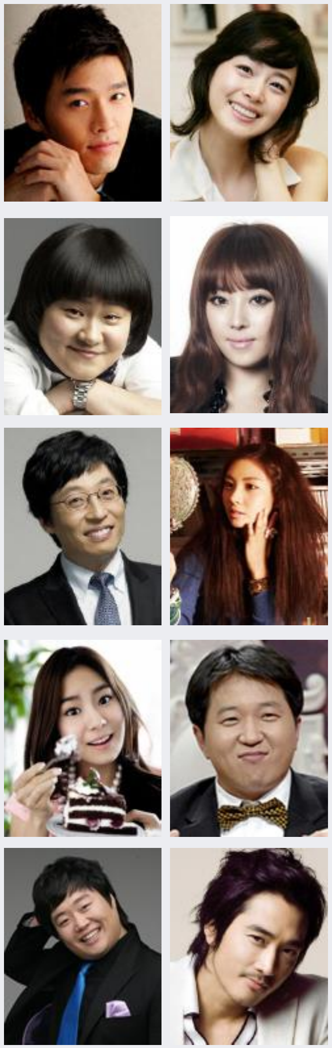
Lecture 7: k-Nearest Neighbor

Pilsung Kang
School of Industrial Management Engineering
Korea University

AGENDA

k-Nearest Neighbor Classification

- Gender classification



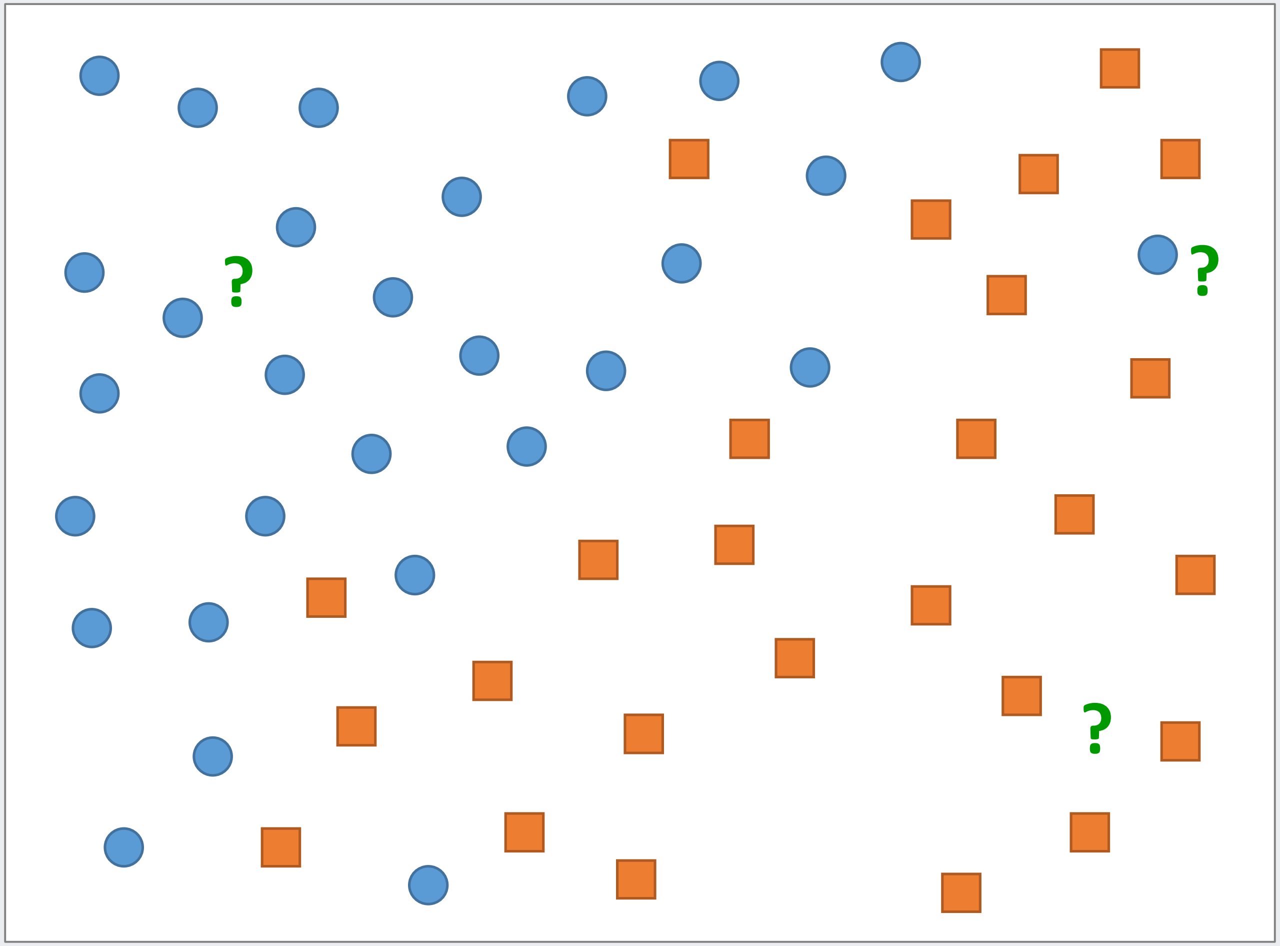
Men

Vs.

Women

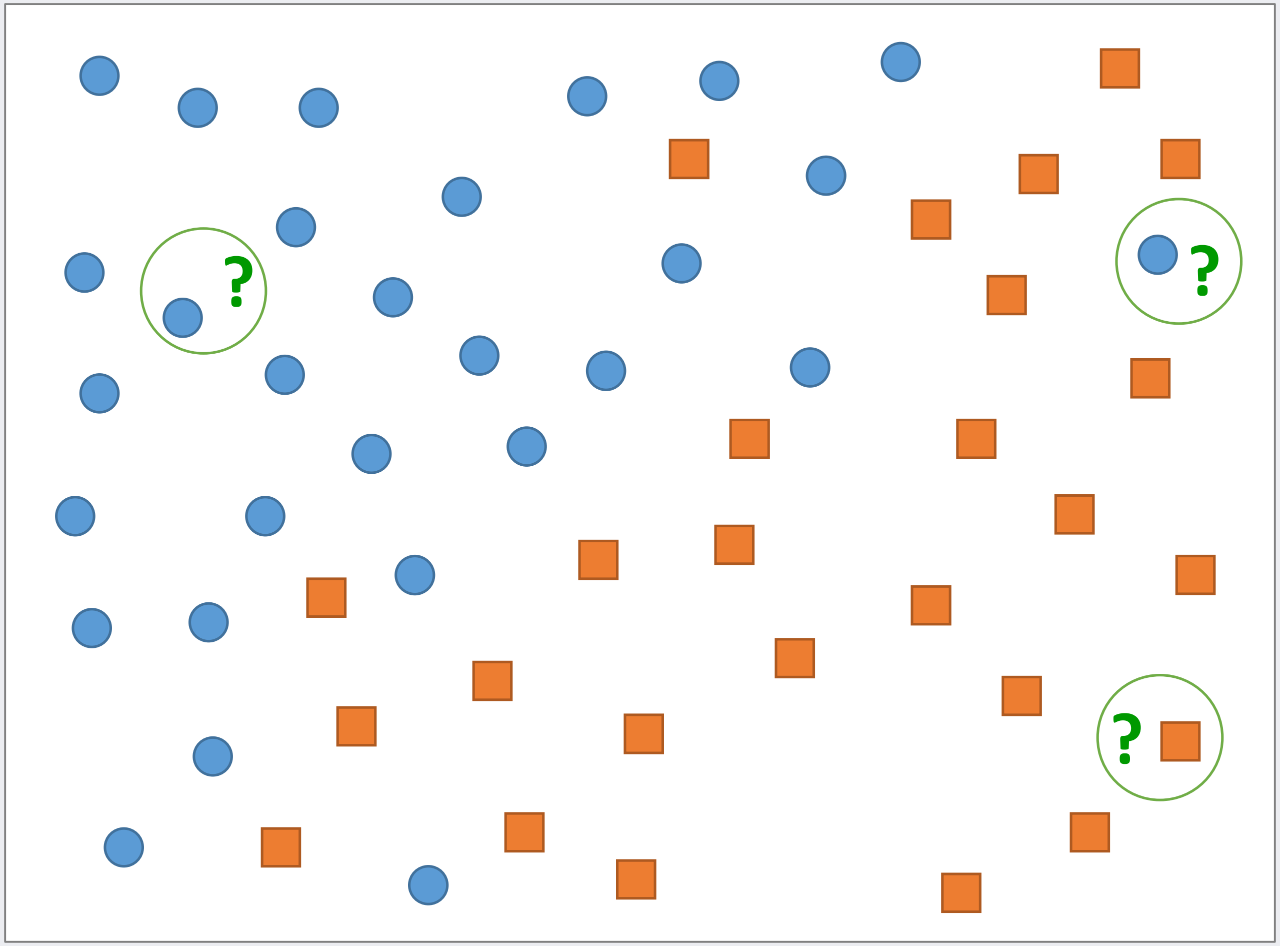
k-Nearest Neighbor Classification

- Which class does the question mark belong to?



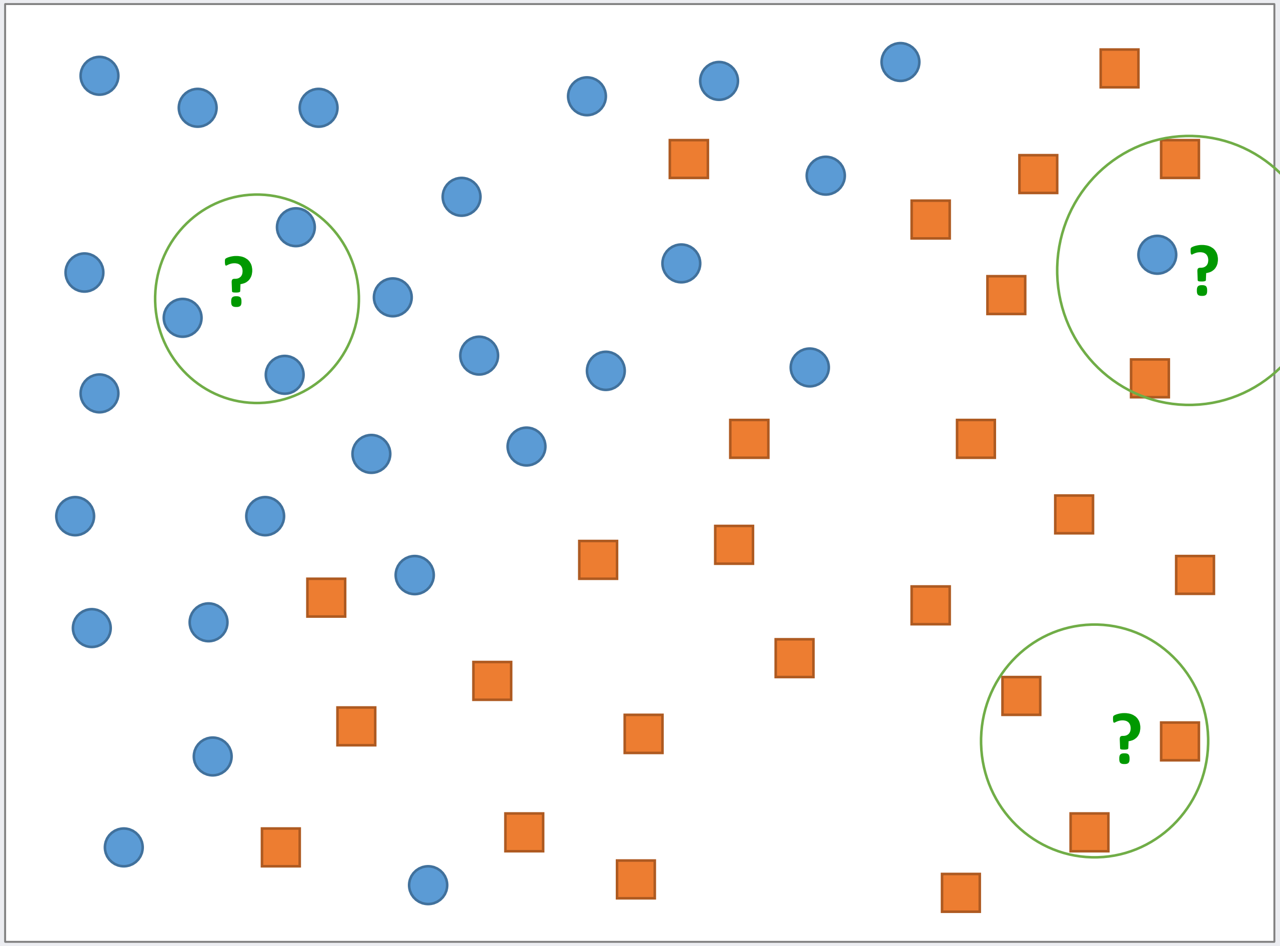
k-Nearest Neighbor Classification

- Which class does the question mark belong to?



k-Nearest Neighbor Classification

- Which class does the question mark belong to?



k-Nearest Neighbor Classification

- Motivation

類類相從　近墨者黑

"Birds of a feather flock together"

"A Man is known by the company he keeps"

k-Nearest Neighbor Classification

k-NN Classification Process

- Step 1: Prepare the reference data
  - ✓ Define attributes
    - Height, Weight, Body Fat Statistics (BFS)
  - ✓ Collect sufficient number of records from each class

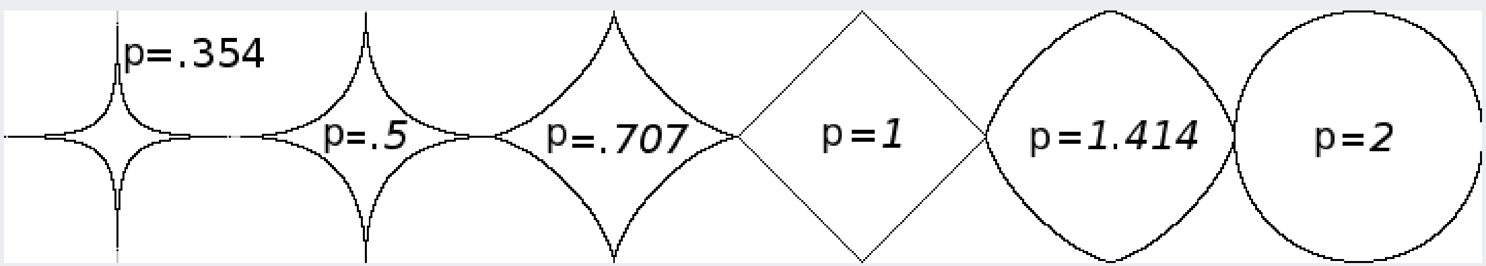
No.	Height	Weight	BFS	Class (Gender)
1	187	93	15	M
2	165	51	25	F
3	174	68	14	M
4	156	48	29	F
…	…	…	…	…
N	168	59	12	M

k-Nearest Neighbor Classification

k-NN Classification Process

- Step 2: Define the similarity measure
  - ✓ Similarity ∝ 1/distance
  - ✓ Minkovski distance with order p

$$\text{distance}\Big(P = (x_1, x_2, \ldots, x_n)\,,\, Q(y_1, y_2, \ldots, y_n)\Big) = \left(\sum\nolimits_{i=1}^{n} | x_i - y_i |^p\right)^{\frac{1}{p}}$$



  - ✓ p=2: Euclidean distance
  - ✓ p=1: Mahattan distance

k-Nearest Neighbor Classification

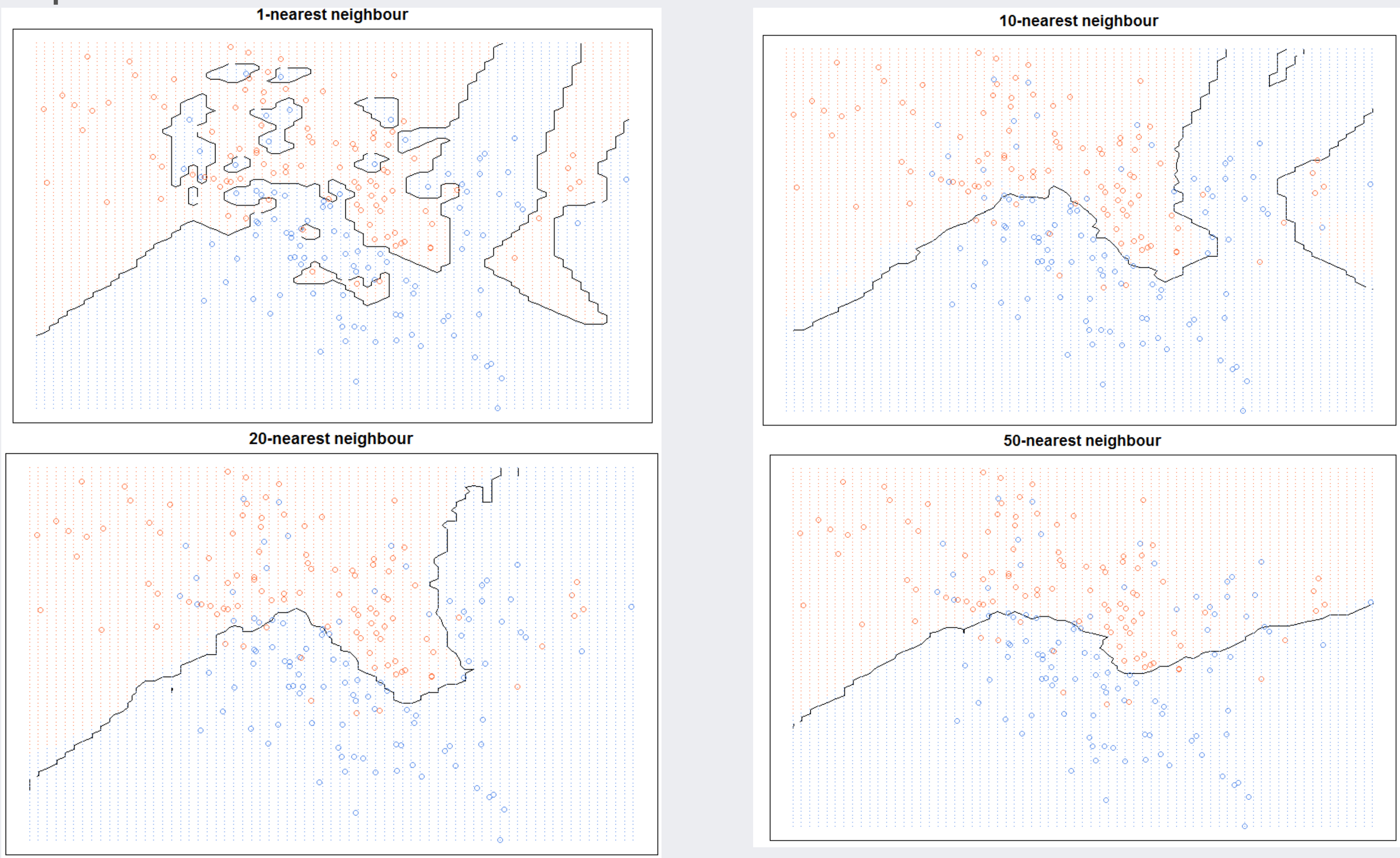
k-NN Classification Process

- Step 3: Initialize the set of candidate values for k
  - ✓ If k is too small, then the classification will be highly locally sensitive (over-fitting).
  - ✓ If k is too large, then it will lose the ability to capture the local structure (under-fitting).
  - ✓ A proper k should be chosen among a set of candidates.
  - ✓ Use the validation data.

k-Nearest Neighbor Classification

k-NN Classification Process

- Step 3: Initialize the set of candidate values for k



k-Nearest Neighbor Classification

k-NN Classification Process

- Step 4: Determine the combining rule
  - ✓ Majority voting vs. Weighted voting

For a new data **X**

Neighbor	Class	Distance	1/distance	Weight
N1	M	1	1.00	0.44
N2	F	2	0.50	0.22
N3	M	3	0.33	0.15
N4	F	4	0.25	0.11
N5	F	5	0.20	0.08

  - ✓ Majority voting: P(X=M) = 2/5 = 0.4
  - ✓ Weighted voting: P(X=M) = 0.59
  - ✓ If the cut-off is set to 0.5 X is classified as F by the majority voting while classified as M by the weighted voting

k-Nearest Neighbor Classification

k-NN Classification Process

- Step 5: Find the best k using the validation dataset

Value of k	% Error Training	% Error Validation	
1	0.00	33.33	
2	16.67	33.33	
3	11.11	33.33	
4	22.22	33.33	
5	11.11	33.33	
6	27.78	33.33	
7	22.22	33.33	
8	22.22	16.67	<--- Best k
9	22.22	16.67	
10	22.22	16.67	
11	16.67	33.33	
12	16.67	16.67	
13	11.11	33.33	
14	11.11	16.67	
15	5.56	33.33	
16	16.67	33.33	
17	11.11	33.33	
18	50.00	50.00	

k-Nearest Neighbor Classification

- k-NN Issue 1: Normalization
  - ✓ Normalization or scaling must be done before finding k-nearest neighbors
  - ✓ If not, variables with large measuring units are over-emphasized while variables with small measuring units are under-evaluated

[Before Normalization]

No.	Height	Weight	BFS	Gender
1	187	93	15	M
2	165	51	25	F
3	174	68	14	M
4	156	48	29	F
…	…	…	…	…
N	168	59	12	M
Avg.	165	65	20	-
Stdev.	15	10	5	-

[After Normalization]

No.	Height	Weight	BFS	Gender
1	1.47	2.80	-1.00	M
2	0.00	-1.40	1.00	F
3	0.60	0.30	-1.20	M
4	-0.60	-1.70	1.80	F
…	…	…	…	…
N	0.20	-0.60	-1.60	M

k-Nearest Neighbor Classification

- k-NN Issue 2: Cut-off
  - ✓ Consider the prior probability of each class
  - ✓ Assume that N(CM) = 100, N(CF) = 400

For a new data **X**

Neighbor	Class
N1	M
N2	F
N3	M
N4	F
N5	F

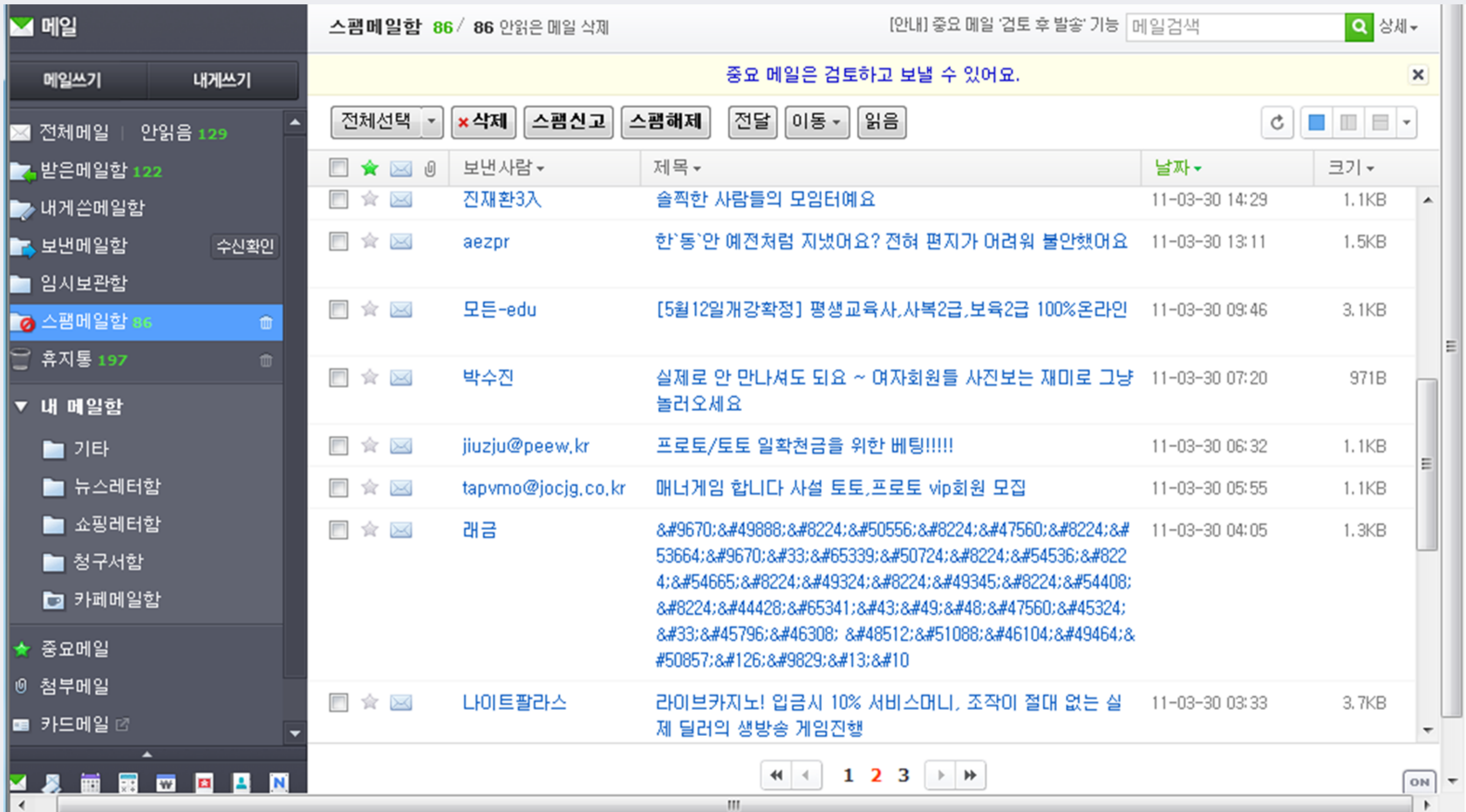
Majority voting

P(X=M)=0.4

  - ✓ If the cut-off is set to 0.5 (assuming equal class distribution), then X is classified as F.
  - ✓ If the cut-off is set to 0.2 (proportion of M among the people), then X is classified as M.

k-Nearest Neighbor Classification

- k-NN Classification Application: Spam Filtering



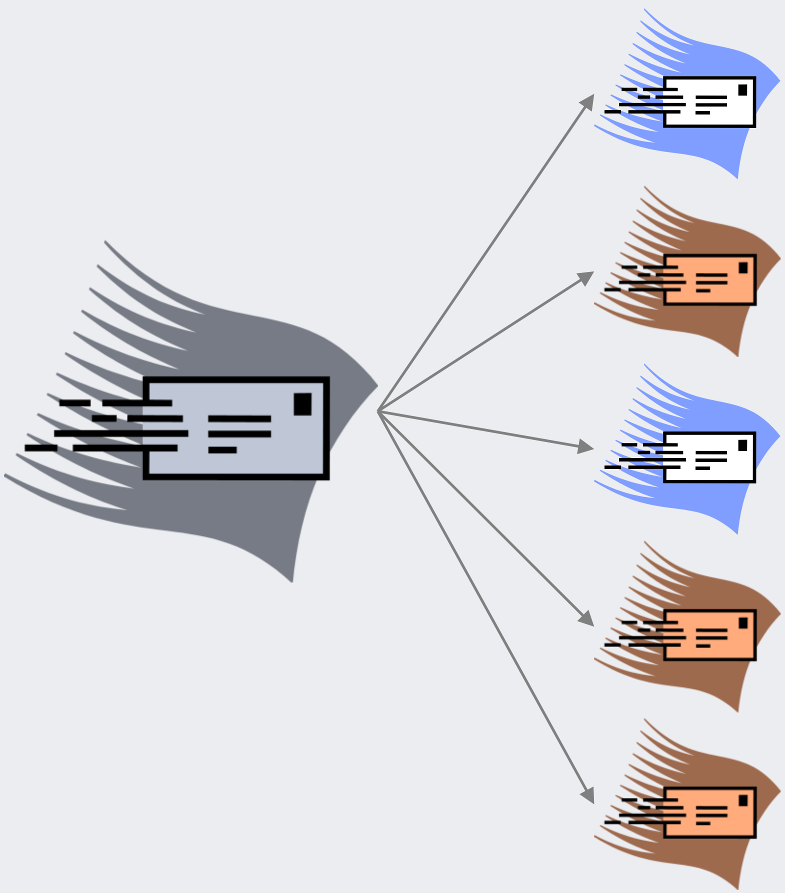
k-Nearest Neighbor Classification

- k-NN Classification Application: Spam Filtering
  - ✓ Attributes: frequency of a set of keywords

Mail	회의	수정	기안	보고	대박	머니	외로워	미팅	…	스팸?
1	2	3	1	0	0	0	0	0	…	N
2	1	0	2	3	0	0	1	0	…	N
3	2	2	3	1	0	0	0	1	…	N
4	0	0	0	0	3	2	0	0	…	Y
5	0	0	0	1	0	0	2	3	…	Y
…	…	…	…	…	…	…	…	…	…	…

k-Nearest Neighbor Classification

- k-NN Classification Application: Spam Filtering
  - ✓ For a given new mail, find 5 similar existing mails



- Neighbor 1 (no spam)
- Similarity = 0.9

- Neighbor 2 (spam)
- Similarity = 0.7

- Neighbor 3 (no spam)
- Similarity = 0.6

- Neighbor 4 (spam)
- Similarity = 0.3

- Neighbor 5 (spam)
- Similarity = 0.2

If we use the majority voting, then classify the mail as "spam"

If we use the weighted voting, then classify the mail as "no spam"

AGENDA

k-Nearest Neighbor Regression

- Regression problem revisited

Predict

One's

BFS

k-Nearest Neighbor Regression

k-NN Regression Process

- Step 1: Prepare the reference data
  - ✓ Define attributes
    - Height, Weight, Gender
  - ✓ Collect sufficient number of records from each class

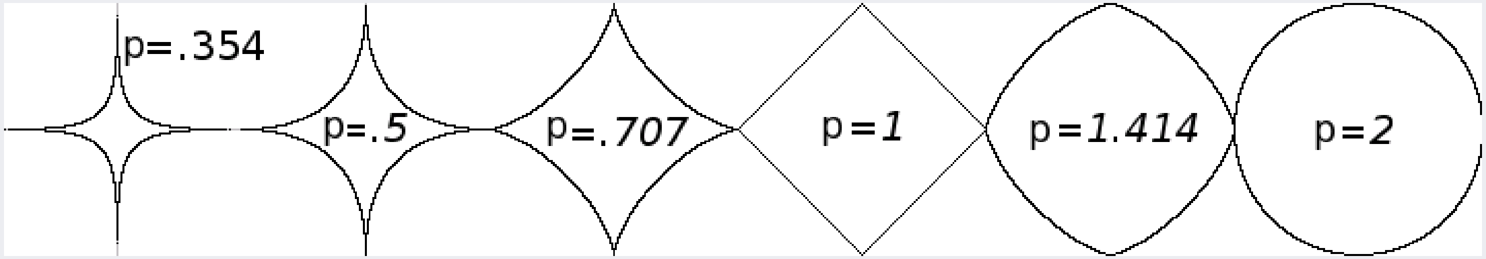
No.	Height	Weight	Gender (M=1)	BFS
1	187	93	1	15
2	165	51	0	25
3	174	68	1	14
4	156	48	0	29
…	…	…	…	…
N	168	59	1	12

k-Nearest Neighbor Regression

k-NN Regression Process

- Step 2: Define the similarity measure
  - ✓ Similarity $\propto$ 1/distance
  - ✓ Minkovski distance with order p

$$\text{distance}\Big(P = (x_1, x_2, \dots, x_n)\,, Q(y_1, y_2, \dots, y_n)\Big) = \left(\sum_{i=1}^{n} | x_i - y_i |^p\right)^{\frac{1}{p}}$$



  - ✓ p=2: Euclidean distance
  - ✓ p=1: Mahattan distance

k-Nearest Neighbor Regression

k-NN Regression Process

- Step 3: Initialize the set of candidate values for k
  - ✓ If k is too small, then the classification will be highly locally sensitive (over-fitting).
  - ✓ If k is too large, then it will lose the ability to capture the local structure (under-fitting).
  - ✓ A proper k should be chosen among a set of candidates.
  - ✓ Use the validation data.

k-Nearest Neighbor Regression

k-NN Regression Process

- Step 4: Determine the combining rule
  - ✓ Simple average vs. Weighted average

For a new data **X**

Neighbor	BFS	Distance	1/distance	Weight
N1	15.4	1	1.00	0.44
N2	17.2	2	0.50	0.22
N3	12.3	3	0.33	0.15
N4	11.5	4	0.25	0.11
N5	10.9	5	0.20	0.08

  - ✓ Simple average
    - BFS of X = (15.4+17.2+12.3+11.5+10.9)/5 = 13.46
  - ✓ Weighted average
    - BFS of X = 0.44*15.4+0.22*17.2+0.15*12.3+0.11*11.5+0.08*10.9 = 14.54

k-Nearest Neighbor Regression

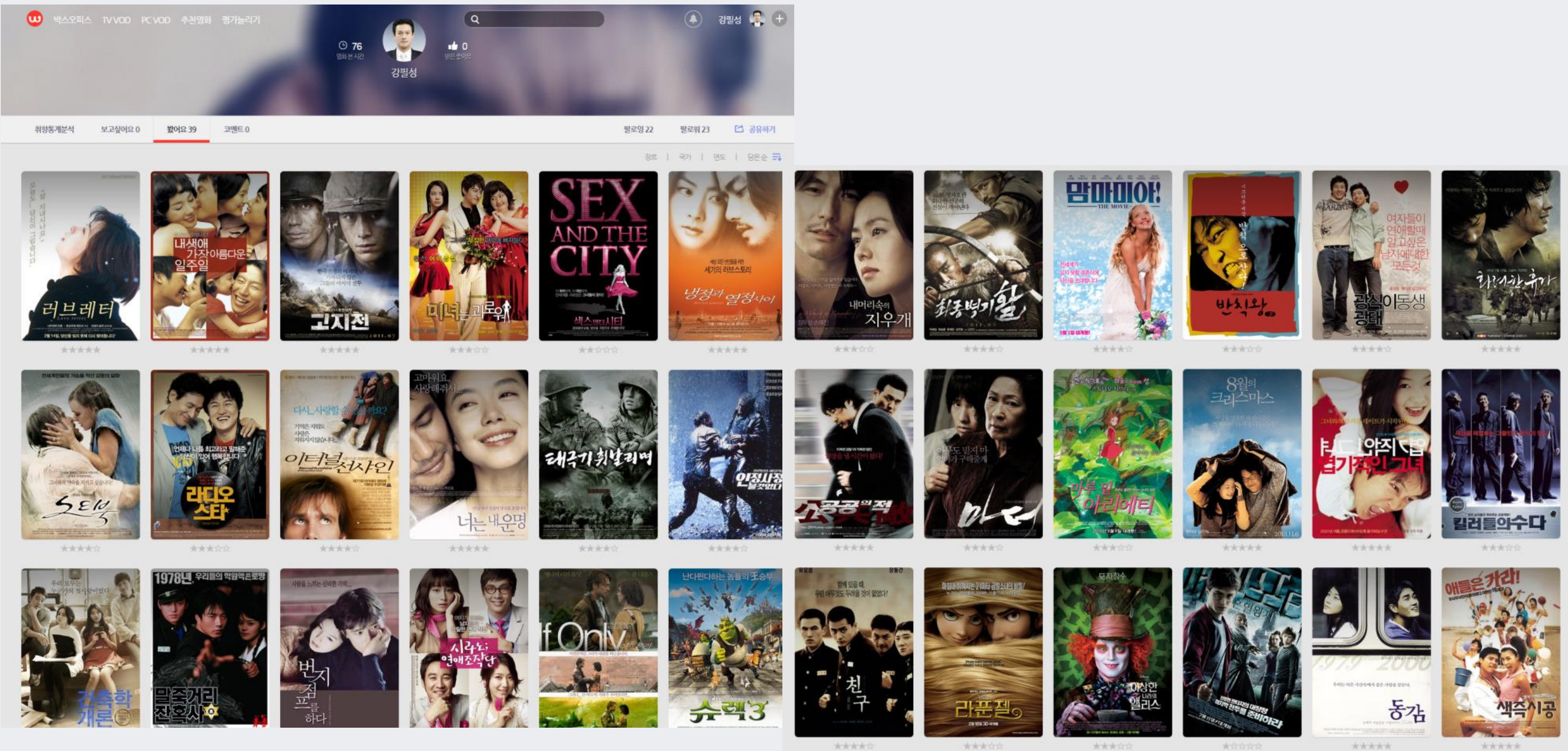
k-NN Regression Process

- Step 5: Find the best k using the validation dataset

Value of k	% Error Training	% Error Validation	
1	0.00	33.33	
2	16.67	33.33	
3	11.11	33.33	
4	22.22	33.33	
5	11.11	33.33	
6	27.78	33.33	
7	22.22	33.33	
8	22.22	16.67	<--- Best k
9	22.22	16.67	
10	22.22	16.67	
11	16.67	33.33	
12	16.67	16.67	
13	11.11	33.33	
14	11.11	16.67	
15	5.56	33.33	
16	16.67	33.33	
17	11.11	33.33	
18	50.00	50.00	

k-Nearest Neighbor Regression

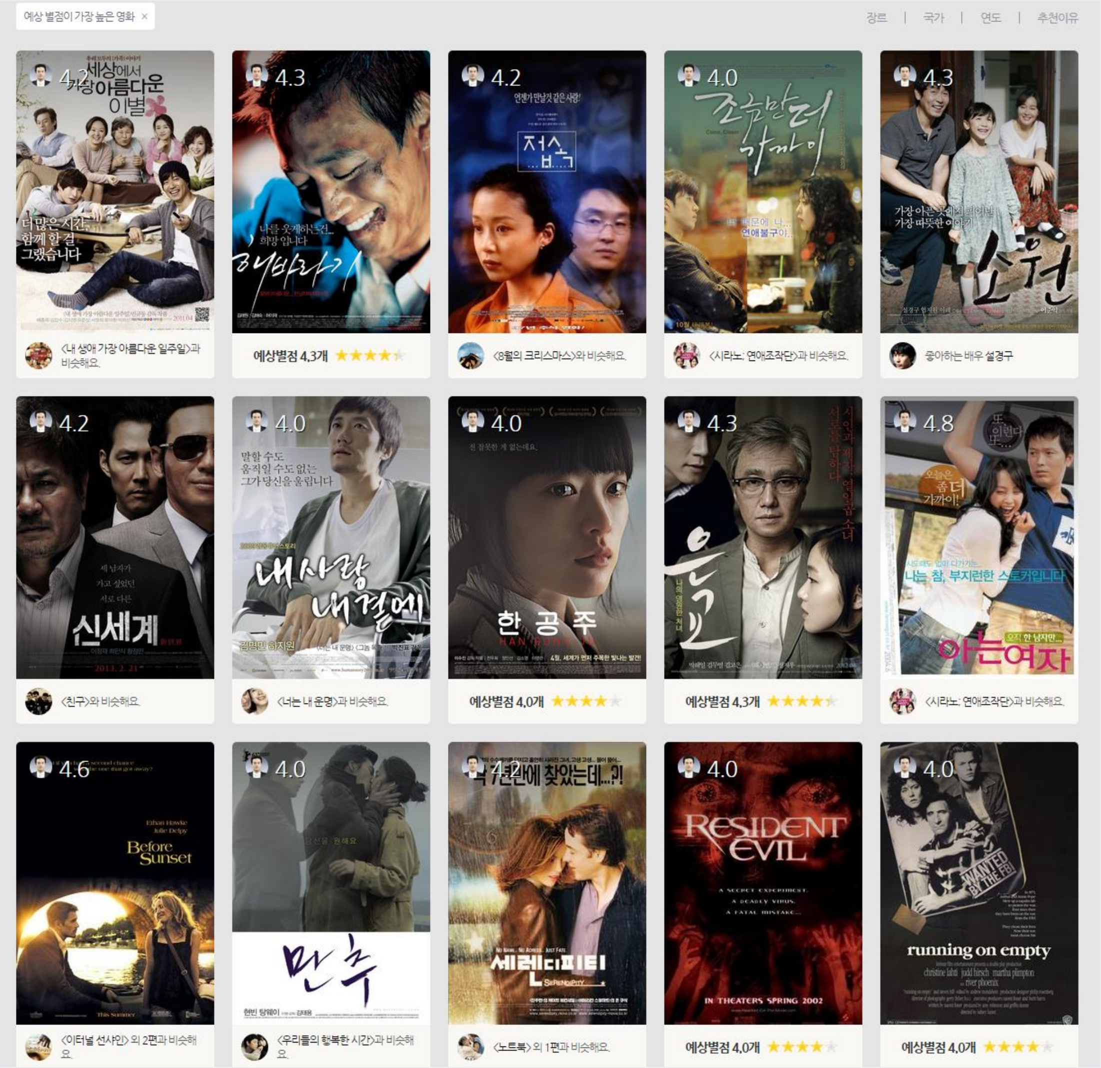
- k-Nearest Neighbor Regression Application
  - ✓ Collaborative Filtering-based Recommendation



http://watcha.net

k-Nearest Neighbor Regression

- Recommendation List



k-Nearest Neighbor Regression

- Collaborative Filtering

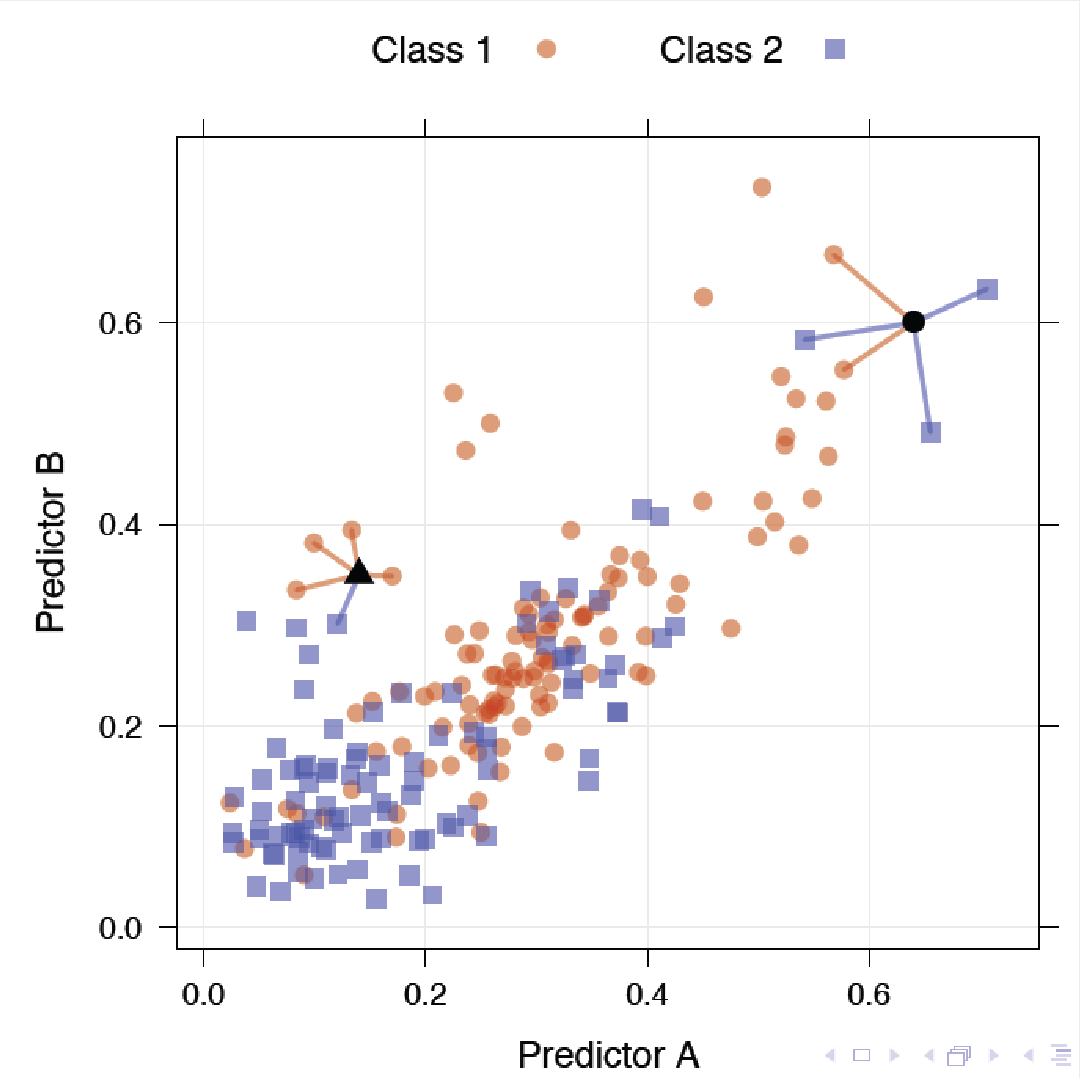
	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	…	Movie N
PSKang	10	9	5	6	9	…	? **9**

Cust.	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	…	Movie N
1	10	8	4	7	10	…	10
2	8	5	7	9	4	…	5
3	10	9	6	5	8	…	9
4	4	2	10	10	5	…	3
5	7	4	6	8	5	…	3
6	5	2	10	10	10	…	6
7	10	8	6	6	8	…	8
…	…	…	…	…	…	…	…
N	5	7	1	5	4	…	7

AGENDA

R Exercise

- k-NN Illustration



R Exercise

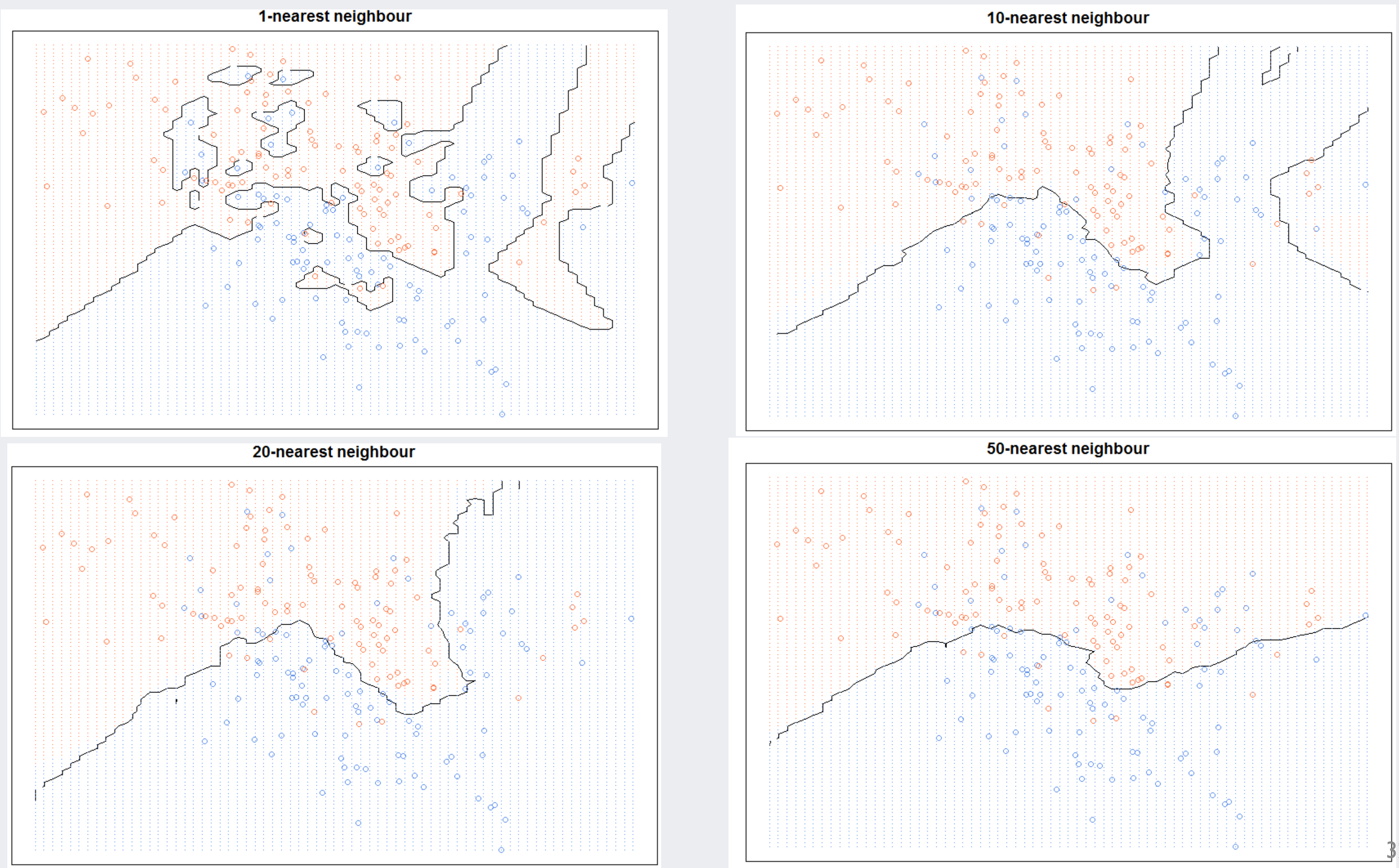
- Illustration with 2-D synthetic data

```r
# k-Nearest Neighbor Illustration ----------------------------------------
install.packages("ElemStatLearn", dependencies = TRUE)
install.packages("class", dependencies = TRUE)
library(ElemStatLearn)
library(class)

# 2-D artificial data example with k=1
x <- mixture.example$x
g <- mixture.example$y
xnew <- mixture.example$xnew
mod1 <- knn(x, xnew, g, k=50, prob=TRUE)
prob1 <- attr(mod1, "prob")
prob1 <- ifelse(mod1=="1", prob1, 1-prob1)
px1 <- mixture.example$px1
px2 <- mixture.example$px2
prob1 <- matrix(prob1, length(px1), length(px2))
par(mar=rep(2,4))
contour(px1, px2, prob1, levels=0.5, labels="", xlab="", ylab="", main= "50-nearest neighbour", axes=FALSE)
points(x, col=ifelse(g==1, "coral", "cornflowerblue"))
gd <- expand.grid(x=px1, y=px2)
points(gd, pch=".", cex=1.2, col=ifelse(prob1>0.5, "coral", "cornflowerblue"))
box()
```

R Exercise

- Decision boundaries with regard to different k values



R Exercise

- k-NN Classification & Regression
  - ✓ Classification: package "kknn"
  - ✓ Regression: package "FNN"
- Classification dataset: Wisconsin breast cancer data
  - ✓ 569 patients (malignant: 357, benign: 212)
  - ✓ To determine whether the current patient is malignant or benign based on the 30 variables extracted from the 3D image of cell nucleus

```
Ten real-valued features are computed for each cell nucleus:

        a) radius (mean of distances from center to points on the perimeter)
        b) texture (standard deviation of gray-scale values)
        c) perimeter
        d) area
        e) smoothness (local variation in radius lengths)
        f) compactness (perimeter^2 / area - 1.0)
        g) concavity (severity of concave portions of the contour)
        h) concave points (number of concave portions of the contour)
        i) symmetry
        j) fractal dimension ("coastline approximation" - 1)
```

R Exercise

- k-NN Classification
  - ✓ Divide the entire dataset into training (70%) and validation (30%) and evaluate the performance of the classifier with k=1
    - Install the package and load the dataset

```r
# k-Nearest Neigbor Learning (Classification) ----------------------------
# kknn package install & call
install.packages("kknn", dependencies = TRUE)
library(kknn)

# Load the wdbc data
RawData <- read.csv("wdbc.csv", header = FALSE)
head(RawData)
```

```
> head(RawData)
      V1    V2    V3    V4      V5      V6      V7       V8     V9     V10    V11    V12   V13    V14      V15      V16
1 13.540 14.36 87.46 566.3 0.09779 0.08129 0.06664 0.047810 0.1885 0.05766 0.2699 0.7886 2.058 23.560 0.008462 0.014600
2 13.080 15.71 85.63 520.0 0.10750 0.12700 0.04568 0.031100 0.1967 0.06811 0.1852 0.7477 1.383 14.670 0.004097 0.018980
3  9.504 12.44 60.34 273.9 0.10240 0.06492 0.02956 0.020760 0.1815 0.06905 0.2773 0.9768 1.909 15.700 0.009606 0.014320
4 13.030 18.42 82.61 523.8 0.08983 0.03766 0.02562 0.029230 0.1467 0.05863 0.1839 2.3420 1.170 14.160 0.004352 0.004899
5  8.196 16.84 51.71 201.9 0.08600 0.05943 0.01588 0.005917 0.1769 0.06503 0.1563 0.9567 1.094  8.205 0.008968 0.016460
6 12.050 14.63 78.04 449.3 0.10310 0.09092 0.06592 0.027490 0.1675 0.06043 0.2636 0.7294 1.848 19.870 0.005488 0.014270
      V17      V18     V19      V20    V21   V22   V23   V24     V25     V26     V27     V28    V29     V30 V31
1 0.02387 0.013150 0.01980 0.002300 15.110 19.26 99.70 711.2 0.14400 0.17730 0.23900 0.12880 0.2977 0.07259   M
2 0.01698 0.006490 0.01678 0.002425 14.500 20.49 96.09 630.5 0.13120 0.27760 0.18900 0.07283 0.3184 0.08183   M
3 0.01985 0.014210 0.02027 0.002968 10.230 15.66 65.13 314.9 0.13240 0.11480 0.08867 0.06227 0.2450 0.07773   M
4 0.01343 0.011640 0.02671 0.001777 13.300 22.81 84.46 545.9 0.09701 0.04619 0.04833 0.05013 0.1987 0.06169   M
5 0.01588 0.005917 0.02574 0.002582  8.964 21.96 57.26 242.2 0.12970 0.13570 0.06880 0.02564 0.3105 0.07409   M
6 0.02322 0.005660 0.01428 0.002422 13.760 20.70 89.88 582.6 0.14940 0.21560 0.30500 0.06548 0.2747 0.08301   M
```

R Exercise

- k-NN Classification
  - ✓ Divide the entire dataset into training (70%) and validation (30%) and evaluate the performance of the classifier with k=1
    - Normalize the variables and divide the dataset

```
# Divide the dataset into the training (70%) and Validation (30%) datasets
trn_idx <- sample(1:length(Class), round(0.7*length(Class)))
trnInputs <- ScaledInputData[trn_idx,]
trnTargets <- Class[trn_idx]
valInputs <- ScaledInputData[-trn_idx,]
valTargets <- Class[-trn_idx]

trnData <- data.frame(trnInputs, trnTargets)
colnames(trnData)[31] <- "Target"
valData <- data.frame(valInputs, valTargets)
colnames(valData)[31] <- "Target"
```

trnData	398 obs. of 31 variables
trnInputs	num [1:398, 1:30] 0.577 0.233 -0.658 -0.865 -0.484 ...
valData	171 obs. of 31 variables
valInputs	num [1:171, 1:30] -0.138 -1.123 -1.447 -0.169 -0.36 ...

R Exercise

- k-NN Classification
  - ✓ Divide the entire dataset into training (70%) and validation (30%) and evaluate the performance of the classifier with k=1
    - Perform k-NN

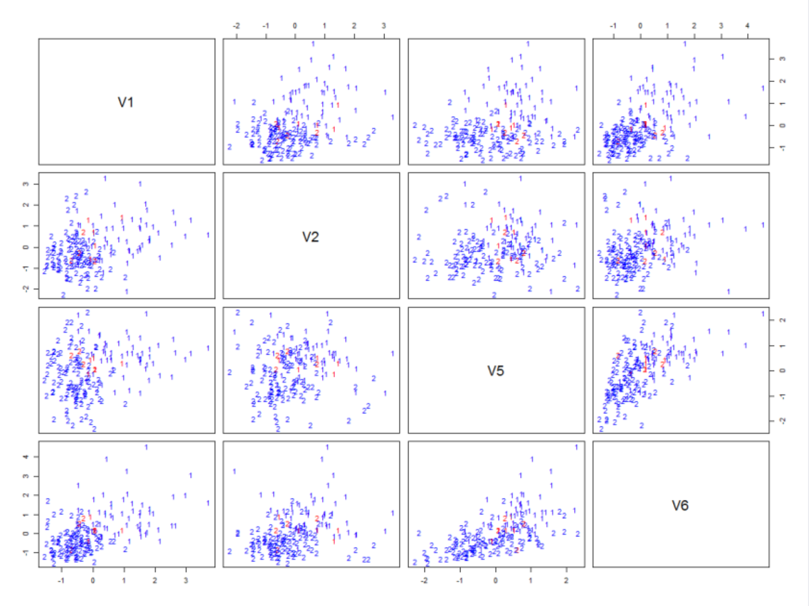
```
# Perform k-nn classification with k=1, Distance = Euclidean, and weighted scheme = majority voting
kknn <- kknn(Target ~ ., trnData, valData, k=1, distance=2, kernel = "rectangular")

# View the k-nn results
summary(kknn)
kknn$CL
kknn$W
kknn$D
```

R Exercise

- k-NN Classification
  - ✓ Divide the entire dataset into training (70%) and validation (30%) and evaluate the performance of the classifier with k=1
    - Visualize the result

```
# Visualize the classification results
knnfit <- fitted(kknn)
table(valTargets, knnfit)
pcol <- as.character(as.numeric(valTargets))
pairs(valData[c(1,2,5,6)], pch = pcol, col = c("blue", "red")[(valTargets != knnfit)+1])
```



R Exercise

- k-NN Classification
  - ✓ Divide the entire dataset into training (70%) and validation (30%) and evaluate the performance of the classifier with k=1
    - Check the validation error

```
table(valTargets, kknn$fitted.values)
```

```
> table(valTargets, kknn$fitted.values)

valTargets   B   M
         B  57   5
         M   4 105
```

R Exercise

- k-NN Classification

  ✓ Find the best k based on leave-one-out (LOO) validation

```
# Leave-one-out validation for finding the best k
knntr <- train.kknn(Target ~ ., trnData, kmax=10, distance=2, kernel="rectangular")

knntr$MISCLASS
knntr$best.parameters
```

```
> knntr$MISCLASS
   rectangular
1   0.05778894
2   0.06783920
3   0.03517588
4   0.03517588
5   0.04020101
6   0.03015075
7   0.03266332
8   0.03517588
9   0.03768844
10  0.04020101
```

```
> knntr$best.parameters
$kernel
[1] "rectangular"

$k
[1] 6
```

R Exercise

- k-NN Classification
  - ✓ Train the model with the best k

```
# Perform k-nn classification with the best k, Distance = Euclidean, and weighted scheme = majority voting
kknn_opt <- kknn(Target ~ ., trnData, valData, k=knntr$best.parameters$k, distance=2, kernel = "rectangular")
fit_opt <- fitted(kknn_opt)
cfmatrix <- table(valTargets, fit_opt)
cfmatrix
```

```
> cfmatrix
          fit_opt
valTargets   B   M
         B  59   3
         M   2 107
```

R Exercise

- k-NN Classification
  - ✓ Evaluate the classification performance

```
# Summarize the classification performances
Cperf <- matrix(0, nrow=1,ncol=3)
colnames(Cperf) <- c("Accuracy", "BCR", "F1")

# Simple Accuracy
Cperf[1,1] <- (cfmatrix[1,1]+cfmatrix[2,2])/sum(cfmatrix)

# Balanced correction rate (BCR)
Cperf[1,2] <- sqrt((cfmatrix[1,1]/(cfmatrix[1,1]+cfmatrix[1,2]))*(cfmatrix[2,2]/(cfmatrix[2,1]+cfmatrix[2,2])))

# F1-measure
Recall <- cfmatrix[2,2]/(cfmatrix[2,1]+cfmatrix[2,2])
Precision <- cfmatrix[1,1]/(cfmatrix[1,1]+cfmatrix[1,2])
Cperf[1,3] <- 2*Recall*Precision/(Recall+Precision)

Cperf
```

```
> Cperf
      Accuracy       BCR        F1
[1,] 0.9707602 0.9665155 0.9663988
```

R Exercise

- k-NN Regression
  - ✓ Dataset: Concrete Strength

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

 Concrete Compressive Strength	Multivariate	Regression	Real	1030	9	2007

Name -- Data Type -- Measurement -- Description

Cement (component 1) -- quantitative -- kg in a m3 mixture -- Input Variable
Blast Furnace Slag (component 2) -- quantitative -- kg in a m3 mixture -- Input Variable
Fly Ash (component 3) -- quantitative -- kg in a m3 mixture -- Input Variable
Water (component 4) -- quantitative -- kg in a m3 mixture -- Input Variable
Superplasticizer (component 5) -- quantitative -- kg in a m3 mixture -- Input Variable
Coarse Aggregate (component 6) -- quantitative -- kg in a m3 mixture -- Input Variable
Fine Aggregate (component 7) -- quantitative -- kg in a m3 mixture -- Input Variable
Age -- quantitative -- Day (1~365) -- Input Variable
Concrete compressive strength -- quantitative -- MPa -- Output Variable

R Exercise

- k-NN Regression
  - ✓ Install the package and normalize the data

```
# k-Nearest Neighbor Learning (Regression) --------------------------------
install.packages("FNN", dependencies = TRUE)
library(FNN)
# Concrete strength data
concrete <- read.csv("concrete.csv")

RegX <- concrete[,1:8]
RegY <- concrete[,9]

# Data Normalization
RegX <- scale(RegX, center = TRUE, scale = TRUE)

# Combine X and Y
RegData <- as.data.frame(cbind(RegX, RegY))
```

R Exercise

- k-NN Regression
  - ✓ Divide the dataset into the training (70%) and the test (30%) at random

```
# Split the data into the training/test sets
trn_idx <- sample(1:1029, round(0.7*1029))
trn_data <- RegData[trn_idx,]
test_data <- RegData[-trn_idx,]
```

Data	
concrete	1029 obs. of 9 variables
RegData	1029 obs. of 9 variables
RegX	num [1:1029, 1:8] 2.485 0.495 0.495 -0.79 -0.143 ...
test_data	309 obs. of 9 variables
trn_data	720 obs. of 9 variables

R Exercise

- k-NN Regression
  - ✓ Find the best k based on LOO validation

```r
# Find the best k using leave-one-out validation
nk <- c(1:10)
trn.n <- dim(trn_data)[1]
trn.v <- dim(trn_data)[2]

val.rmse <- matrix(0,length(nk),1)

for (i in 1:length(nk)){

  cat("k-NN regression with k:", nk[i], "\n")
  tmp_residual <- matrix(0,trn.n,1)

  for (j in 1:trn.n){

    # Data separation for leave-one-out validation
    tmptrnX <- trn_data[-j,1:(trn.v-1)]
    tmptrnY <- trn_data[-j,trn.v]
    tmpvalX <- trn_data[j,1:(trn.v-1)]
    tmpvalY <- trn_data[j,trn.v]

    # Train k-NN & evaluate
    tmp.knn.reg <- knn.reg(tmptrnX, test = tmpvalX, tmptrnY, k=nk[i])
    tmp_residual[j,1] <- tmpvalY - tmp.knn.reg$pred

  }

  val.rmse[i,1] <- sqrt(mean(tmp_residual^2))
}
```

```
> val.rmse
          [,1]
 [1,] 9.151637
 [2,] 8.897865
 [3,] 8.524996
 [4,] 8.838078
 [5,] 9.019768
 [6,] 9.087615
 [7,] 9.212508
 [8,] 9.458858
 [9,] 9.441347
[10,] 9.507920
```

R Exercise

- k-NN Regression
  - ✓ Use the best k for the final model

```
# find the best k
best.k <- nk[which.min(val.rmse)]

# Evaluate the k-NN with the test data
test.knn.reg <- knn.reg(trn_data[,1:ncol(trn_data)-1], test = test_data[,1:ncol(test_data)-1],
                        trn_data[,ncol(trn_data)], k=best.k)

tgt.y <- test_data[,ncol(trn_data)]
knn.haty <- test.knn.reg$pred
```

  - ✓ Train the MLR for benchmark

```
# Train the MLR for comparison
full_model <- lm(RegY ~ ., data = trn_data)
mlr.haty <- predict(full_model, newdata = test_data)
```

R Exercise

- k-NN Regression
  - ✓ Compare the performance of k-NN and MLR

```
# Regression performance comparison in terms of MAE
mean(abs(tgt.y-knn.haty))
mean(abs(tgt.y-mlr.haty))

# Plot the result
plot(tgt.y, knn.haty, pch = 1, col = 1, xlim = c(0,80), ylim = c(0, 80))
points(tgt.y, mlr.haty, pch = 2, col = 4, xlim = c(0,80), ylim = c(0,80))
abline(0,1,lty=3)
```

```
> mean(abs(tgt.y-knn.haty))
[1] 6.949288
> mean(abs(tgt.y-mlr.haty))
[1] 8.239489
```

R Exercise

- k-NN Regression
  - ✓ Compare the performance of k-NN and MLR