20152950 Kangminseok Datamining HW1

April 17, 2020

1 20152950 Kang Minseok HW1

[ ]:
```

```

1.0.1 Read data

[1]:
```
data=read.table('C:/Users/Administrator/Desktop/datamining/HW1/data2.
↪txt',header= TRUE)
```

1.1 split into 300 test datas and 700 train datas

[2]:
```
set.seed(1)
train_ind <- sample(seq_len(nrow(data)), size = 700)

train <- data[train_ind, ]
test <- data[-train_ind, ]
```

2 a) Determine the best polynomial regression model

2.1 validation method

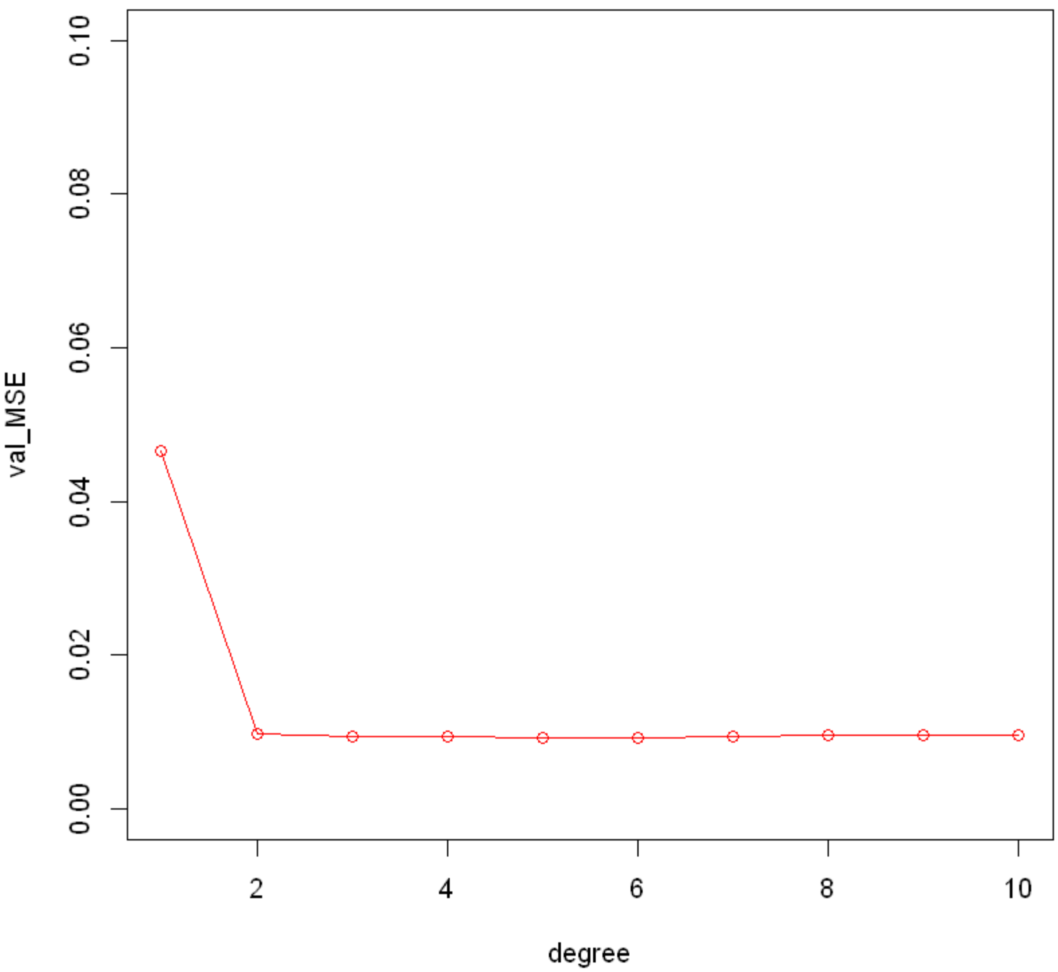
[3]:
```
validation_ind <- sample(seq_len(nrow(train)), size=350)
val_set<-train[validation_ind,]
train2<-train[-validation_ind,]
head(train2)
head(val_set)
```

	y	x
679	11.19222	2.981615
930	11.52814	4.283470
978	11.72022	6.124232
187	11.89680	7.581031
307	10.71828	1.763507
597	11.64581	4.798358

	y	x
437	12.01530	7.396417
420	11.83963	8.227933
193	10.39658	1.008731
824	11.87077	8.196983
582	11.95556	9.489382
260	11.89417	9.346914

[4]:

```r
set.seed(20152950)

for (j in 1:10) {
  vs.error = rep(0,10)
  for (i in 1:10){
    fit = lm(y ~ poly(x,i),data=train2)
    vs.error[i] = mean((val_set$y - predict(fit,val_set))^2)
  }
  if (j<2){
    plot(vs.error, type='o', ylab="val_MSE", xlab="degree",ylim=c(0,0.1))
  } else{
    lines(vs.error, type='o',col=j,ylim=c(0,0.2))
  }
}
```



[5]:
```
vs.error
which.min(vs.error)
```

1. 0.0465721891548336 2. 0.0098031641525759 3. 0.00942826179225358 4. 0.00942974521196896 5. 0.00934990449144303 6. 0.00934315580929329 7. 0.00949343218372707 8. 0.00964203671920841 9. 0.00965395033851972 10. 0.0096740429675282

6

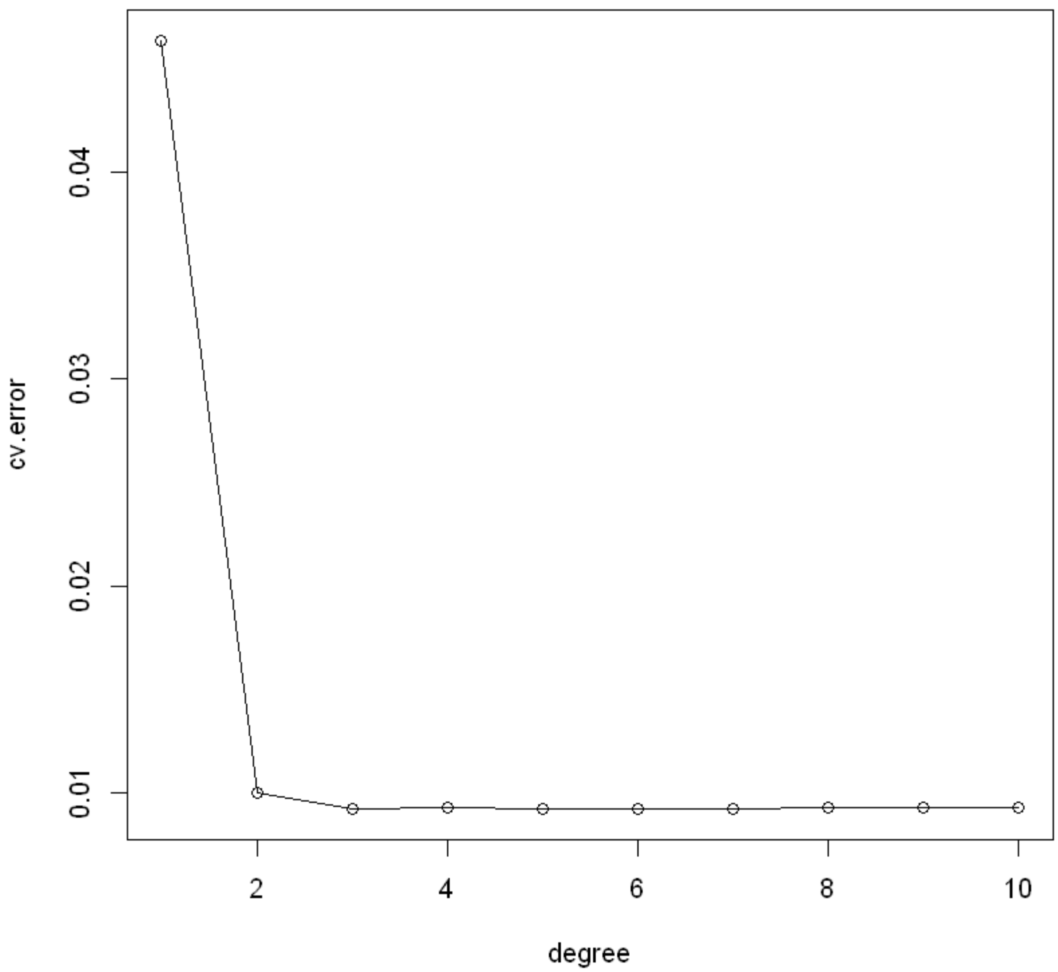
2.1.1 6-degree model looks fine

2.2 LOOCV method

[6]:
```
set.seed(20152950)

library(boot)

cv.error=rep(0,10)

for(i in 1:10){
  glm.fit=glm(y~poly(x,i),data=train)
  cv.error[i]=cv.glm(train,glm.fit)$delta[1]
}
```

```
Warning message:
"package 'boot' was built under R version 3.6.3"
```

[7]:
```
plot(cv.error, type='o', xlab="degree")
```



[8]:
```
cv.error
which.min(cv.error)
```

1. 0.0463845957391451 2. 0.00999340140811508 3. 0.0092249510630756 4. 0.00925259953339787 5. 0.00920037087855594 6. 0.0092153235096801 7. 0.0092372151748105 8. 0.00926227496058895 9. 0.00928805002808778 10. 0.00930717390840043

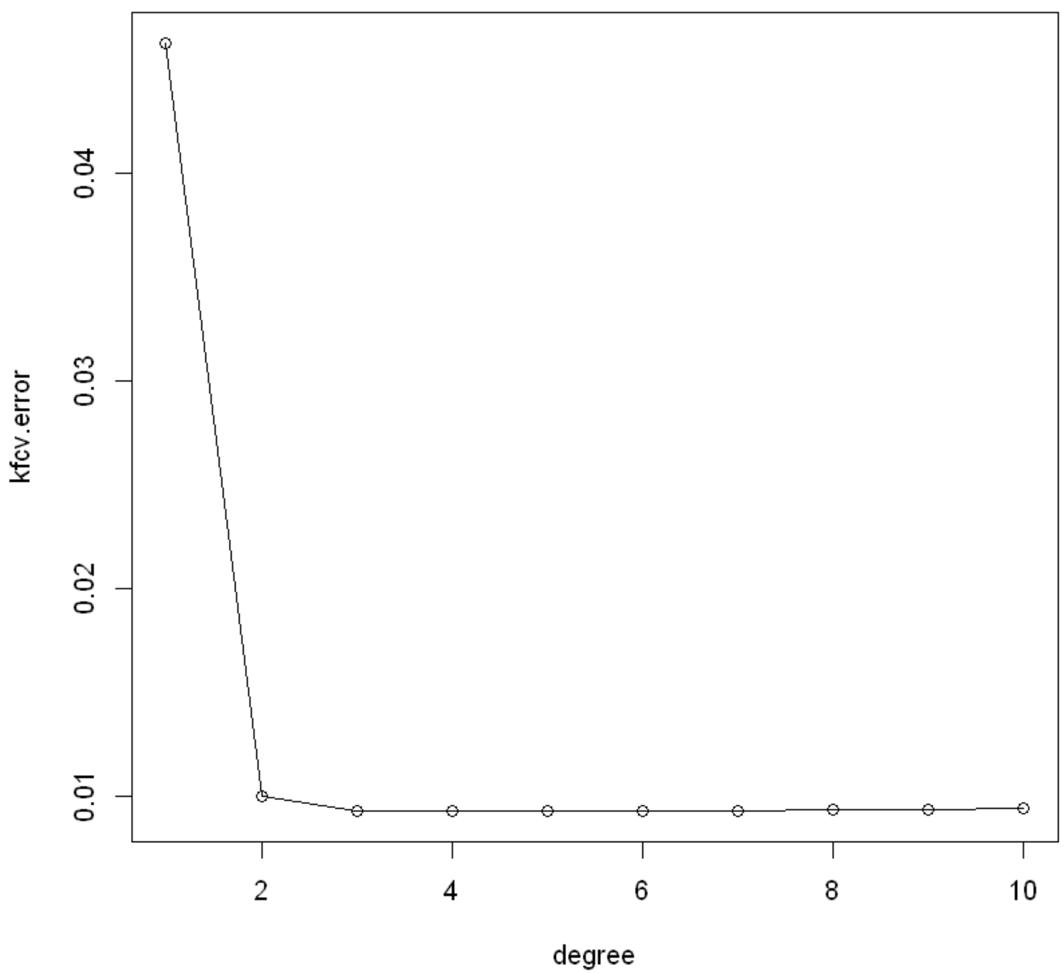
5

2.2.1 5 degree model looks good

3 10-fold CV approach

[9]:
```
set.seed(20152950)



kfcv.error=rep(0,10)

for (i in 1:10){
  glm.fit=glm(y~poly(x,i),data=train)
  kfcv.error[i]=cv.glm(train,glm.fit,K=10)$delta[1]
}
```

[10]:
```
plot(kfcv.error, type='o', xlab="degree")
```



[11]:
```
kfcv.error
which.min(kfcv.error)
```

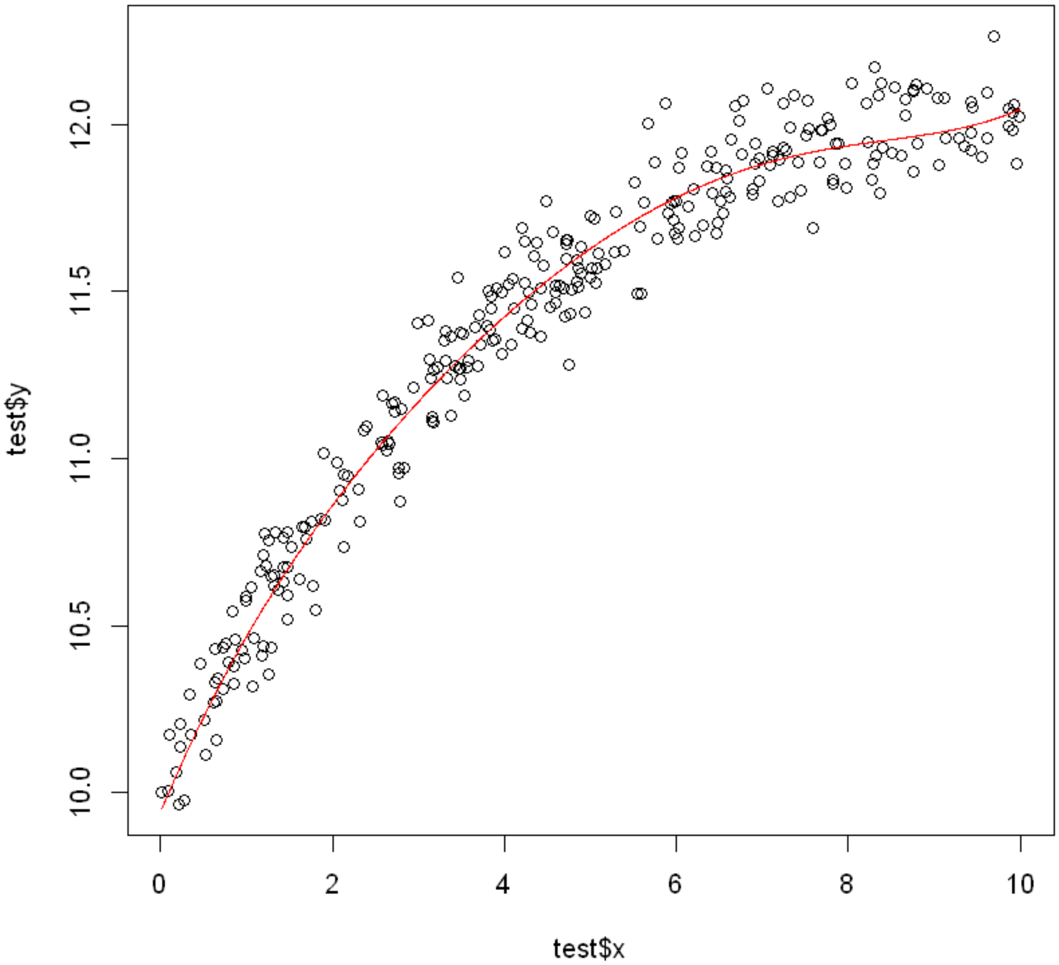
1. 0.0463020236139796 2. 0.00998262784890129 3. 0.00926343614584298 4. 0.00928493505348354 5. 0.00925511413941544 6. 0.00925473368305222 7. 0.00927848155619971 8. 0.00929882595527234 9. 0.00931257797900688 10. 0.00940248464018602

6

3.0.1 6 degree model looks good

4 Let's make 5 degree model

[12]:
```
fit=glm(y~poly(x,5),data=train)
x <- with(test, seq(min(x), max(x), length.out=2000))
y <- predict(fit, newdata = data.frame(x= x))
plot(test$x,test$y)
lines(x, y, col = "red")
```



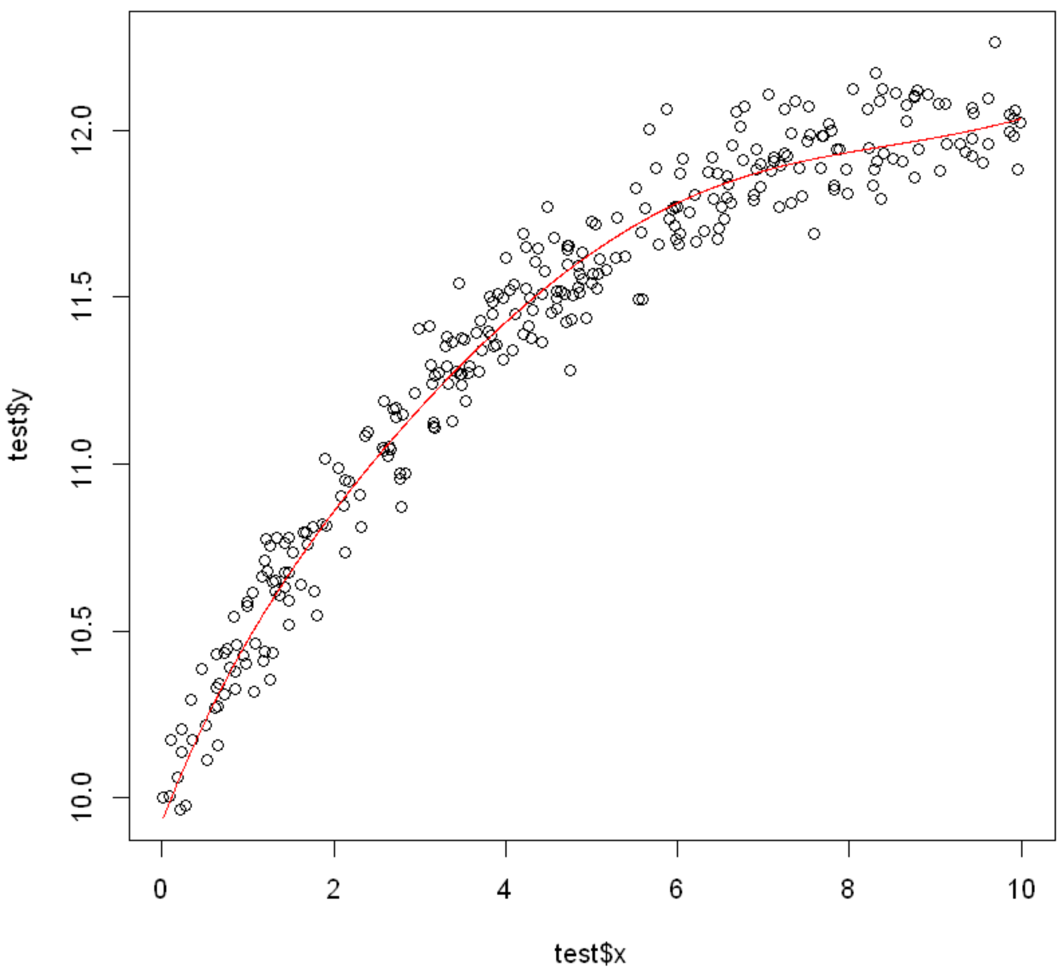
[13]:
```
predict=predict(fit,test)
mse5=mean((test$y-predict)^2)
mse5
```

0.0114262134970114

mse is 0.0114262134970114

5 Now, make 6 degree model

[14]:
```
fit=glm(y~poly(x,6),data=train)
x <- with(test, seq(min(x), max(x), length.out=2000))
y <- predict(fit, newdata = data.frame(x= x))
plot(test$x,test$y)
lines(x, y, col = "red")
```



[15]:
```
predict=predict(fit,test)
mse6=mean((test$y-predict)^2)
mse6
```

0.0114474847420273

mse is 0.0114474847420273

[16]:
```
mse5
```

0.0114262134970114

[17]:
```
mse6
```

0.0114474847420273

6 So it seems that 5-degree polynomial model is the best

[ ]:
```

```

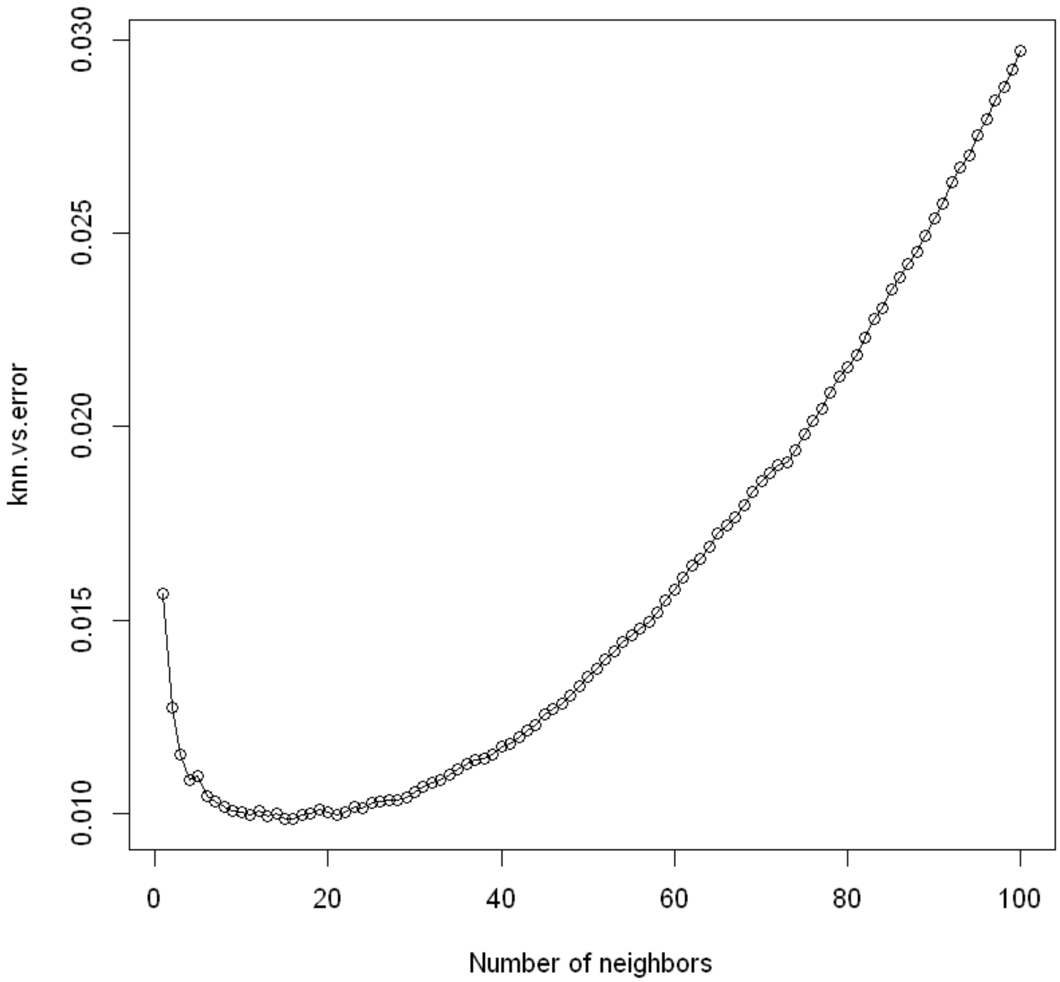
7 Let's do KNN regression

[18]:
```
library(FNN)
```

```
Warning message:
"package 'FNN' was built under R version 3.6.3"
```

8 Validatiom method

[19]:
```
set.seed(20152950)

knn.vs.error=rep(0,100)

for (i in 1:100){
  fit=knn.reg(train=as.data.frame(train2$x),test=as.data.frame(val_set$x), y=␣
 ↪train2$y, k=i)
  knn.vs.error[i] = mean((val_set$y - fit$pred)^2)
}
plot(knn.vs.error,type='o',xlab = c('Number of neighbors'))
```



[20]:
```
which.min(knn.vs.error)
```

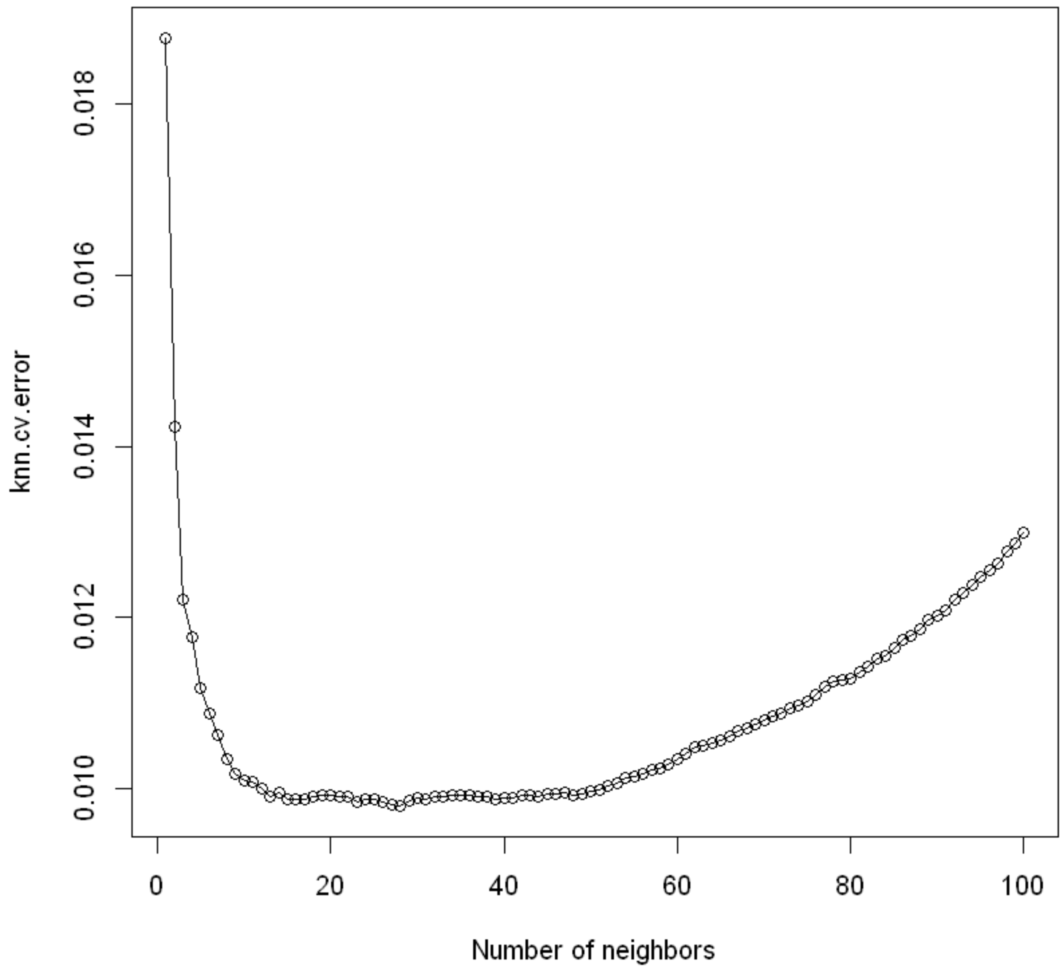
16

8.1 k=16 looks like it is the best

9 LOOCV

[21]:
```
knn.cv.error=rep(0,100)

for (i in 1:100){
    fit=knn.reg(train=as.data.frame(train$x), y= train$y, k=i)
```

```
    # in knn.reg, if test is not supplied, Leave one out cross-validation is
↪performed and R-square is the predicted R-square.
    knn.cv.error[i] = mean((train$y - fit$pred)^2)
  }
plot(knn.cv.error,type='o',xlab = c('Number of neighbors'))
```



[22]:
```
which.min(knn.cv.error)
```

28

9.1 K=28 model looks good

10 10-fold cross validation method

[23]:
```
library(caret)
```

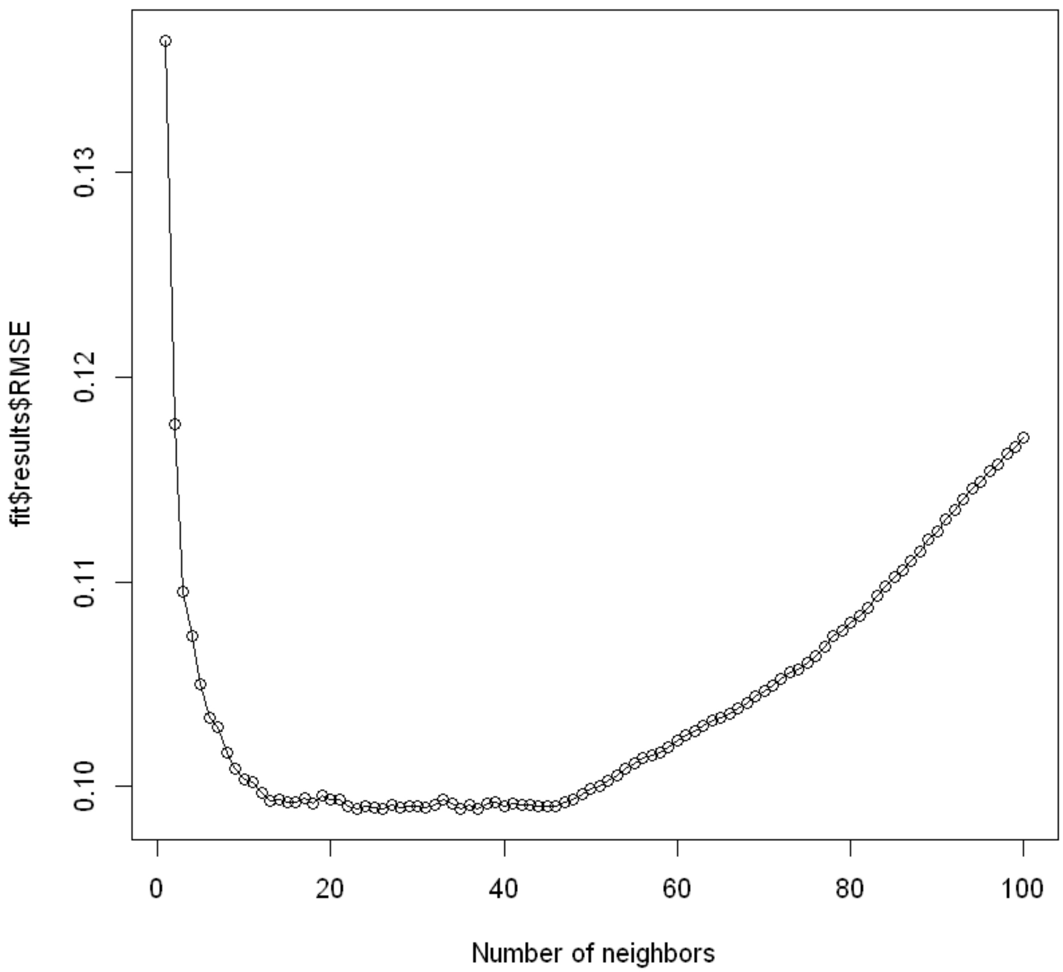
```
Warning message:
"package 'caret' was built under R version 3.6.3"Loading required package:
lattice

Attaching package: 'lattice'

The following object is masked from 'package:boot':

    melanoma

Loading required package: ggplot2
Warning message:
"package 'ggplot2' was built under R version 3.6.3"
```

[ ]:
```

```

[24]:
```
trControl <- trainControl(method  = "cv",
                          number  = 10)
```

[25]:
```
fit <- train(y~x,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:100),
             trControl  = trControl,
             data       = train)
```

[26]:
```
plot(fit$results$RMSE,type='o',xlab = c('Number of neighbors'))
```



[27]:
```
which.min(fit$results$RMSE)
```

37

10.1 Looks like k=37 model is the best!

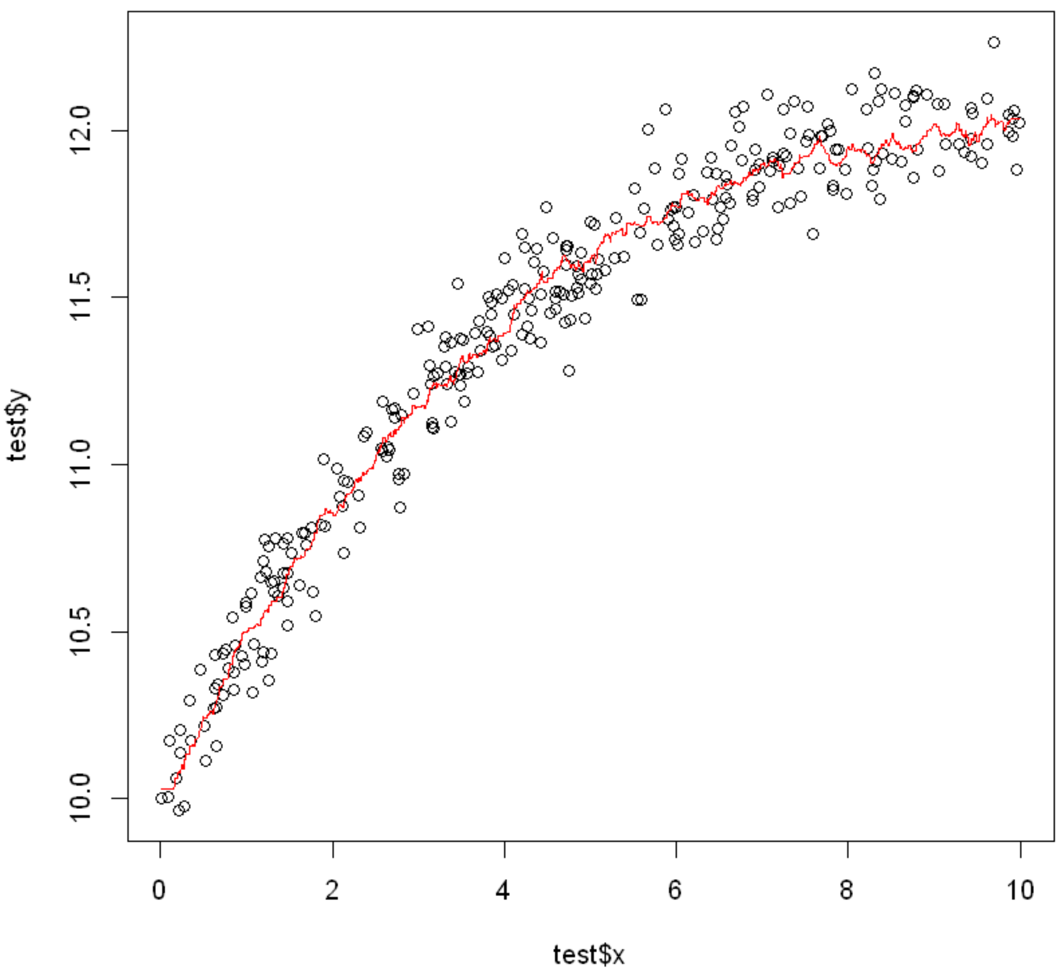
[ ]:
```

```

11 Let's test!

11.1 k=16

[28]:
```
fit=knn.reg(train = as.data.frame(train$x),test=as.data.
↪frame(test$x),y=train$y,k=16)
mse16=mean((test$y - fit$pred)^2)
```

[29]:
```
x <- with(test, seq(min(x), max(x), length.out=2000))
fit2=knn.reg(train = as.data.frame(train$x),test=as.data.
↪frame(x),y=train$y,k=16)
y <- fit2$pred
plot(test$x,test$y)
lines(x, y, col = "red")
```



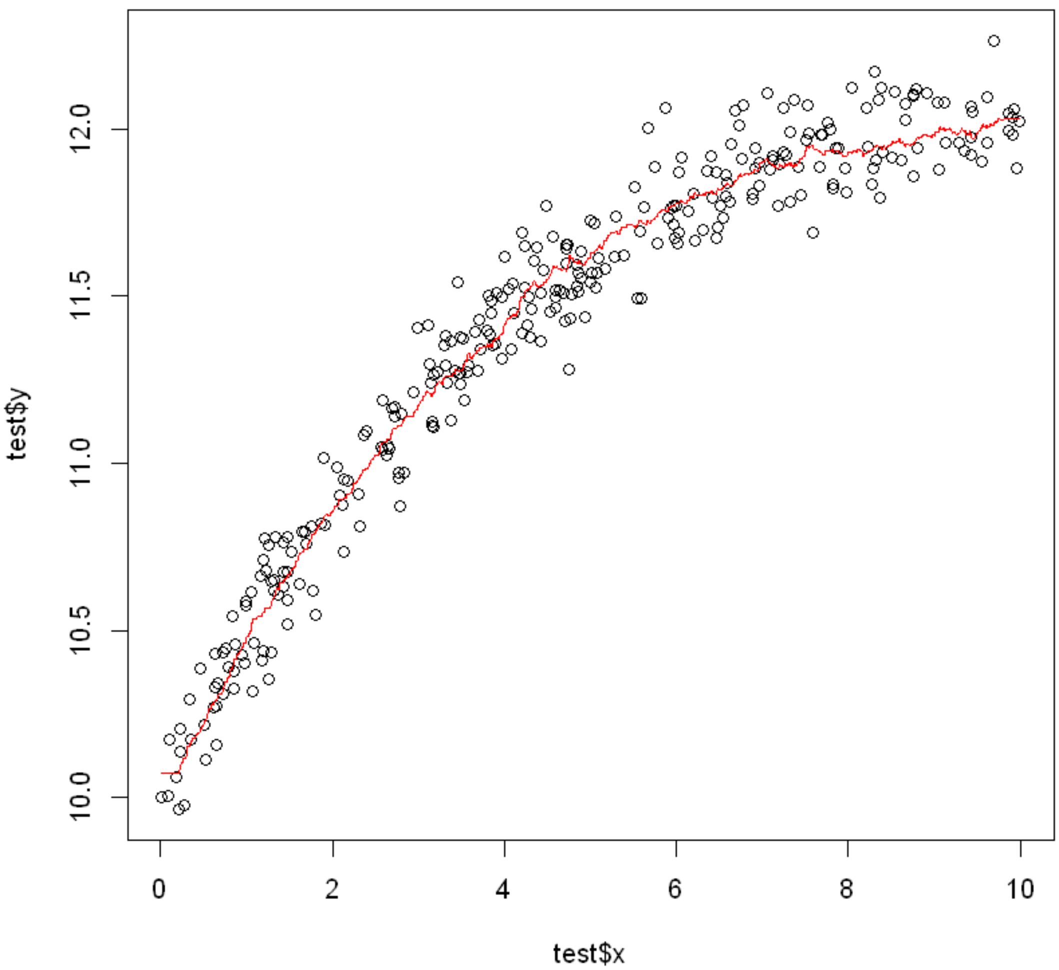
[30]:
```
mse16
```

0.011866644165692

11.2 mse is 0.011866644165692

12 k=28

[31]:
```
fit=knn.reg(train = as.data.frame(train$x),test=as.data.
 ↪frame(test$x),y=train$y,k=28)
mse28=mean((test$y - fit$pred)^2)



x <- with(test, seq(min(x), max(x), length.out=2000))
fit2=knn.reg(train = as.data.frame(train$x),test=as.data.
 ↪frame(x),y=train$y,k=28)
y <- fit2$pred
plot(test$x,test$y)
lines(x, y, col = "red")
```




```
[32]: mse28
```

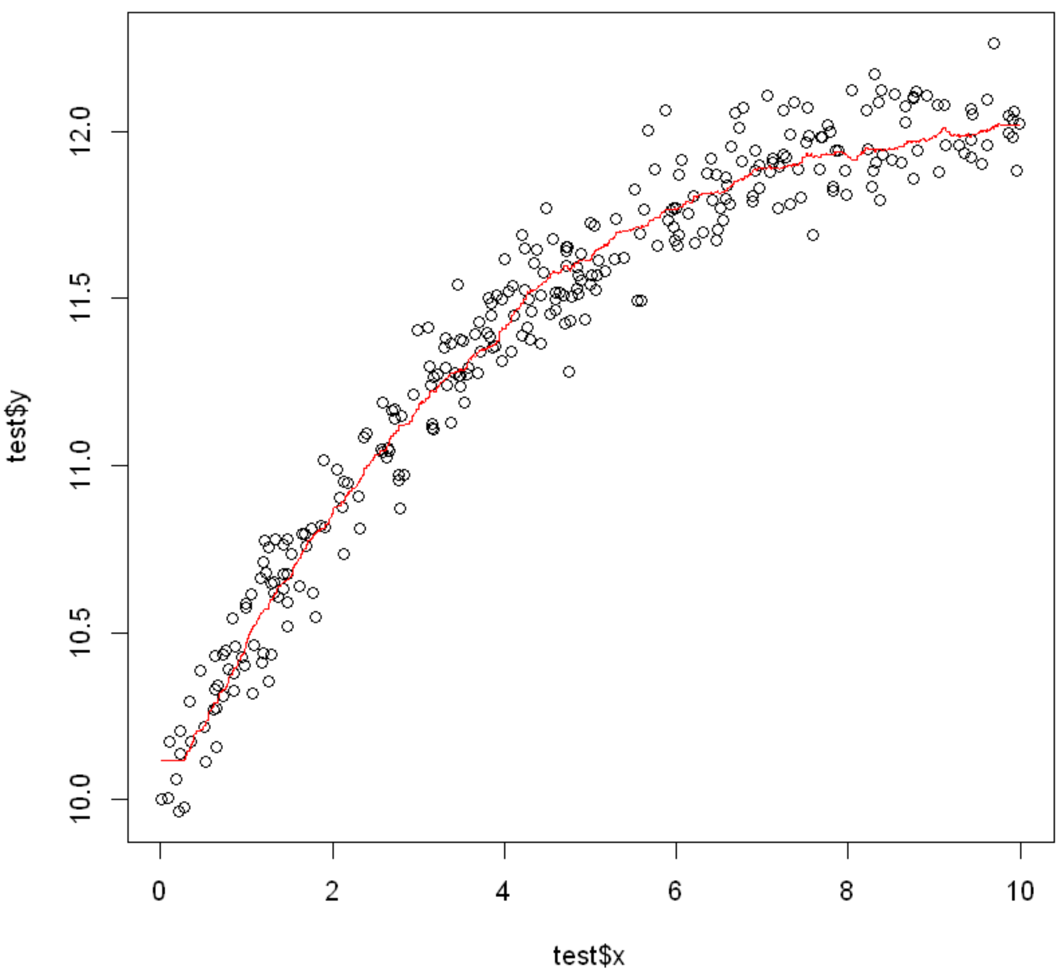
0.0117155454070362

mse is 0.011755454070362

13 k=37 model

```
[33]: fit=knn.reg(train = as.data.frame(train$x),test=as.data.
       ↪frame(test$x),y=train$y,k=37)
      mse37=mean((test$y - fit$pred)^2)
```

```
x <- with(test, seq(min(x), max(x), length.out=2000))
fit2=knn.reg(train = as.data.frame(train$x),test=as.data.
 ↪frame(x),y=train$y,k=37)
y <- fit2$pred
plot(test$x,test$y)
lines(x, y, col = "red")
```



[34]:

```
mse37
```

0.0116169596371103

mse is 0.011616596371103

[35]:

```
mse16
mse28
mse37
```

0.011866644165692

0.0117155454070362

0.0116169596371103

14 So k=37 model is the best knn model

[ ]:

```
```