

Memorization Method - part 2

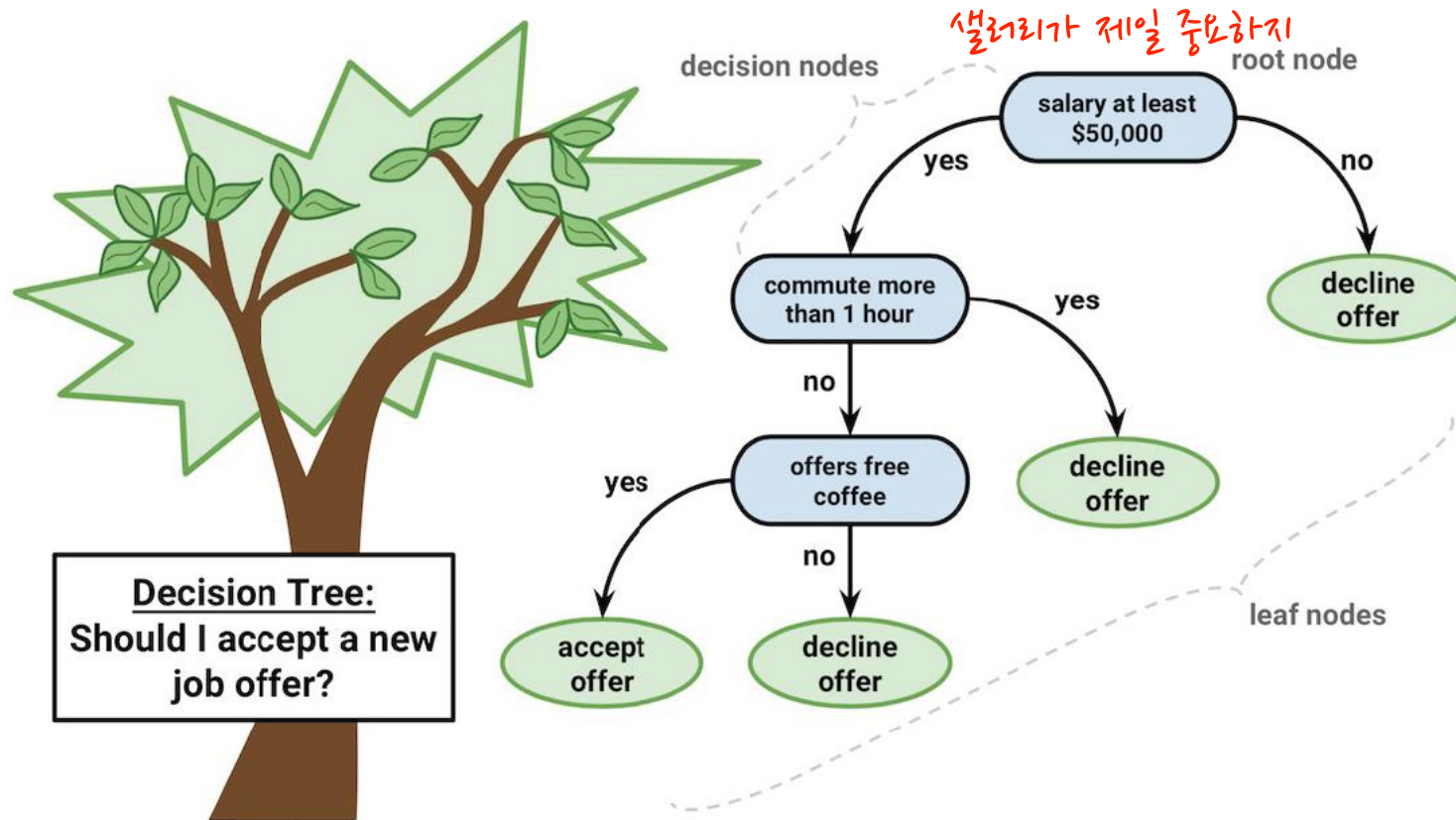
A series of horizontal lines in teal and light blue colors, with varying lengths and offsets, creating a layered, step-like effect across the middle of the slide.

Contents

- Decision Tree
- K-nearest neighbor
- Naïve Bayes

Decision Tree

Decision Tree Model



Before accepting a job offer, You may consider several **conditions**

Tree model organizes the **conditions** to make a **decision**

Conditions are organized from most significant condition to less significant ones

Loan Applications



Check Your Rate

Get a custom rate for your **\$35,000** loan in **1 click**

First Name

Last Name

Street Address

City

State

Zip Code

Date of Birth

Month	Day	Year
-------	-----	------

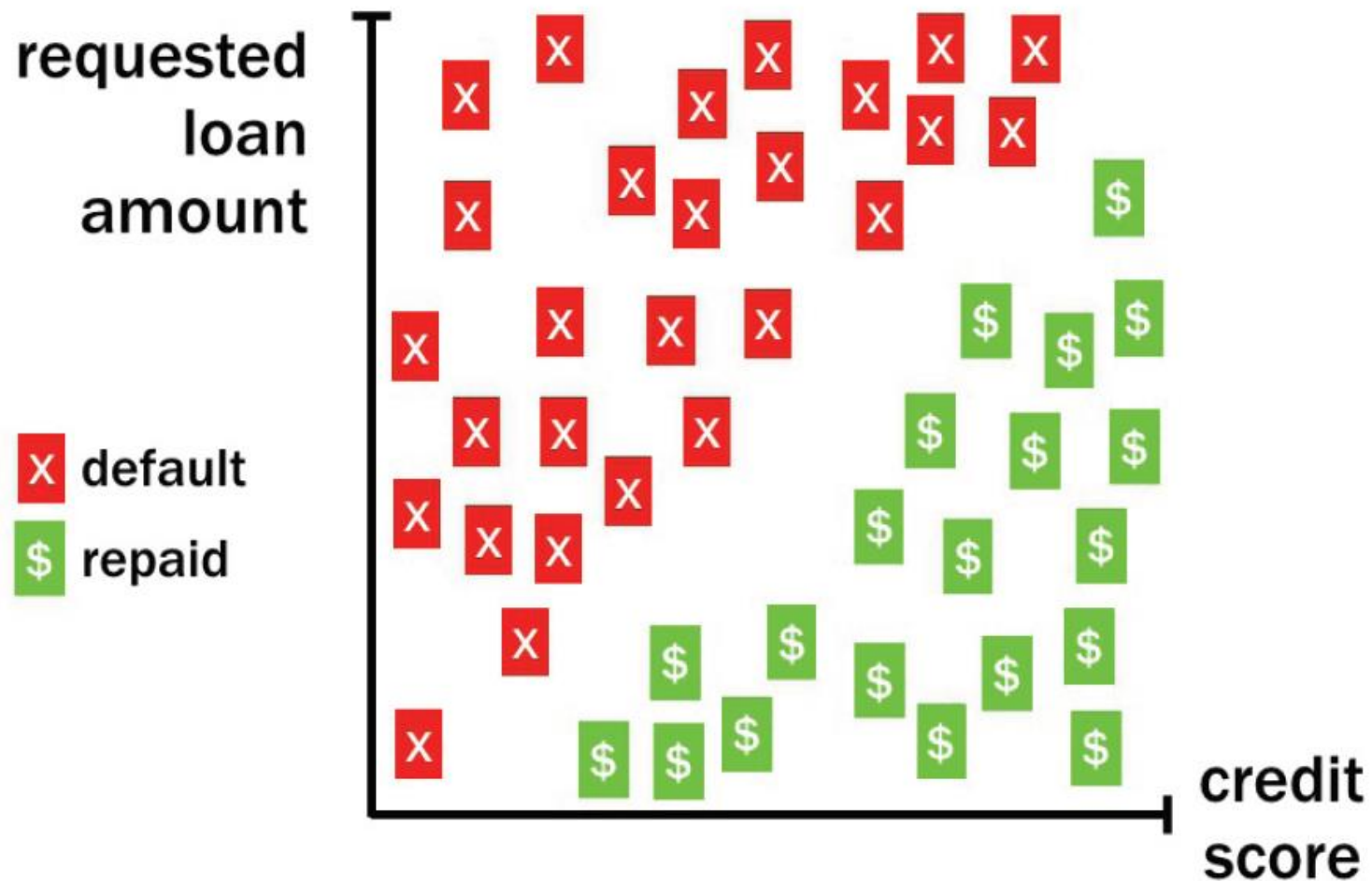
considering:
amount of loan
purpose
credit history
income

...

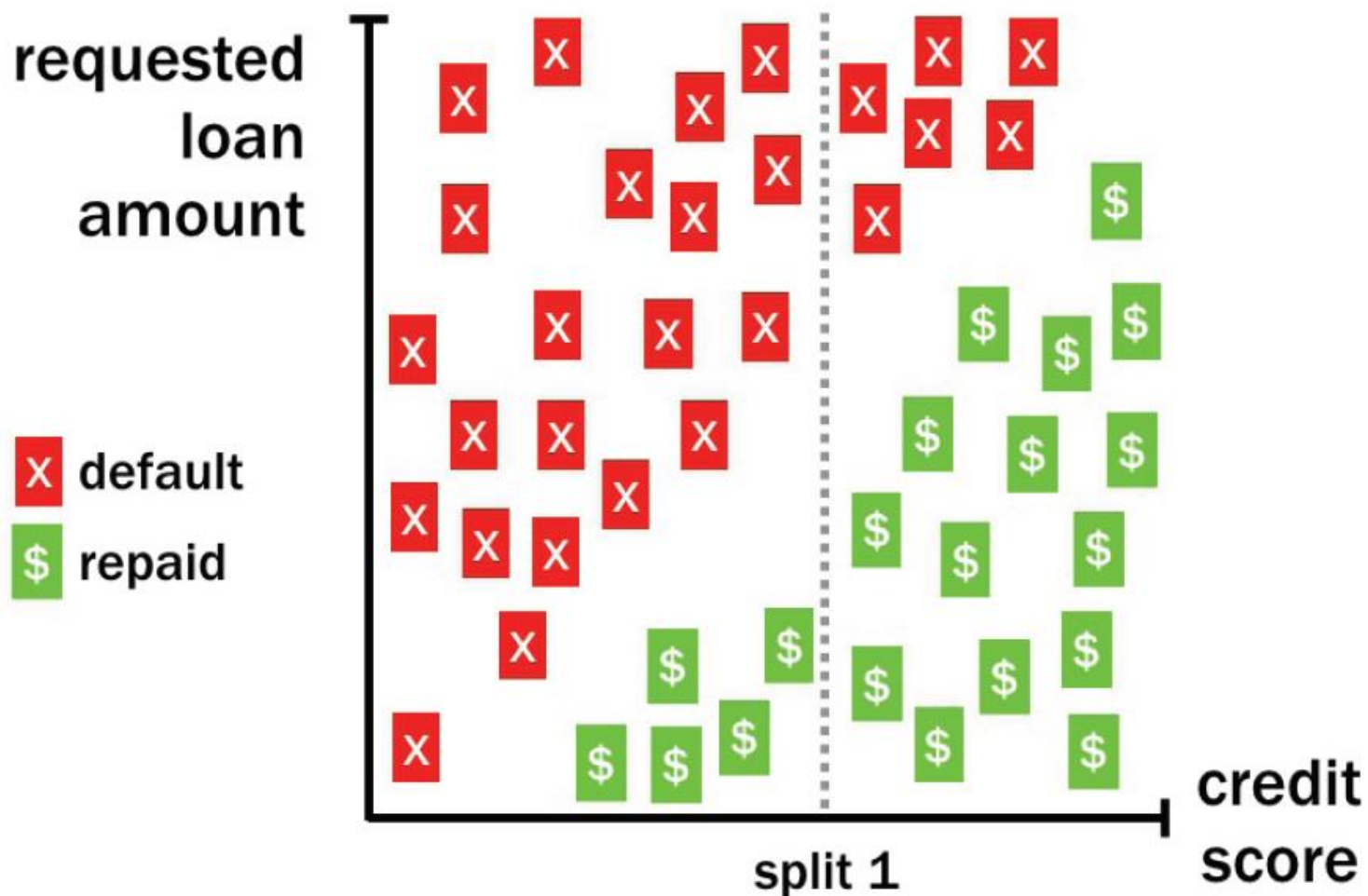
Decide:
approve or **decline**

decision tree model is useful when deciding process is better to be **interpretable** and hence **transparent**

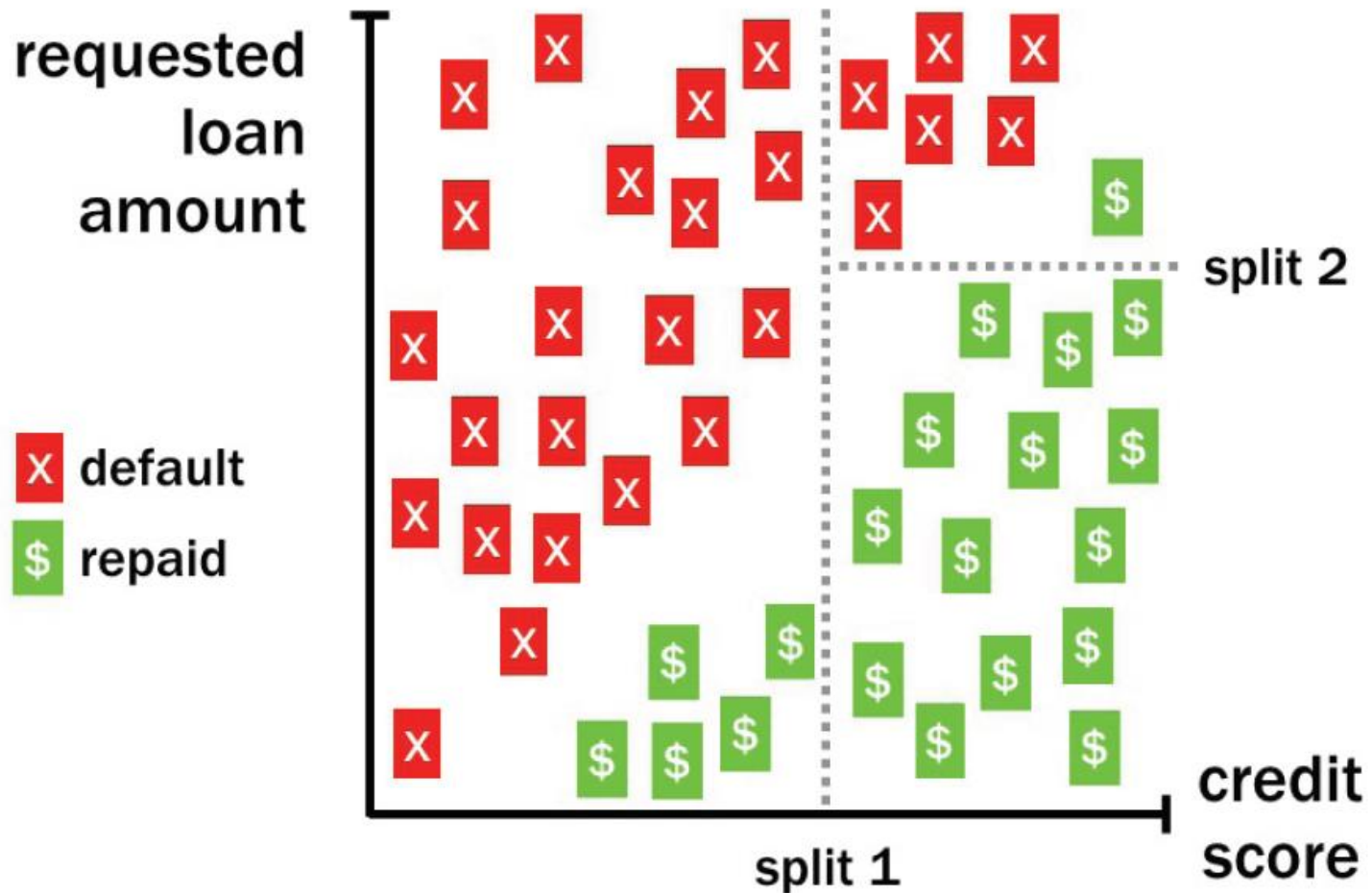
Divide and Conquer



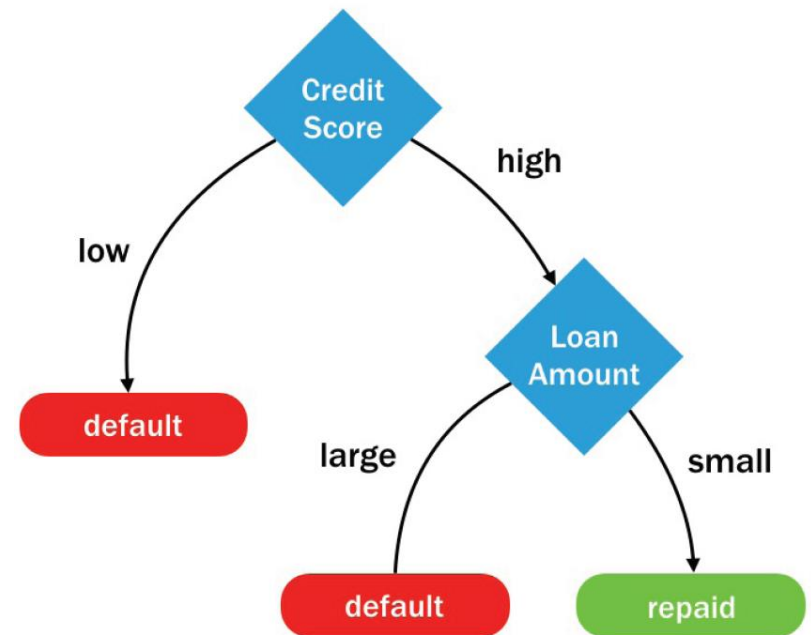
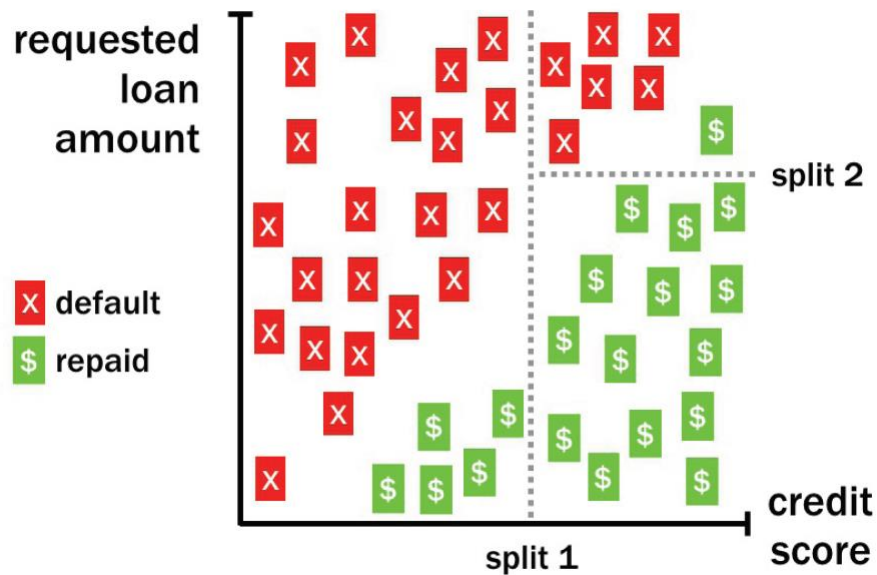
Divide and Conquer



Divide and Conquer



The resulting tree



Building a simple decision tree

```
load(url('https://github.com/hbchoi/SampleData/blob/master/dtree_data.RData?raw=true'))

str(loans)

## 'data.frame':    11312 obs. of  14 variables:
## $ loan_amount      : Factor w/ 3 levels "HIGH","LOW","MEDIUM": 2 2 2 3 2 3 3 2 1 3 ...
## $ emp_length       : Factor w/ 5 levels "< 2 years","10+ years",...: 2 1 4 3 1 1 3 2 2 1 ...
## $ home_ownership    : Factor w/ 4 levels "MORTGAGE","OTHER",...: 4 4 4 3 4 4 4 1 4 4 ...
## $ income           : Factor w/ 3 levels "HIGH","LOW","MEDIUM": 2 2 3 3 2 2 1 1 1 3 ...
## $ loan_purpose        : Factor w/ 14 levels "car","credit_card",...: 2 1 1 12 10 3 10 7 3 7 ...
## $ debt_to_income    : Factor w/ 3 levels "AVERAGE","HIGH",...: 2 3 3 3 1 1 3 1 1 3 ...
## $ credit_score      : Factor w/ 3 levels "AVERAGE","HIGH",...: 1 1 3 1 1 1 1 2 1 1 ...
## $ recent_inquiry    : Factor w/ 2 levels "NO","YES": 2 2 2 2 1 2 2 1 1 2 ...
## $ delinquent        : Factor w/ 3 levels "IN PAST 2 YEARS",...: 3 3 3 3 3 3 3 3 3 3 ...
## $ credit_accounts   : Factor w/ 3 levels "AVERAGE","FEW",...: 2 2 2 1 2 2 3 3 1 1 ...
## $ bad_public_record : Factor w/ 2 levels "NO","YES": 1 1 1 1 1 1 1 1 1 1 ...
## $ credit_utilization: Factor w/ 3 levels "HIGH","LOW","MEDIUM": 1 2 1 3 3 1 3 2 1 3 ...
## $ past_bankrupt     : Factor w/ 2 levels "NO","YES": 1 1 1 1 1 1 1 1 1 1 ...
## $ outcome           : Factor w/ 2 levels "default","rapid": 2 1 2 1 1 1 1 2 1 1 ...
```

The **loans** dataset contains 11,312 randomly-selected people who were applied for and later received loans from Lending Club, a US-based peer-to-peer lending company.

Building a simple decision tree

```
library(rpart)
```

```
loan_model <- rpart(outcome ~ loan_amount + credit_score, data = loans, method = "class",
control = rpart.control(cp = 0))
```

```
# Examine the loan_model object
```

```
loan_model
```

```
## n= 11312
```

```
##
```

```
## node), split, n, loss, yval, (yprob)
```

```
##      * denotes terminal node
```

```
##
```

```
## 1) root 11312 5654 rapid (0.4998232 0.5001768)
```

```
## 2) credit_score=AVERAGE,LOW 9490 4437 default (0.5324552 0.4675448)
```

```
## 4) credit_score=LOW 1667 631 default (0.6214757 0.3785243) *
```

```
## 5) credit_score=AVERAGE 7823 3806 default (0.5134859 0.4865141)
```

```
## 10) loan_amount=HIGH 2472 1079 default (0.5635113 0.4364887) *
```

```
## 11) loan_amount=LOW,MEDIUM 5351 2624 rapid (0.4903756 0.5096244)
```

```
## 22) loan_amount=LOW 1810 874 default (0.5171271 0.4828729) *
```

```
## 23) loan_amount=MEDIUM 3541 1688 rapid (0.4767015 0.5232985) *
```

```
## 3) credit_score=HIGH 1822 601 rapid (0.3298573 0.6701427) *
```

We will build a decision tree to try to learn patterns in the outcome of these loans (either **repaid** or **default**) based on the requested loan amount and credit score at the time of application.

Visualizing the Model

```
# Load the rpart.plot package
```

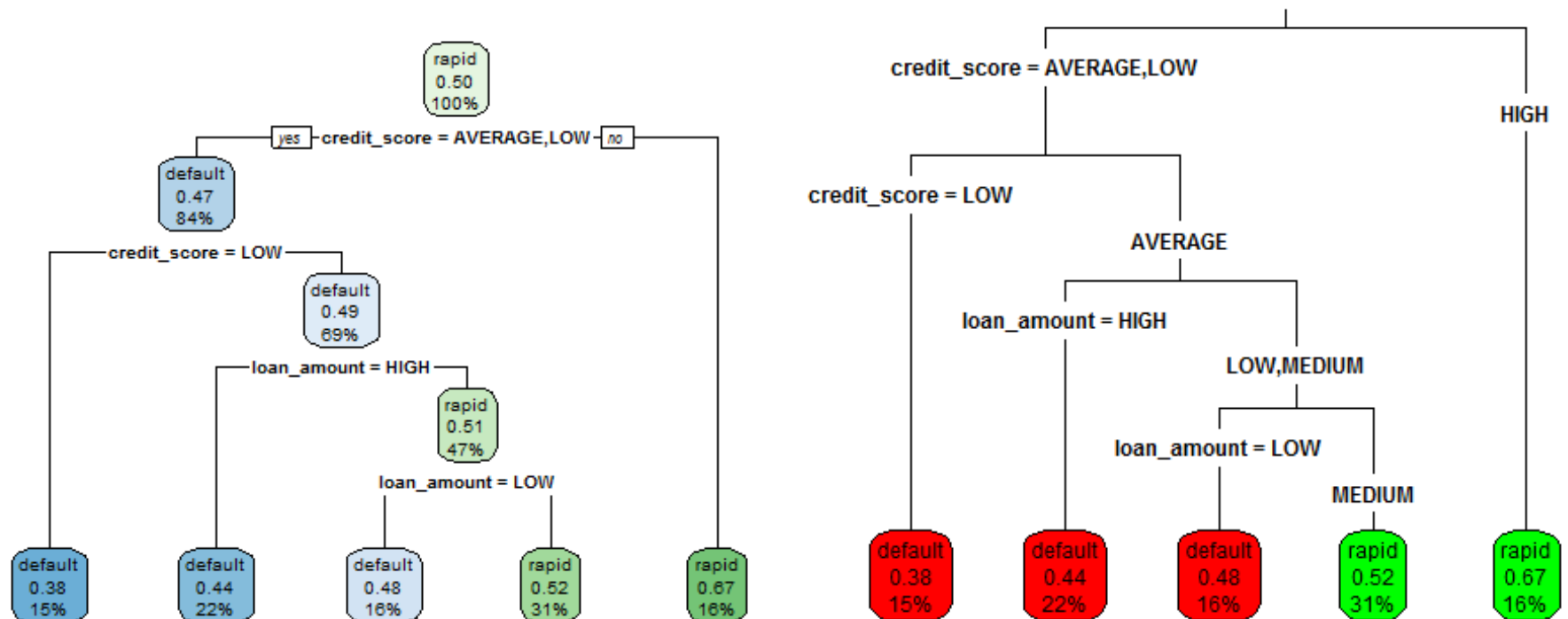
```
library(rpart.plot)
```

```
# Plot the loan_model with default settings
```

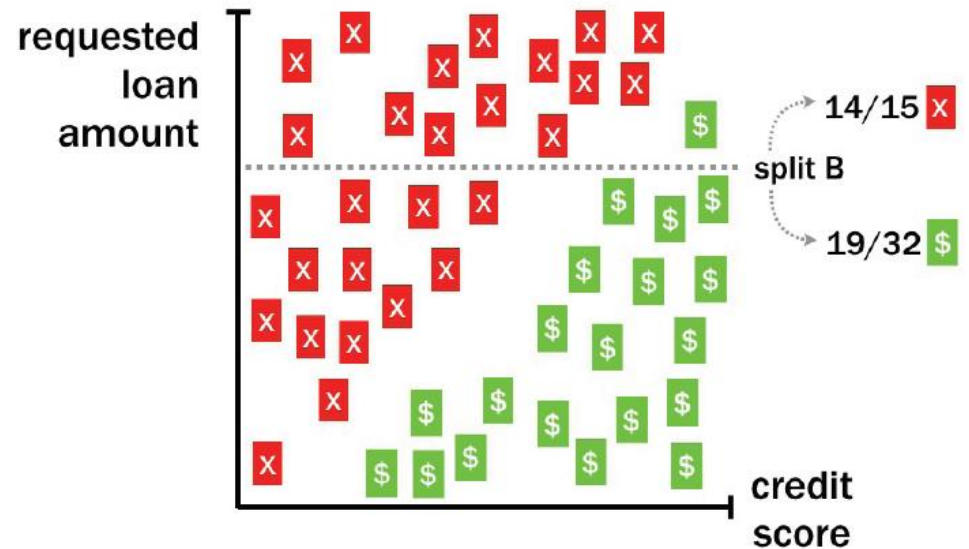
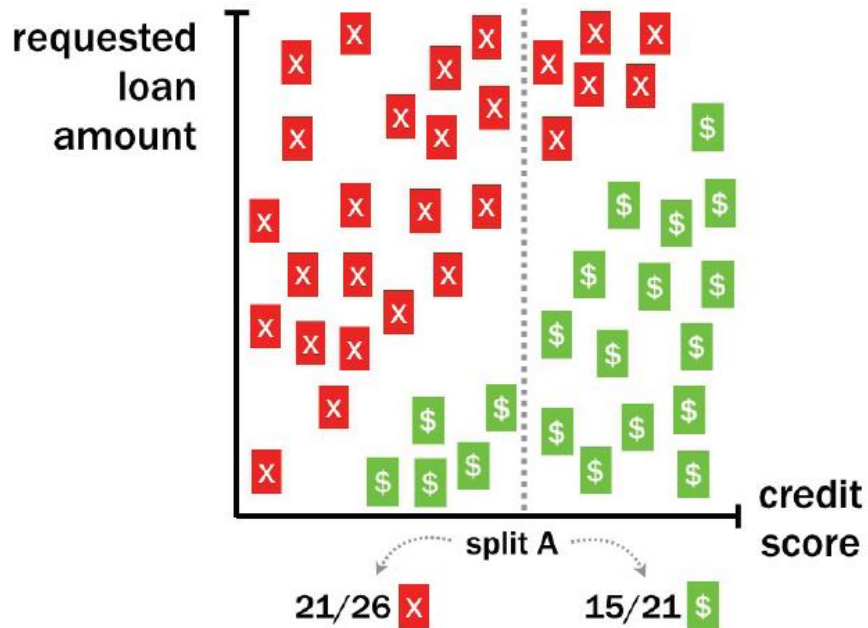
```
rpart.plot(loan_model)
```

```
# Plot the loan_model with customized settings
```

```
rpart.plot(loan_model, type = 3, box.palette = c("red", "green"), fallen.leaves = TRUE)
```



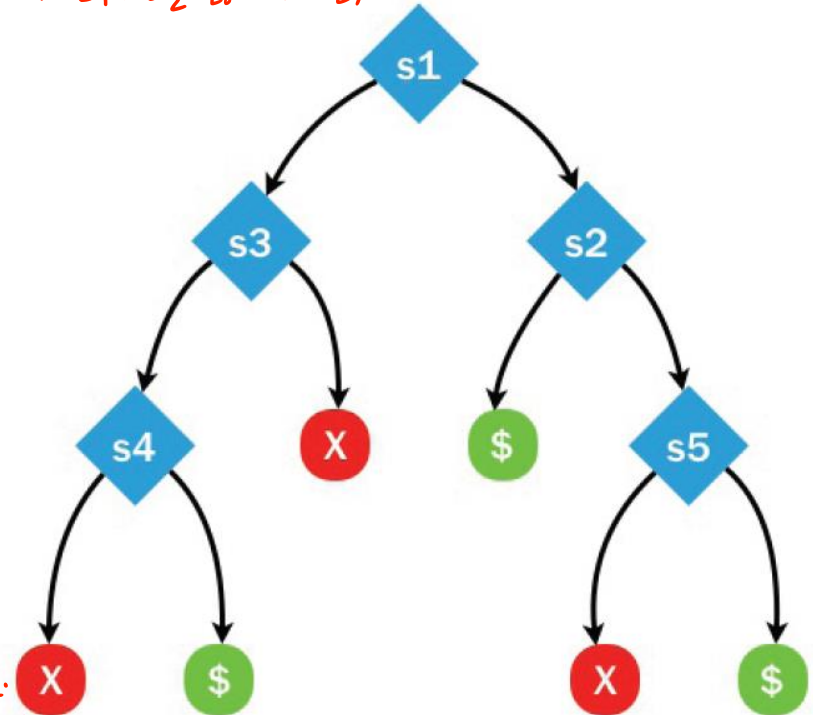
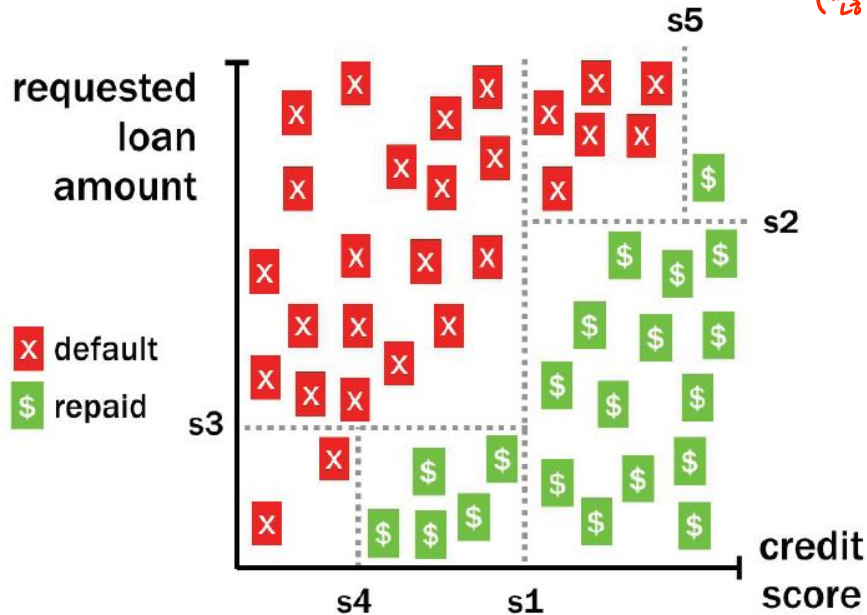
Choosing where to split



어떻게 split을 하는게 혼잡도(entropy)를 더 떨어뜨릴 수 있을까 생각하면서 혼잡도를 떨어뜨리는 방향으로 나눠야한다.

The problem of overfitting

decision tree는 상당히 overfitting 되기가 쉽다
(많은 변수를 써서 split을 많이 하면)

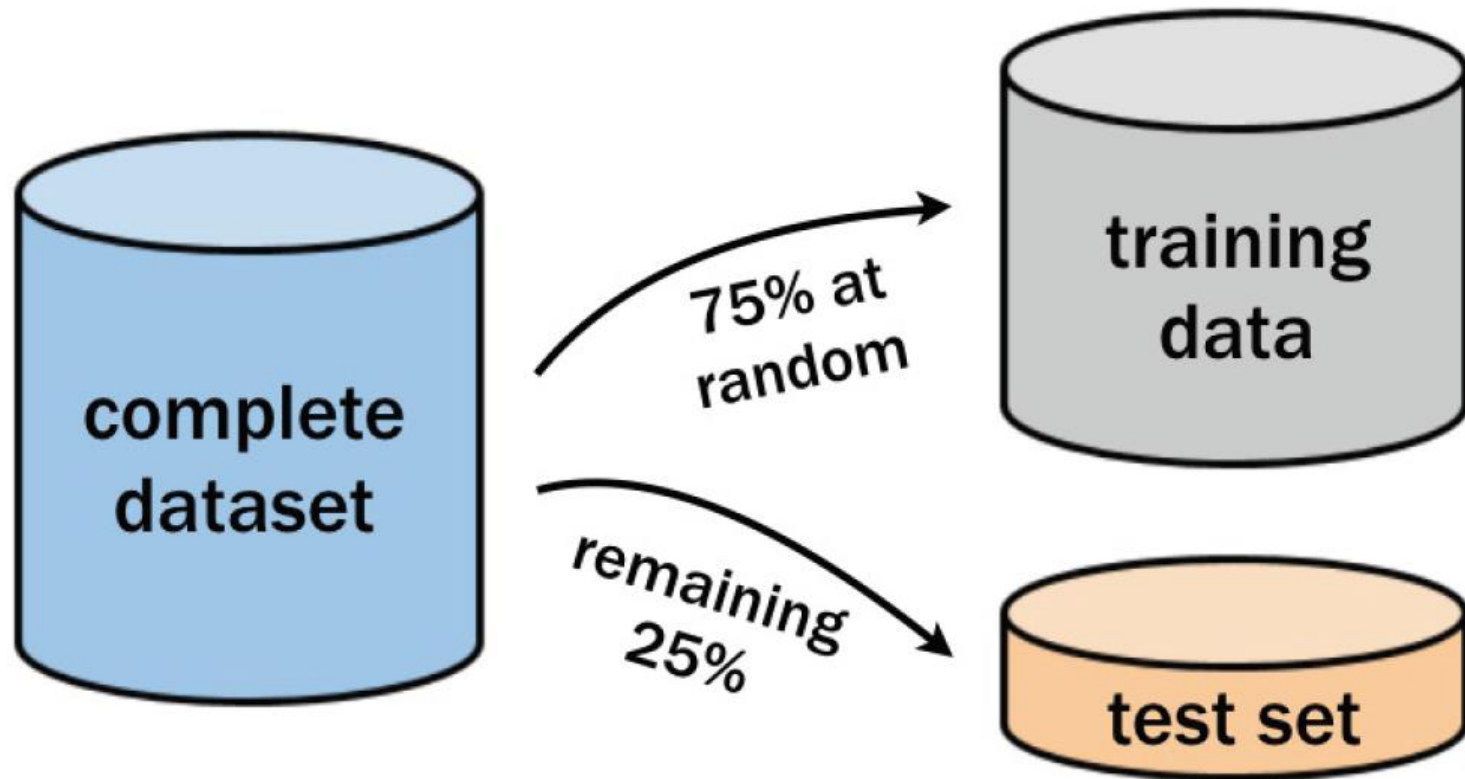


실제로 써먹어야 하는 상황에서 써먹지 못하는 그러한 상황들.
sample 수가 너무 적기 때문에 하나만 보고 선택을 판단을 내릴지도 몰라

Some over-grown decision trees capture errors that is too specific to the training dataset, which cannot be observed in general cases

Predicting very well on training data does not always guarantee that work well in the real world problem.

Evaluating model performance



performance on train data \gg perf. on test data \Rightarrow **high** likely to be over-fitted
performance on train data \approx perf. on test data \Rightarrow **less** likely to be over-fitted
We should limit the **complexity** somehow to avoid **overfitting problem**

```

loan_model <- rpart(outcome ~ ., data = loans_train, method = "class", control = rpart.control(cp = 0))

# Make predictions on the training dataset
loans_train$pred <- predict(loan_model, loans_train, type = 'class')

# Examine the confusion matrix
table(loans_train$outcome, loans_train$pred)

##
##           default rapid
## default      2932  1269
## rapid        1084  3199

# Compute the accuracy on the training dataset
mean(loans_train$outcome == loans_train$pred)

## [1] 0.7226544

# Make predictions on the test dataset
loans_test$pred <- predict(loan_model, loans_test, type = 'class')

# Examine the confusion matrix
table(loans_test$outcome, loans_test$pred)

##
##           default rapid
## default         821   632
## rapid           546   829

# Compute the accuracy on the test dataset
mean(loans_test$outcome == loans_test$pred)

## [1] 0.5834512

loans_train <- loans_train[-15]
loans_test <- loans_test[-15]

```

Lending Club has additional information about the applicants, such as home ownership status, length of employment, loan purpose, and past bankruptcies, that may be useful for making more accurate predictions.

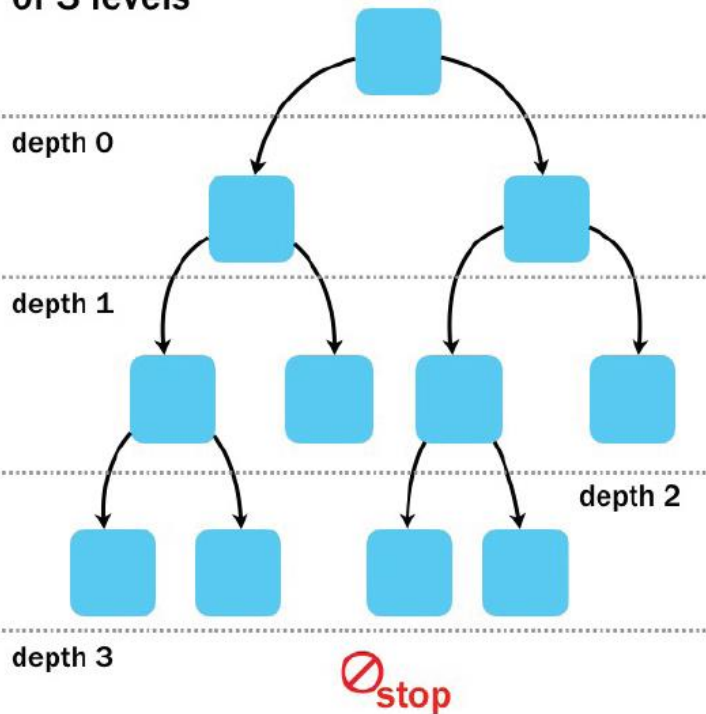
Using all of the available applicant data, we build a more sophisticated lending model using the training dataset and test the model on training and test datasets.

Limiting Tree Model's Complexity

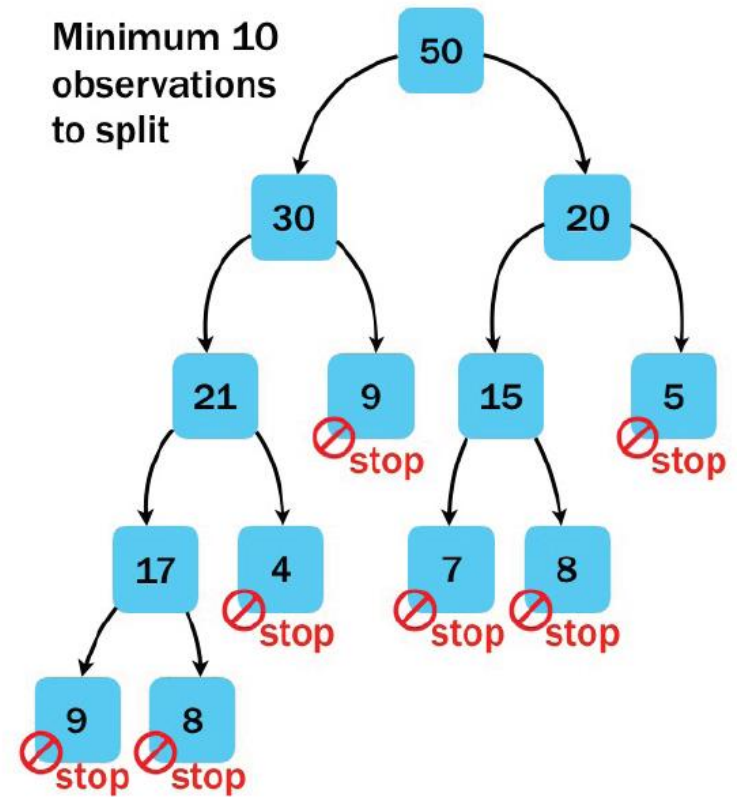
- Two Approaches
 - Pre-pruning
 - Restrict complexity before growing the model
 - Post-Pruning
 - Cut out two complex branches after having tree model

Pre-pruning

Maximum depth
of 3 levels

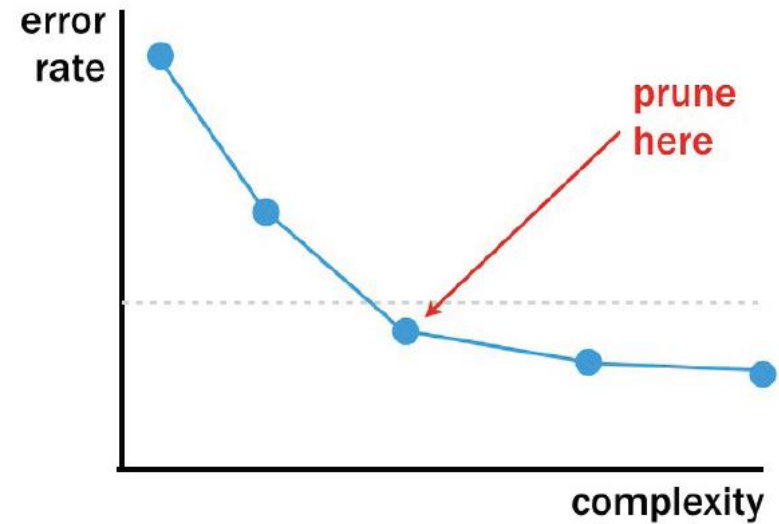
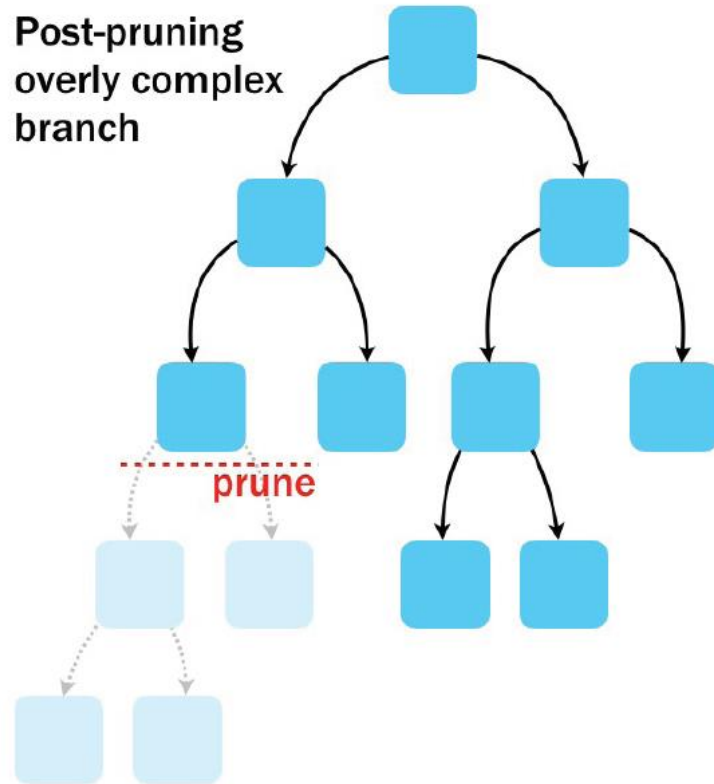


Minimum 10
observations
to split



Post-pruning

entropy가 어느 $\frac{1}{n}$ 만큼 안 떨어지면 그만하겠다!



Pre-pruning

Grow a tree with maxdepth of 6

```
loan_model <- rpart(outcome ~ ., data = loans_train, method = "class",  
control = rpart.control(cp = 0, maxdepth = 6))
```

Compute the accuracy of the simpler tree

```
loans_test$pred <- predict(loan_model, loans_test, type = 'class')  
mean(loans_test$outcome == loans_test$pred)
```

```
## [1] 0.5919378
```

Grow a tree with minsplit of 500

```
loan_model2 <- rpart(outcome ~ ., data = loans_train, method =  
"class", control = rpart.control(cp = 0, minsplit = 500))
```

Compute the accuracy of the simpler tree

```
loans_test$pred2 <- predict(loan_model2, loans_test, type = 'class')  
mean(loans_test$outcome == loans_test$pred2)
```

```
## [1] 0.5922914
```

Post-pruning

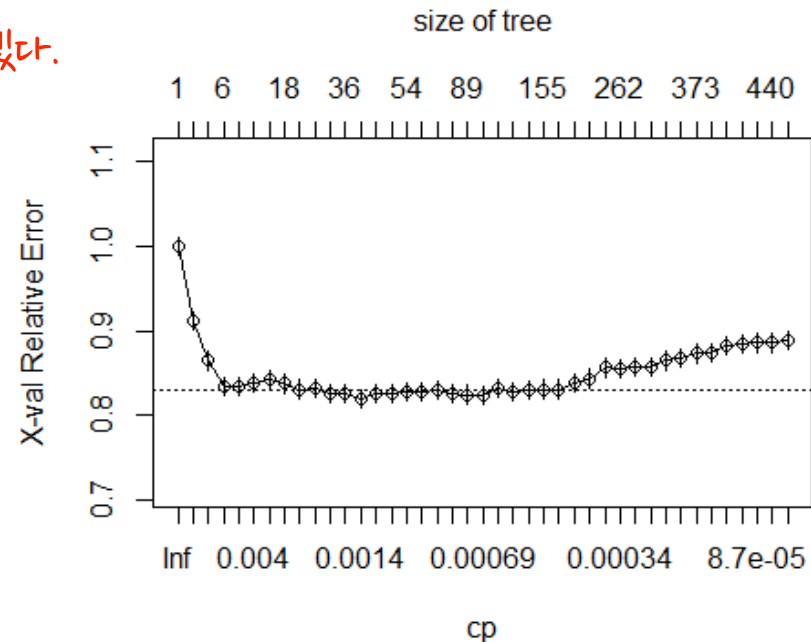
Grow an overly complex tree

```
loan_model <- rpart(outcome ~ ., data = loans_train, method = "class", control =  
rpart.control(cp = 0))
```

조건없이 tree를 계속 키워 나가겠다.

Examine the complexity plot

```
plotcp(loan_model)
```



Prune the tree

```
loan_model_pruned <- prune(loan_model, cp = 0.0014)
```

Compute the accuracy of the pruned tree

```
loans_test$pred <- predict(loan_model_pruned, loans_test, type = 'class')  
mean(loans_test$outcome == loans_test$pred)
```

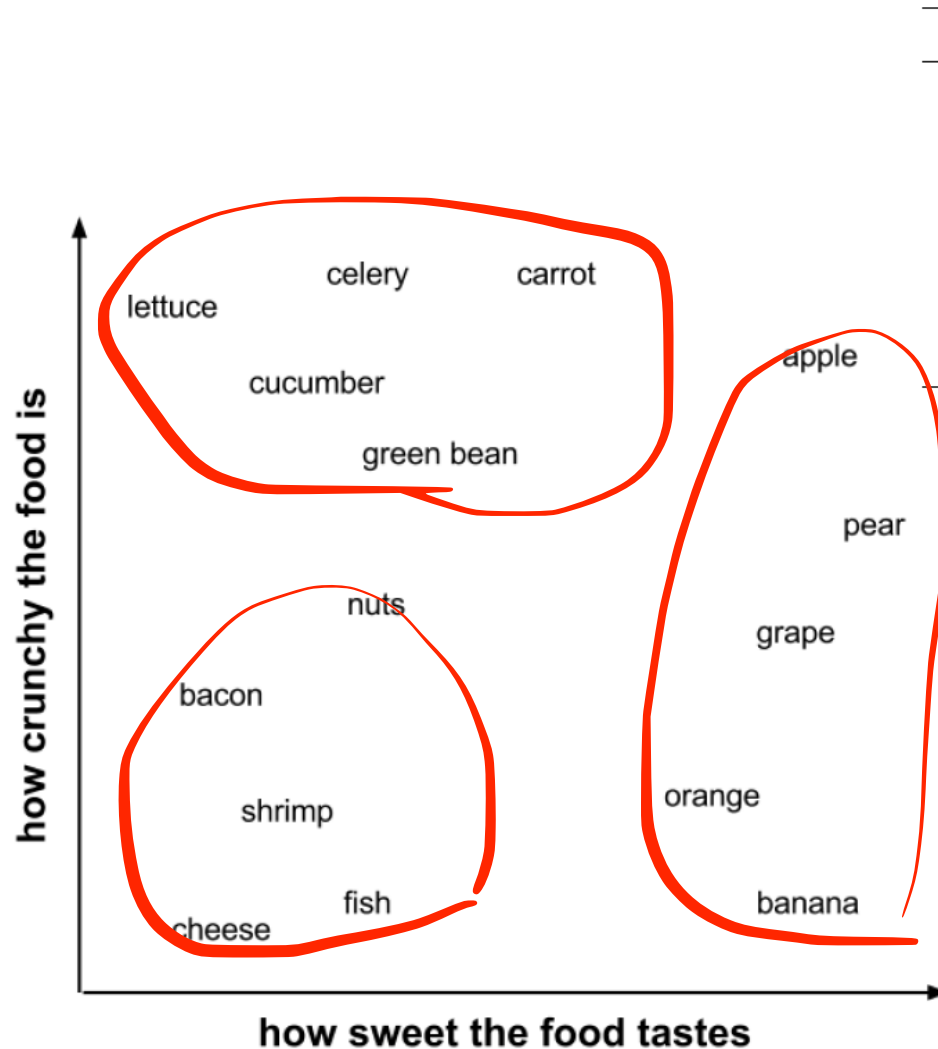
```
## [1] 0.6007779
```

Ref.

- rpart manual: <https://cran.r-project.org/web/packages/rpart/rpart.pdf>
- An Introduction to Recursive Partitioning Using the RPART Routines:
<https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf>
- Plotting rpart trees with the rpart.plot package:
<http://www.milbo.org/rpart-plot/prp.pdf>

k Nearest Neighbors

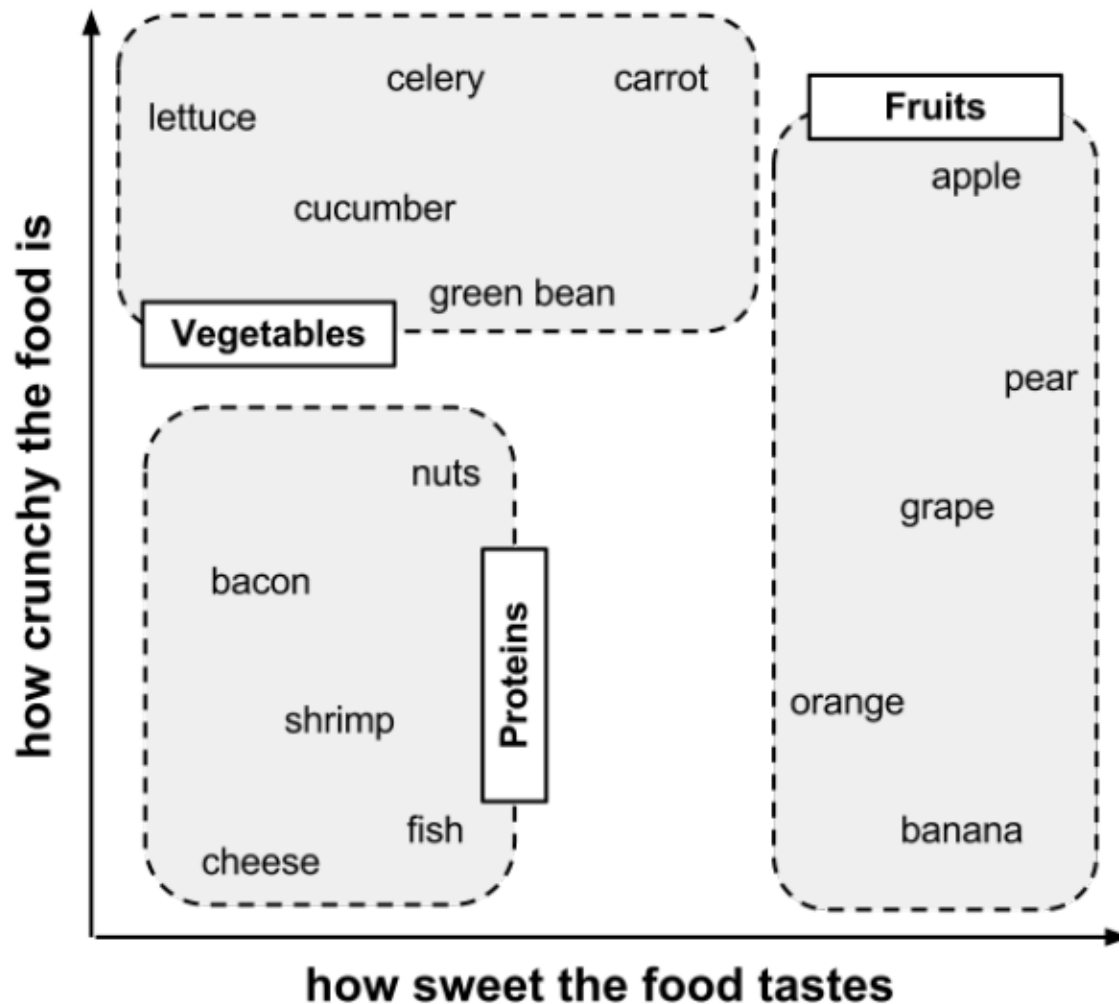
k-Nearest Neighbors (kNN)



ingredient	sweetness	crunchiness	food type
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit
carrot	7	10	vegetable
celery	3	10	vegetable
cheese	1	1	protein

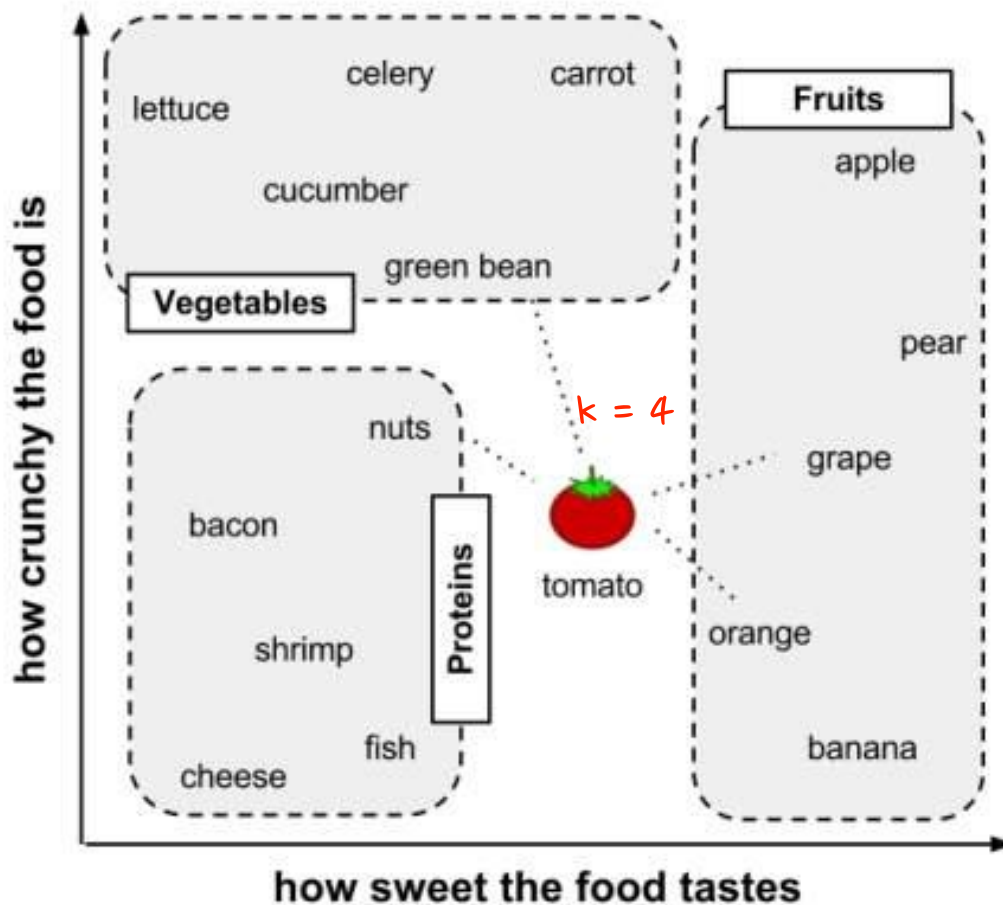
Similar Foods are located near to each other geographically

k-Nearest Neighbors (kNN)



Similar Foods are located near to each other geographically

k-Nearest Neighbors (kNN)



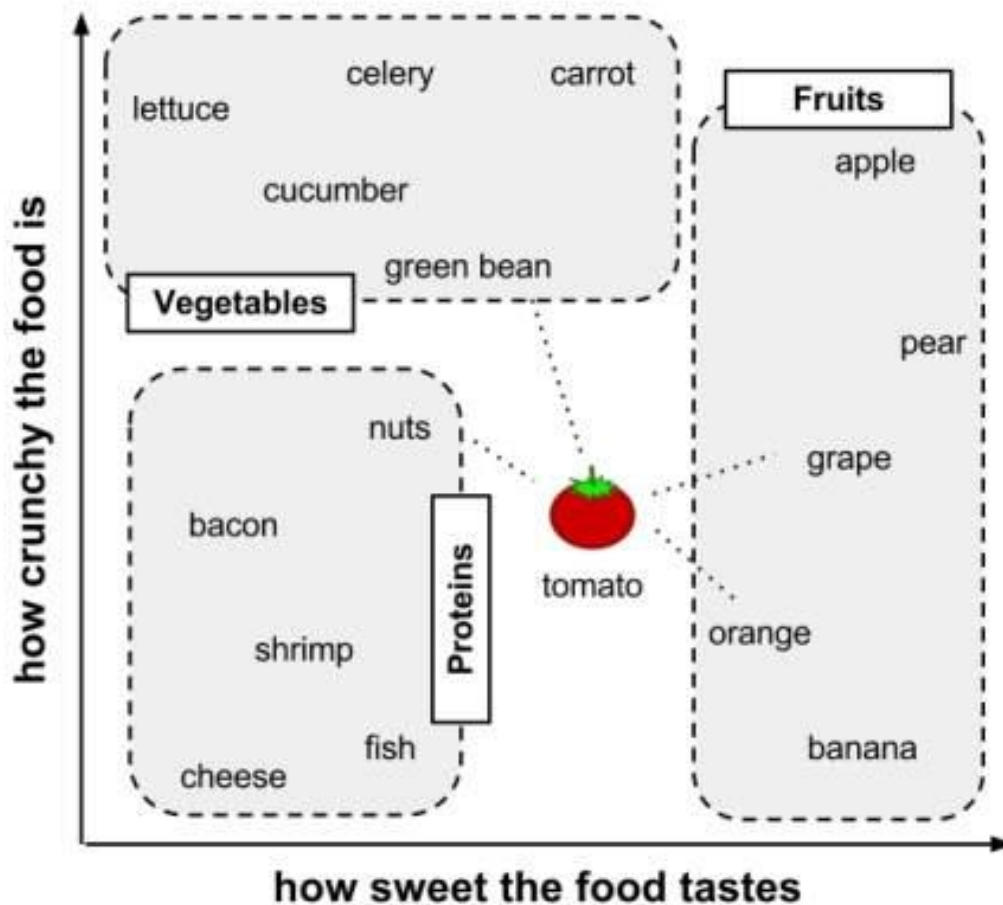
Then what is **tomato**?

Tomato is close to green **bean**, **nuts**, **orange**, and **grape**

1 Veg, 1 Protein, 2 Fruits

We may consider **tomato** belongs to **Fruits** class

k-Nearest Neighbors (kNN)



In **kNN** method, we decide according to most similar examples = **nearest neighbors**

Majority class would be our answer

The portion could be **estimated probability**

$$p(\text{fruit}) = 2/4$$

How many neighbors do we need to consider?

k -> # of neighbor to be consider for classification (or regression)

Measuring similarity with distance

distance between object p and q

Euclidean distance

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_d - q_d)^2}$$

p_1, p_2, \dots, p_d and q_1, q_2, \dots, q_d are feature sets of p and q

Other distance measures:

Manhattan distance, Cosign Similarity, Minkowski, ...

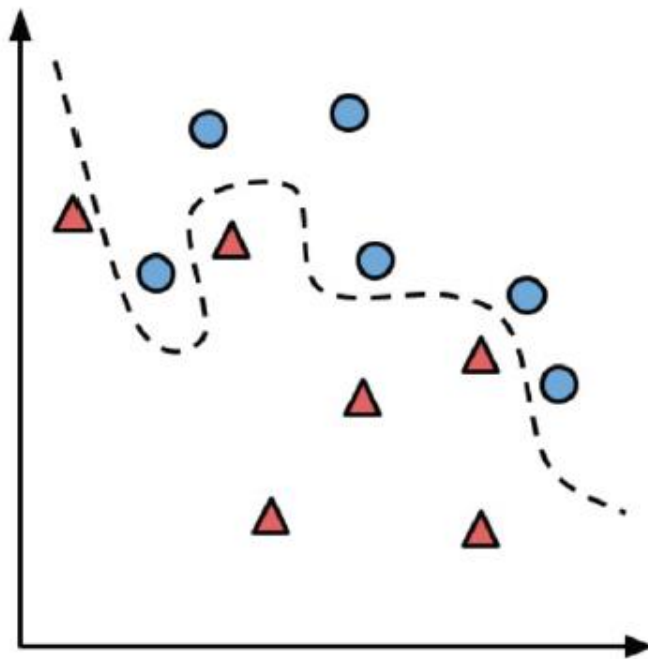
Applying nearest neighbors in R

```
library(class)
pred <- knn(training_data, testing_data, training_labels)
```

known data unknown data

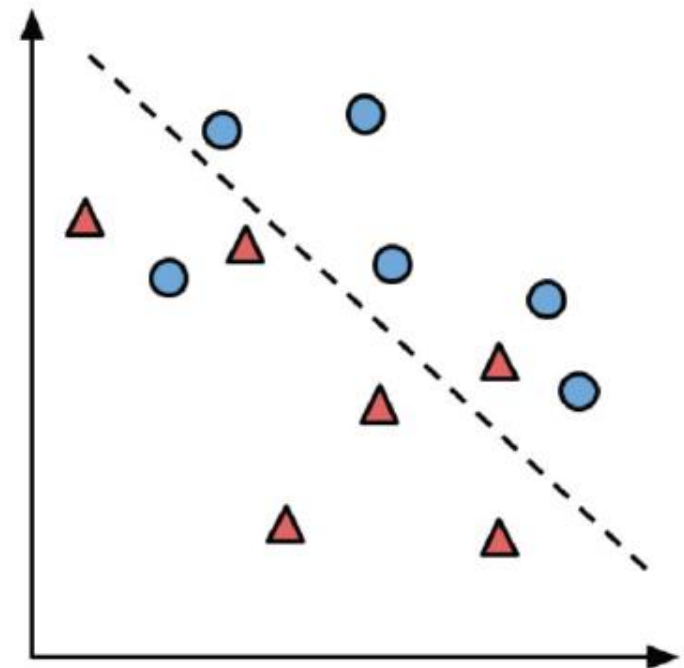
Choosing k

Bigger 'k' is not always better



Smaller k

overfitting이 발생 할 수도..



Larger k

예측을 잘 못해

Choosing k

There is no **right k** for all the problems

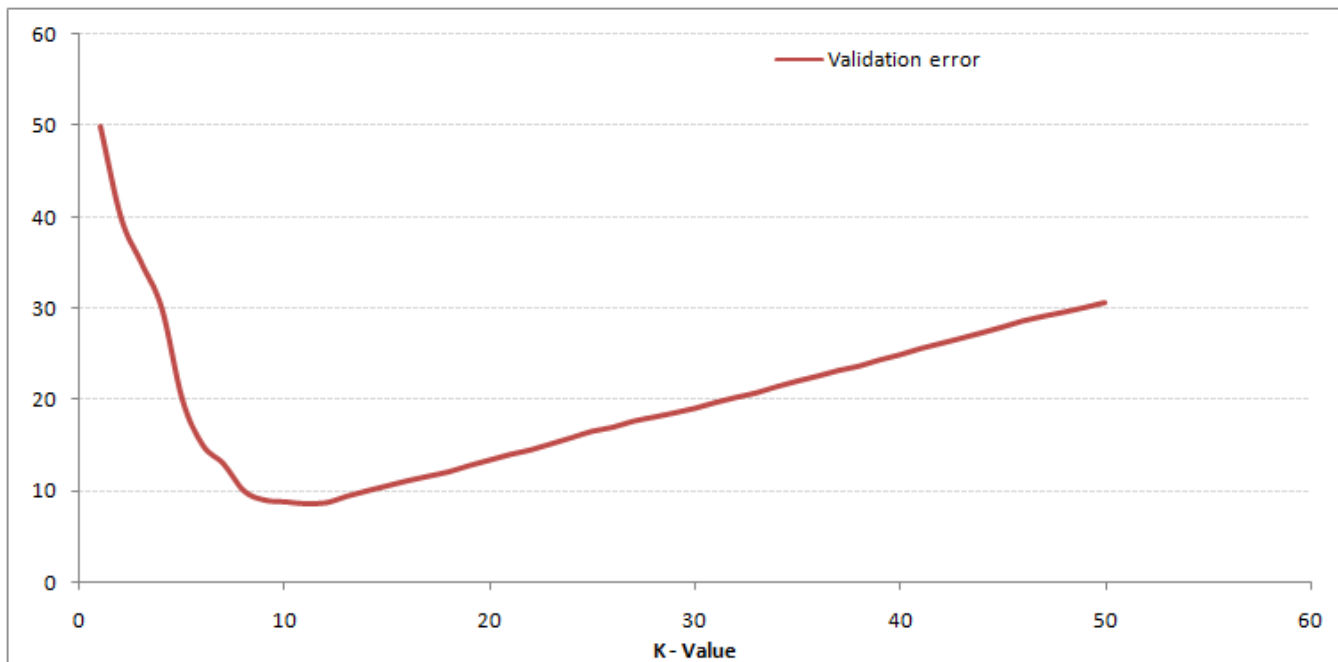
You may try different k to find the best

Suggesting square root of # of training examples, empirically

When a certain class is very **rare**, **larger k** would be better to see sufficient number of example (about 10?) in each class

실험을 많이 해보니까 루트값이 잘 나오더라

imbalanced data일 때는 k 값을 크게 잡아서



kNN – example

- Diagnosing Breast Cancer with KNN
- "Breast Cancer Wisconsin Diagnostic" dataset from the *UCI Machine Learning Repository*

Learning Repository

- 30 numeric measurements comprise the mean, standard error, and worst (that is, largest) value for 10 different characteristics of the digitized cell nuclei.

Radius

Texture

Perimeter

Area

Smoothness

Compactness

Concavity

Concave points

Symmetry

Fractal dimension

Loading Data

```
wbcd <- read.csv("https://github.com/hbchoi/SampleData/raw/master/wisc_bc_data.csv",  
stringsAsFactors = F)
```

```
str(wbcd)
```

```
## 'data.frame':    569 obs. of  32 variables:  
##  $ id              : int  87139402 8910251 905520 868871 9012568 906539 925291  
## 87880 862989 89827 ...  
##  $ diagnosis       : chr   "B" "B" "B" "B" ...  
##  $ radius_mean     : num   12.3 10.6 11 11.3 15.2 ...  
##  $ texture_mean    : num   12.4 18.9 16.8 13.4 13.2 ...  
##  $ perimeter_mean  : num   78.8 69.3 70.9 73 97.7 ...  
##  $ area_mean       : num   464 346 373 385 712 ...  
## ...  
##  $ points_worst    : num   0.0939 0.0793 0.0743 0.0861 0.0818 ...  
##  $ symmetry_worst  : num   0.283 0.294 0.3 0.21 0.249 ...  
##  $ dimension_worst : num   0.0677 0.0759 0.0788 0.0678 0.0677 ...
```

Data Preparation

Var. ID is not helpful to determine the type of tumor, hence remove it

```
# removing ID variable
```

```
wbcd <- wbcd[,-1]
```

```
table(wbcd$diagnosis)
```

```
##
```

```
##    B    M
```

```
## 357 212
```

```
# changing value for clear interpretation
```

```
wbcd$diagnosis <- ifelse(wbcd$diagnosis == 'B', 'Benign', 'Malignant')
```

Normalization

각 변수마다 값의 범위가 다르기 때문에 정규화를 해주어야 한다.

Var. in larger scale contributes more to the distance than Var. in smaller scale
e.g. area_mean has bigger effect on Euclidean distance than smoothness_mean

Having all variables in same scale is important preprocessing for **kNN**

- *normalization*

거리를 계산할 때도 정규화를 하지 않으면 차이가 크기 때문에

```
summary(wbcd[c("radius_mean", "area_mean", "smoothness_mean")])
```

##	radius_mean	area_mean	smoothness_mean
##	Min. : 6.981	Min. : 143.5	Min. : 0.05263
##	1st Qu.: 11.700	1st Qu.: 420.3	1st Qu.: 0.08637
##	Median : 13.370	Median : 551.1	Median : 0.09587
##	Mean : 14.127	Mean : 654.9	Mean : 0.09636
##	3rd Qu.: 15.780	3rd Qu.: 782.7	3rd Qu.: 0.10530
##	Max. : 28.110	Max. : 2501.0	Max. : 0.16340

min-max normalization

setting the scale of variables to 0~1

0 for minimum value, 1 for maximum value

```
minmax_norm <- function(x) {  
  (x-min(x))/(max(x)-min(x))  
}
```

```
wbcd_norm <- sapply(wbcd[, -1], minmax_norm)
```

```
summary(wbcd_norm[, c("radius_mean", "area_mean", "smoothness_mean")])
```

##	radius_mean	area_mean	smoothness_mean
##	Min. :0.0000	Min. :0.0000	Min. :0.0000
##	1st Qu.:0.2233	1st Qu.:0.1174	1st Qu.:0.3046
##	Median :0.3024	Median :0.1729	Median :0.3904
##	Mean :0.3382	Mean :0.2169	Mean :0.3948
##	3rd Qu.:0.4164	3rd Qu.:0.2711	3rd Qu.:0.4755
##	Max. :1.0000	Max. :1.0000	Max. :1.0000

kNN – split test and training data

```
# split data into train and test set
```

```
dim(wbcd_norm)
```

```
## [1] 569  30
```

```
wbcd_train <- wbcd_norm[1:469, ]
```

```
wbcd_test <- wbcd_norm[470:569, ]
```

```
wbcd_train_label <- wbcd[1:469, 1]
```

```
wbcd_test_label <- wbcd[470:569, 1]
```

```
# choosing proper k
```

```
sqrt(nrow(wbcd_train))
```

```
## [1] 21.65641
```

kNN – Making Prediction

```
library(class)
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_label,
k = 21)
```

```
wbcd_test_pred
```

```
##      [1] Benign      Malignant Benign      Benign      Malignant Benign      Malignant
##      [8] Benign      Malignant Benign      Malignant Benign      Malignant Malignant
##     [15] Benign      Benign      Malignant Benign      Malignant Benign      Malignant
##     [22] Malignant Malignant Malignant Benign      Benign      Benign      Benign
##     [29] Malignant Malignant Malignant Malignant Malignant Malignant Benign
##     [36] Benign      Benign      Benign      Benign      Malignant Malignant Benign
##     [43] Malignant Malignant Benign      Malignant Malignant Malignant Malignant
##     [50] Malignant Malignant Benign      Benign      Benign      Benign      Benign
##     [57] Benign      Benign      Malignant Benign      Benign      Benign      Benign
##     [64] Benign      Malignant Malignant Benign      Benign      Benign      Benign
##     [71] Benign      Malignant Benign      Benign      Malignant Malignant Benign
##     [78] Benign      Benign      Malignant Benign      Benign      Benign      Malignant
##     [85] Benign      Benign      Malignant Benign      Benign      Benign      Benign
##     [92] Malignant Benign      Benign      Benign      Benign      Benign      Malignant
##    [99] Benign      Malignant
## Levels: Benign Malignant
```

Performance

#accuracy

```
mean(wbcd_test_label == wbcd_test_pred)
```

```
## [1] 0.98
```

#confusion matrix

```
cmat <- table(wbcd_test_label, wbcd_test_pred)
```

```
cmat
```

```
##                wbcd_test_pred
## wbcd_test_label Benign Malignant
##      Benign      61          0
##      Malignant    2          37
```

이 환자들이 굉장히 위험해

#precision

```
cmat[2,2] / sum(cmat[,2])
```

```
## [1] 1
```

#recall

```
cmat[2,2] / sum(cmat[2,])
```

```
## [1] 0.9487179
```

We hope to avoid **false negative** rather than false positive.

How can?

kNN – Probabilistic Interpretation

```
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_label, k =
21, prob = TRUE)
```

```
head(wbcd_test_pred)
```

```
## [1] Benign    Benign    Benign    Benign    Malignant Benign
## Levels: Benign Malignant
```

```
head(attributes(wbcd_test_pred)$prob)
```

```
## [1] 1.000000 0.952381 0.952381 0.952381 1.000000 1.000000
      P(Benign) P(Benign) P(Benign) P(Benign) P(Malignant) P(Benign)
```

```
# converting all Prob to P(Malignant)
```

```
wbcd_test_pred_prob <- ifelse(wbcd_test_pred == 'Malignant',
                             attributes(wbcd_test_pred)$prob,
                             1-attributes(wbcd_test_pred)$prob)
```

```
head(wbcd_test_pred_prob)
```

```
## [1] 0.00000000 0.04761905 0.04761905 0.04761905 1.00000000 0.00000000
```


ROC curve and AUC

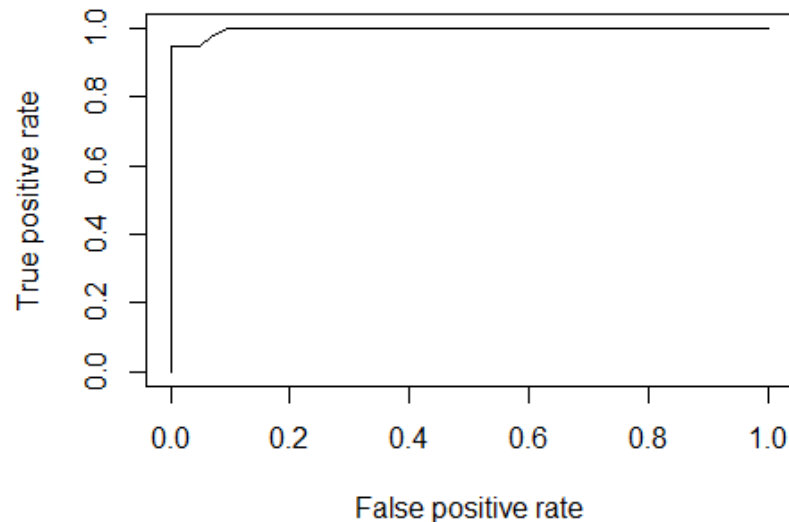
```
library(ROCR)

plot(performance(prediction(wbcd_test_pred_prob, wbcd_test_label == 'Malignant'),
  'tpr', 'fpr'))

calAUC <- function(predCol, targetCol){
  perf <- performance(prediction(predCol, targetCol), 'auc')
  as.numeric(perf@y.values)
}

# AUC for our kNN
calAUC(wbcd_test_pred_prob, wbcd_test_label == 'Malignant')

## [1] 0.9964271
```



Adjusting Threshold

```
# set lower threshold
threshold <- 0.1
wbcd_test_pred_new <- ifelse(wbcd_test_pred_prob > threshold,
                             'Malignant', 'Benign')
cmat <- table(wbcd_test_label, wbcd_test_pred_new)
cmat

##                wbcd_test_pred_new
## wbcd_test_label Benign Malignant
##      Benign      54          7
##      Malignant    0          39

#accuracy
mean(wbcd_test_label == wbcd_test_pred_new)

## [1] 0.93

#precision
cmat[2,2] / sum(cmat[,2])

## [1] 0.8478261

#recall
cmat[2,2] / sum(cmat[2,])

## [1] 1
```

단순히 precision을 높이는 것이 중요한게 아니라
문제에 맞게 recall과 precision을 조정할 줄 알아야한다.

Accuracy is lower,
but **zero false negative !**

One more Thing

Euclidean distance makes sense only when input variables are **numeric**

-> Non-numeric Variables need to be converted to numeric, i.e. ***dummy coding***

dummy coding one-hot encoding

```
sample_df <- data.frame(blood_type = c('A', 'B', 'A', 'O', 'AB'),
                        skin_color = c('black', 'white', 'yellow', 'red',
                                      'black'),
                        age = c(22, 35, 21, 26, 70))
```

```
library(caret)
```

```
predict(dummyVars(~ blood_type+skin_color, data = sample_df), sample_df)
```

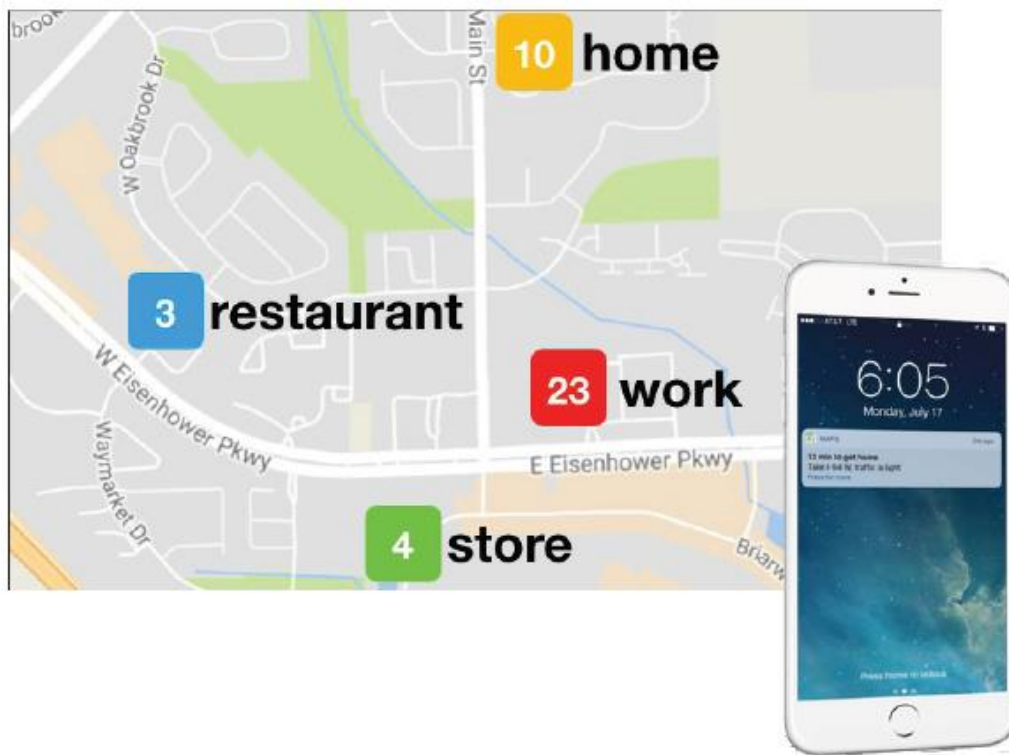
model이라는 함수 써두 땀

sample_df									
##	blood_type	skin_color	age	##	blood_type.A	blood_type.AB	blood_type.B	blood_type.O	skin_color.black
## 1	A	black	22	## 1	1	0	0	0	1
## 2	B	white	35	## 2	0	0	1	0	0
## 3	A	yellow	21	## 3	1	0	0	0	0
## 4	O	red	26	## 4	0	0	0	1	0
## 5	AB	black	70	## 5	0	1	0	0	1
				##	skin_color.red	skin_color.white	skin_color.yellow		
				## 1	0	0	0		
				## 2	0	1	0		
				## 3	0	0	1		
				## 4	1	0	0		
				## 5	0	0	0		

dummy encoding

Naïve Bayes

Estimating probability

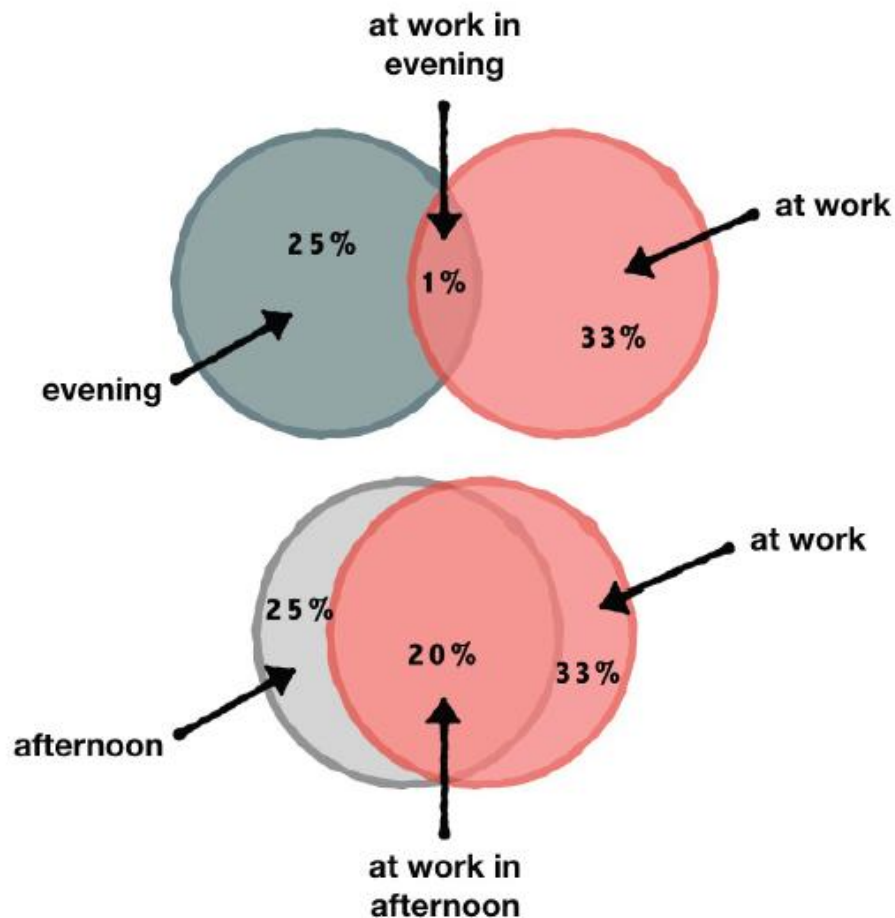


The **probability** of A is denoted $P(A)$

$$P(\text{work}) = 23 / 40 = 57.5\%$$

$$P(\text{store}) = 4 / 40 = 10.0\%$$

Joint probability and independent events



The **joint probability** of events A and B is denoted $P(A \text{ and } B)$

- $P(\text{work and evening}) = 1\%$
- $P(\text{work and afternoon}) = 20\%$

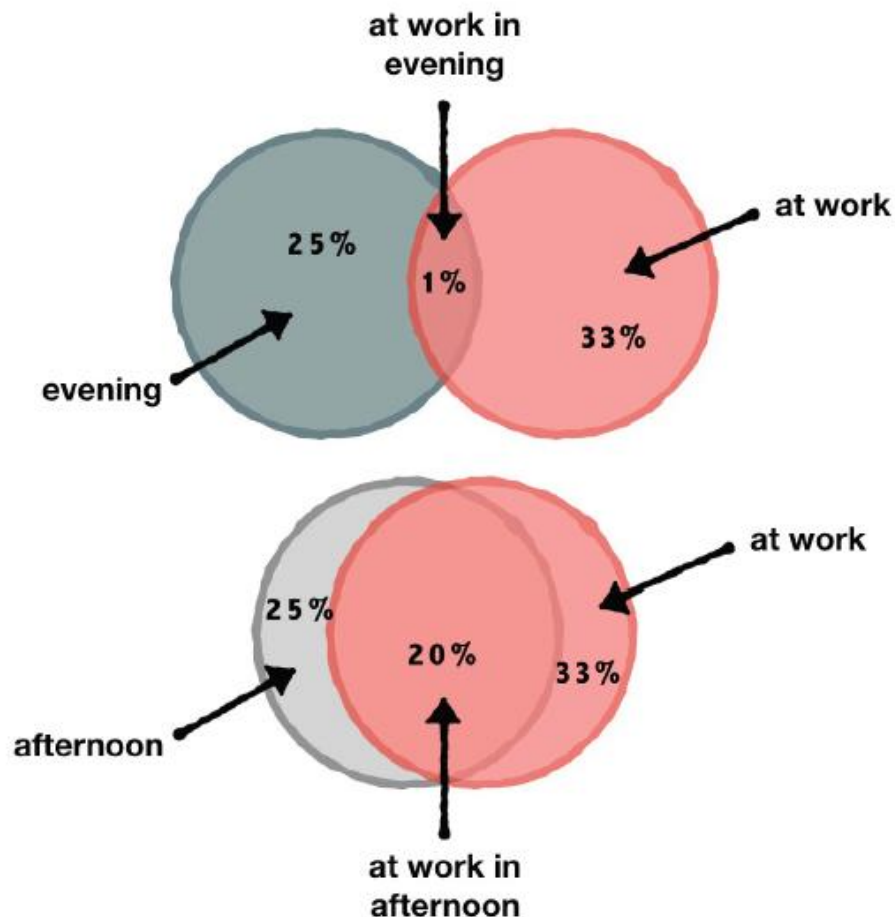
Event A and B are **independent**
 -> occurring A does not affect $P(B)$
 and vice versa

$P(\text{evening}) = 25\%$

$p(\text{weekend}) = 28.5\%$

then $P(\text{evening and weekend}) =$
 $P(\text{evening}) \times p(\text{weekend}) = 7.1\%$

Conditional probability and dependent events



The conditional probability of events A and B is denoted $P(A | B)$

$$P(A | B) = P(A \text{ and } B) / P(B)$$

$$P(\text{work} | \text{evening}) = 1 / 25 = 4\%$$

$$P(\text{work} | \text{afternoon}) = 20 / 25 = 80\%$$

Event A and B are **dependent**

$P(A)$ over event B

hence $P(A) \neq P(A | B)$

Naïve Bayes

we want to know:

$P(\text{work} \mid \text{afternoon})$, $P(\text{home} \mid \text{afternoon})$, ...

$$\begin{aligned}
 P(\text{work} \mid \text{afternoon}) &= \frac{P(\text{work and afternoon})}{P(\text{afternoon})} \\
 &= \frac{P(\text{work and afternoon})}{P(\text{afternoon})} \cdot \frac{P(\text{work})}{P(\text{work})} = \frac{P(\text{work and afternoon})}{P(\text{work})} \cdot \frac{P(\text{work})}{P(\text{afternoon})} \\
 &= P(\text{afternoon} \mid \text{work}) \cdot \frac{P(\text{work})}{P(\text{afternoon})}
 \end{aligned}$$

Prior

Probability

Likelihood

$$P(\text{outcome} = \text{True} \mid a = A) = \frac{P(\text{outcome} = \text{True}) \times P(a = A \mid \text{outcome} = \text{True})}{P(a = A)}$$

Posterior

Probability

Marginal Likelihood

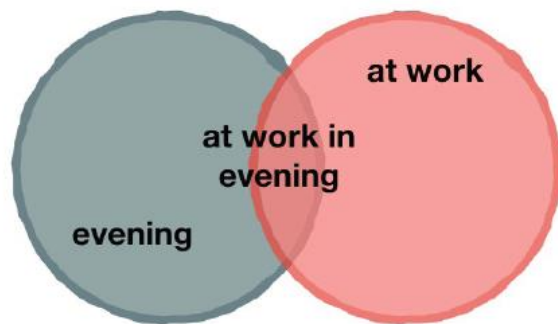
Naïve Bayes

$$P(\text{work} \mid \text{afternoon}) = P(\text{afternoon} \mid \text{work}) \cdot \frac{P(\text{work})}{P(\text{afternoon})}$$

frequency	morning	after noon	evening	total
work	14	14	2	30
home	5	5	25	35
church	5	8	2	15
downtown	2	6	12	20
total	26	33	41	100

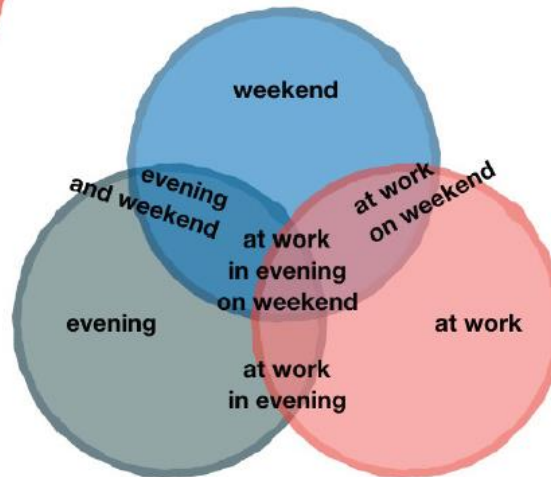
likelihood	morning	after noon	evening	total
work	14/30	14/30	2/30	30
home	5/35	5/35	25/35	35
church	5/15	8/15	2/15	15
downtown	2/20	6/20	12/20	20
total	26/100	33/100	41/100	100

The challenge of multiple predictors



$$P(\text{work} \mid \text{weekend and evening})$$

$$= P(\text{weekend and evening} \mid \text{work}) \cdot \frac{P(\text{work})}{P(\text{weekend and evening})}$$



$P(\text{work} \mid \text{weekend and evening}), P(\text{work} \mid \text{weekend and morning}),$
 $P(\text{work} \mid \text{weekend and afternoon}), P(\text{work} \mid \text{weekday and evening}), P(\text{work} \mid \text{weekday and morning}),$

...

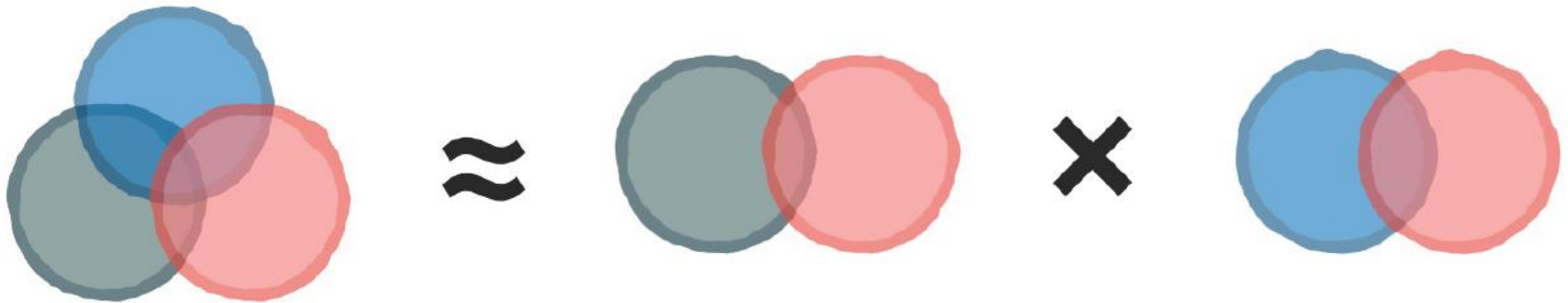
More input variables we have, more combination we need to consider...

Computationally expensive!!

Naivety Assumption

Naïve Bayes assumes that all input variables are conditionally independent

Which simplifies the problem



$$P(ev_1 \& \dots \& ev_N | y = T) \approx P(ev_1 | y = T) \times P(ev_2 | y = T) \times \dots \times P(ev_N | y = T)$$

$$P(ev_1 \& \dots \& ev_N | y = F) \approx P(ev_1 | y = F) \times P(ev_2 | y = F) \times \dots \times P(ev_N | y = F)$$

$$P(y = T | ev_1 \& \dots \& ev_N) \approx \frac{P(y = T) \times (P(ev_1 | y = T) \times \dots \times P(ev_N | y = T))}{P(ev_1 \& \dots \& ev_N)}$$

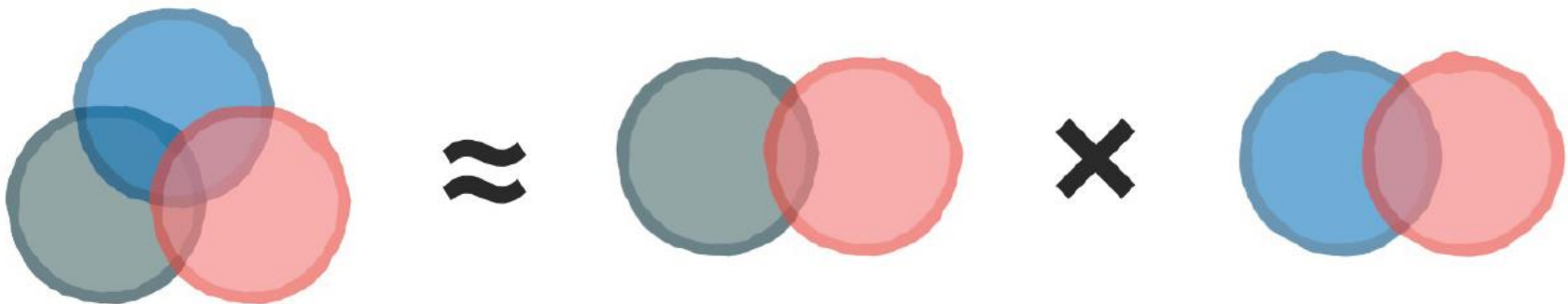
$$P(y = F | ev_1 \& \dots \& ev_N) \approx \frac{P(y = F) \times (P(ev_1 | y = F) \times \dots \times P(ev_N | y = F))}{P(ev_1 \& \dots \& ev_N)}$$

Naivety Assumption

The independence assumption is rarely true in practice

However it performs admirably on many real-world tasks

Naïve Bayes is good approximation of complex probability model



sample dataset for trial

```
load(url('https://github.com/hbchoi/SampleData/raw/master/nb_data.RData'))
head(locations)
```

```
##   month day  weekday daytype hour hourtype location
## 1     1   4 wednesday weekday   0    night     home
## 2     1   4 wednesday weekday   1    night     home
## 3     1   4 wednesday weekday   2    night     home
## 4     1   4 wednesday weekday   3    night     home
```

```
str(locations)
```

```
## 'data.frame':   2184 obs. of  7 variables:
## $ month      : int  1 1 1 1 1 1 1 1 1 1 ...
## $ day        : int  4 4 4 4 4 4 4 4 4 4 ...
## $ weekday    : Factor w/ 7 levels "friday","monday",...: 7 7 7 7 7 7 7 7 7 7 ...
## $ daytype    : Factor w/ 2 levels "weekday","weekend": 1 1 1 1 1 1 1 1 1 1 ...
## $ hour       : int  0 1 2 3 4 5 6 7 8 9 ...
## $ hourtype   : Factor w/ 4 levels "afternoon","evening",...: 4 4 4 4 4 4 3 3 3 3 ...
## $ location   : Factor w/ 7 levels "appointment",...: 3 3 3 3 3 3 3 3 4 ...
```

```
head(when9am)
```

```
##   daytype location
## 1 weekday  office
## 2 weekday  office
## 3 weekday  office
## 4 weekend   home
## 5 weekend   home
## 6 weekday  campus
```

simple Naïve Bayes model

```
library(naivebayes)

locmodel <- naive_bayes(location ~ daytype, data = where9am)
locmodel

## ===== Naive Bayes =====
## Call:
## naive_bayes.formula(formula = location ~ daytype, data = where9am)
##
## A priori probabilities:
##
## appointment      campus      home      office
## 0.01098901 0.10989011 0.45054945 0.42857143
##
## Tables:
##
## daytype  appointment      campus      home      office
## weekday  1.0000000 1.0000000 0.3658537 1.0000000
## weekend   0.0000000 0.0000000 0.6341463 0.0000000
```

make prediction

```
test_simple
```

```
##    daytype
```

```
## 1 weekday
```

```
## 2 weekend
```

```
predict(locmodel, newdata = test_simple)
```

```
## [1] office home
```

```
## Levels: appointment campus home office
```

```
predict(locmodel, newdata = test_simple, type = 'prob')
```

```
##      appointment      campus      home office
```

```
## [1,] 0.01538462 0.1538462 0.2307692    0.6
```

```
## [2,] 0.00000000 0.0000000 1.0000000    0.0
```

A more sophisticated location model

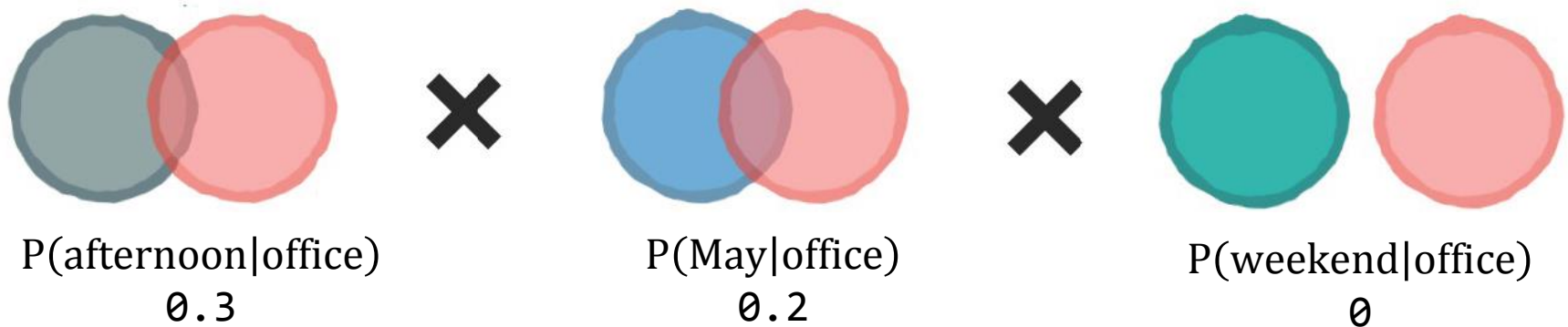
```
dim(locations_train)
## [1] 2088    7

dim(locations_test)
## [1] 96    7

locmodel <- naive_bayes(location ~ daytype + hourtype, data = locations_train)
pred <- predict(locmodel, locations_test)

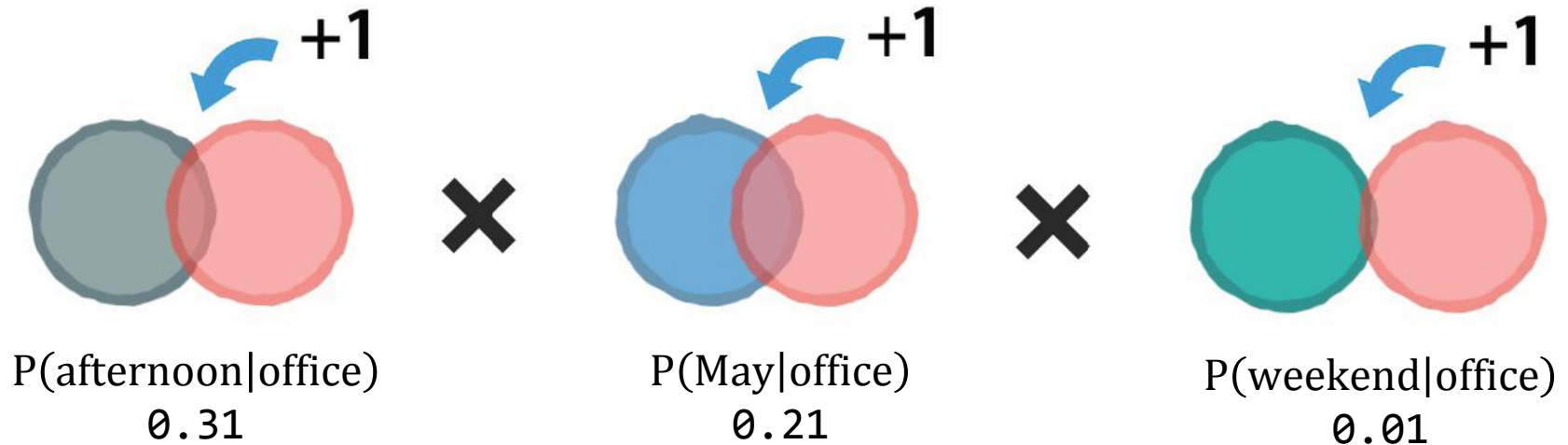
## accuracy of our model
mean(locations_test$location == pred)
## [1] 0.8854167
```


An "infrequent" problem



Joint Event that never happened before, nullify effect of all other events

The Laplace correction



```
locmodel <- naive_bayes(location ~ daytype + hourtype, data = locations_train)
pred <- predict(locmodel, locations_test, type = 'prob')
head(pred, n = 4)
```

```
##      appointment campus      home office  restaurant      store theater
## [1,]           0      0 0.9978926      0 0.0007773246 0.001330089      0
## [2,]           0      0 0.9978926      0 0.0007773246 0.001330089      0
## [3,]           0      0 0.9978926      0 0.0007773246 0.001330089      0
## [4,]           0      0 0.9978926      0 0.0007773246 0.001330089      0
```

```
locmodel
```

```
## ===== Naive Bayes =====
```

```
...
```

```
##
```

```
## daytype  appointment  campus      home      office restaurant      store
## weekday   0.5000000  1.0000000 0.6396517 1.0000000  0.8051948 0.6666667
## weekend    0.5000000  0.0000000 0.3603483 0.0000000  0.1948052 0.3333333
```

```
##
```

```
## daytype  theater
## weekday  0.0000000
## weekend   1.0000000
```

```
##
```

```
##
```

```
## hourtype  appointment  campus      home      office restaurant
## afternoon 0.50000000 0.54166667 0.11587408 0.66666667 0.51948052
## evening   0.25000000 0.01388889 0.18687207 0.03307888 0.16883117
## morning    0.25000000 0.44444444 0.23241795 0.30025445 0.29870130
## night      0.00000000 0.00000000 0.46483590 0.00000000 0.01298701
```

```
##
```

```
## hourtype      store      theater
## afternoon 0.10256410 0.00000000
## evening   0.87179487 1.00000000
## morning    0.00000000 0.00000000
## night      0.02564103 0.00000000
```

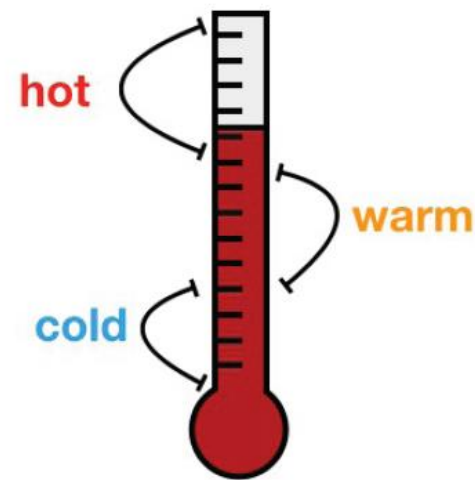
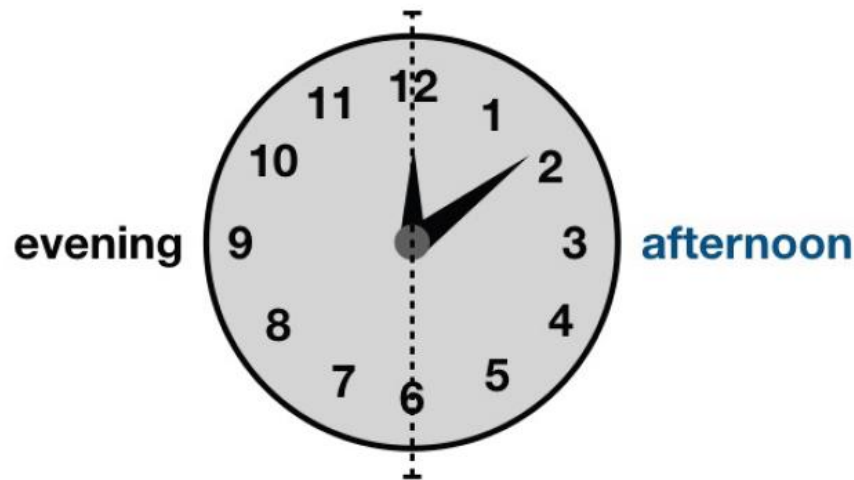
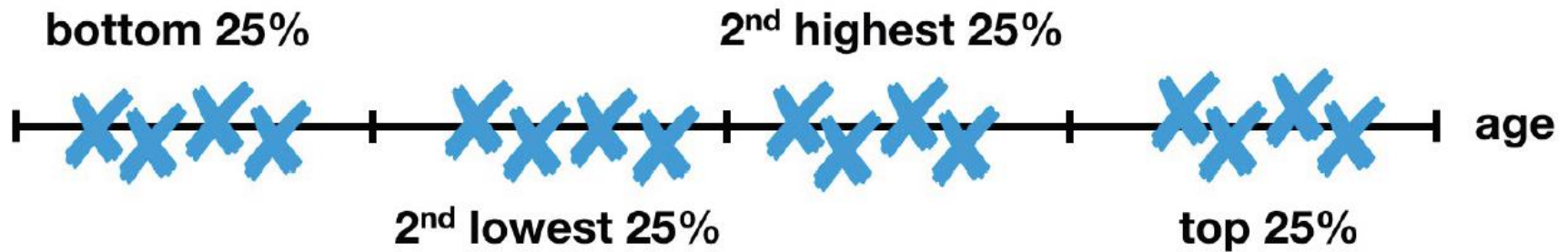
```
# Laplace corrected model
locmodel_lap <- naive_bayes(location ~ daytype + hourtype, data = locations_train, laplace = 1)
pred <- predict(locmodel_lap, locations_test, type = 'prob')
head(pred, n = 4)
```

##		appointment	campus	home	office	restaurant
##	[1,]	0.0007114617	4.61692e-05	0.9947752	9.829844e-06	0.001397514
##	[2,]	0.0007114617	4.61692e-05	0.9947752	9.829844e-06	0.001397514
##	[3,]	0.0007114617	4.61692e-05	0.9947752	9.829844e-06	0.001397514
##	[4,]	0.0007114617	4.61692e-05	0.9947752	9.829844e-06	0.001397514
##		store	theater			
##	[1,]	0.00206529	0.000994573			
##	[2,]	0.00206529	0.000994573			
##	[3,]	0.00206529	0.000994573			
##	[4,]	0.00206529	0.000994573			

locmodel_lap

##						
##	daytype	appointment	campus	home	office	restaurant
##	weekday	0.33333333	0.92405063	0.63733333	0.98500000	0.75000000
##	weekend	0.33333333	0.01265823	0.35933333	0.00250000	0.19047619
##						
##	daytype	store	theater			
##	weekday	0.58695652	0.07692308			
##	weekend	0.30434783	0.53846154			
##						
##	hourtype	appointment	campus	home	office	restaurant
##	afternoon	0.33333333	0.50632911	0.11600000	0.65750000	0.48809524
##	evening	0.20000000	0.02531646	0.18666667	0.03500000	0.16666667
##	morning	0.20000000	0.41772152	0.23200000	0.29750000	0.28571429
##	night	0.06666667	0.01265823	0.46333333	0.00250000	0.02380952
##						
##	hourtype	store	theater			
##	afternoon	0.10869565	0.07692308			
##	evening	0.76086957	0.53846154			
##	morning	0.02173913	0.07692308			
##	night	0.04347826	0.07692308			

Binning numeric data for Naive Bayes



Naïve Bayes

Strengths

- Simple, fast, and very effective
- Does well with noisy and missing data
- Requires relatively few examples for training, but also works well with very large numbers of examples
- Easy to obtain the estimated probability for a prediction

Weaknesses

- Relies on an often-faulty assumption of equally important and independent features
 - Not ideal for datasets with large numbers of numeric features
 - Estimated probabilities are less reliable than the predicted classes
-

References

- Practical Data Science with R, by Nina Zumel and John Mount
- R을 이용한 데이터 분석 실무, 서민구, 길벗
- [DBGUIDE 연재] ggplot2를 이용한 R 시각화
 - <http://freesearch.pe.kr/archives/3134>

Appendix - Example

- SMS (short message service) spam filtering

- Spam – spam

Congratulations ur awarded 500 of CD vouchers or 125gift guaranteed & Free entry 2 100 wkly draw txt MUSIC to 87066

December only! Had your mobile 11mths+? You are entitled to update to the latest colour camera mobile for Free! Call The Mobile Update Co FREE on 08002986906

Valentines Day Special! Win over £1000 in our quiz and take your partner on the trip of a lifetime! Send GO to 83600 now. 150p/msg rcvd.

- Non-spam – ham

Better. Made up for Friday and stuffed myself like a pig yesterday. Now I feel bleh. But at least its not writhing pain kind of bleh.

If he started searching he will get job in few days. He have great potential and talent.

I got another job! The one at the hospital doing data analysis or something, starts on monday! Not sure when my thesis will got finished

Data Loading

- SMS (short message service) spam filtering

```
> sms_raw<-read.csv("sms_spam.csv",stringsAsFactors = FALSE)
```

	type	text
1	ham	Hope you are having a good week. Just checking in
2	ham	K..give back my thanks.
3	ham	Am also doing in cbe only. But have to pay.
4	spam	complimentary 4 STAR Ibiza Holiday or ₩10,000 cash n...
5	spam	okmail: Dear Dave this is your final notice to collect yo...
6	ham	Aiya we discuss later lar... Pick u up at 4 is it?
7	ham	Are you this much buzy
8	ham	Please ask mummy to call father
9	spam	Marvel Mobile Play the official Ultimate Spider-man gam...
10	ham	fyi I'm at usf now, swing by the room whenever
11	ham	Sure thing big man. i have hockey elections at 6, shoul...

Data preparation

```
> sms_raw$type<-factor(sms_raw$type)
> str(sms_raw$type)
  Factor w/ 2 levels "ham","spam": 1 1 1 2 2 1 1 1 2 1 ...
> table(sms_raw$type)

ham spam
4811  747
```

Data preparation

- text -> bag of words
 - text mining package “tm”

```
> install.packages("tm")
```

```
Installing package into 'C:/Users/Hyebyong Choi/Documents/R/win-library/3.2'
(as 'lib' is unspecified)
```

```
also installing the dependencies 'NLP', 'slam'
```

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/NLP_0.1-9.zip'
Content type 'application/zip' length 278734 bytes (272 KB)
downloaded 272 KB
```

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/slam_0.1-34.zip'
Content type 'application/zip' length 111493 bytes (108 KB)
downloaded 108 KB
```

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/tm_0.6-2.zip'
Content type 'application/zip' length 710657 bytes (694 KB)
downloaded 694 KB
```

```
package 'NLP' successfully unpacked and MD5 sums checked
package 'slam' successfully unpacked and MD5 sums checked
package 'tm' successfully unpacked and MD5 sums checked
```

```
The downloaded binary packages are in
```

```
C:/Users/Hyebyong Choi/AppData/Local/Temp/RtmpUHLcnN/downloaded_packages
```

```
> library(tm)
```

```
필요한 패키지를 로딩중입니다: NLP
```

Data preparation

- text -> bag of words
 - text mining package “tm”

```
> sms_corpus <- Corpus(VectorSource(sms_raw$text))
```

This command uses two functions. First, the `Corpus()` function creates an R object to store text documents. This function takes a parameter specifying the format of the text documents to be loaded. Since we have already read the SMS messages and stored them in an R vector, we specify `VectorSource()`, which tells `Corpus()` to use the messages in the vector `sms_train$text`. The `Corpus()` function stores the result in an object named `sms_corpus`.



The `Corpus()` function is extremely flexible and can read documents from many different sources such as PDFs and Microsoft Word documents. To learn more, examine the *Data Import* section in the `tm` package vignette using the command: `print(vignette("tm"))`

Data preparation


```
> print(sms_corpus)
<<VCorpus>>
Metadata: corpus specific: 0, document level (indexed): 0
Content: documents: 5558
> inspect(sms_corpus[1:2])
<<VCorpus>>
Metadata: corpus specific: 0, document level (indexed): 0
Content: documents: 2

[[1]]
<<PlainTextDocument>>
Metadata: 7
Content: chars: 49

[[2]]
<<PlainTextDocument>>
Metadata: 7
Content: chars: 23

> sms_corpus[[1]]
<<PlainTextDocument>>
Metadata: 7
Content: chars: 49
> sms_corpus[[1]]$content
[1] "Hope you are having a good week. Just checking in"
> sms_corpus[[1]]$meta
author      : character(0)
datetimestamp: 2016-05-25 16:00:51
description  : character(0)
heading      : character(0)
id           : 1
language     : en
origin       : character(0)
```

Data preparation

hello!, HELLO..., and Hello  count as all “hello”

- remove numbers, punctuation, white space, and stop words.
- to make all lower-cased

```
> corpus_clean<-tm_map(sms_corpus,content_transformer(tolower))
> corpus_clean<-tm_map(corpus_clean,content_transformer(removeNumbers))
> corpus_clean<-tm_map(corpus_clean,content_transformer(removeWords),stopwords())
> corpus_clean<-tm_map(corpus_clean,content_transformer(removePunctuation))
> corpus_clean<-tm_map(corpus_clean,content_transformer(stripWhitespace))

> sms_corpus[[1]]$content
[1] "Hope you are having a good week. Just checking in"
> corpus_clean[[1]]$content
[1] "hope good week just checking "
> sms_corpus[[1234]]$content
[1] "Can u all decide faster cos my sis going home liao.."
> corpus_clean[[1234]]$content
[1] "can u decide faster cos sis going home liao"
```

Data preparation

- to document and term matrix (tokenizing)

A	B	C	D	E	F	G
	balloon	balls	bam	bambling	band	bandages
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

```
> sms_dtm <- DocumentTermMatrix(corpus_clean)
```

```
> inspect(sms_dtm[1:10, 1000:1010])
```

```
<<DocumentTermMatrix (documents: 10, terms: 11)>>
```

```
Non-/sparse entries: 0/110
```

```
Sparsity : 100%
```

```
Maximal term length: 10
```

```
Weighting : term frequency (tf)
```

```
> dim(sms_dtm)
```

```
[1] 5558 7989
```

```

      Terms
Docs caps captain captaining car card cardiff cardin cards care careabout cared
1      0      0      0      0      0      0      0      0      0      0      0
2      0      0      0      0      0      0      0      0      0      0      0
3      0      0      0      0      0      0      0      0      0      0      0
4      0      0      0      0      0      0      0      0      0      0      0
5      0      0      0      0      0      0      0      0      0      0      0
6      0      0      0      0      0      0      0      0      0      0      0
7      0      0      0      0      0      0      0      0      0      0      0

```

Splitting Training and Test set

```
> sms_raw_train <- sms_raw[1:4169, ]  
> sms_raw_test  <- sms_raw[4170:5559, ]
```

Then the document-term matrix:

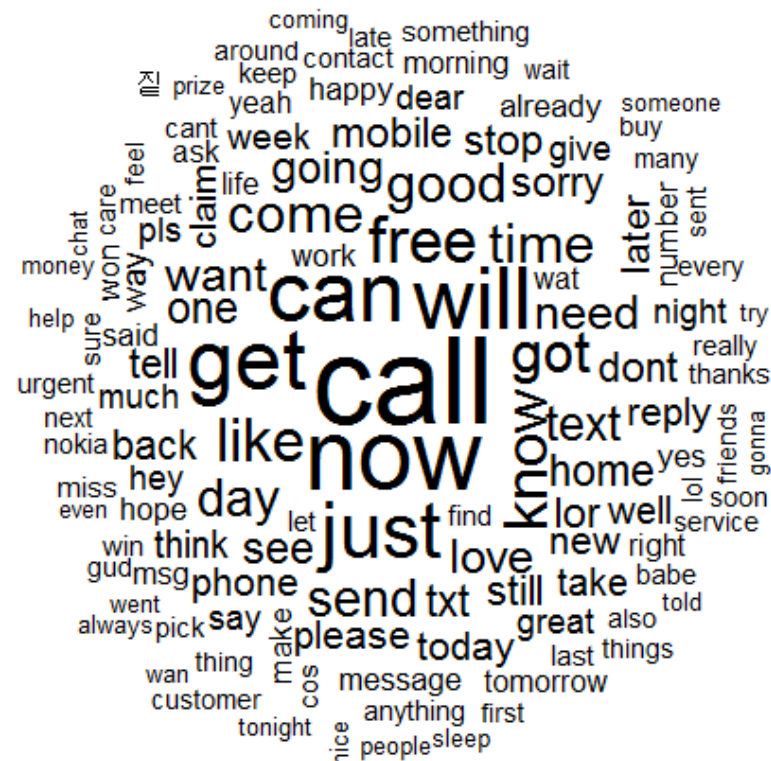
```
> sms_dtm_train <- sms_dtm[1:4169, ]  
> sms_dtm_test  <- sms_dtm[4170:5559, ]
```

And finally, the corpus:

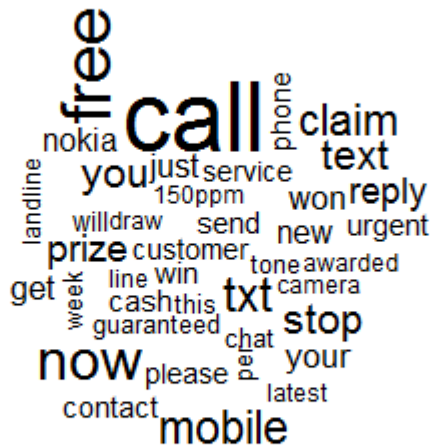
```
> sms_corpus_train <- corpus_clean[1:4169]  
> sms_corpus_test  <- corpus_clean[4170:5559]  
  
> prop.table(table(sms_raw_train$type))  
      ham      spam  
0.8647158 0.1352842  
  
> prop.table(table(sms_raw_test$type))  
      ham      spam  
0.8683453 0.1316547
```



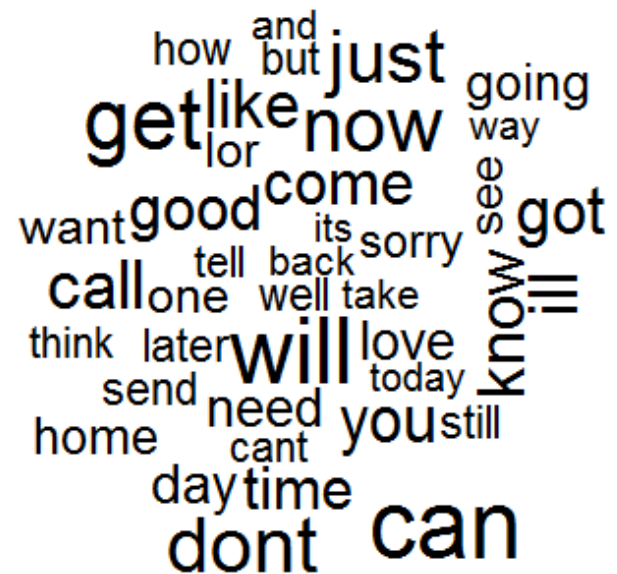
```
> wordcloud(sms_corpus_train,min.freq = 40,random.order = FALSE)
```



```
> spam<-subset(sms_raw_train,type=="spam")
> ham<-subset(sms_raw_train,type=="ham")
> wordcloud(spam$text,min.freq=20,max.words = 40,scale=c(3,0.5))
> wordcloud(ham$text,max.words = 40,scale=c(3,0.5))
```



spam



ham

Filtering frequent terms

- Remove terms that appear less than 5 times

```
findFreqTerms(sms_dtm_train,5)
```

```
sms_dict<-findFreqTerms(sms_dtm_train,5)
```

```
sms_train<-DocumentTermMatrix(sms_corpus_train,list(dictionary=sms_dict))
```

```
sms_test<-DocumentTermMatrix(sms_corpus_test,list(dictionary=sms_dict))
```

- term frequency -> term occurrence

```
> convert_counts <- function(x) {  
+   x <- ifelse(x > 0, 1, 0)  
+   x <- factor(x, levels = c(0, 1), labels = c("No", "Yes"))  
+ }  
> sms_train <- apply(sms_train, MARGIN = 2, convert_counts)  
> sms_test  <- apply(sms_test, MARGIN = 2, convert_counts)
```

Naïve Bayes

Naive Bayes classification syntax

using the `naiveBayes()` function in the `e1071` package

Building the classifier:

```
m <- naiveBayes(train, class, laplace = 0)
```

- `train` is a data frame or matrix containing training data
- `class` is a factor vector with the class for each row in the training data
- `laplace` is a number to control the Laplace estimator (by default, 0)

The function will return a naive Bayes model object that can be used to make predictions.

Making predictions:

```
p <- predict(m, test, type = "class")
```

- `m` is a model trained by the `naiveBayes()` function
- `test` is a data frame or matrix containing test data with the same features as the training data used to build the classifier
- `type` is either `"class"` or `"raw"` and specifies whether the predictions should be the most likely class value or the raw predicted probabilities

The function will return a vector of predicted class values or raw predicted probabilities depending upon the value of the `type` parameter.

Example:

```
sms_classifier <- naiveBayes(sms_train, sms_type)
sms_predictions <- predict(sms_classifier, sms_test)
```

Naïve Bayes

```
> install.packages("e1071")
Installing package into 'C:/Users/Hyebong Choi/Documents/R/win-library/3.2'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/e1071_1.6-7.zip'
Content type 'application/zip' length 814476 bytes (795 KB)
downloaded 795 KB

package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
      C:\Users\Hyebong Choi\AppData\Local\Temp\RtmpUHLcnN\downloaded_packages
> library(e1071)
Warning message:
패키지 'e1071'는 R 버전 3.2.5에서 작성되었습니다

> sms_classifier<-naiveBayes(sms_train,sms_raw_train$type)
> sms_test_pred<-predict(sms_classifier,sms_test)
```

Naïve Bayes

```
> library(gmodels)
> CrossTable(sms_test_pred, sms_raw_test$type,
  prop.chisq = FALSE, prop.t = FALSE,
  dnn = c('predicted', 'actual'))
```

This produces the following table:

Total observations in Table: 1390

predicted	actual		Row Total
	ham	spam	
ham	1203 0.997	32 0.175	1235
spam	4 0.003	151 0.825	155
Column Total	1207 0.868	183 0.132	1390

Improvement

```
> sms_classifier2 <- naiveBayes(sms_train, sms_raw_train$type, laplace = 1)
> sms_test_pred2 <- predict(sms_classifier2, sms_test)
> CrossTable(sms_test_pred2, sms_raw_test$type,
+           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
+           dnn = c('predicted', 'actual'))
```

Total Observations in Table: 1390

predicted	actual		Row Total
	ham	spam	
ham	1204 0.998	31 0.169	1235
spam	3 0.002	152 0.831	155
Column Total	1207 0.868	183 0.132	1390