random Forests

How to make classification

Run the new data down the trees that we made. Conclusion are made based on which option received more votes. This process of combining results from multiple models is called aggregation.

BMI	College Graduate	Diabetes	Health Outcome
25	Yes	No	

New Sample

Health Outcome		
Good	Poor	Excellent
4	2	1

Random Forests

Evaluating a random forest

1. For the original data that is not in the bootstrapped datasets, we can run these data through the trees that were built without it and see if it correctly classifies the sample.
2. Measure how accurate our random forest is by the proportion of samples that were correctly classified by the random forest.
3. Deciding the number of the variables at each step based on the accuracy of the random forest.

BMI	College Graduate	Diabetes	Health Outcome
30	No	Yes	Poor

Out-Of-Bootstrap Sample

Out-Of-Bootstrap Sample		
Good	Poor	Excellent
1	2	0

Algorithm Properties: Naive Bayes & kNN

	Naive Bayes	**kNN**
Features	Categorical & Continuous	Categorical & Continuous
Outcomes	Binary/Multi-label classification	Binary/Multi-label Classification Regression
Effect of Number of Features & Sample Size	Small change in training speed and computational resources	Computational expense scales with sample size and number of dimensions, method of calculating distances must be considered
Train/Test Cost	Train cost higher than testing but very efficient compared to other methods	All computation performed during prediction, test cost scales with number of samples, dimensions, and value of k
Data Normalization	Not Needed	Needed
Sensitivity to Data Structure Changes	Yes	Yes
Data Imbalance Bias	Yes	Yes
Outliers	Cat: Zero probabilities Cont: Effects the conditional distribution	Outlier effect can be influenced by choice of k
Overfitting	Mostly robust to overfitting there are some instances where it can be an issue	Small value of k more likely to overfit
Explainability	Easy	Easy. More visually intuitive in low dimensional applications

Algorithm Properties: Decision Trees & Random Forests

	Decision Trees	Random Forest
Features	Categorical & Continuous	Categorical & Continuous
Outcomes	Binary/Multi-label Classification and Regression	Binary/Multi-label Classification and Regression
Effect of Number of Features & Sample Size	Computational load is very sensitive to change in sample size, even more sensitive to the number of features.	Computationally expensive and time-consuming with increasing number of features and sample size
Train/Test Cost	Training is time-consuming, but testing is inexpensive.	Training is time-consuming, and testing is easy with the current computational power.
Data Normalization	Not needed. Decision tree does not post assumptions on feature distribution.	Not needed
Sensitivity to Data Structure Changes	Yes	Yes
Data Imbalance Bias	Yes	Yes
Outliers	Mostly robust	Robust
Overfitting	High. But can be controlled with pruning	Low
Explainability	Conceptually easy, but not scalable with number of features.	Difficult. Random forests work like a black-box.