



Sentiment Analysis

감정사전 & 감정점수 만들기

김형준

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



퀀트랩 소개

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 이해하기
- 회귀분석 이해하기
- 모형평가 이해하기

왜 감정분석을 하는가?

설문지의 단점

- 1) 조사 비용 발생
- 2) 미리 정해진 문항만 측정 가능
- 3) 사회적 바람직성 등 편향 발생

감정분석

텍스트에서 감정 단어를 추출하여 점수화

1) 기계 학습 (Machine Learning)

2) 단어 사전 기반

사전 기반 분석

장점

- 사용하기 간편

단점

- 주제에 따라 사전이 달라 짐
- 동음이의어 처리 힘들 e.g) bank

기계학습 기반 분석

장점

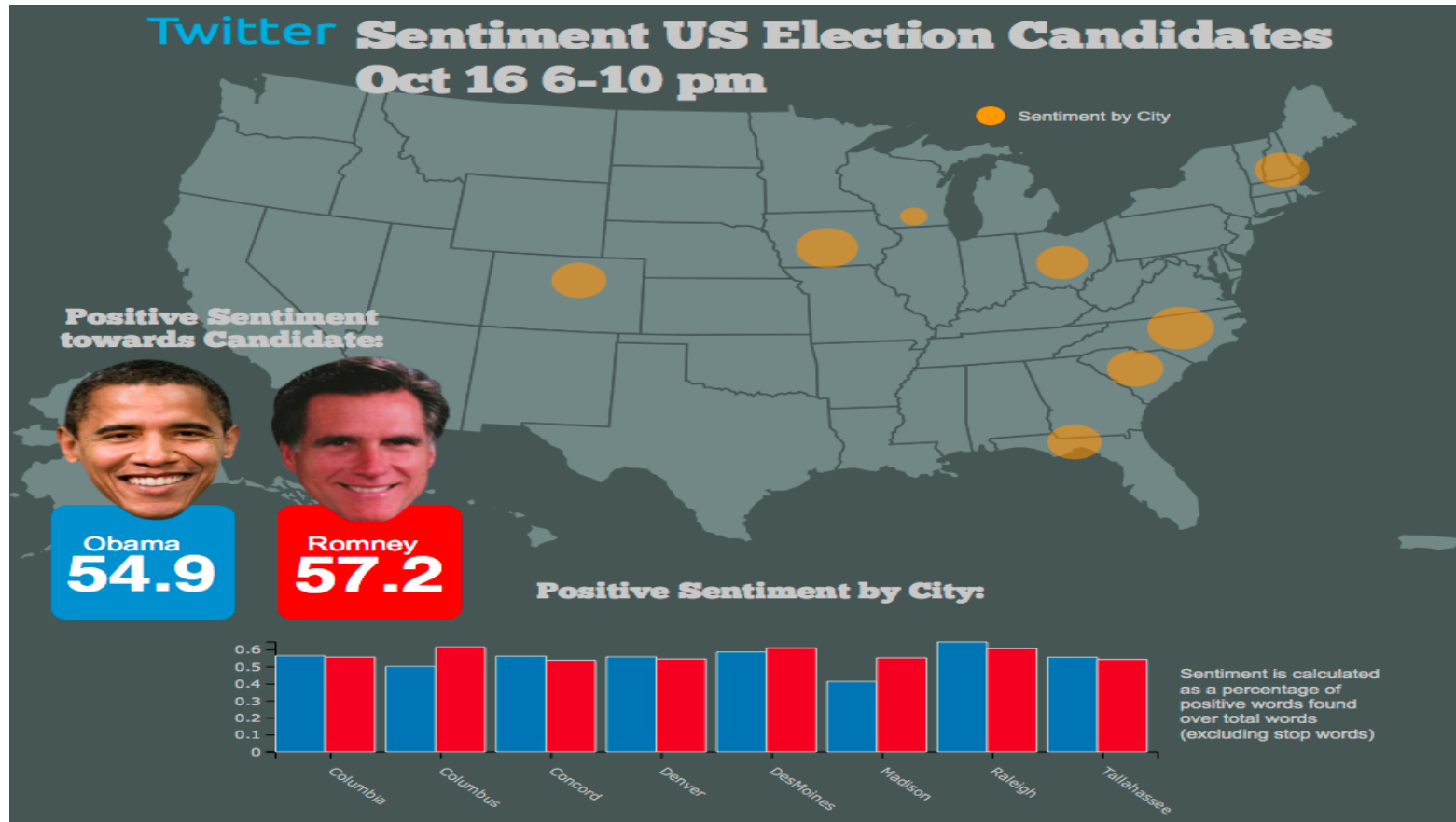
- 높은 정확도

단점

- Over-fitting 해결
- 많은 데이터 필요

예) 나이브 베이즈 / 최대 엔트로피 / 서포트지지벡터머신 /
랜덤 포레스트 / 토픽 모델

감정 분석 예시



감정 분석 예시



사전 지식

예측이란 무엇?

자기자신(Y) : Y가 변화하는 추세

다른변수(X,Y) : X가 Y를 예측

- 키로 몸무게를 예측!

- 키로 성적을 예측?

예측이 잘 되려면 서로 상관(관련성)이 높아야 함

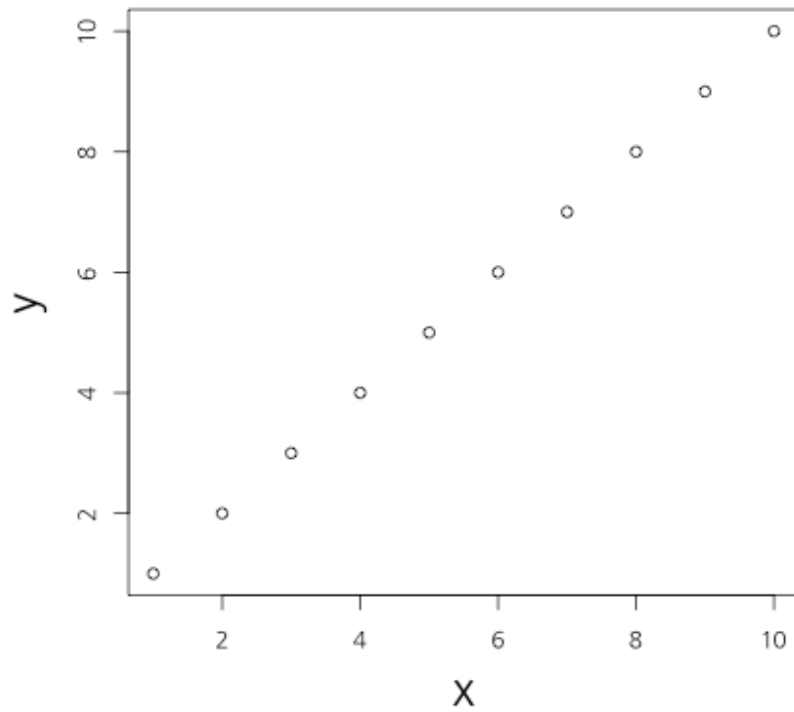
-> 감정단어로 영화 평점을 예측

회귀분석(선형(직선) 모형)

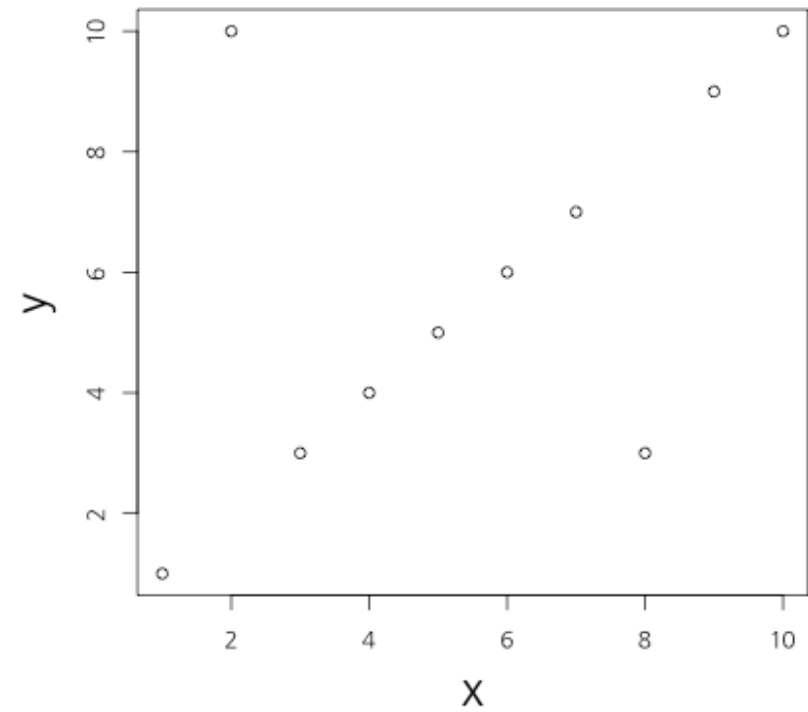
예시

- 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 선형적으로 증가(혹은 감소)하는 정도

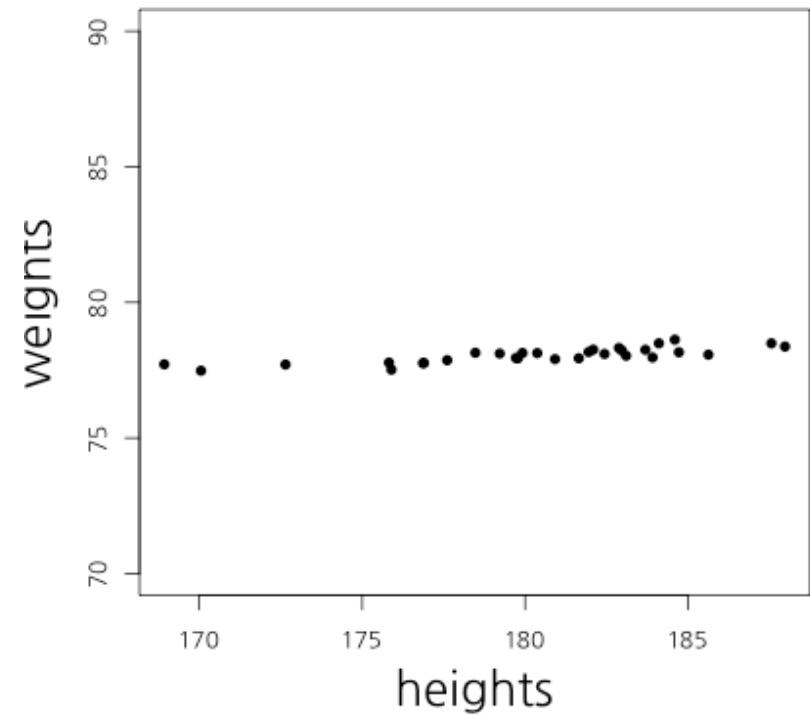
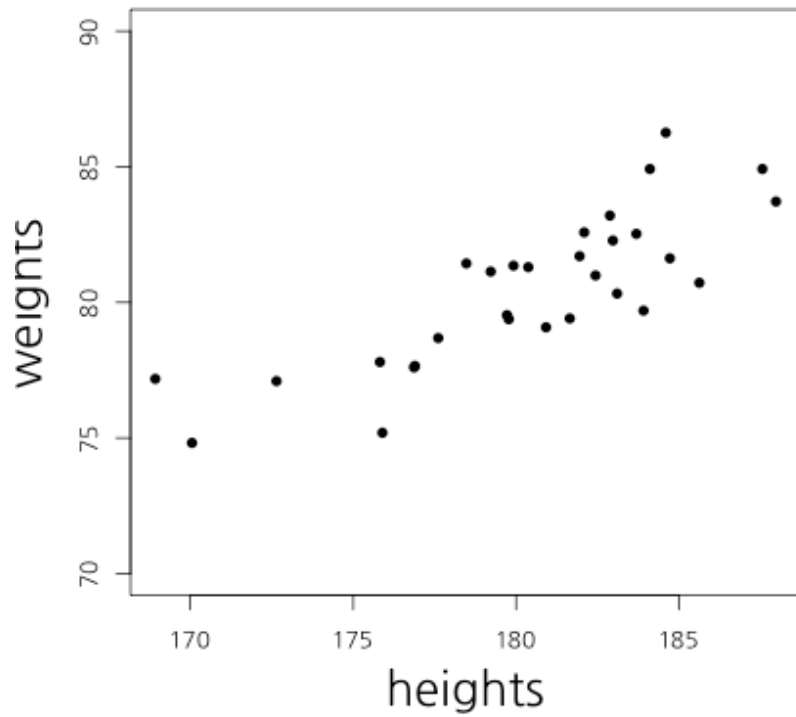
scale

키가 만약 cm라면, 키가 1cm 증가하면 몸무게는 1kg증가

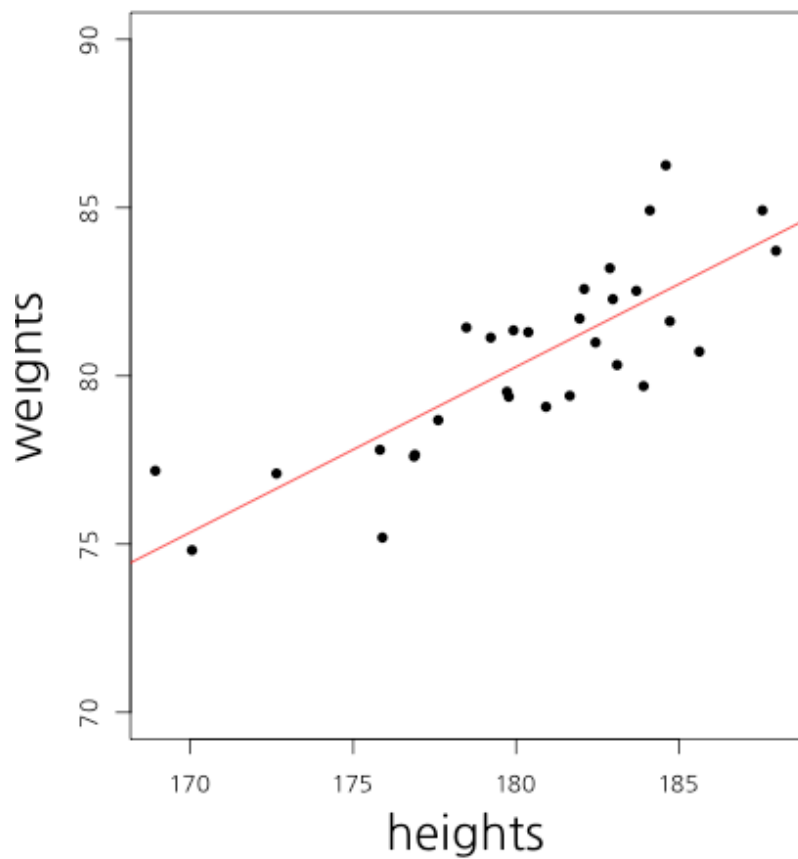
키가 만약 mm라면, 키가 1mm 증가하면 몸무게는 0.1kg 증가

-> 표준화해야 한다

둘 중 무엇이 상관이 더 클까요?



상관관계 및 회귀분석

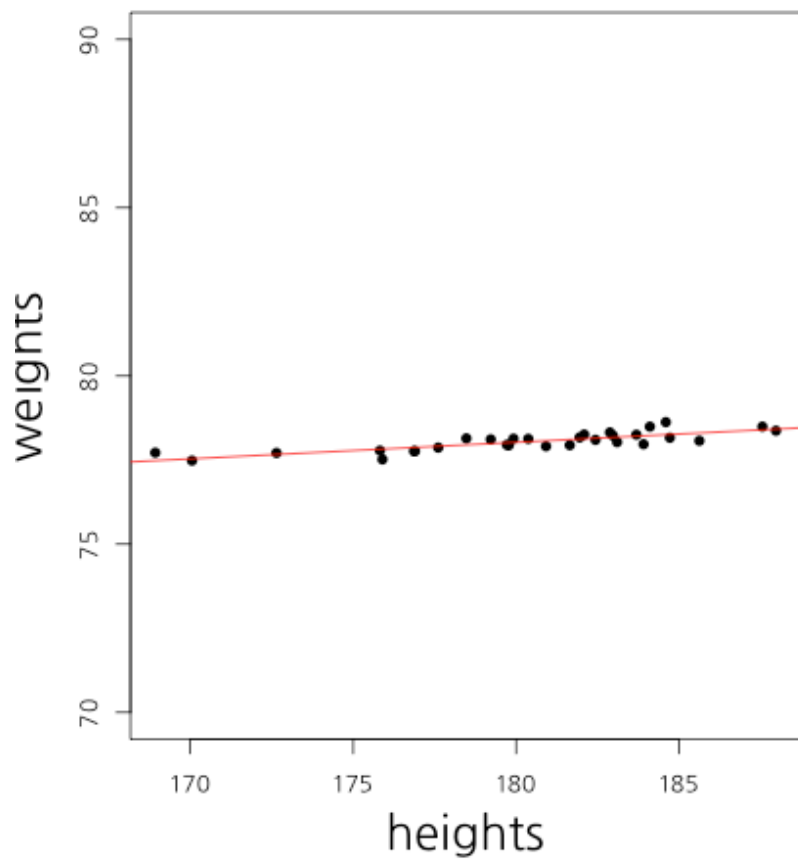


	ESTIMATE	STD. ERROR	T VALUE	PR(> T)
(Intercept)	-8.29	11.74	-0.71	0.49
heights	0.49	0.07	7.56	0.00

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

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


상관관계 및 회귀분석

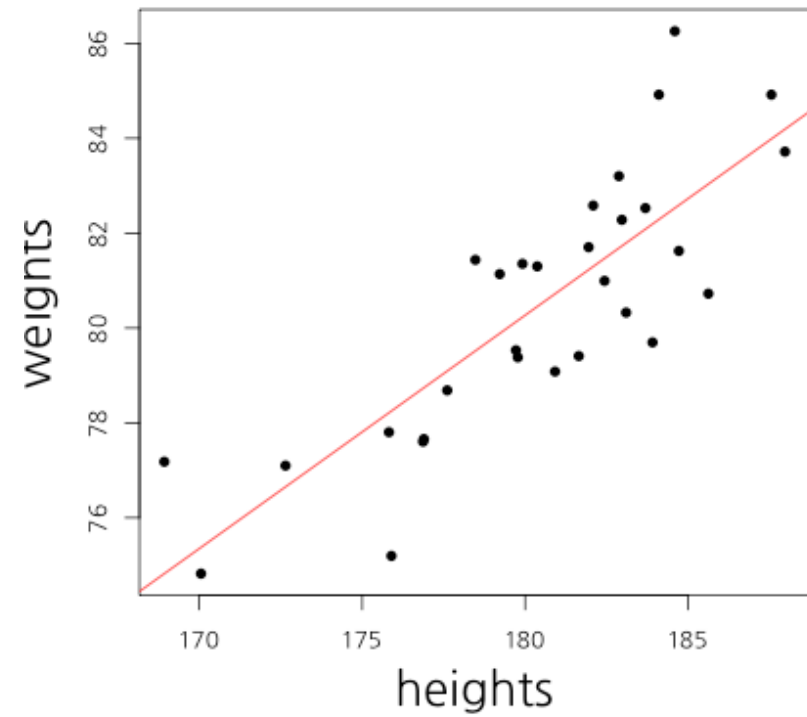
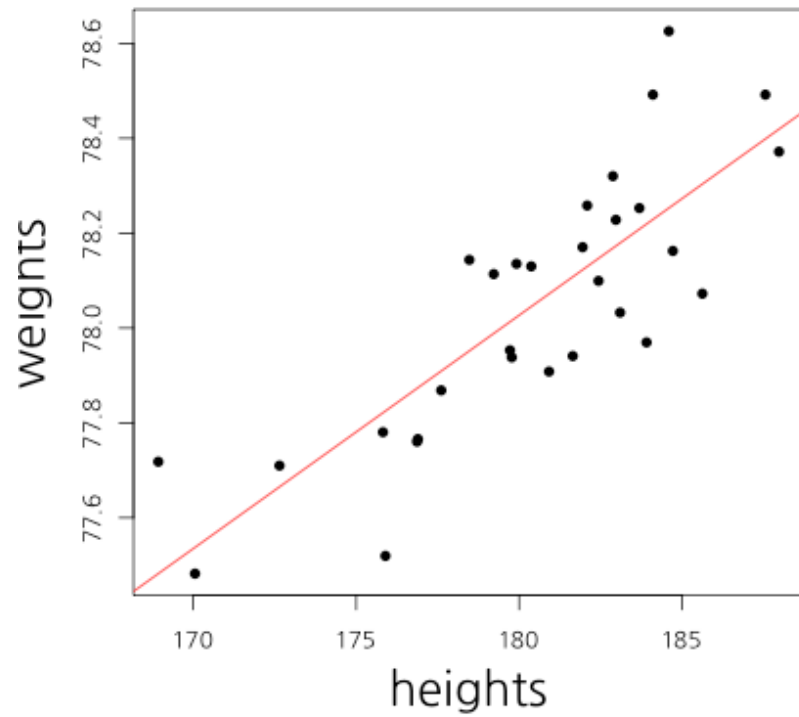


	ESTIMATE	STD. ERROR	T VALUE	PR(> T)
(Intercept)	69.17	1.17	58.93	0.00
heights	0.05	0.01	7.56	0.00

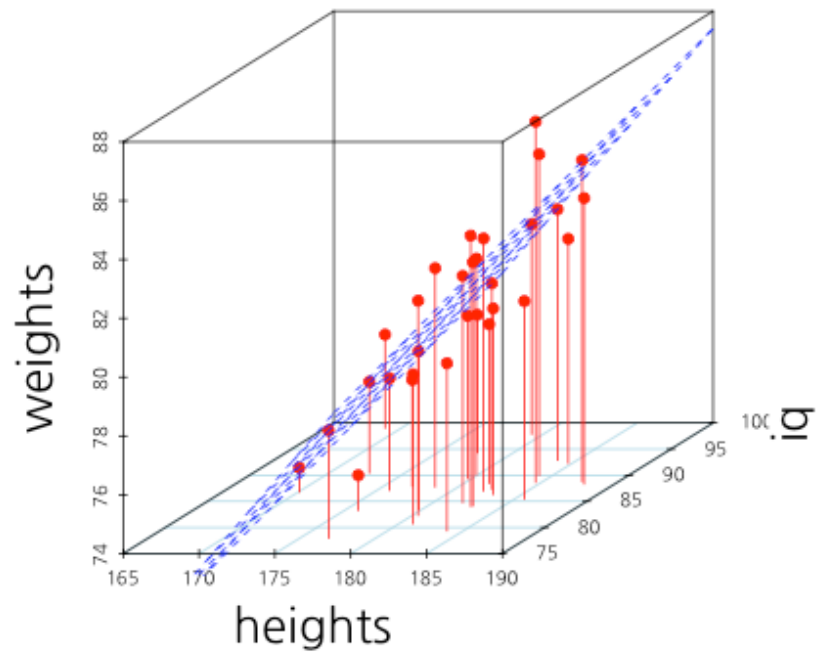
```
cor(weights, heights)
```

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

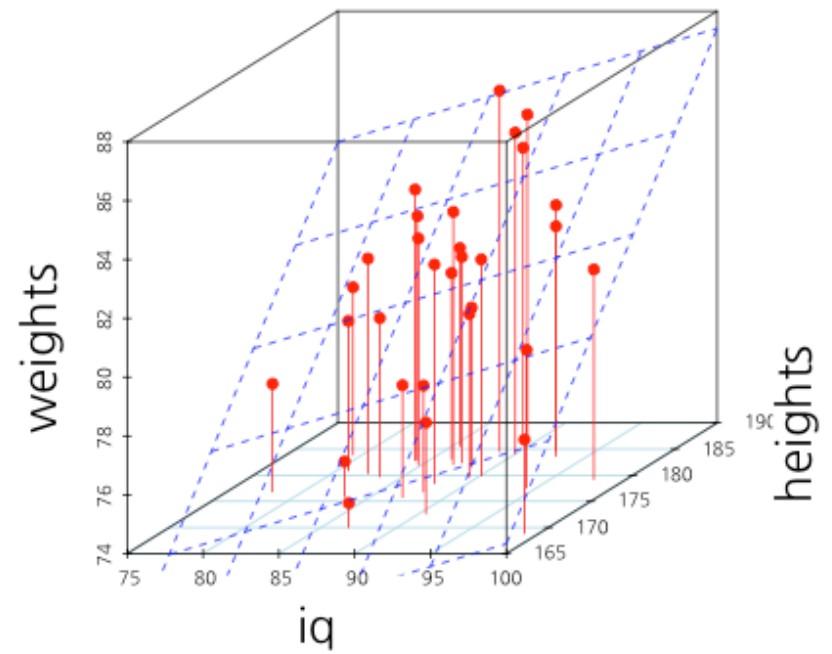
상관관계 및 회귀분석



X가 2개라면?



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



```
## [1] 0.09818667
```

다중회귀분석

	ESTIMATE	STD. ERROR	T VALUE	PR(> T)
(Intercept)	-27.49	12.81	-2.15	0.04
iq	0.15	0.06	2.68	0.01
heights	0.52	0.06	8.72	0.00

예측력

MSE(Mean of Square Error)

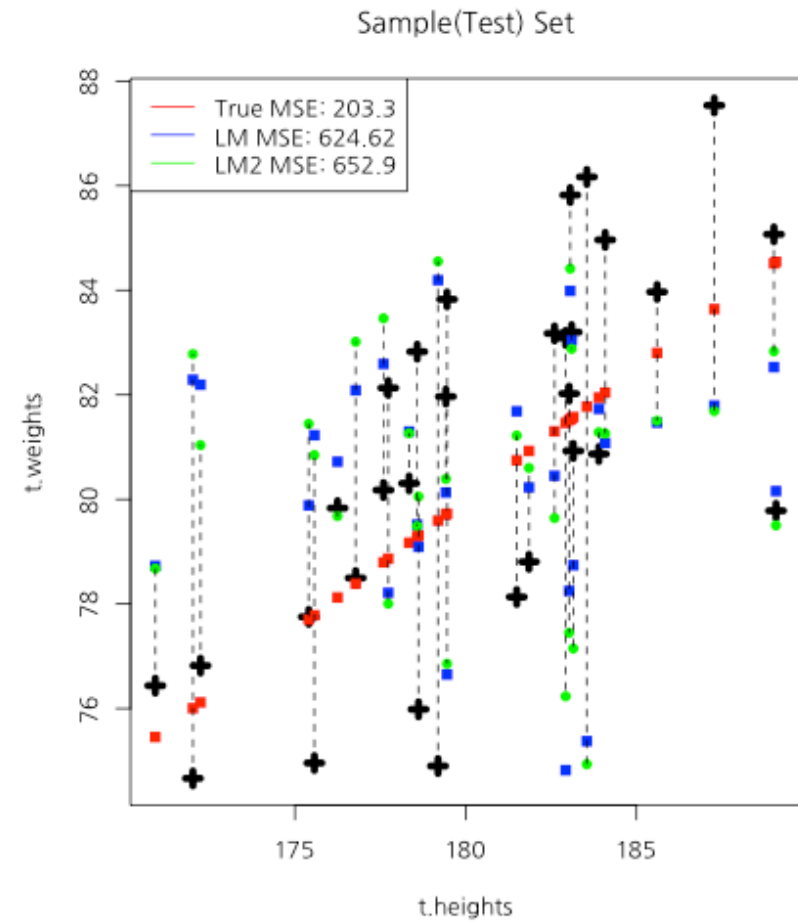
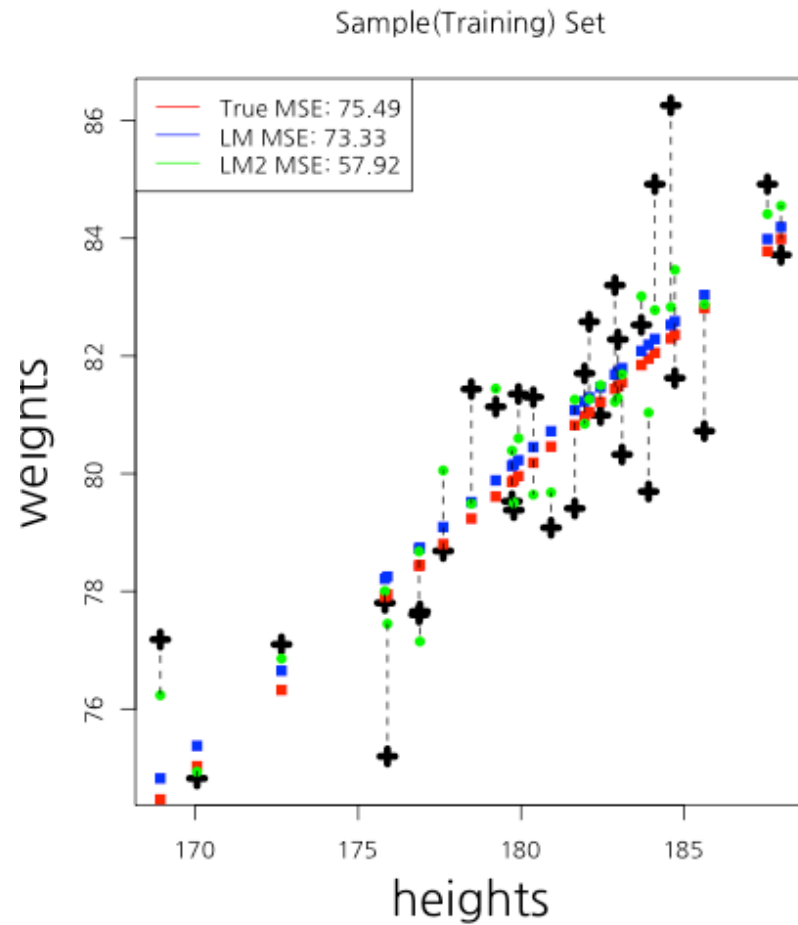
$$MSE = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

정확도(Accracy)

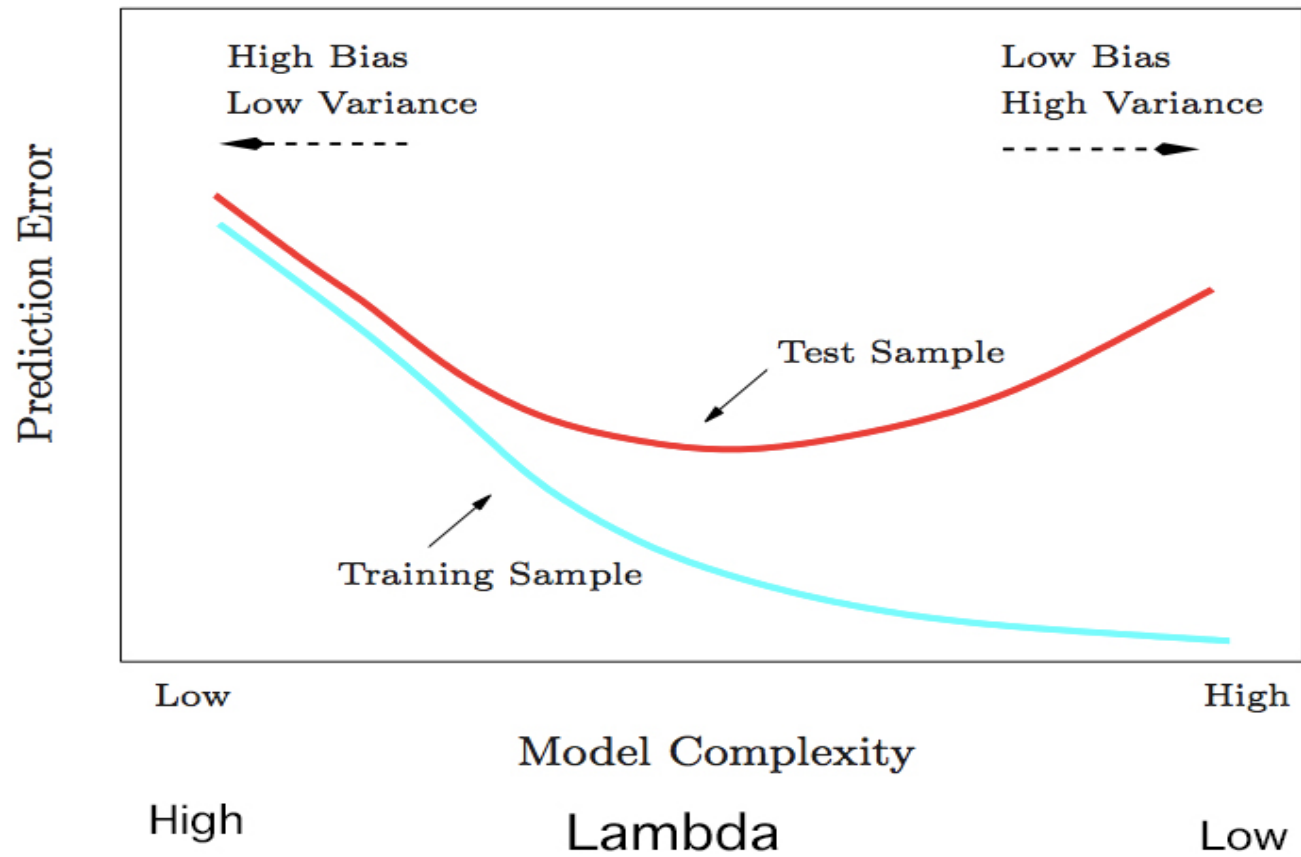
		실제 라벨	
		긍정 문서	부정 문서
모형이 예측한 라벨	긍정 문서	True Positive	False Positive
	부정 문서	False Negatives	True Negatives

$$\text{정확도} = (TP + TN) / (TP + FP + TN + FN)$$

Traning Vs Test



Over-fitting

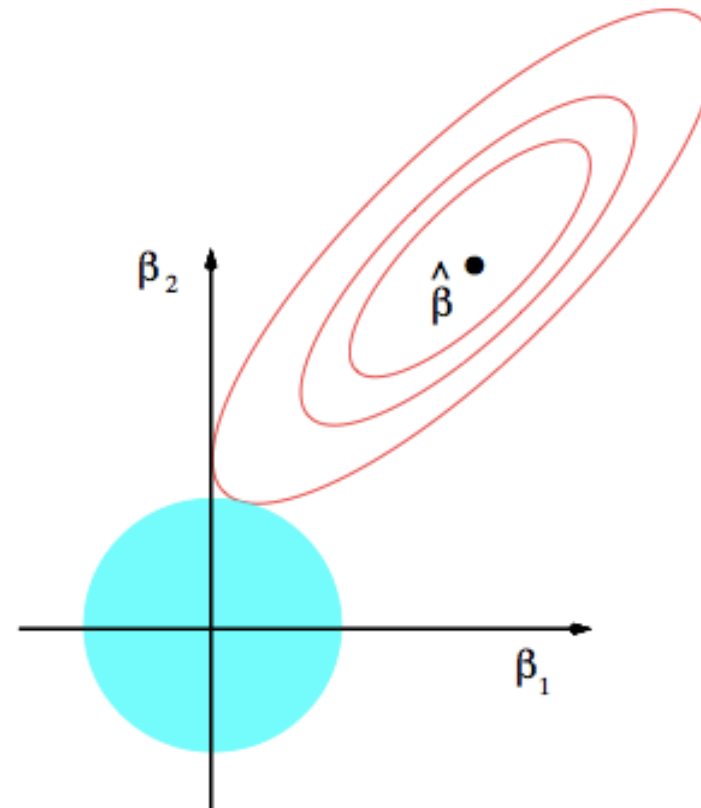
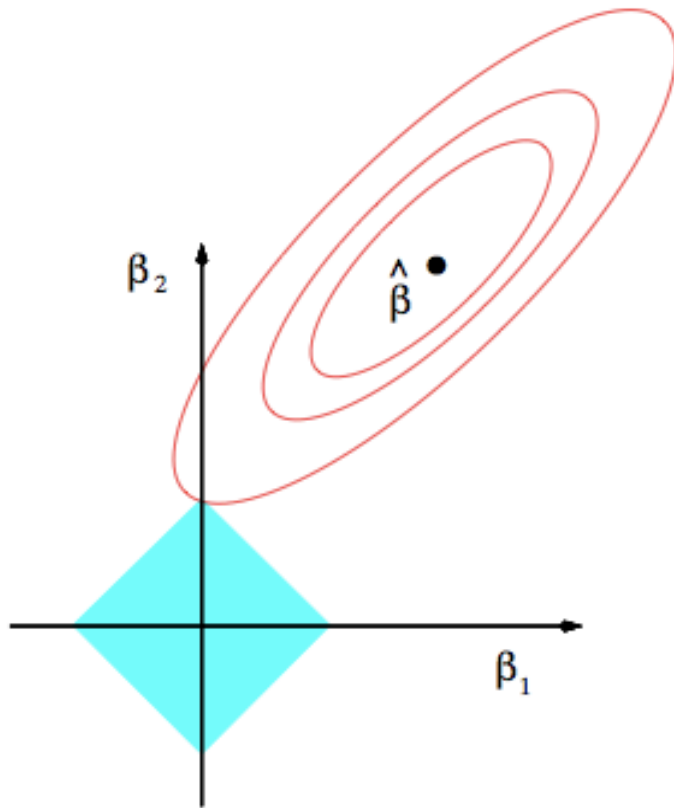


Over-fitting(과적합)

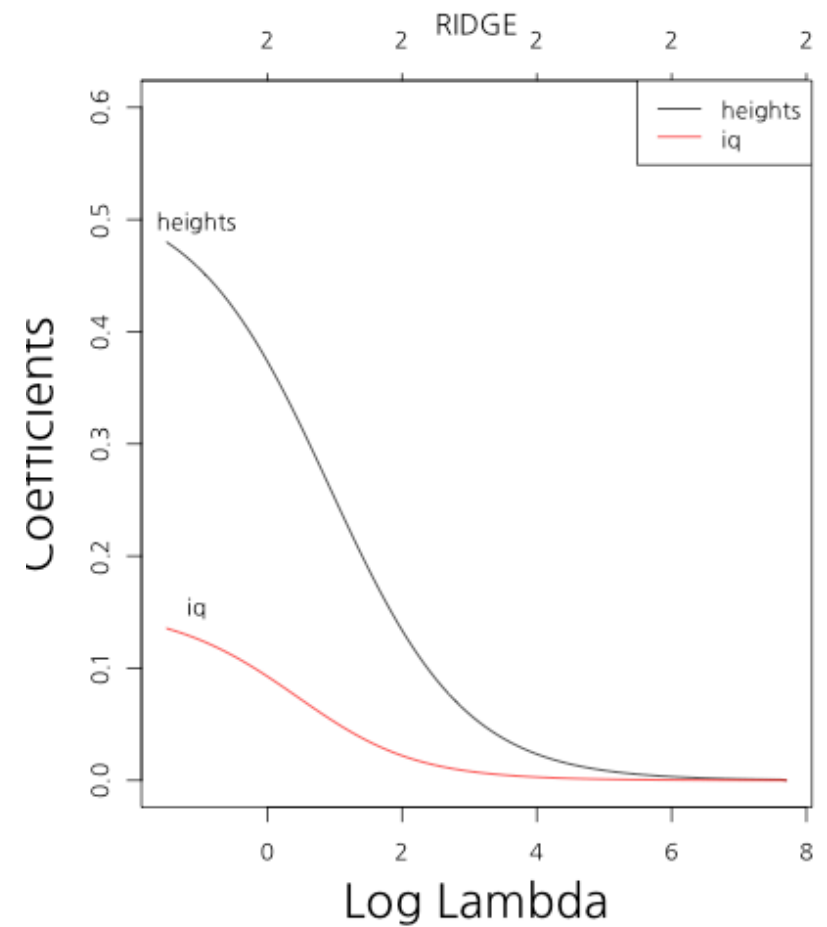
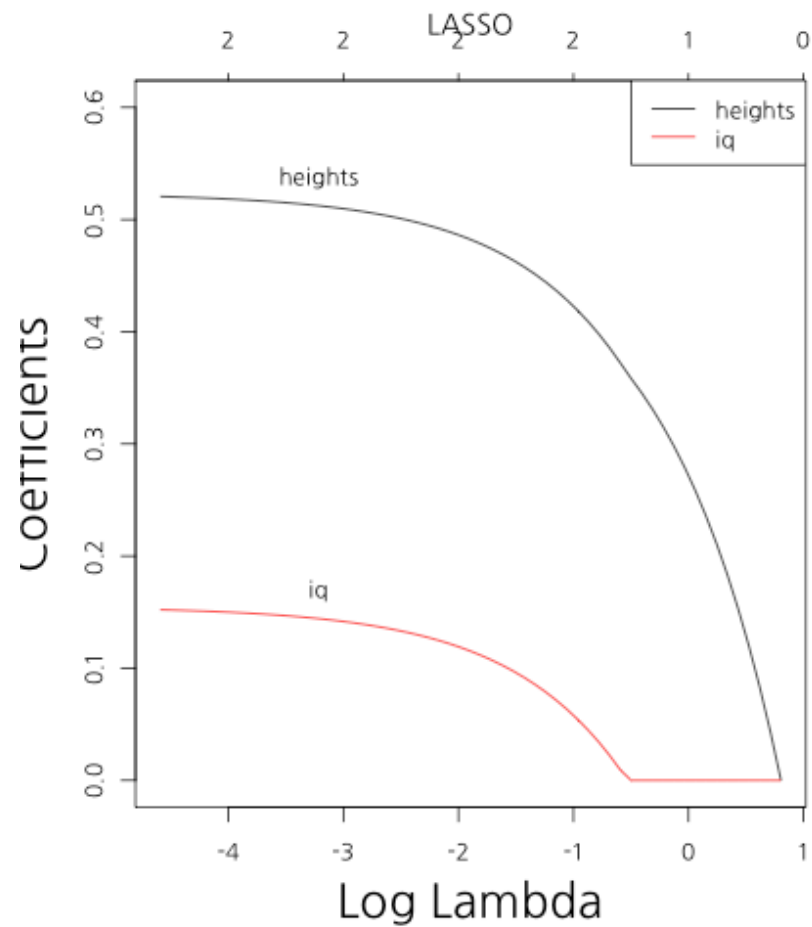
How to avoid Over-fitting

- Penalty 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

Training Set 과 Test Set 분할

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

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

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

Training 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(tolower = T,  
                                              removePunctuation = T,  
                                              removeNumbers = T,  
                                              stopwords=stopwords("SMART")))
```

주요 단어 10000개 사용

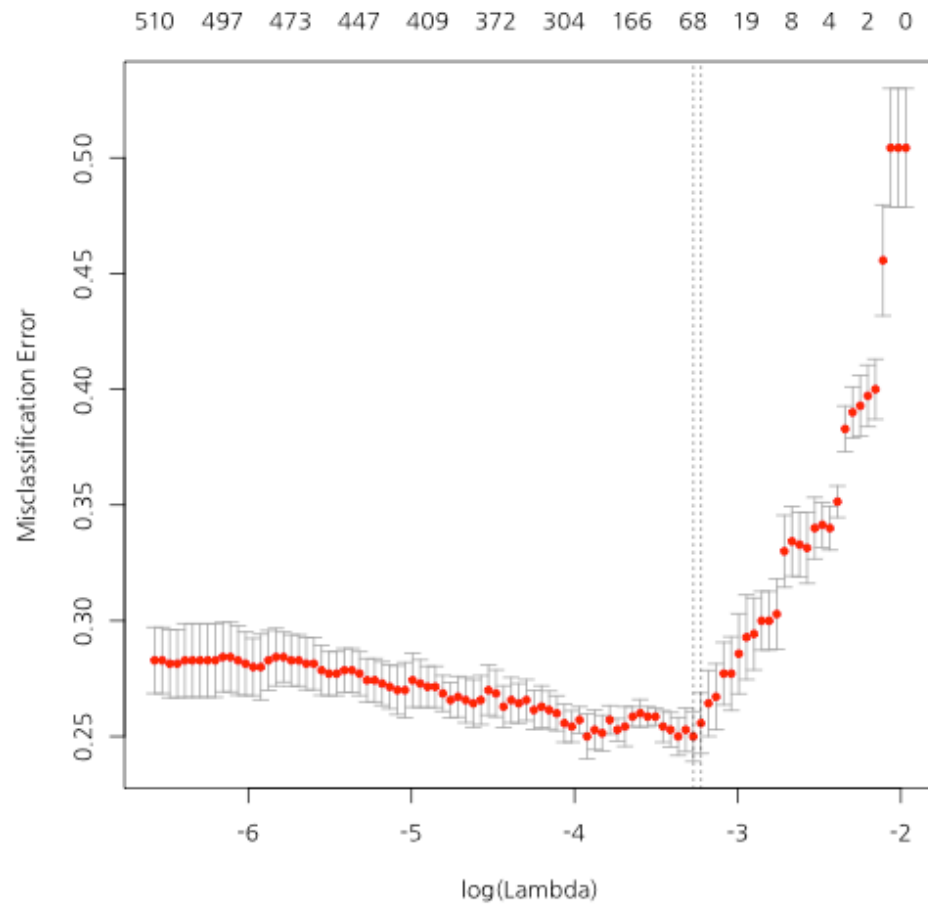
```
library(slam)
word.count = as.array(rollup(tdm.train, 2))
word.order = order(word.count, decreasing = T)
freq.word = word.order[1 : 10000]
tdm.train <- tdm.train[freq.word, ]
```

LASSO Regression

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

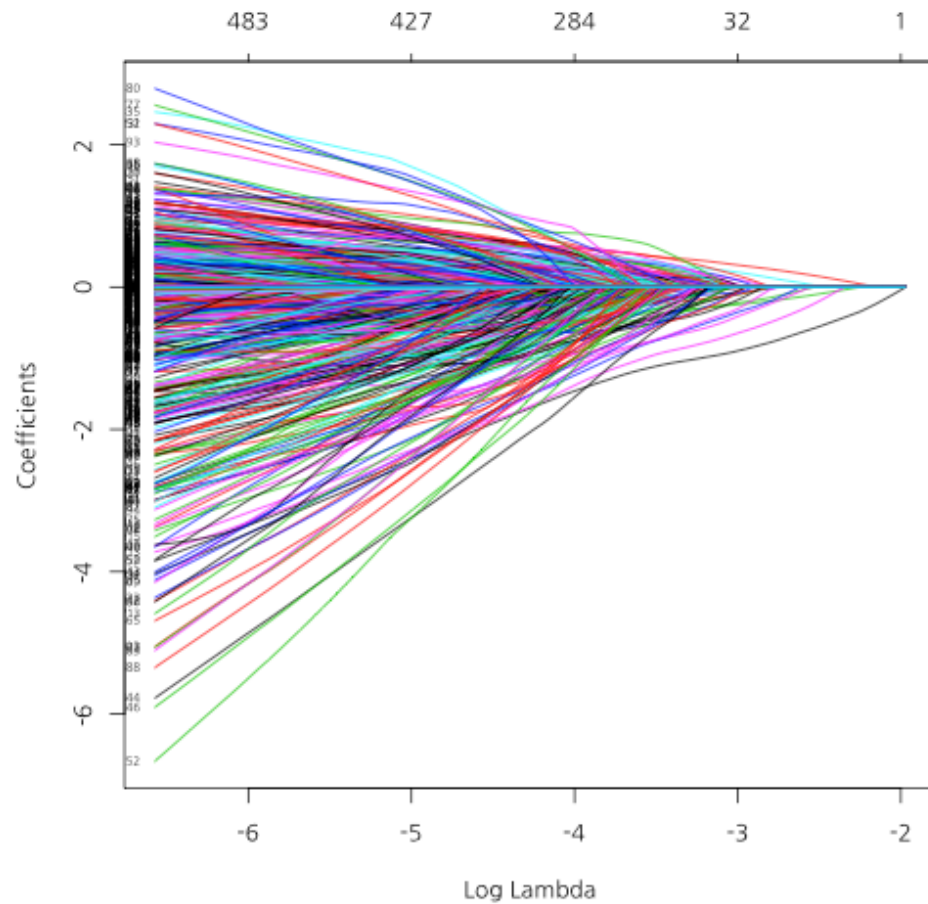

LASSO Regression

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



LASSO Regression

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

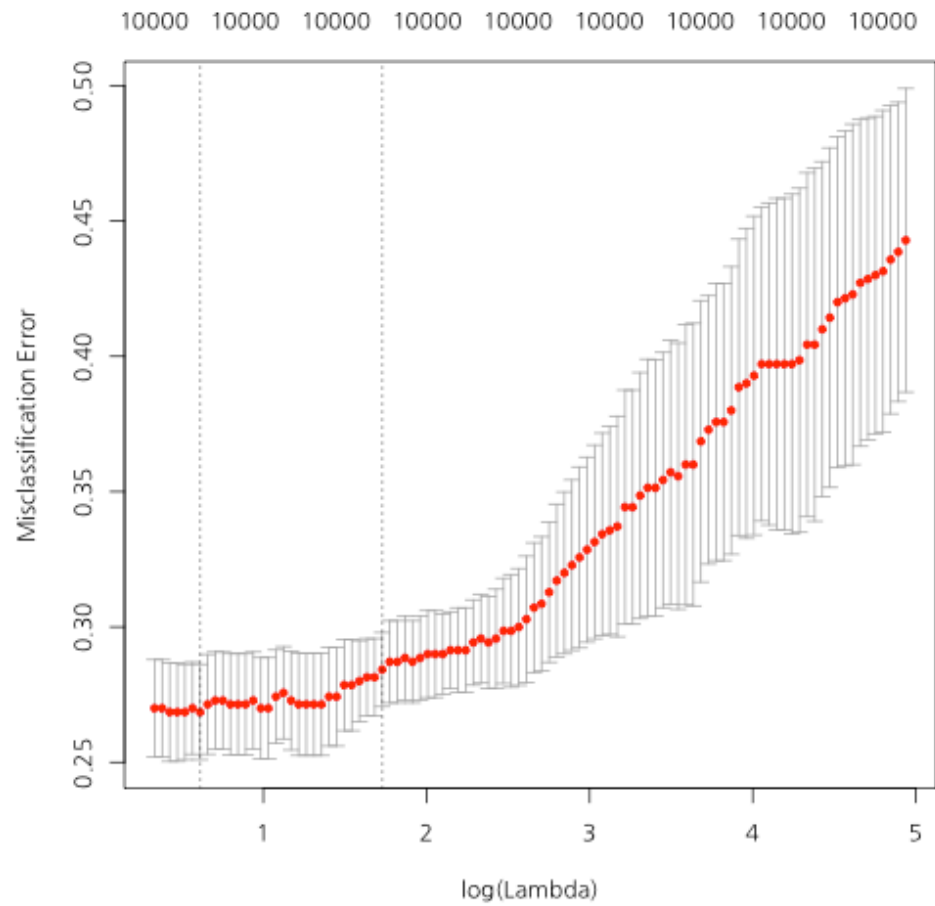


Ridge Regression

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

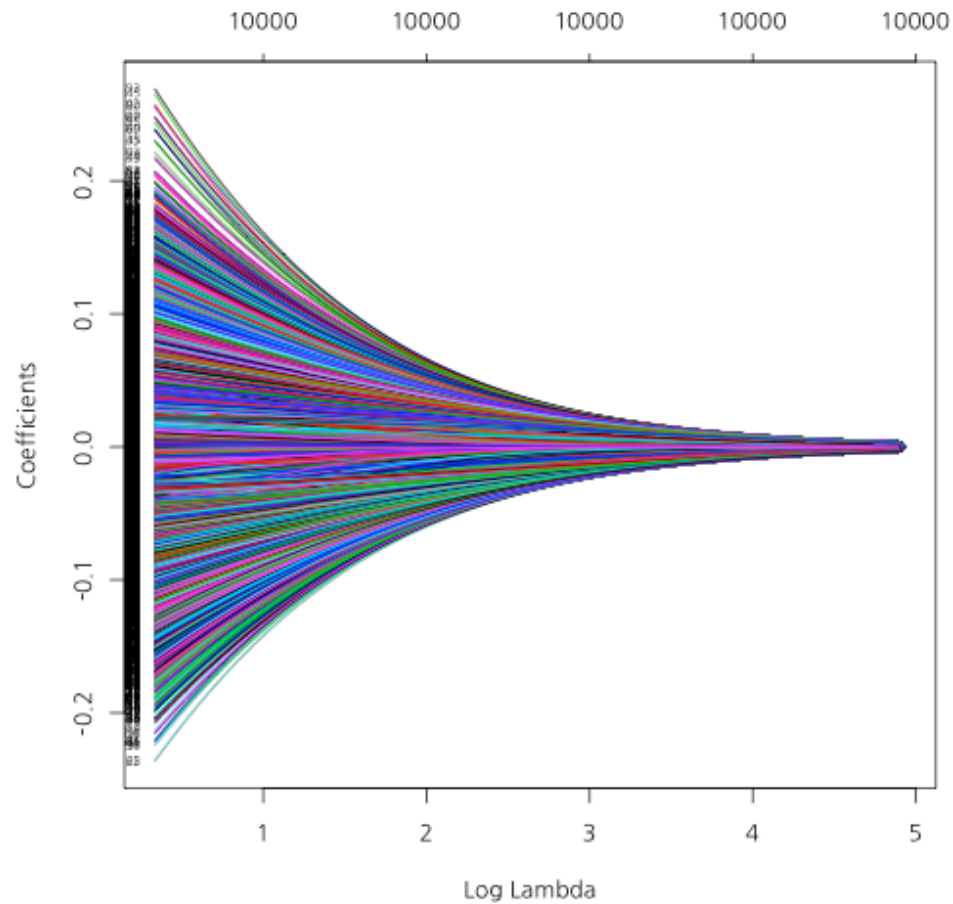
RIDGE Regression

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



RIDGE Regression

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

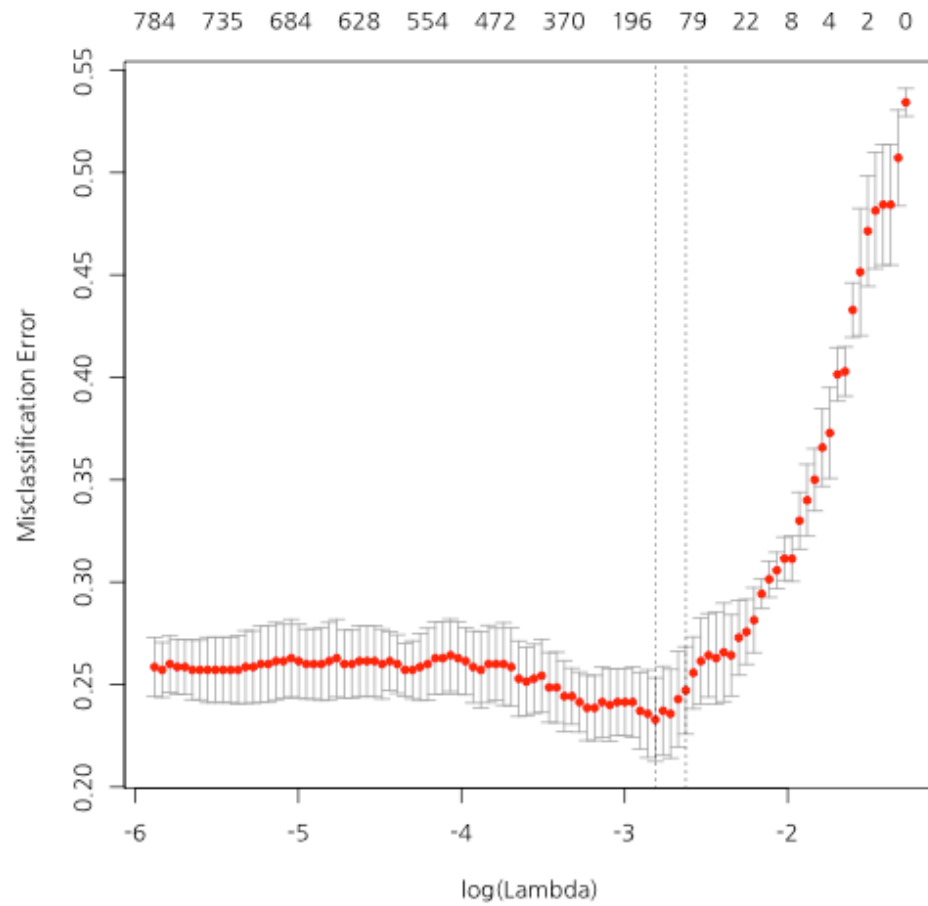


ElasticNet Regression

```
alpha <- .5
cv.elastic <- cv.glmnet(as.matrix(t(tdm.train)), data.train$sentiment,
                        type.measure = "class",
                        nfolds = 4,
                        family = "binomial",
                        alpha = alpha)
```

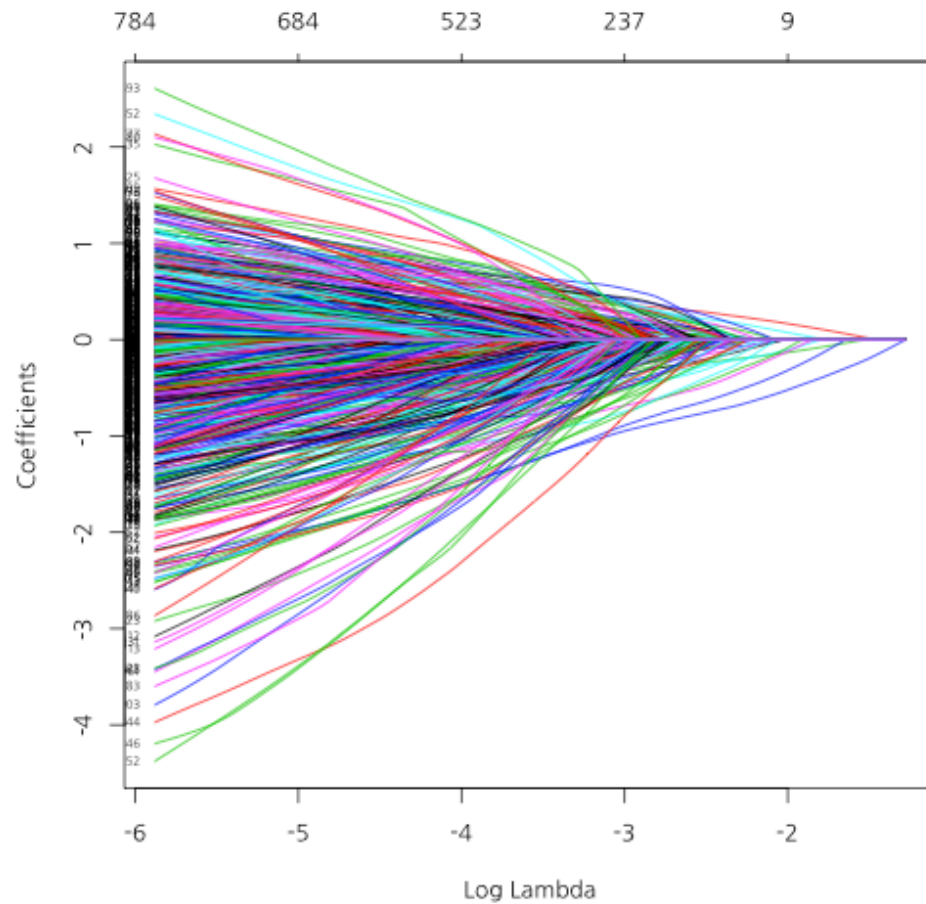
ElasticNet Regression

```
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.lasso[1:5]
```

```
##    hilarious      drama      life      world      bit  
## 0.004800054 0.008611723 0.038371849 0.044125055 0.051063664
```

```
neg.lasso[1:5]
```

```
##      worst      waste      save      wouldnt whatsoever  
## -1.0178384 -0.8242384 -0.6190533 -0.5708966 -0.5378581
```

감정 단어 추출

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

```
pos.ridge[1:5]
```

```
##           hints      fathers    scripted  frequently    obsession  
## 2.801103e-06 1.891073e-05 4.397610e-05 4.478976e-05 5.434588e-05
```

```
neg.ridge[1:5]
```

```
##           binks           coughed    betterhard bottomofthebarrel  
##      -0.1925666      -0.1925665      -0.1845028      -0.1845024  
## capturelike  
##      -0.1835726
```

감정 단어 추출

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

```
pos.elastic[1:5]
```

```
##    brilliant      dark    american      naive      humour  
## 0.002717850 0.005426420 0.008651522 0.008926515 0.010264251
```

```
neg.elastic[1:5]
```

```
##      worst      waste    wouldnt whatsoever      save  
## -0.8711227 -0.7880778 -0.6300778 -0.6222940 -0.6193373
```

감정 단어 점수화

```
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] 0.2
```

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

```
## [1] 0.05882353
```

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

```
## [1] 0.05882353
```

```
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),
                                              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  
##           0 131  59  
##           1  36  74  
##  
##           Accuracy : 0.6833  
##           95% CI : (0.6274, 0.7356)  
##           No Information Rate : 0.5567  
##           P-Value [Acc > NIR] : 4.879e-06  
##
```

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  45
##              1  37  88
##
##              Accuracy : 0.7267
##              95% CI : (0.6725, 0.7763)
##              No Information Rate : 0.5567
##              P-Value [Acc > NIR] : 9.443e-10
##
##              Kappa : 0.4428
##              Mcnemar's Test P-Value : 0.4395
##
```

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  45
##              1  37  88
##
##              Accuracy : 0.7267
##              95% CI : (0.6725, 0.7763)
##              No Information Rate : 0.5567
##              P-Value [Acc > NIR] : 9.443e-10
##
##              Kappa : 0.4428
##              Mcnemar's Test P-Value : 0.4395
##
```

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] 0.1043392
```

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

```
## [1] 0.03085342
```

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

```
## [1] 0.1470654
```

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    3    3
##           1 164 130
##
##           Accuracy : 0.4433
```

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 90 74
##           1 77 59
##
##           Accuracy : 0.4967
##           95% CI : (0.4387, 0.5547)
##           No Information Rate : 0.5567
##           P-Value [Acc > NIR] : 0.9840
##
##           Kappa : -0.0174
##           Mcnemar's Test P-Value : 0.8707
##
```

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
```

```
##           0 120 100
```

```
##           1  47  33
```

```
##
```

```
##           Accuracy : 0.51
```

```
##           95% CI : (0.4519, 0.5679)
```

```
## No Information Rate : 0.5567
```

```
## P-Value [Acc > NIR] : 0.9537
```

```
##
```

```
##           Kappa : -0.0347
```

```
## McNemar's Test P-Value : 1.796e-05
```

```
##
```