Sentiment Analysis

감정사전 & 감정점수 만들기

김형준
Analytic Director / (주) 퀀트랩 / kim@mindscale.kr

퀀트랩 소개

- 2011년 설립
- 데이터 분석, 직무역량평가, 전문성 개발 전문 컨설팅 기업

members

유재명
서울대학교 산업공학과
서울대학교 인지과학 박사(수료)

황창주
서울대학교 심리학과
서울대학교 심리학 박사(수료)

김형준
서울대학교 인류학과 / 심리학과
서울대학교 인지과학 석사

clients

- LG생활건강
- LG U+
- NC소프트
- SK플래닛
- 교통안전공단
- 삼성전자
- 이지웰페어
- 웅진씽크빅
- 중소기업진흥공단
- 한화
- 현대자동차

워크숍 관련 온라인 사이트

http://course.mindscale.kr/course/text-analysis

Facebook 아이디에 연결되어 있습니다 연결 해제

코스

현재 수강 중인 코스입니다.

제목
텍스트에서 여론과 감정을 발견하기 : R을 이용한 텍스트 데이터 분석 (05/30)
텍스트에서 여론과 감정을 발견하기 : R을 이용한 텍스트 데이터 분석
토픽 분석
R을 이용한 웹 크롤링

오늘의 목표

감정 사전 만들기

감정 점수 만들기

상관관계

회귀분석

모형평가

감정분석

- 감정 사전을 기반으로 텍스트 자료에서 긍정 단어와 부정 단어의 비율을 계산
- 감정 사전을 어떻게 만들 것인가?

사전지식

예측이란 무엇?

자기자신 : Y가 변화하는 추세
다른변수 : X가 Y를 예측

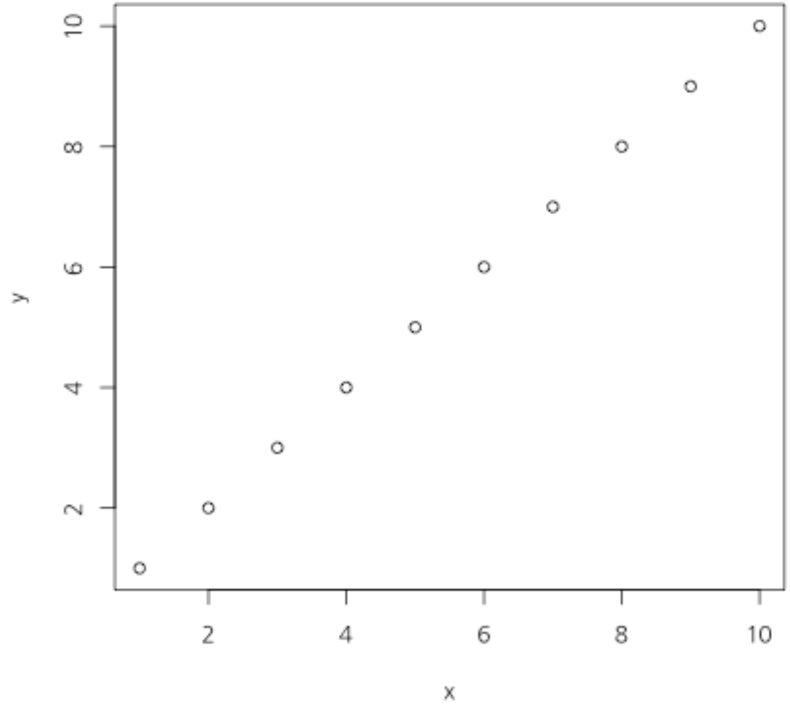
- 키로 몸무게를 예측!
- 키로 성적을 예측?
- 예측이 잘 되려면 서로 상관(관련성)이 높아야 함
  -> 감정단어로 영화 평점을 예측

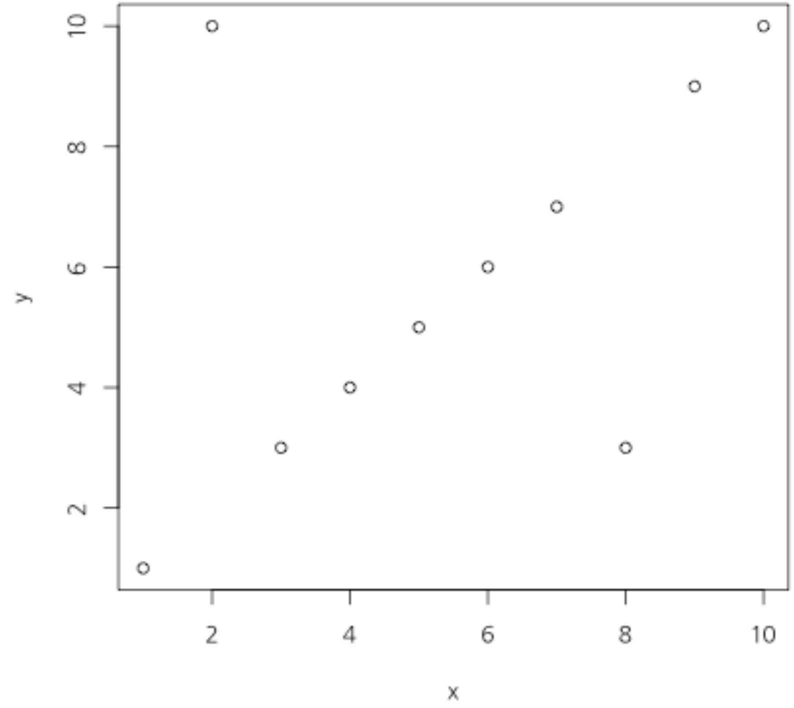
회귀분석(선형(직선) 모형)

예시

- 키가 1cm 증가할 때마다 몸무게가 1kg 증가
- 월 소득이 100만원 증가할 때마다 몸무게가 1kg 감소
- 부정단어가 1개 증가할 때 마다 평점 .1점 감점
- 긍정단어가 1개 증가할 때 마다 평점 .1점 증가

상관관계






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

```
## [1] 0.4885042
```

상관관계

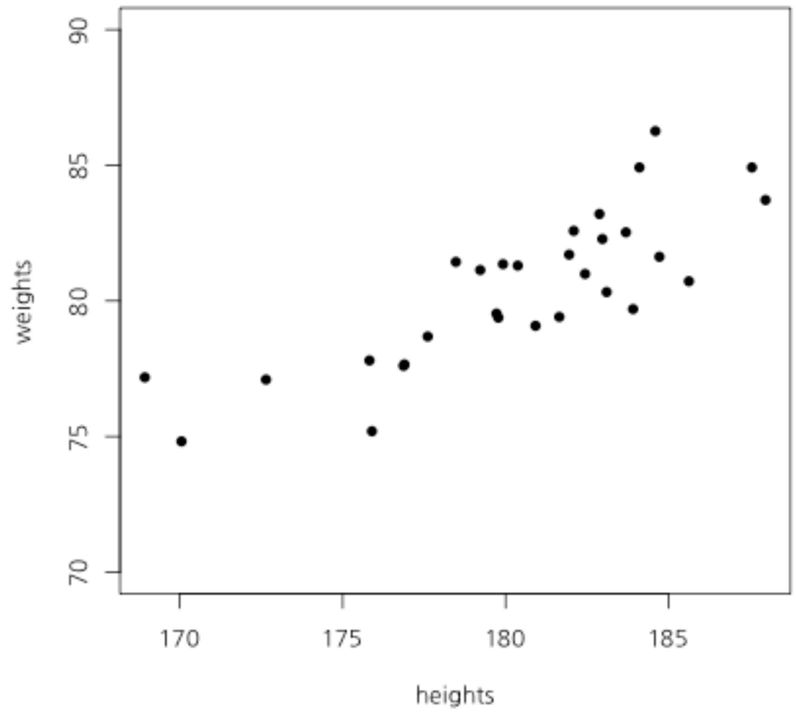
- x가 증가(혹은 감소)할때 y가 증가(혹은 감소)하는 정도

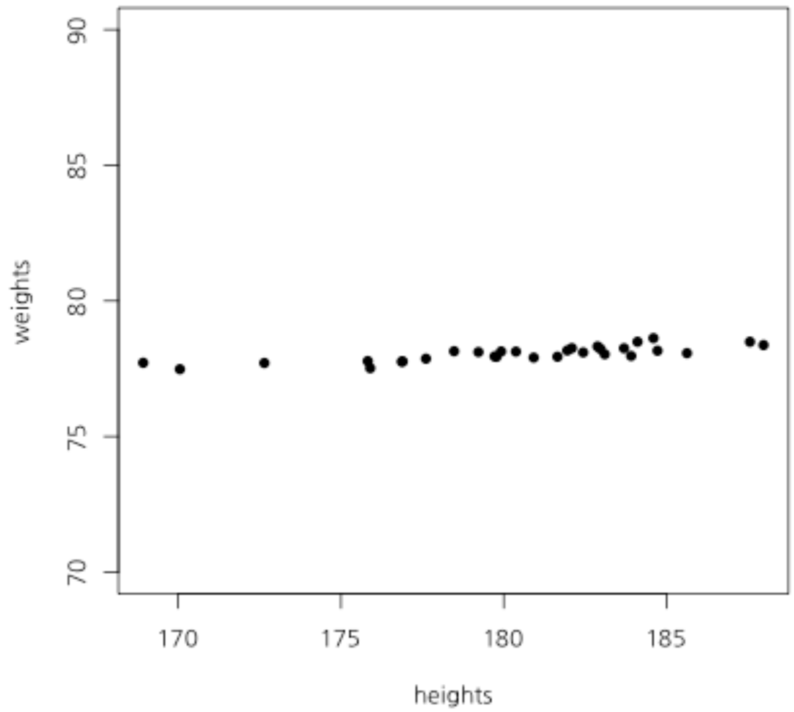
scale

키가 만약 cm라면, 키가 1cm 증가하면 몸무게는 1kg증가
키가 만약 mm라면, 키가 1mm 증가하면 몸무게는 0.1kg 증가

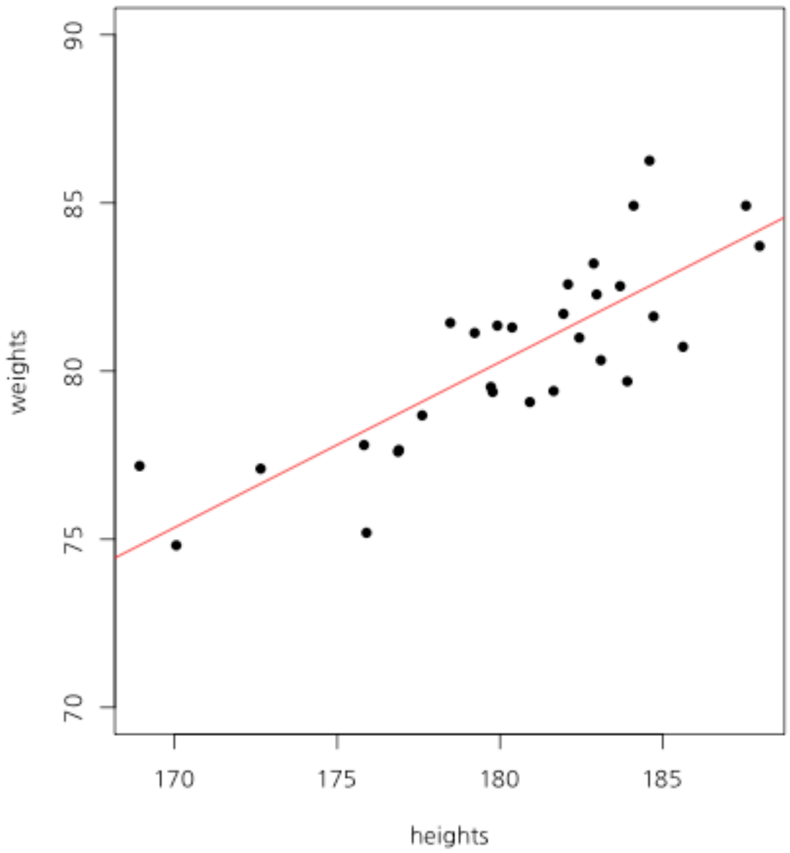
-> 표준화해야 한다

둘 중 무엇이 상관이 더 클까요?





상관관계 및 회귀분석

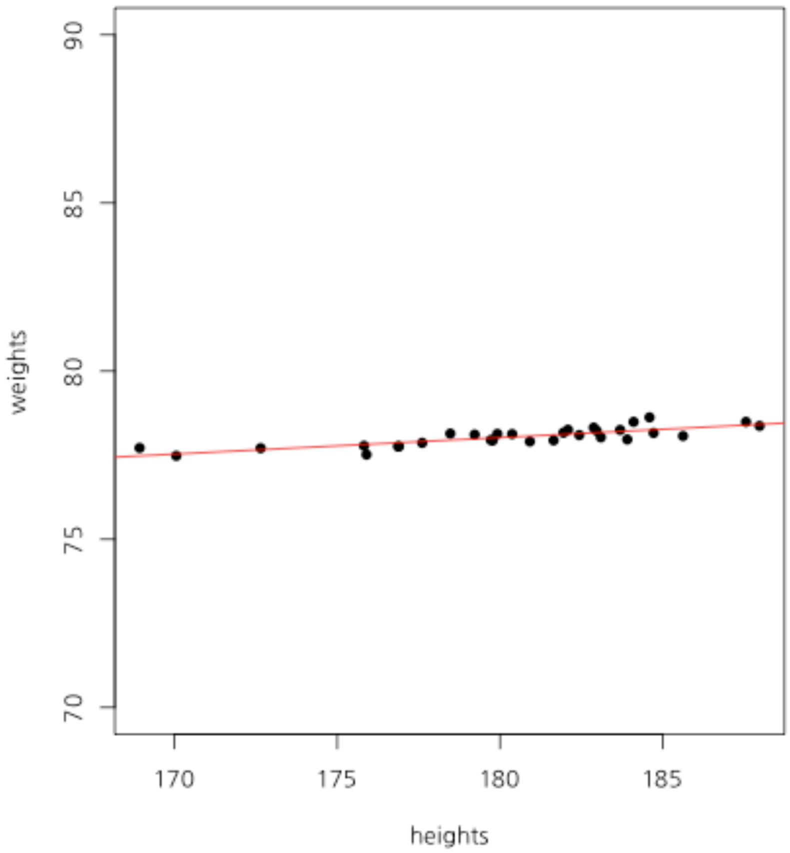


	ESTIMATE	STD. ERROR	T VALUE	PR(>|T|)
(Intercept)	-8.29	11.74	-0.71	0.49
heights	0.49	0.07	7.56	0.00

```
cor(weights, heights)
```

```
## [1] 0.8194181
```

상관관계 및 회귀분석

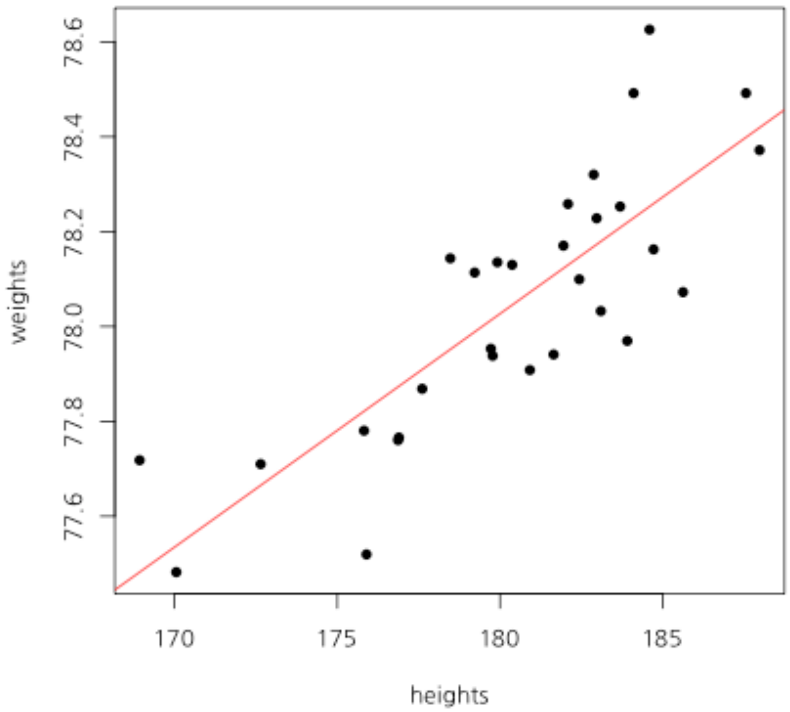


	ESTIMATE	STD. ERROR	T VALUE	PR(>|T|)
(Intercept)	69.17	1.17	58.93	0.00
heights	0.05	0.01	7.56	0.00

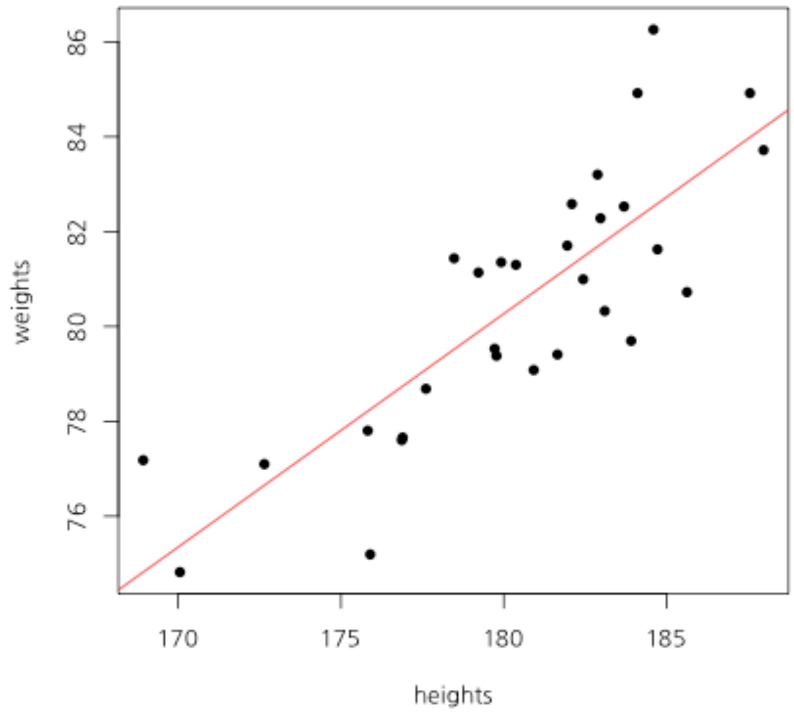
```
cor(weights, heights)
```

```
## [1] 0.8194181
```

상관관계 및 회귀분석

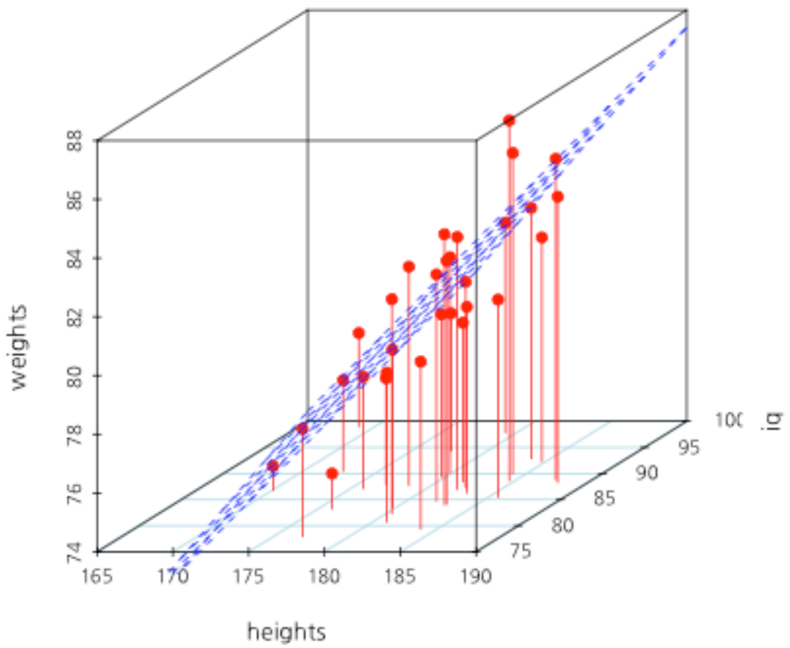


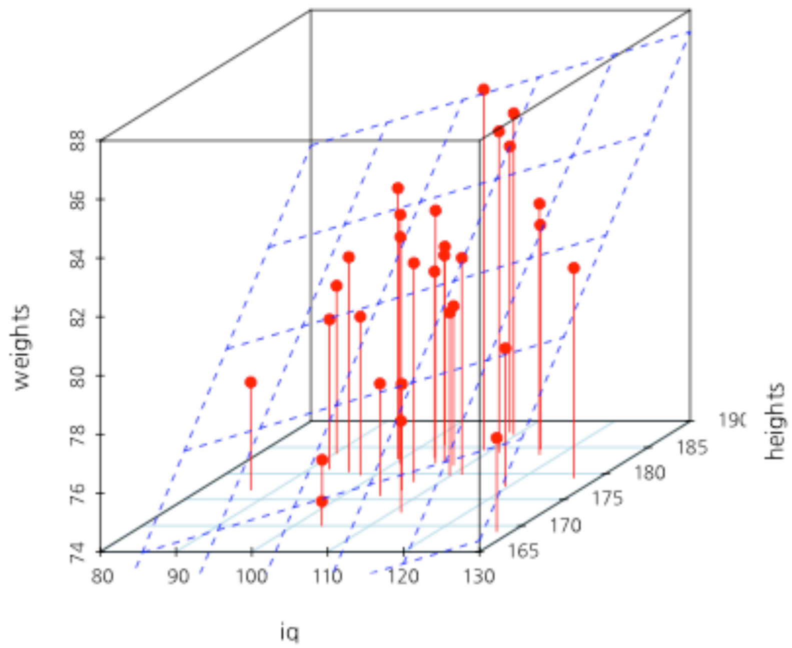
weights
78.6
78.4
78.2
78.0
77.8
77.6
170
175
180
185
heights



weights
86
84
82
80
78
76
170
175
180
185
heights

X가 2개라면?





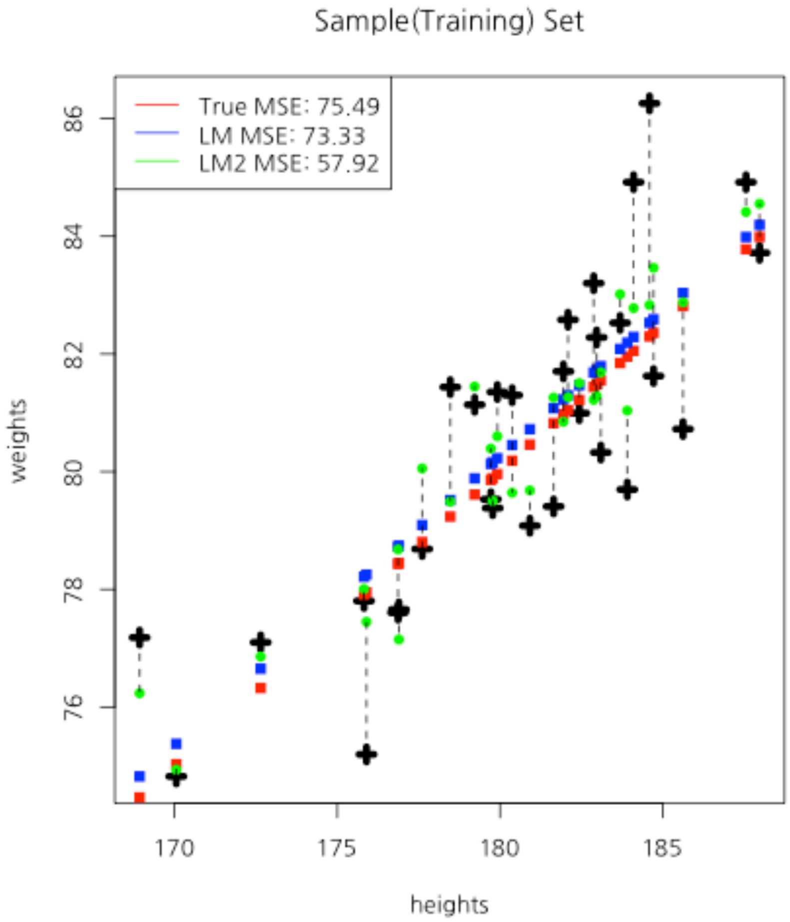

```
## [1] 0.8194181
```

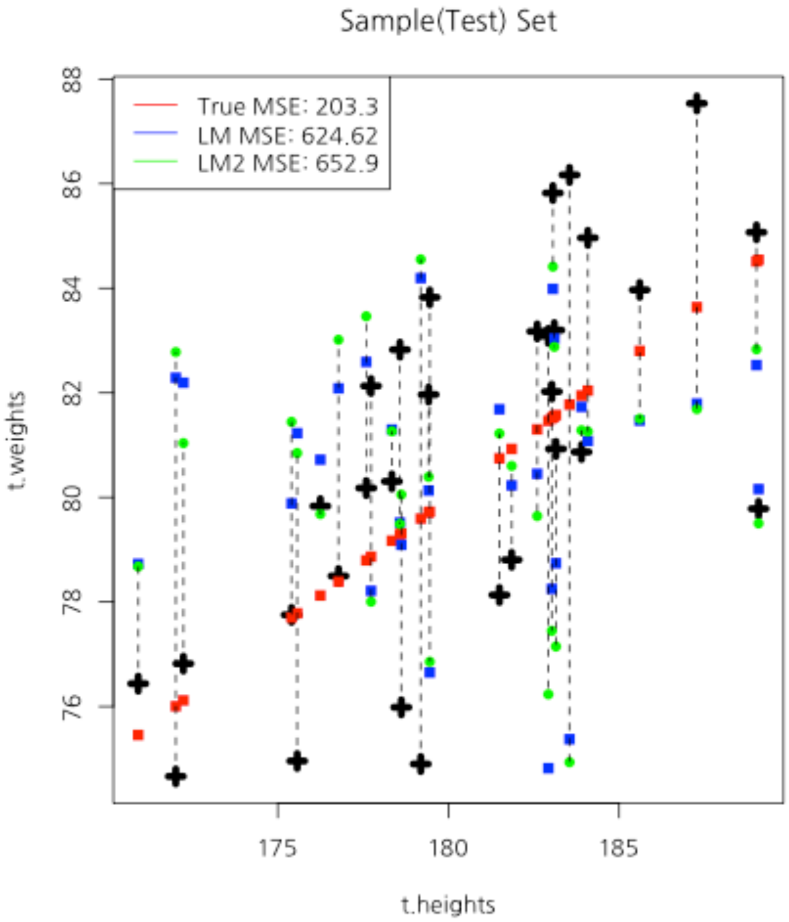
```
## [1] 0.1387562
```

다중회귀분석

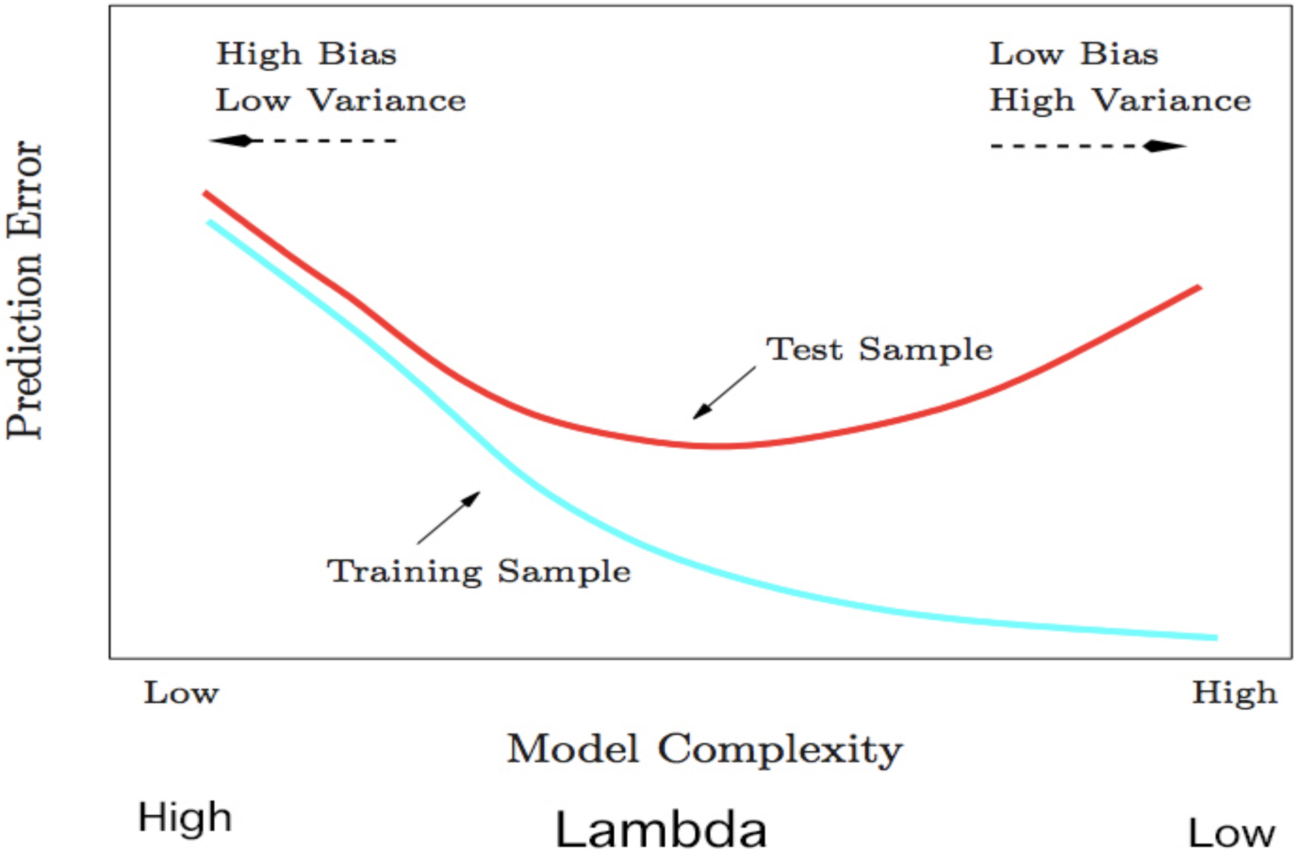
	ESTIMATE	STD. ERROR	T VALUE	PR(>|T|)
(Intercept)	-20.76	11.60	-1.79	0.08
iq	0.08	0.03	2.68	0.01
heights	0.52	0.06	8.66	0.00

Traninig Vs Test





Over-fitting

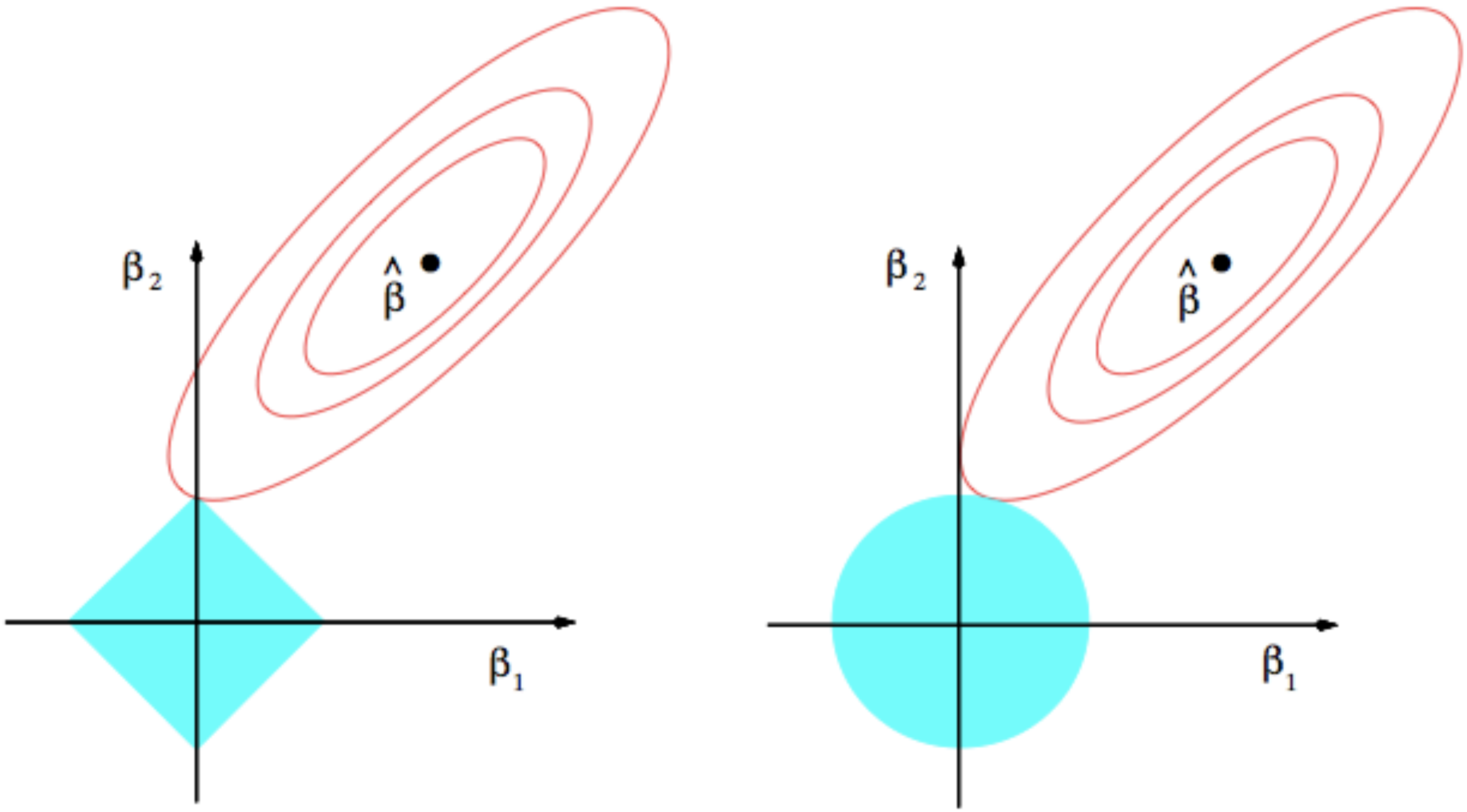


Over-fitting(과적합)

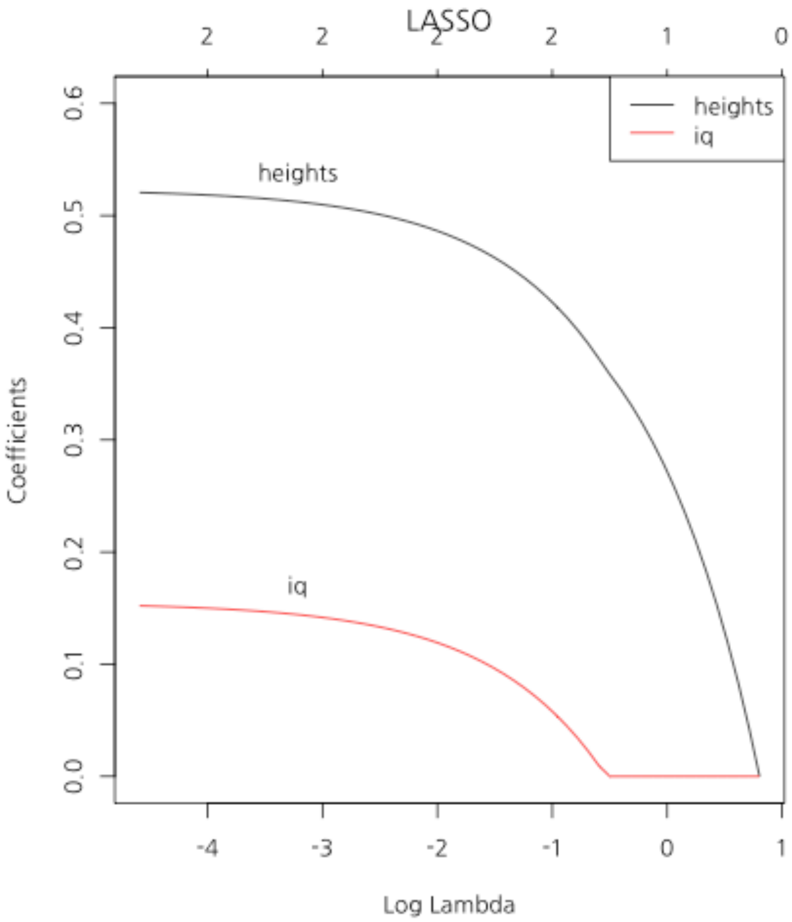
How to avoid Over-fitting

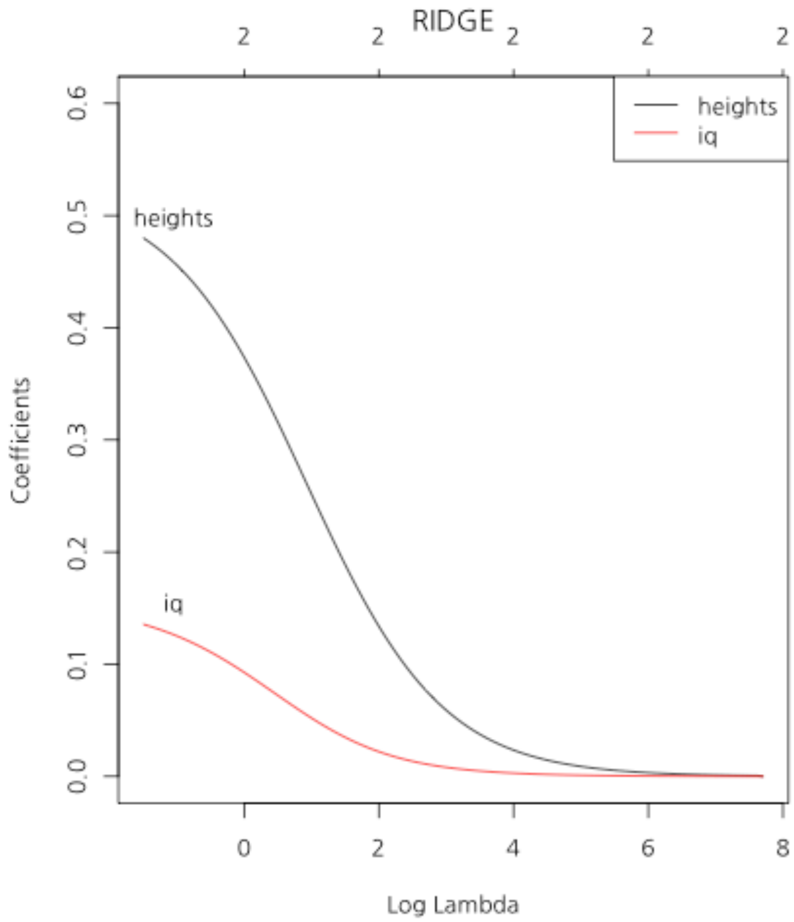
- Penality of Model Complexity (MSE 보정)
  - Regulization (Lasso, Ridge, Elastic Net)
- Bayesian
- Drop Out, Bagging, Feature Bagging

Lasso Vs Ridge



Lasso Vs Ridge





감정분석

Data

25,000 IMDB movie reviews 중에서 1,000개만

Training Vs Test = 7 Vs 3

Traing Set 과 Test Set 분리

```
fileName <- "data/IMDBmovie/labeledTrainData.tsv"
data <- read.csv(fileName, header=T, sep="\t", quote="")
nrow(data)
```

```
## [1] 25000
```

```
data <- data[1:1000, ]
```

Traing Set 과 Test Set 분할

```
totalNum <- 1:nrow(data)
set.seed(12345)
shuffledNum <- sample(totalNum, nrow(data), replace = F)
trainingNum <- shuffledNum[1:700]
testNum <- shuffledNum[701:1000]
data.train <- data[trainingNum, ]
data.test <- data[testNum, ]
```

Term-DocumentMatrix

```
library(tm)
```

```
corpus <- Corpus(VectorSource(data.train$review))
tdm.train <- TermDocumentMatrix(corpus,
                                control=list(stemming = T,
                                             tolower = T,
                                             removePunctuation = T,
                                             removeNumbers = T,
                                             stopwords=stopwords("SMART")))
```

주요 단어 10000개 사용

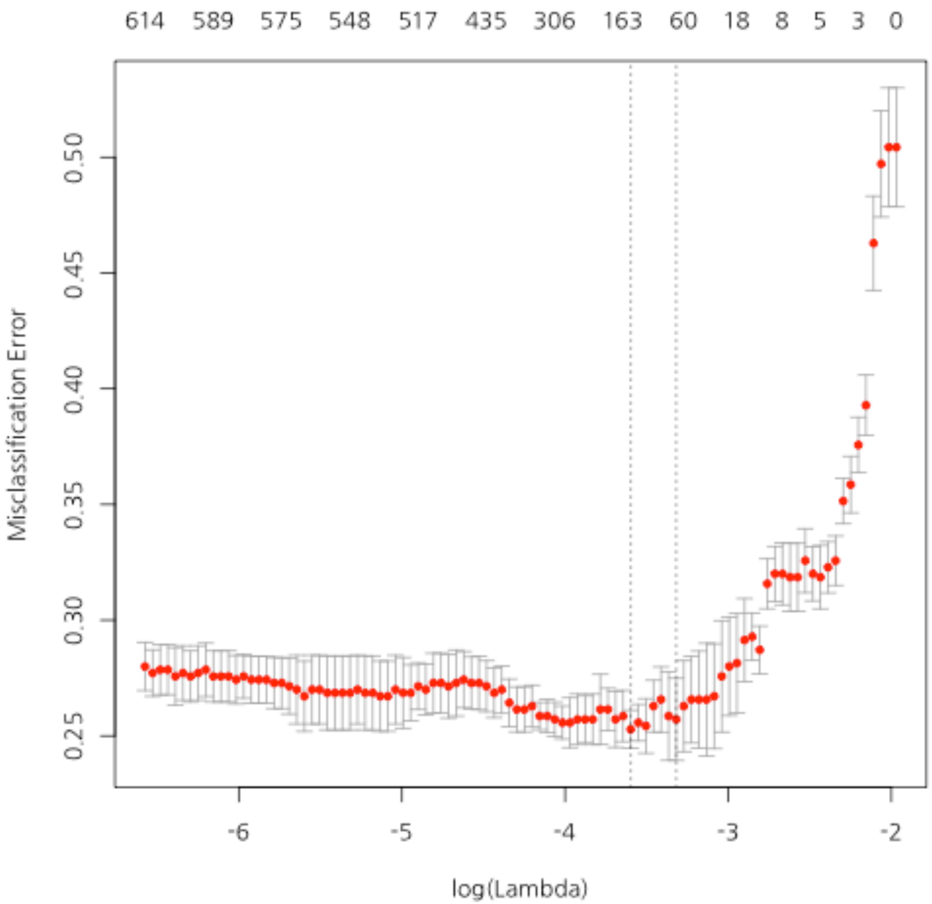
```r
library(slam)
word.count = as.array(rollup(tdm.train, 2))
word.order = order(word.count, decreasing = T)
freq.word = word.order[1 : 10000]
tdm.train <- tdm.train[freq.word, ]
```

LASSO Regression

```r
alpha <- 1
cv.lasso <- cv.glmnet(as.matrix(t(tdm.train)), data.train$sentiment,
                      type.measure = "class",
                      nfolds = 4,
                      family = "binomial",
                      alpha = alpha)
```

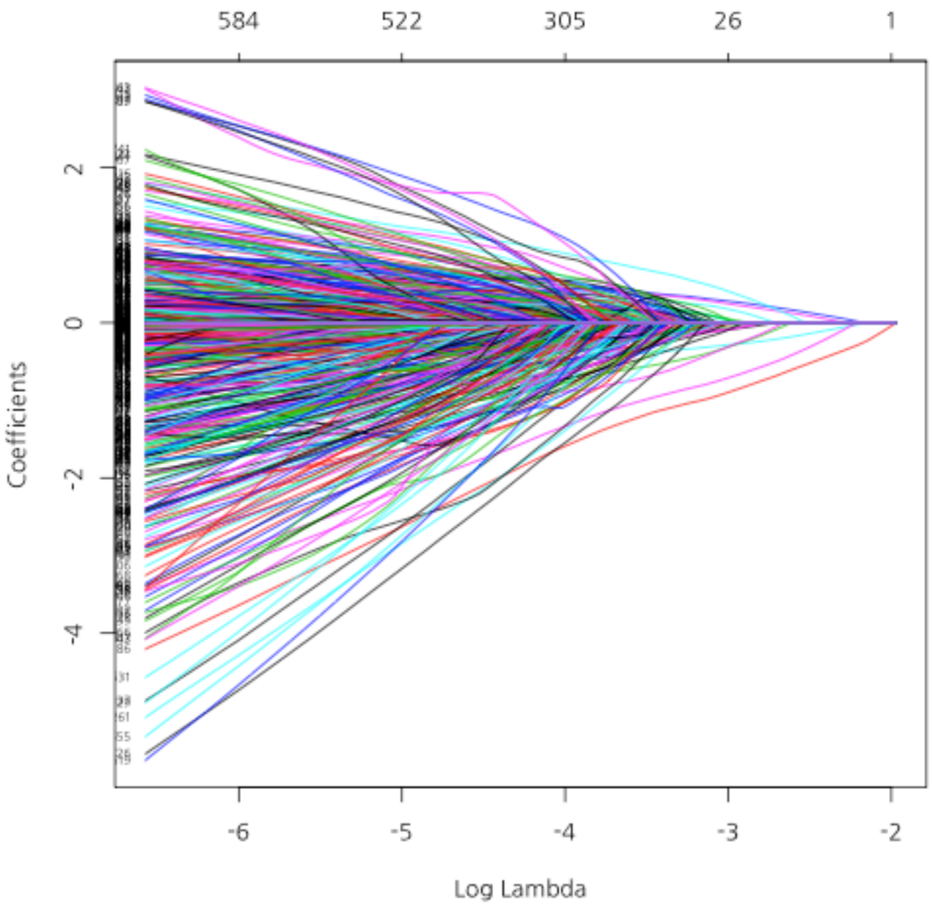
LASSO Regression

```
plot(cv.lasso)
```



LASSO Regression

```
plot(cv.lasso$glmnet.fit, "lambda", label=TRUE)
```

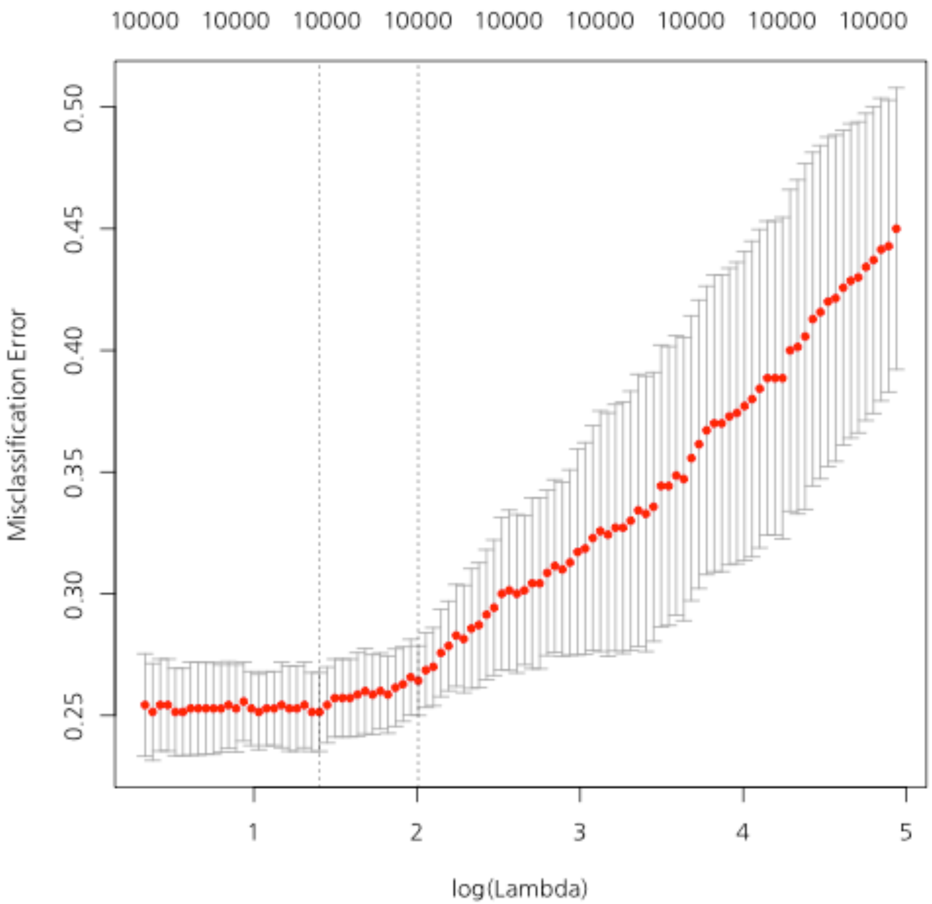


Ridge Regression

```r
alpha <- 0
cv.ridge <- cv.glmnet(as.matrix(t(tdm.train)), data.train$sentiment,
                      type.measure = "class",
                      nfolds = 4,
                      family = "binomial",
                      alpha = alpha)
```

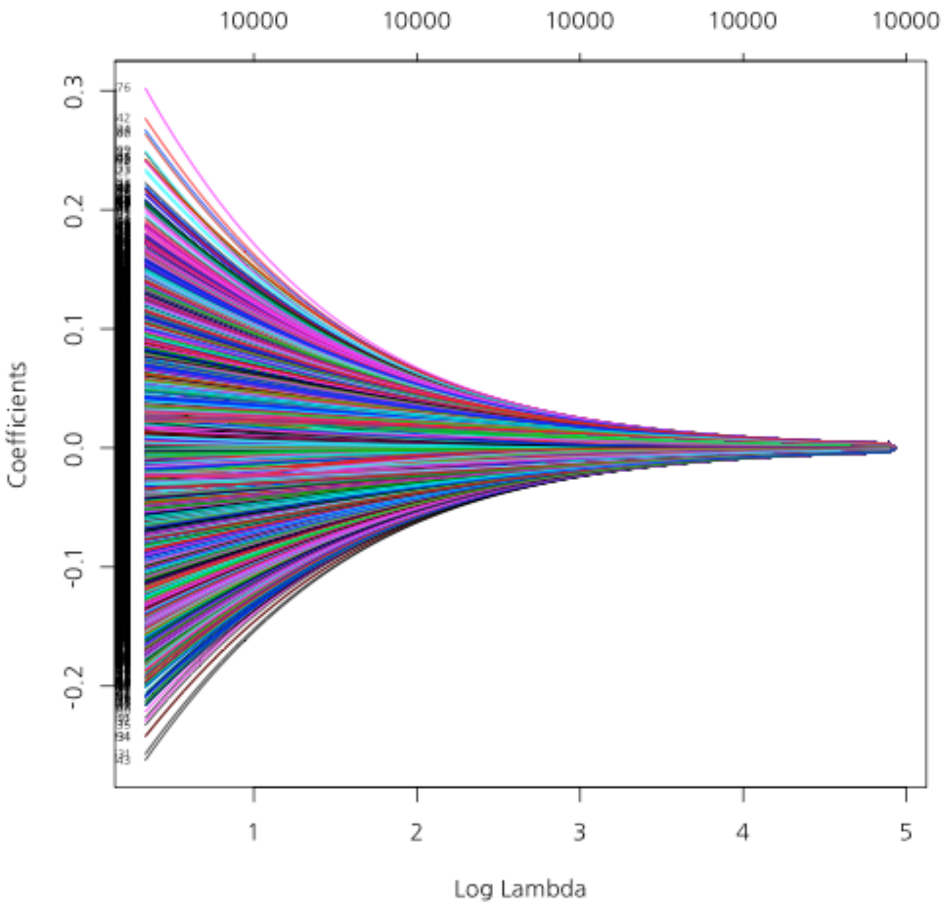
RIDGE Regression

```
plot(cv.ridge)
```



RIDGE Regression

```
plot(cv.ridge$glmnet.fit, "lambda", label=TRUE)
```

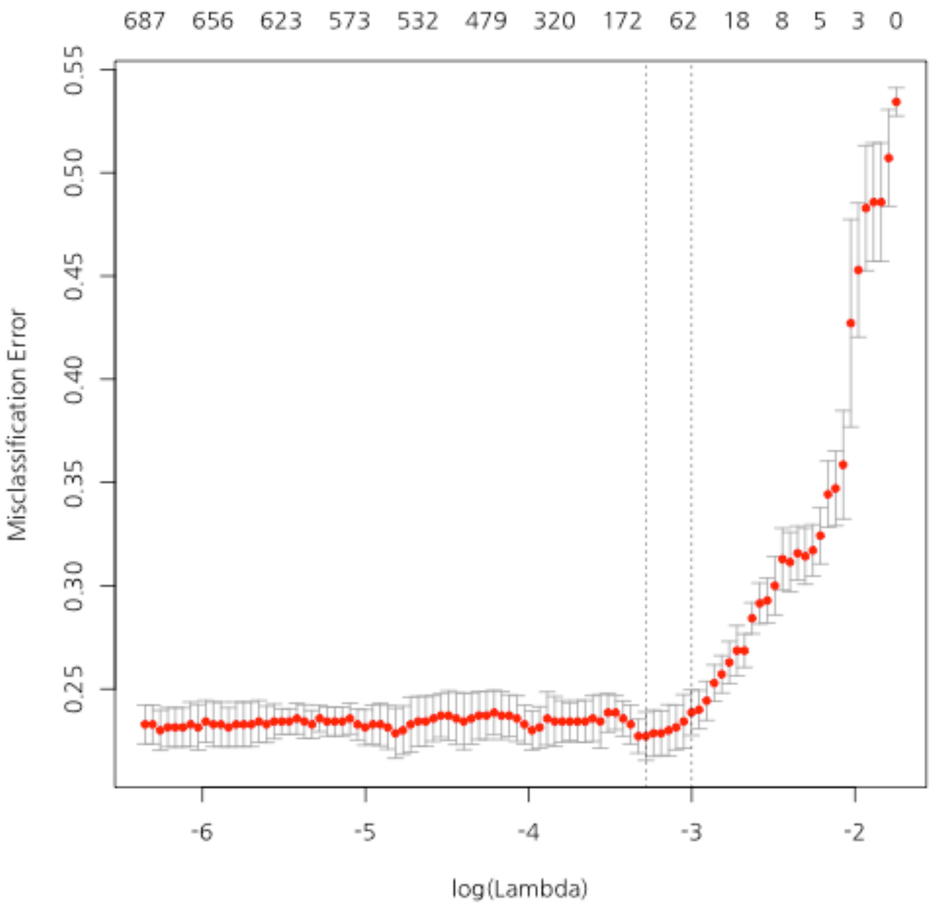


ElasticNet Regression

```
alpha <- .8
cv.elastic <- cv.glmnet(as.matrix(t(tdm.train)), data.train$sentiment,
                        type.measure = "class",
                        nfolds = 4,
                        family = "binomial",
                        alpha = alpha)
```

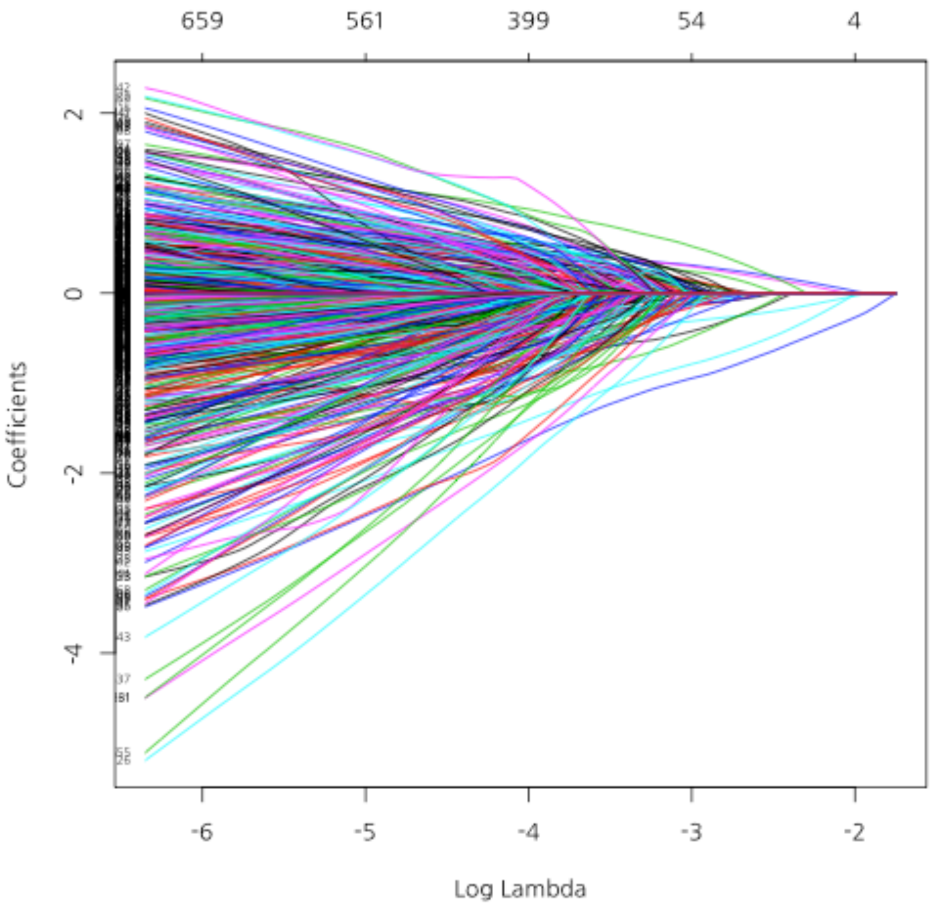
ElasticNet Regression

```
plot(cv.elastic)
```



ElasticNet Regression

```
plot(cv.elastic$glmnet.fit, "lambda", label=TRUE)
```



감정 단어 추출

```
coef.lasso <- coef(cv.lasso, s = "lambda.min")[,1]
coef.ridge <- coef(cv.ridge, s = "lambda.min")[,1]
coef.elastic <- coef(cv.elastic, s = "lambda.min")[,1]
```

감정 단어 추출

```
pos.lasso <- sort(coef.lasso[coef.lasso > 0])
neg.lasso <- sort(coef.lasso[coef.lasso < 0])

pos.ridge <- sort(coef.ridge[coef.ridge > 0])
neg.ridge <- sort(coef.ridge[coef.ridge < 0])

pos.elastic <- sort(coef.elastic[coef.elastic > 0])
neg.elastic <- sort(coef.elastic[coef.elastic < 0])
```

```
library(tm.plugin.sentiment)
```

```
score.lasso <- polarity(tdm.train, names(pos.lasso), names(neg.lasso))
score.ridge <- polarity(tdm.train, names(pos.elastic), names(neg.elastic))
score.elastic <- polarity(tdm.train, names(pos.elastic), names(neg.elastic))
```

CUT-POINT

```
findCutpoint(data.train$sentiment, score.lasso)
```

```
## [1] 0.09090909
```

```
findCutpoint(data.train$sentiment, score.ridge)
```

```
## [1] 0.3333333
```

```
findCutpoint(data.train$sentiment, score.elastic)
```

```
## [1] 0.3333333
```

```
cut.lasso <- findCutpoint(data.train$sentiment, score.lasso)
cut.ridge <- findCutpoint(data.train$sentiment, score.ridge)
cut.elastic <- findCutpoint(data.train$sentiment, score.elastic)
```

Test Set

```
corpus <- Corpus(VectorSource(data.test$review))

tdm.test <- TermDocumentMatrix(corpus,
                               control=list(dictionary = Terms(tdm.train),
                                            stemming = T,
                                            tolower = T,
                                            removePunctuation = T,
                                            removeNumbers = T,
                                            stopwords=stopwords("SMART")))
```

Test Set

```
score.lasso <- polarity(tdm.test, names(pos.lasso), names(neg.lasso))
score.ridge <- polarity(tdm.test, names(pos.elastic), names(neg.elastic))
score.elastic <- polarity(tdm.test, names(pos.elastic), names(neg.elastic))
```

Test Set

```
library(caret)
```

```
score.lasso.b <- rep(0, length(score.lasso))
score.lasso.b[score.lasso >= cut.lasso] <- 1
confusionMatrix(score.lasso.b, data.test$sentiment)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 118  30
##          1  49 103
##
##                Accuracy : 0.7367
##                  95% CI : (0.683, 0.7856)
##     No Information Rate : 0.5567
##     P-Value [Acc > NIR] : 8.952e-11
##
```

Test Set

```
score.ridge.b <- rep(0, length(score.ridge))
score.ridge.b[score.ridge >= cut.ridge] <- 1
confusionMatrix(score.ridge.b, data.test$sentiment)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 130  42
##          1  37  91
##
##                Accuracy : 0.7367
##                  95% CI : (0.683, 0.7856)
##     No Information Rate : 0.5567
##     P-Value [Acc > NIR] : 8.952e-11
##
##                   Kappa : 0.4644
##  Mcnemar's Test P-Value : 0.6527
##
```

Test Set

```
score.elastic.b <- rep(0, length(score.elastic))
score.elastic.b[score.elastic >= cut.elastic] <- 1
confusionMatrix(score.elastic.b, data.test$sentiment)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 130  42
##          1  37  91
##
##                Accuracy : 0.7367
##                  95% CI : (0.683, 0.7856)
##     No Information Rate : 0.5567
##     P-Value [Acc > NIR] : 8.952e-11
##
##                   Kappa : 0.4644
##  Mcnemar's Test P-Value : 0.6527
##
```

glmnet 활용

```
score.lasso <- predict(cv.lasso, as.matrix(t(tdm.train)), s = "lambda.min")
score.ridge <- predict(cv.ridge, as.matrix(t(tdm.train)), s = "lambda.min")
score.elastic <- predict(cv.elastic, as.matrix(t(tdm.train)), s = "lambda.min")
```

```
findCutpoint(data.train$sentiment, score.lasso)
```

```
## [1] 0.1426384
```

```
findCutpoint(data.train$sentiment, score.ridge)
```

```
## [1] -0.03759213
```

```
findCutpoint(data.train$sentiment, score.elastic)
```

```
## [1] 0.1195488
```

glmnet 활용

```
score.lasso <- predict(cv.lasso, as.matrix(t(tdm.test)), s = "lambda.min")
score.ridge <- predict(cv.ridge, as.matrix(t(tdm.test)), s = "lambda.min")
score.elastic <- predict(cv.elastic, as.matrix(t(tdm.test)), s = "lambda.min")
```

Test Set

```
score.lasso.b <- rep(0, length(score.lasso))
score.lasso.b[score.lasso >= cut.lasso] <- 1
confusionMatrix(score.lasso.b, data.test$sentiment)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 148 110
##          1  19  23
##
##                Accuracy : 0.57
```

Test Set

```
score.ridge.b <- rep(0, length(score.ridge))
score.ridge.b[score.ridge >= cut.ridge] <- 1
confusionMatrix(score.ridge.b, data.test$sentiment)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 85 71
##          1 82 62
##
##                Accuracy : 0.49
##                  95% CI : (0.4321, 0.5481)
##     No Information Rate : 0.5567
##     P-Value [Acc > NIR] : 0.9912
##
##                   Kappa : -0.0246
##  Mcnemar's Test P-Value : 0.4188
##
```

Test Set

```
score.elastic.b <- rep(0, length(score.elastic))
score.elastic.b[score.elastic >= cut.elastic] <- 1
confusionMatrix(score.elastic.b, data.test$sentiment)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 147 110
##          1  20  23
##
##                Accuracy : 0.5667
##                  95% CI : (0.5085, 0.6235)
##     No Information Rate : 0.5567
##     P-Value [Acc > NIR] : 0.3865
##
##                   Kappa : 0.0571
##  Mcnemar's Test P-Value : 5.912e-15
##
```