Classification-HW

Franky Zhang

2/1/2022

4.6

(a)

```r
b0 <- -6
b1 <- 0.05
b2 <- 1
hours_studied <- 40
undergrad_GPA <- 3.5
percent(invlogit(b0 + b1*hours_studied + b2*undergrad_GPA))
```

```
## [1] 37.75%
```

$$Pr(reveiveA) = invlogit(-6 + 0.05 \times HoursStudied + 1 \times UndergradGPA)$$

plug hours_studied <- 40 & undergrad_GPA <- 3.5 into algorithm, the prob of this student to get an A is 37.75%.

(b)

```r
(logit(0.5) - b0 - b2*undergrad_GPA)/b1
```

```
## [1] 50
```

plug $Pr(reveiveA) = 0.5$ into equation, and calculate the hours need to study to have 50% chance of getting an A is 50.

4.8

Although the error rate for 1-nearset neighbors is 18%, it is an average. Assume the training error for this KNN model is p_1 and test error is p_2, then $0.18 = (p_1 + p_2)/2$. However the training rate for KNN under $K = 1$ is 0, so the test error here is actually 36%, which is higher than logistic regression(30%). Thus, I prefer logistic regression!

4.9

(a)

```
odds = 0.37
percent(odds/(1+odds))
```

```
## [1] 27.01%
```

$$odds = \frac{Pr(Default)}{1 - Pr(Default)}$$

plug $odds = 0.37$ into the equation, and get the fraction of peoplel get default is 27.01%

(b)

```
p = 0.16
percent(p/(1-p))
```

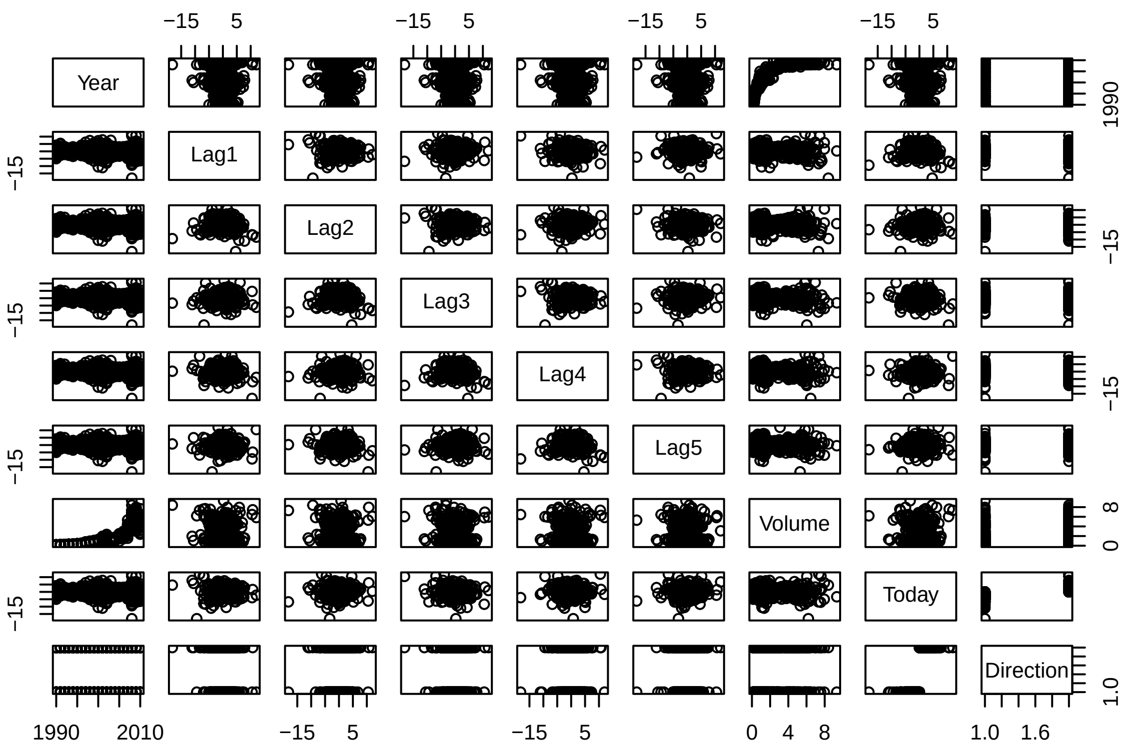
```
## [1] 19.05%
```

plug $prob = 0.16$ into the equation, and get the odds equal to 19.05%

4.13

(a)

```r
# Weekly
pairs(Weekly)
```




```r
cor(Weekly[, -9])
```

```
##               Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

covariance between the lag variables and today’s returns are close to zero, which indicates weak collinearity.

(b)

```r
glm.fits <- glm(Direction~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family = binomial)
summary(glm.fits)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

Lag2 appears to be statistically significant while other predictors fail to reject null hypothesis.

(c)

```r
glm.probs <- predict(glm.fits, type = "response")
glm.pred <- rep("Down", length(glm.probs))
glm.pred[glm.probs > .5] <- "Up"
table(glm.pred, Weekly$Direction)
```

```
##
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557
```

```r
percent(1 - mean(glm.pred == Weekly$Direction))
```

```
## [1] 43.89%
```

the confusion matrix tells me the overall error rate is 43.89% and the model has predict too much "Up" direction which should be "Down" in original data.

(d)

```r
train <- (Weekly$Year < 2009)
Weekly.test <- Weekly[!train, ]
Direction.test <- Weekly.test$Direction
glm.fits <- glm(Direction~Lag2, data = Weekly, family = binomial, subset = train)
glm.probs <- predict(glm.fits, Weekly.test, type = "response")
glm.pred <- rep("Down", length(glm.probs))
glm.pred[glm.probs > .5] <- "Up"
table(glm.pred, Direction.test)
```

```
##         Direction.test
## glm.pred Down Up
##     Down    9  5
##     Up     34 56
```

By GLM model, the overall fraction of correct predictions for held data is $(9+56)/(9+5+34+56) = 62.50\%$

(e)

```r
lda.fits <- lda(Direction~Lag2, data = Weekly, subset = train)
lda.pred <- predict(lda.fits, Weekly.test)
lda.class <- lda.pred$class
table(lda.class, Direction.test)
```

```
##          Direction.test
## lda.class Down Up
##      Down    9  5
##      Up     34 56
```

By LDA model, the overall fraction of correct predictions for held data is $(9+56)/(9+5+34+56) = 62.50\%$

(f)

```r
qda.fits <- qda(Direction~Lag2, data = Weekly, subset = train)
qda.pred <- predict(qda.fits, Weekly.test)
qda.class <- qda.pred$class
table(qda.class, Direction.test)
```

```
##          Direction.test
## qda.class Down Up
##      Down    0  0
##      Up     43 61
```

By QDA model, the overall fraction of correct predictions for held data is $(61)/(43+61) = 58.65\%$

(g)

```
train.X <- cbind(Weekly$Lag2[train])
test.X <- cbind(Weekly$Lag2[!train])
Direction <- Weekly$Direction
Direction.train <- Direction[train]
set.seed(2)
knn.pred <- knn(test = test.X, train = train.X, cl = Direction.train, k = 1)
table(knn.pred,  Direction.test)
```

```
##         Direction.test
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

By KNN model($k = 1$), the overall fraction of correct predictions for held data is $(21+31)/(21+31+22+30)$ $= 50\%$

(h)

```
nb.fits <- naiveBayes(Direction~Lag2, data = Weekly, subset = train)
nb.class <- predict(nb.fits, Weekly.test)
table(nb.class,  Direction.test)
```

```
##         Direction.test
## nb.class Down Up
##     Down    0  0
##     Up     43 61
```

By naive Bayes model, the overall fraction of correct predictions for held data is $(61)/(43+61) = 58.65\%$

(i)

glm and LDA model provide the best results on this data

(j)

```
lda.fits <- lda(Direction~Lag2 + Lag3, data = Weekly, subset = train)
lda.pred <- predict(lda.fits, Weekly.test)
lda.class <- lda.pred$class
table(lda.class, Direction.test)
```

```
##           Direction.test
## lda.class Down Up
##      Down    8  4
##      Up     35 57
```

```
qda.fits <- qda(Direction~Lag1 + Lag3, data = Weekly, subset = train)
qda.pred <- predict(qda.fits, Weekly.test)
qda.class <- qda.pred$class
table(qda.class, Direction.test)
```

```
##           Direction.test
## qda.class Down Up
##      Down   10  7
##      Up     33 54
```

```
nb.fits <- naiveBayes(Direction~Lag2 + Lag3, data = Weekly, subset = train)
nb.class <- predict(nb.fits, Weekly.test)
table(nb.class,  Direction.test)
```

```
##          Direction.test
## nb.class Down Up
##     Down    0  0
##     Up     43 61
```

```
knn.pred <- knn(test = test.X, train = train.X, cl = Direction.train, k = 3)
table(knn.pred,  Direction.test)
```

```
##          Direction.test
## knn.pred Down Up
##     Down   16 19
##     Up     27 42
```

LDA: when includes Lag2 and Lag3 as predictors, LDA model gives best results, correct prediction for held data is 62.5%.

QDA: QDA model give best results (61.5%) including Lag1 and Lag3 as predictors.

Naive Bayes: including Lag2 and Lag3, Naive Bayes gives best prediction results: 58.7%

KNN: when k = 3, KNN classification give best result,

4.14

(a)

```
mpg01 <- rep(1, length(Auto$mpg))
mpg01[Auto$mpg<median(Auto$mpg)] <- 0
```

(b)

```r
names(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

```r
cylinders <- ggplot(data = Auto, mapping = aes(x = cylinders, y = mpg01)) +
  geom_point() + geom_jitter()
displacement <- ggplot(data = Auto, mapping = aes(x = displacement, y = mpg01)) +
  geom_point()
horsepower <- ggplot(data = Auto, mapping = aes(x = horsepower, y = mpg01)) +
  geom_point()
weight <- ggplot(data = Auto, mapping = aes(x = weight, y = mpg01)) +
  geom_point()
acceleration <- ggplot(data = Auto, mapping = aes(x = acceleration, y = mpg01)) +
  geom_point()  # no clear relationship
year <- ggplot(data = Auto, mapping = aes(x = year, y = mpg01)) +
  geom_point() + geom_jitter() # no clear relationship
origin <- ggplot(data = Auto, mapping = aes(x = origin, y = mpg01)) +
  geom_point() + geom_jitter()
Auto01 <- cbind(mpg01, Auto)
cor(Auto01[, -10])
```

```
##                   mpg01        mpg  cylinders displacement horsepower
## mpg01         1.0000000  0.8369392 -0.7591939   -0.7534766 -0.6670526
## mpg           0.8369392  1.0000000 -0.7776175   -0.8051269 -0.7784268
## cylinders    -0.7591939 -0.7776175  1.0000000    0.9508233  0.8429834
## displacement -0.7534766 -0.8051269  0.9508233    1.0000000  0.8972570
## horsepower   -0.6670526 -0.7784268  0.8429834    0.8972570  1.0000000
## weight       -0.7577566 -0.8322442  0.8975273    0.9329944  0.8645377
## acceleration  0.3468215  0.4233285 -0.5046834   -0.5438005 -0.6891955
## year          0.4299042  0.5805410 -0.3456474   -0.3698552 -0.4163615
## origin        0.5136984  0.5652088 -0.5689316   -0.6145351 -0.4551715
##                  weight acceleration       year     origin
## mpg01        -0.7577566    0.3468215  0.4299042  0.5136984
## mpg          -0.8322442    0.4233285  0.5805410  0.5652088
## cylinders     0.8975273   -0.5046834 -0.3456474 -0.5689316
## displacement  0.9329944   -0.5438005 -0.3698552 -0.6145351
## horsepower    0.8645377   -0.6891955 -0.4163615 -0.4551715
## weight        1.0000000   -0.4168392 -0.3091199 -0.5850054
## acceleration -0.4168392    1.0000000  0.2903161  0.2127458
## year         -0.3091199    0.2903161  1.0000000  0.1815277
## origin       -0.5850054    0.2127458  0.1815277  1.0000000
```

```r
Auto01$mpg01 <- factor(Auto01$mpg01)
```

cylinders, horsepower, weight, acceleration and origin seems to be useful for predicting *mpg01*

(c)

```r
# Auto$year
train <- (Auto01$year < 81)
Auto01.train <- Auto01[train, ]
Auto01.test <- Auto01[!train, ]
mpg01.train <- Auto01.train$mpg01
mpg01.test <- Auto01.test$mpg01
```

(d)

```r
lda.fits <- lda(mpg01~cylinders+displacement+weight, data = Auto01.train)
lda.pred <- predict(lda.fits, newdata = Auto01.test)
lda.class <- lda.pred$class
table(lda.class, mpg01.test)
```

```
##          mpg01.test
## lda.class  0  1
##         0  4  7
##         1  0 47
```

```r
1 - percent(51/58)
```

```
## [1] 12.07%
```

```r
# lda.pred <- predict(lda.fits, newdata = Auto01.train)
# lda.class <- lda.pred$class
# table(lda.class, mpg01.train)
# (166 + 134)/334
```

the test error of this LDA model is 12.07%

(e)

```r
qda.fits <- qda(mpg01~cylinders+displacement+weight, data = Auto01.train)
qda.pred <- predict(qda.fits, newdata = Auto01.test)
qda.class <- qda.pred$class
table(qda.class, mpg01.test)
```

```
##          mpg01.test
## qda.class  0  1
##         0  4  8
##         1  0 46
```

```r
1 - percent(52/58)
```

```
## [1] 10.34%
```

```r
# qda.pred <- predict(qda.fits, newdata = Auto01.train)
# qda.class <- qda.pred$class
# table(qda.class, mpg01.train)
# (173 + 130)/334
```

the test error of this QDA model is 10.34%

(f)

```r
glm.fits <- glm(mpg01~cylinders+displacement+weight, data = Auto01.train, family = binomial)
glm.probs <- predict(glm.fits, newdata =  Auto01.test, type = "response")
glm.pred <- rep(1, length(glm.probs))
glm.pred[glm.probs < .5] <- 0
table(glm.pred, mpg01.test)
```

```
##          mpg01.test
## glm.pred  0  1
##        0  4  9
##        1  0 45
```

```r
1 - percent(49/58)
```

```
## [1] 15.52%
```

the test error of this GLM model is 15.52%

(g)

```r
nb.fits <- naiveBayes(mpg01~cylinders+displacement+weight, data = Auto01.train)
glm.pred <- predict(nb.fits, newdata = Auto01.test)
table(glm.pred, mpg01.test)
```

```
##          mpg01.test
## glm.pred  0  1
##        0  4  7
##        1  0 47
```

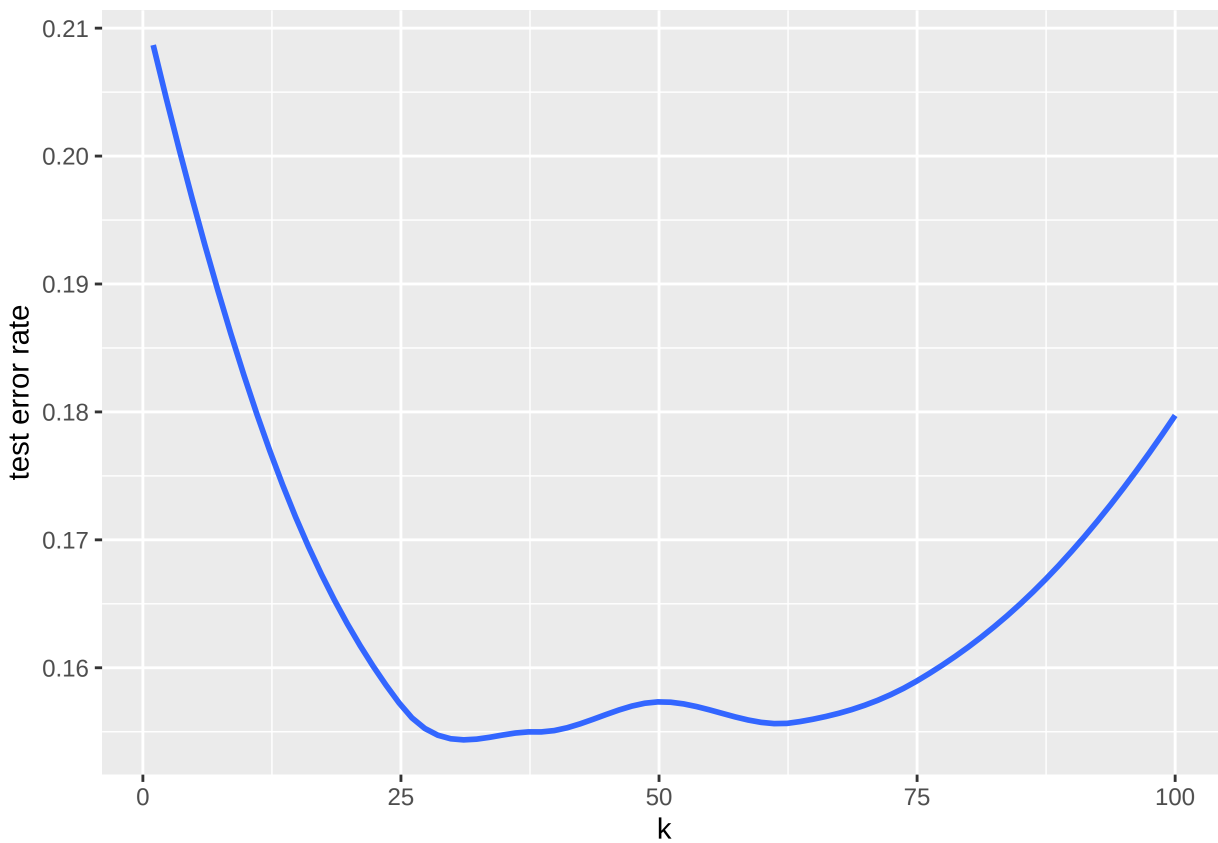
```r
1 - percent(51/58)
```

```
## [1] 12.07%
```

the test error of this naive Bayes model is 12.07%

(h)

```r
train.X <- cbind(cylinders=Auto$cylinders,
                 displacement=Auto$displacement, weight=Auto$weight)[train, ]
test.X <- cbind(cylinders=Auto$cylinders,
                displacement=Auto$displacement, weight=Auto$weight)[!train, ]
knn.error <- c()
for (i in 1:100) {
  # i = 1
  knn.pred <- knn(train = train.X, test = test.X, cl = mpg01.train, k = i)
  knn.error <- rbind(knn.error, c(i,percent(1-(table(knn.pred,  mpg01.test)[1,1] +
      table(knn.pred,  mpg01.test)[2,2])/58)))
}
knn.error <- data.frame(k = knn.error[, 1], error.rate = knn.error[, 2])
ggplot(data = knn.error, aes(x = k, y = error.rate)) +
  geom_smooth(method = 'loess', formula = 'y ~ x', se = FALSE) +
  xlab("k") + ylab("test error rate")
```



when k = 30, KNN model performs best on *Auto* data, reach a test rate down to 15.51%

4.15

(a)

```r
Power <- function(a){
  print(a^3)
}
# Power(2)
```

(b)

```r
Power2 <- function(x, a){
  print(x^a)
}
Power2(3, 8)
```

```
## [1] 6561
```

(c)

```r
Power2(10, 3)
```

```
## [1] 1000
```

```r
Power2(8, 17)
```

```
## [1] 2.2518e+15
```

```r
Power2(131, 3)
```

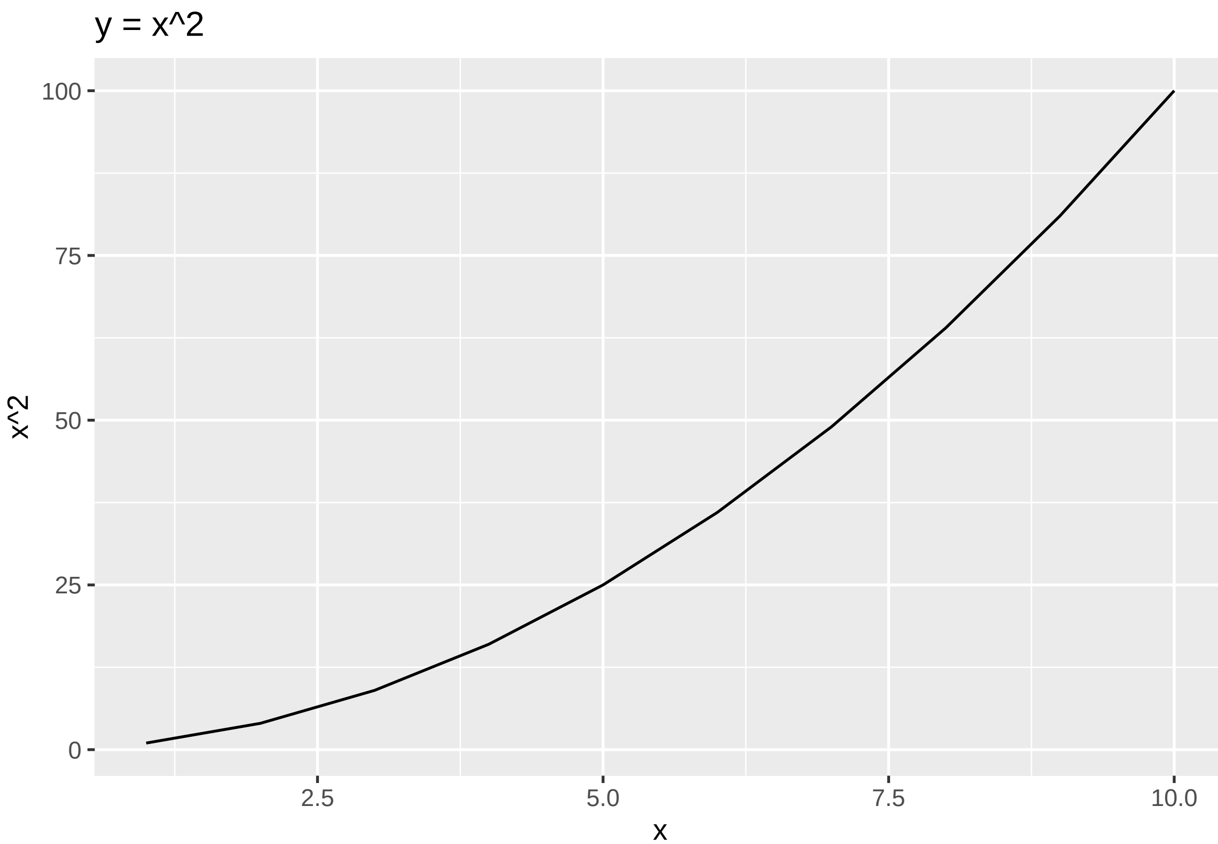
```
## [1] 2248091
```

(d)

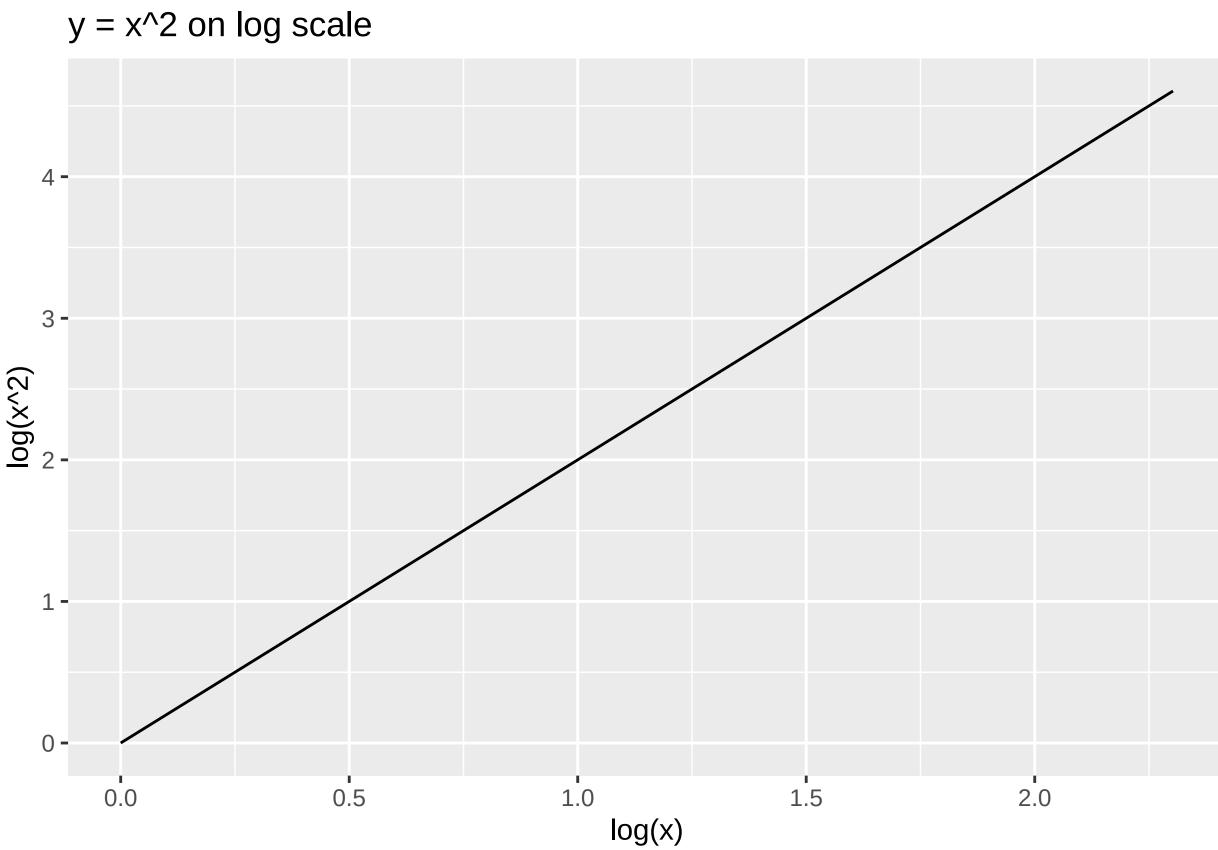
```r
Power3 <- function(x, a){
  R <- x^a
  return(R)
}
```

(e)

```
x <- c(1:10)
ggplot(mapping = aes(x = x, y = Power3(x, 2))) +
  geom_line() + xlab("x") + ylab("x^2") +
  ggtitle("y = x^2")
```

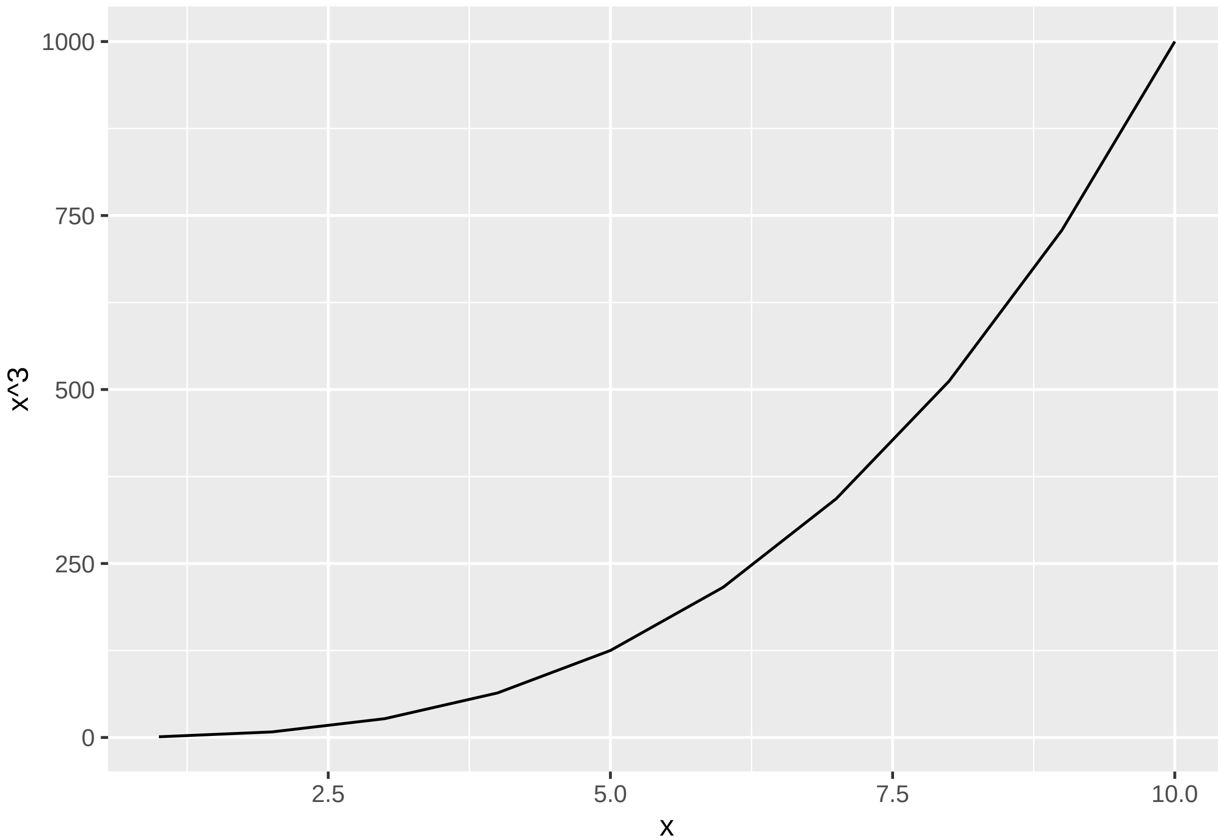



```
ggplot(mapping = aes(x = log(x), y = log(Power3(x, 2)))) +
  geom_line() + xlab("log(x)") + ylab("log(x^2)") +
  ggtitle("y = x^2 on log scale")
```



(f)

```
PlotPower <- function(vector, power){
  plot.data <- data.frame(x = vector, y = vector^3)
  ggplot(data = plot.data, aes(x =x, y = y)) +
    geom_line() + xlab("x") + ylab("x^3")
}
PlotPower(vector = c(1:10), power = 3)
```



4.16

```
# Boston
crim.median <- median(Boston$crim)
# create response
crim01 <- rep(1, length(Boston$crim))
crim01[Boston$crim<crim.median] <- 0
Boston01 <- cbind(crim01, Boston)[, -2]
abs(cor(Boston01)>.5)
```

```
##         crim01 zn indus chas nox rm age dis rad tax ptratio black lstat medv
## crim01       1  0     1    0   1  0   1   0   1   1       0     0     0    0
## zn           0  1     0    0   0  0   0   1   0   0       0     0     0    0
## indus        1  0     1    0   1  0   1   0   1   1       0     0     1    0
## chas         0  0     0    1   0  0   0   0   0   0       0     0     0    0
## nox          1  0     1    0   1  0   1   0   1   1       0     0     1    0
## rm           0  0     0    0   0  1   0   0   0   0       0     0     0    1
## age          1  0     1    0   1  0   1   0   0   1       0     0     1    0
## dis          0  1     0    0   0  0   0   1   0   0       0     0     0    0
## rad          1  0     1    0   1  0   0   0   1   1       0     0     0    0
## tax          1  0     1    0   1  0   1   0   1   1       0     0     1    0
## ptratio      0  0     0    0   0  0   0   0   0   0       1     0     0    0
## black        0  0     0    0   0  0   0   0   0   0       0     1     0    0
## lstat        0  0     1    0   1  0   1   0   0   1       0     0     1    0
## medv         0  0     0    0   0  1   0   0   0   0       0     0     0    1
```

according to correlation matrix, pick up *indus*, *nox*, *age*, *rad* and *tax* to be condidate predictors for following steps.

logistic regression

```
Boston01$crim01 <- factor(Boston01$crim01)
glm.fits <- glm(crim01~indus+nox+age+rad+tax, family = binomial, data = Boston01)
summary(glm.fits)
```

```
##
## Call:
## glm(formula = crim01 ~ indus + nox + age + rad + tax, family = binomial,
##     data = Boston01)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.03233  -0.26526  -0.01174   0.00626   2.65985
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -21.356677   2.906609  -7.348 2.02e-13 ***
## indus        -0.057189   0.042324  -1.351  0.17663
## nox          37.900682   6.181155   6.132 8.70e-10 ***
## age           0.011780   0.008603   1.369  0.17089
## rad           0.595216   0.116260   5.120 3.06e-07 ***
## tax          -0.007268   0.002366  -3.072  0.00213 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 701.46  on 505  degrees of freedom
## Residual deviance: 246.74  on 500  degrees of freedom
## AIC: 258.74
##
## Number of Fisher Scoring iterations: 8
```

predictor *nox*, *rad* and *tax* successfully reject null hypothesis, then consider fitting model on training set to compare performances.

```
dim(Boston)
```

```
## [1] 506  14
```

```
set.seed(22)
sample <- sample(size = round(dim(Boston)[1]*.3), x = dim(Boston)[1], replace = FALSE)
test <- c(1:(dim(Boston)[1])) %in% sample
Boston01.train <- Boston01[!test, ]
crim01.train <- Boston01.train$crim01
Boston01.test <- Boston01[test, ]
crim01.test <-  Boston01.test$crim01
```

```
glm.fit1 <- glm(crim01~indus+nox+age+rad+tax, family = binomial, data = Boston01.train)
glm.prob1 <- predict(glm.fit1, newdata = Boston01.test, type = "response")
glm.pred1 <- rep(1, length(glm.prob1))
glm.pred1[glm.prob1<.5] <- 0

compare.table <- data.frame(method = "glm_5",
  test.error = 1-(table(glm.pred1, crim01.test)[1,1] +
  table(glm.pred1, crim01.test)[2,2])/152)

glm.fit2 <- glm(crim01~nox+rad+tax, family = binomial, data = Boston01.train)
glm.prob2 <- predict(glm.fit2, newdata = Boston01.test, type = "response")
glm.pred2 <- rep(1, length(glm.prob2))
glm.pred2[glm.prob2<.5] <- 0
compare.table <- rbind(compare.table,
  c("glm_3", 1-(table(glm.pred2, crim01.test)[1,1] +
  table(glm.pred2, crim01.test)[2,2])/152))
```

LDA

```
lda.fit1 <- lda(crim01~indus+nox+age+rad+tax, data = Boston01.train)
lda.class1 <- predict(lda.fit1, newdata = Boston01.test)$class
compare.table <- rbind(compare.table,
  c("lda_5", 1-(table(lda.class1, crim01.test)[1,1] +
  table(lda.class1, crim01.test)[2,2])/152))

lda.fit2 <- lda(crim01~nox+rad+tax, data = Boston01.train)
lda.class2 <- predict(lda.fit2, newdata = Boston01.test)$class
compare.table <- rbind(compare.table,
  c("lda_3", 1-(table(lda.class2, crim01.test)[1,1] +
  table(lda.class2, crim01.test)[2,2])/152))
```

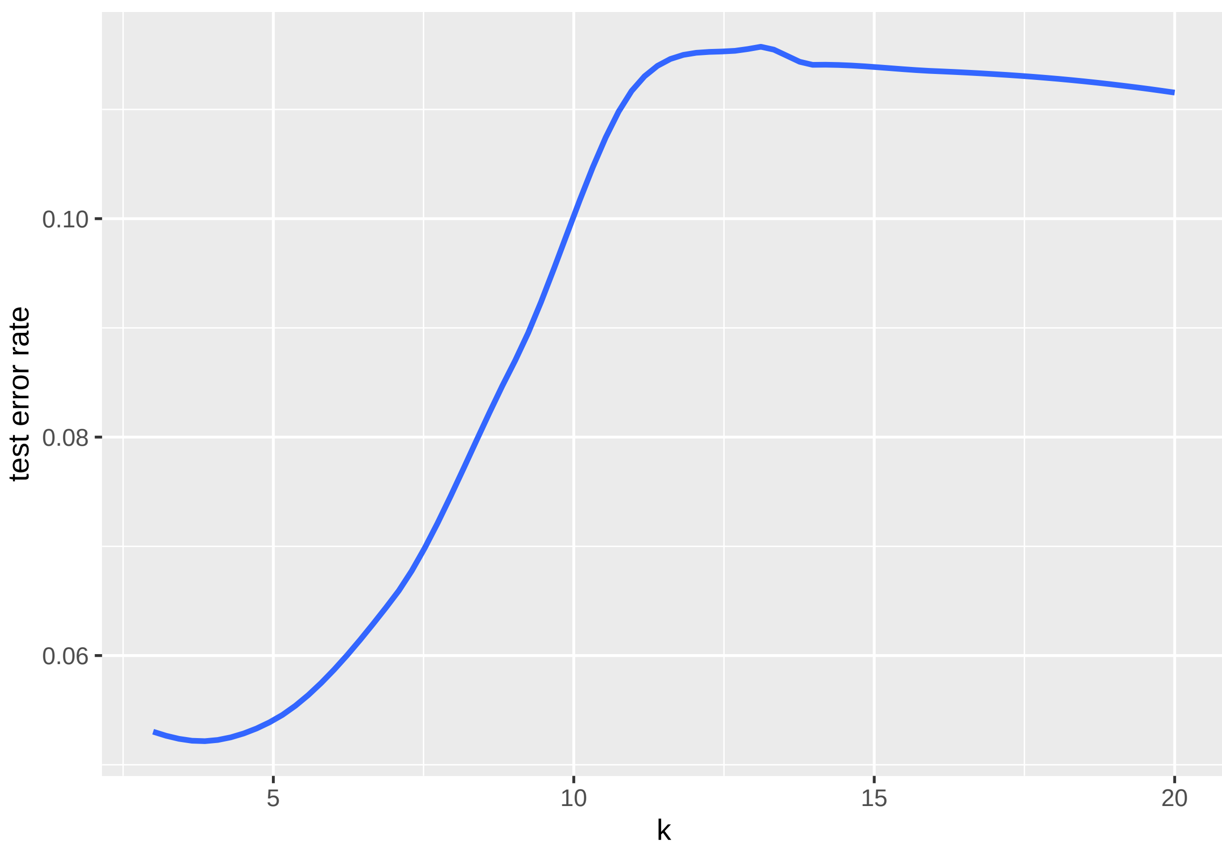
Naive Bayes

```
nb.fit1 <- naiveBayes(crim01~indus+nox+age+rad+tax, data = Boston01.train)
nb.class1 <- predict(nb.fit1, newdata = Boston01.test)
compare.table <- rbind(compare.table,
  c("nb_5", 1-(table(nb.class1, crim01.test)[1,1] +
  table(nb.class1, crim01.test)[2,2])/152))

nb.fit2 <- naiveBayes(crim01~nox+rad+tax, data = Boston01.train)
nb.class2 <- predict(nb.fit2, newdata = Boston01.test)
compare.table <- rbind(compare.table,
  c("nb_3", 1-(table(nb.class2, crim01.test)[1,1] +
  table(nb.class2, crim01.test)[2,2])/152))
```

KNN

```r
# due to the curse of dimension, we prefer 3 dimension predictors
train.X <- cbind(nox = Boston01.train$nox,
                 rad = Boston01.train$rad,
                 tax = Boston01.train$tax)
test.X <- cbind(nox = Boston01.test$nox,
                rad = Boston01.test$rad,
                tax = Boston01.test$tax)
knn.error <- c()
for (i in 3:20) {
  knn.pred <- knn(train = train.X, test = test.X, cl = crim01.train, k = i)
  knn.error <- rbind(knn.error, c(i,percent(1-(table(knn.pred,  crim01.test)[1,1] +
      table(knn.pred,  crim01.test)[2,2])/152)))
}
knn.error <- data.frame(k = knn.error[, 1], error.rate = knn.error[, 2])
ggplot(data = knn.error, aes(x = k, y = error.rate)) +
  geom_smooth(method = 'loess', formula = 'y ~ x', se = FALSE) +
  xlab("k") + ylab("test error rate")
```




```r
compare.table <- rbind(compare.table, c("knn(k=5)", 0.05263158))
compare.table$test.error <- percent(compare.table$test.error, digit = 3)
```

Conclusion

```
compare.table
```

```
##     method test.error
## 1    glm_5    11.842%
## 2    glm_3    11.842%
## 3    lda_5    13.816%
## 4    lda_3    13.158%
## 5     nb_5    13.816%
## 6     nb_3    13.158%
## 7 knn(k=5)     5.263%
```

The result shows that, for glm, lda and naive bayes methods, including 3 statistically significant predictors only gives better results. Overrall, knn performs the bset among these methods with an test error rate around 5%.