KNN Regression and Evaluation

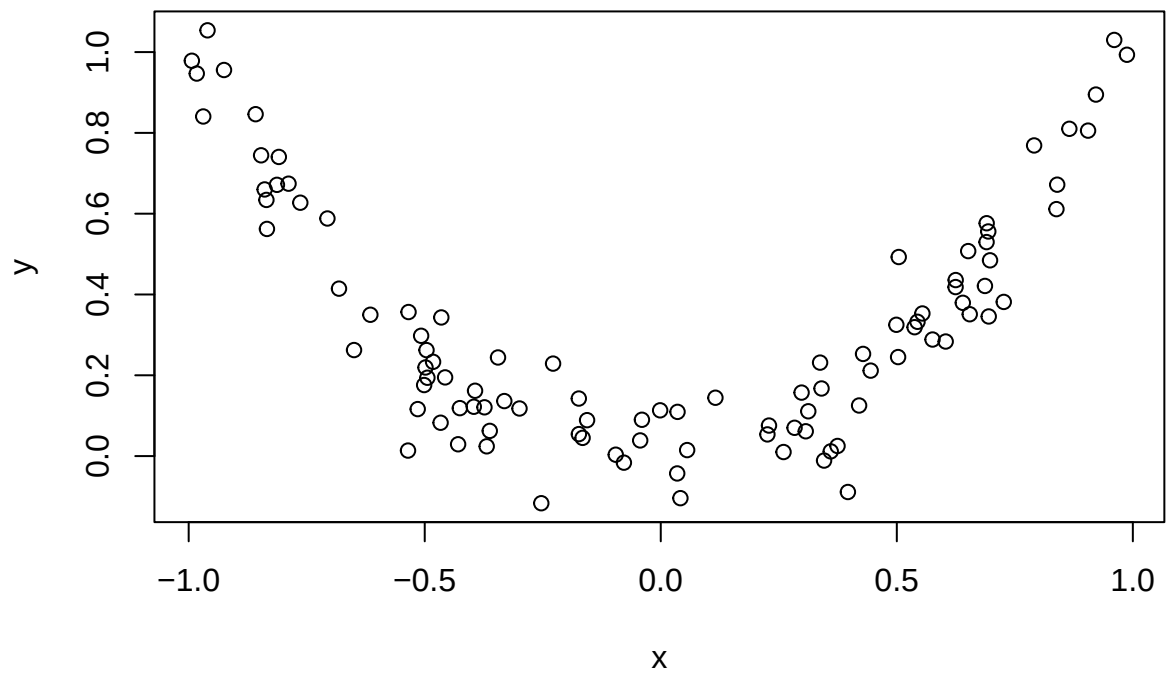
```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

simulation

```
x = runif(100,-1,1)
y = x^2 # f(x) = x^2
e = rnorm(100,0,1/10)
y = y + e
plot(x,y)
```




```
df = data.frame(x=x,y=y)
```

```
flds = createFolds(1:nrow(df),k=2)
flds
```

```
## $Fold1
##  [1]  4  5  7  8 10 13 15 17 20 22 23 24 25 26 28 30 35 36 37 38 40 46 47 48 50
## [26] 51 57 59 61 62 63 66 67 70 72 74 75 76 77 78 79 81 82 85 87 92 93 95 96 97
##
## $Fold2
##  [1]   1   2   3   6   9  11  12  14  16  18  19  21  27  29  31  32  33  34  39
## [20]  41  42  43  44  45  49  52  53  54  55  56  58  60  64  65  68  69  71  73
## [39]  80  83  84  86  88  89  90  91  94  98  99 100
```

test/train split

```
test_df = df[flds[[1]],]
train_df = df[flds[[2]],]
```

```
dim(test_df)
```

```
## [1] 50  2
```

```
dim(train_df)
```

```
## [1] 50  2
```

```
# build on the testing data
knn_mod = knnreg(y~.,data=train_df,k=5)
```

```
train_preds = predict(knn_mod,train_df)
```

```
RMSE_train = sqrt(mean((train_df$y-train_preds)^2))
RMSE_train
```

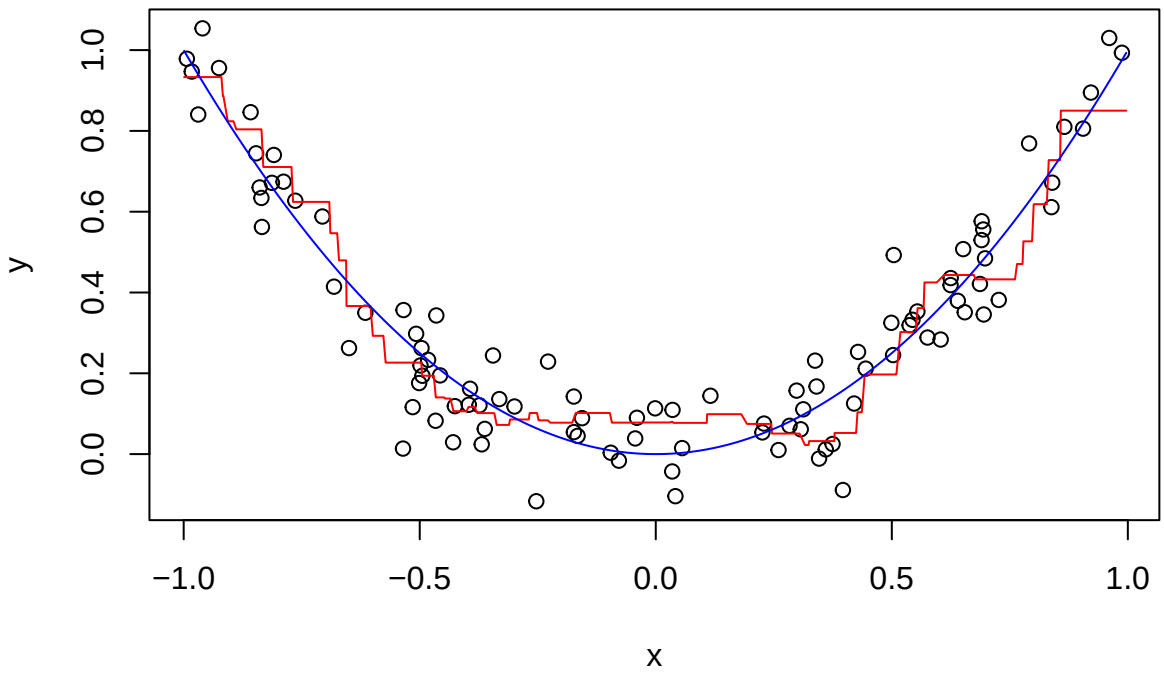
```
## [1] 0.0921929
```

```
test_preds = predict(knn_mod,test_df)
```

```
RMSE_test = sqrt(mean((test_df$y-test_preds)^2))
RMSE_test
```

```
## [1] 0.1058891
```

```
plot(x,y)
xe = data.frame(x=sort(runif(1000,-1,1)))
lines(xe$x,predict(knn_mod,xe),col='red')
lines(xe$x,xe$x^2,col='blue')
```



notice how the training RMSE is typically loweer than the testing RMSE

k fold cross validation

```
flds = createFolds(1:nrow(df),k=5)
flds
```

```
## $Fold1
##  [1]  2  8  9 11 12 26 27 31 36 40 51 53 60 62 73 79 84 87 89 94
##
## $Fold2
##  [1]   3   4   5  18  19  37  41  42  43  45  54  56  61  65  69  76  90  93  99
## [20] 100
##
## $Fold3
##  [1] 14 21 23 24 25 29 32 46 47 49 59 63 64 67 70 78 83 92 95 96
##
## $Fold4
##  [1] 10 16 17 20 22 28 30 35 48 50 52 66 68 71 74 81 82 88 91 98
##
## $Fold5
##  [1]  1  6  7 13 15 33 34 38 39 44 55 57 58 72 75 77 80 85 86 97
```

```
lengths(flds)
```

```
## Fold1 Fold2 Fold3 Fold4 Fold5
##    20    20    20    20    20
```

```
i = 1
test_df = df[flds[[i]],]
train_df = df[unlist(flds[-i]),]
```

```
dim(test_df)
```

```
## [1] 20  2
```

```
dim(train_df)
```

```
## [1] 80  2
```

```
knn_mod = knnreg(y~.,data=train_df,k=10)

train_preds = predict(knn_mod,train_df)
RMSE_train = sqrt(mean((train_df$y-train_preds)^2))

test_preds = predict(knn_mod,test_df)
RMSE_test = sqrt(mean((test_df$y-test_preds)^2))

RMSE_train
```

```
## [1] 0.1084764
```

```
RMSE_test
```

```
## [1] 0.1553873
```

let's put this in a function

```
tt_split_eval = function(train_idx,test_idx){
    test_df = df[test_idx,]
    train_df = df[train_idx,]

    knn_mod = knnreg(y~.,data=train_df,k=5)

    train_preds = predict(knn_mod,train_df)
    RMSE_train = sqrt(mean((train_df$y-train_preds)^2))

    test_preds = predict(knn_mod,test_df)
    RMSE_test = sqrt(mean((test_df$y-test_preds)^2))

    return(data.frame(train=RMSE_train,
        test=RMSE_test
                ))
}
```

```
flds = createFolds(1:nrow(df),k=10)
rmses = lapply(1:length(flds),function(i){
    tdf = tt_split_eval(train_idx = unlist(flds[-i]),test_idx = flds[[i]])
    tdf$i = i
    return(tdf)
})
```

```
rmses[[1]]
```

```
##        train       test i
## 1 0.08515863 0.08253136 1
```

```
rmses[[2]]
```

```
##        train       test i
## 1 0.08216648 0.08965276 2
```

```
RMSE = Reduce('rbind',rmses)
```

```
RMSE
```

```
##         train       test  i
## 1  0.08515863 0.08253136  1
## 2  0.08216648 0.08965276  2
## 3  0.08246252 0.10073391  3
## 4  0.08561567 0.09095266  4
## 5  0.08312587 0.11957416  5
## 6  0.08523740 0.08301801  6
## 7  0.07750446 0.12474830  7
## 8  0.08022885 0.15453855  8
## 9  0.08129165 0.11774150  9
## 10 0.08622975 0.05582486 10
```

```
library('reshape2')
```

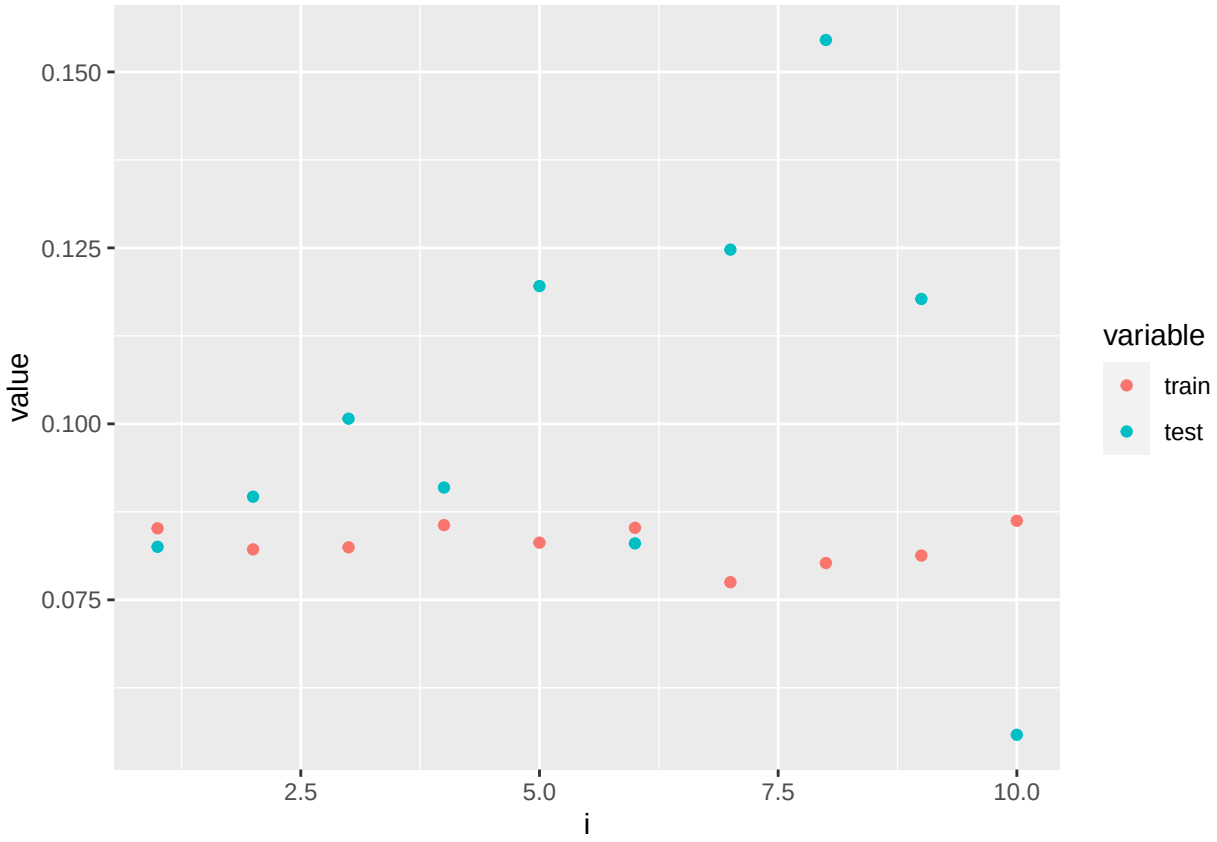
```
mRMSE = reshape2::melt(RMSE,id.vars='i')
mRMSE
```

```
##     i variable      value
## 1   1    train 0.08515863
## 2   2    train 0.08216648
## 3   3    train 0.08246252
## 4   4    train 0.08561567
## 5   5    train 0.08312587
## 6   6    train 0.08523740
## 7   7    train 0.07750446
## 8   8    train 0.08022885
## 9   9    train 0.08129165
## 10 10    train 0.08622975
## 11  1     test 0.08253136
```

```
## 12  2     test 0.08965276
## 13  3     test 0.10073391
## 14  4     test 0.09095266
## 15  5     test 0.11957416
## 16  6     test 0.08301801
## 17  7     test 0.12474830
## 18  8     test 0.15453855
## 19  9     test 0.11774150
## 20 10     test 0.05582486
```

```
library('ggplot2')
```

```
ggplot(data=mRMSE,mapping=aes(x=i,y=value,color=variable))+
    geom_point()
```



in total summary we can summarize the RMSEs

```
median(RMSE$test)
```

```
## [1] 0.09584329
```

How can we use this to choose a value of k for KNN? Use a train/validate/test 3-way split

```
flds = createFolds(1:nrow(df),k=10)
flds
```

```
## $Fold01
##  [1]   6  12  23  31  40  41  70  74  89  93 100
##
## $Fold02
##  [1]  3 13 15 34 37 57 64 79 90 94
##
## $Fold03
##  [1] 22 24 25 36 48 55 67 71 80 91 99
##
## $Fold04
## [1] 20 21 28 42 43 61 66 77 95
##
## $Fold05
## [1]  7 11 29 44 52 72 73 84 86
##
## $Fold06
## [1] 10 16 45 49 53 54 78 85 88
##
## $Fold07
##  [1]  1  2 19 30 38 39 56 75 76 92
##
## $Fold08
##  [1]  8 18 32 35 46 58 63 69 83 87
##
## $Fold09
##  [1]  4  5  9 27 50 51 62 68 82 96 98
##
## $Fold10
##  [1] 14 17 26 33 47 59 60 65 81 97
```

```
i = 1
test_idx = flds[[i]]
trainval_idx = unlist(flds[-i])
```

```
test_df = df[test_idx,]
trainval_df = df[trainval_idx,]
```

```
dim(test_df)
```

```
## [1] 11  2
```

```
dim(trainval_df)
```

```
## [1] 89  2
```

```
tv_flds = createFolds(1:nrow(trainval_df),k=10)
tv_flds
```

```
## $Fold01
## [1] 12 19 33 44 55 60 75 77
##
## $Fold02
## [1] 10 11 16 31 37 59 66 68 79
##
## $Fold03
## [1]  4 13 27 39 47 54 70 73
##
## $Fold04
## [1]  1 17 30 41 53 57 67 76 83
##
## $Fold05
##  [1]  6  7  8 24 36 48 52 84 88 89
##
## $Fold06
## [1] 14 21 22 25 35 50 61 71 78
##
## $Fold07
## [1]  2  3 34 45 49 62 74 80
##
## $Fold08
## [1]  9 20 38 40 43 51 58 69 82
##
## $Fold09
##  [1]  5 15 28 29 46 64 65 72 85 87
##
## $Fold10
## [1] 18 23 26 32 42 56 63 81 86
```

```
j=1
val_idx = tv_flds[[j]]
train_idx = unlist(tv_flds[-j])
```

```
train_df = trainval_df[train_idx,]
val_df = trainval_df[val_idx,]
```

```
dim(train_df)
```

```
## [1] 81  2
```

```
dim(val_df)
```

```
## [1] 8 2
```

```
tt_split_eval_k = function(train_idx,val_idx,k=1){
    train_df = trainval_df[train_idx,]
    val_df = trainval_df[val_idx,]

    knn_mod = knnreg(y~.,data=train_df,k=k)

    train_preds = predict(knn_mod,train_df)
```

```
    RMSE_train = sqrt(mean((train_df$y-train_preds)^2))

    val_preds = predict(knn_mod,val_df)
    RMSE_val = sqrt(mean((val_df$y-val_preds)^2))

    return(data.frame(train=RMSE_train,
        val=RMSE_val
                ))
}
```

```
tt_split_eval_k(train_idx,val_idx,k=5)
```

```
##        train       val
## 1 0.08333261 0.1353021
```

```
tt_split_eval_k(train_idx,val_idx,k=10)
```

```
##      train       val
## 1 0.096993 0.1922101
```

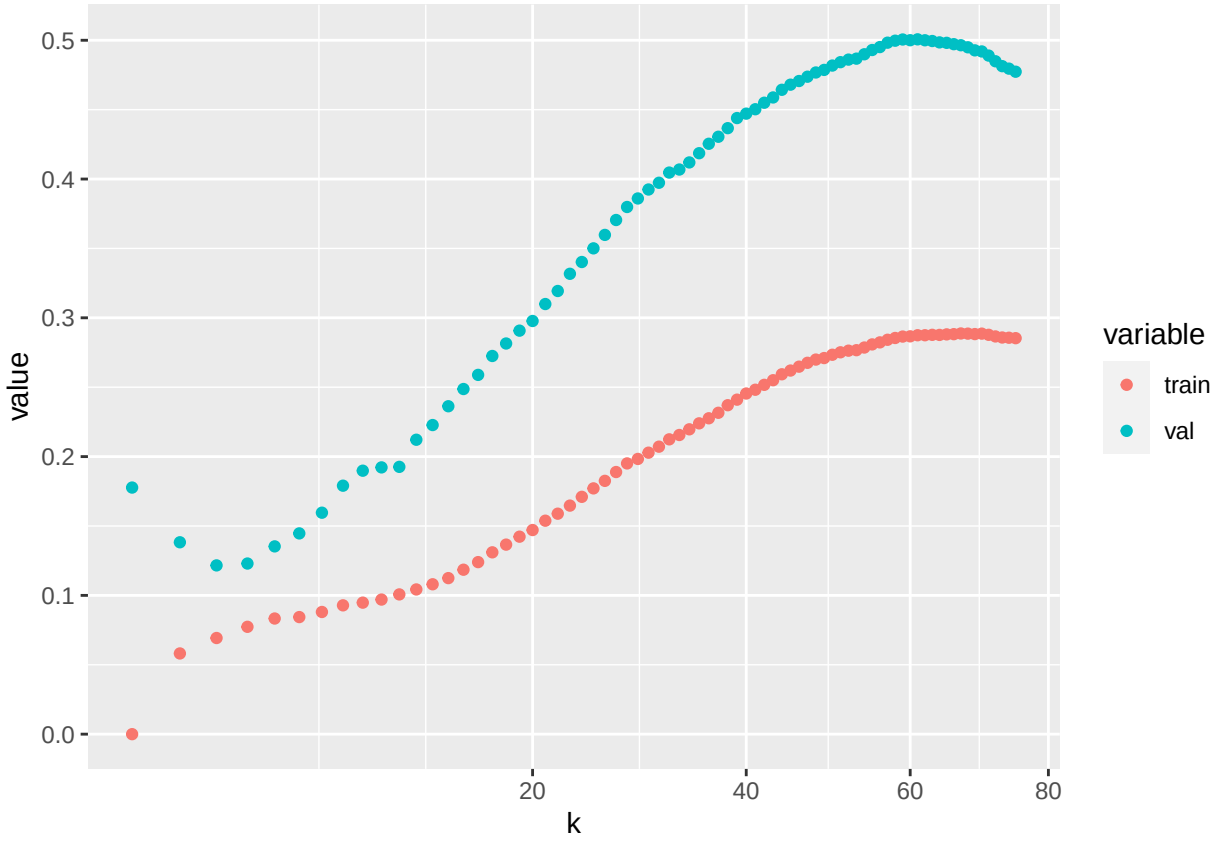
```
RMSE = lapply(1:75,function(k){
    tdf = tt_split_eval_k(train_idx,val_idx,k=k)
    tdf$k = k
    return(tdf)
})
RMSE = Reduce('rbind',RMSE)
head(RMSE)
```

```
##        train       val k
## 1 0.00000000 0.1776882 1
## 2 0.05814963 0.1382572 2
## 3 0.06928599 0.1216110 3
## 4 0.07734408 0.1229542 4
## 5 0.08333261 0.1353021 5
## 6 0.08437332 0.1446567 6
```

```
library(reshape2)
mRMSE = melt(RMSE,id.vars='k')
head(mRMSE)
```

```
##   k variable      value
## 1 1    train 0.00000000
## 2 2    train 0.05814963
## 3 3    train 0.06928599
## 4 4    train 0.07734408
## 5 5    train 0.08333261
## 6 6    train 0.08437332
```

```
ggplot(data=mRMSE,mapping=aes(x=k,y=value,color=variable))+
    geom_point()+
    scale_x_sqrt()
```




```
which.min(RMSE$val)
```

```
## [1] 3
```

```
min_df = RMSE[which.min(RMSE$val),]
min_df
```

```
##        train      val k
## 3 0.06928599 0.121611 3
```

```
knn_mod = knnreg(y~.,data=trainval_df,k=min_df$k)
```

```
test_preds = predict(knn_mod,test_df)
RMSE_val = sqrt(mean((test_df$y-test_preds)^2))
RMSE_val
```

```
## [1] 0.09879314
```

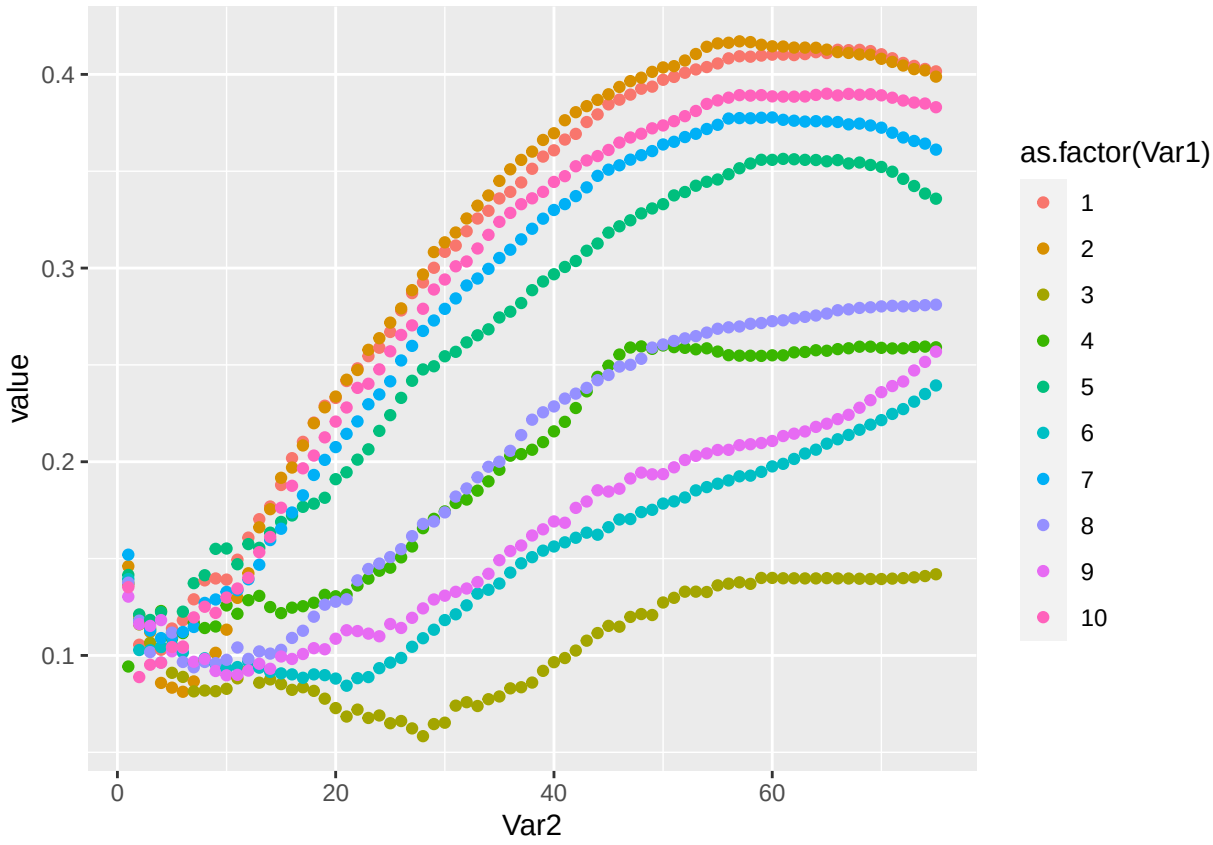
can I do this in a cross validated way? yes use nested cross validation!

```
# outer loop = split into test and trainval datasets
# inner loop = MBP, split into train/val and search over k
```

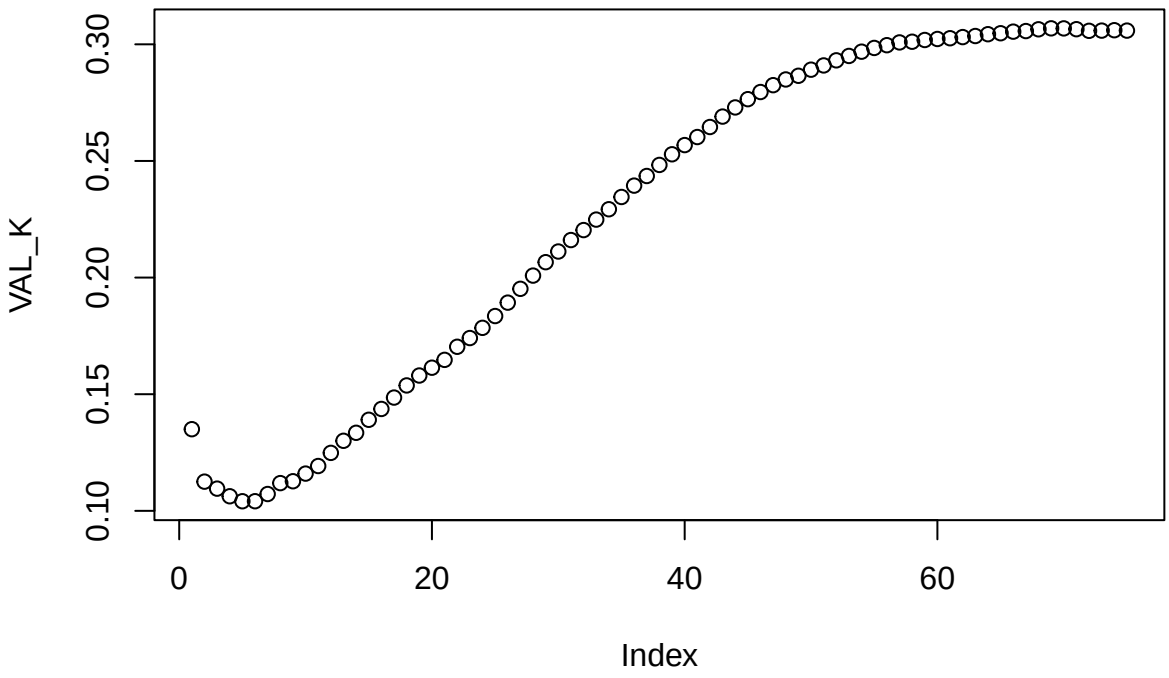
```
TEST_RMSE = rep(NA,length(flds))

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

    # split testing from trainval
    test_idx = flds[[i]]
    trainval_idx = unlist(flds[-i])
    test_df = df[test_idx,]
    trainval_df = df[trainval_idx,]

    #MODEL BUILDING PROCESS
    tv_flds = createFolds(1:nrow(trainval_df),k=10)

    K_seq = seq(1,75)
    VAL_MTX = array(NA,c(length(tv_flds),length(K_seq)))

    for(j in 1:length(tv_flds)){

        val_idx = tv_flds[[j]]
        train_idx = unlist(tv_flds[-j])
        train_df = trainval_df[train_idx,]
        val_df = trainval_df[val_idx,]

        for(k in K_seq){
            knn_mod = knnreg(y~.,data=train_df,k=k)
            val_preds = predict(knn_mod,val_df)
            VAL_MTX[j,k] = sqrt(mean((val_df$y-val_preds)^2))
        }
    }

    VAL_K = apply(VAL_MTX,2,mean)
    K_hat = K_seq[which.min(VAL_K)]

    knn_mod = knnreg(y~.,data=trainval_df,k=K_hat)

    # eval on testing data
    test_preds = predict(knn_mod,test_df)
    TEST_RMSE[i] = sqrt(mean((test_df$y-test_preds)^2))
}
```

consider for a single run:

```
ggplot(data=melt(VAL_MTX),mapping=aes(x=Var2,y=value,color=as.factor(Var1)))+geom_point()
```




```
plot(VAL_K)
```




```
K_hat = K_seq[which.min(VAL_K)]
K_hat
```

```
## [1] 5
```

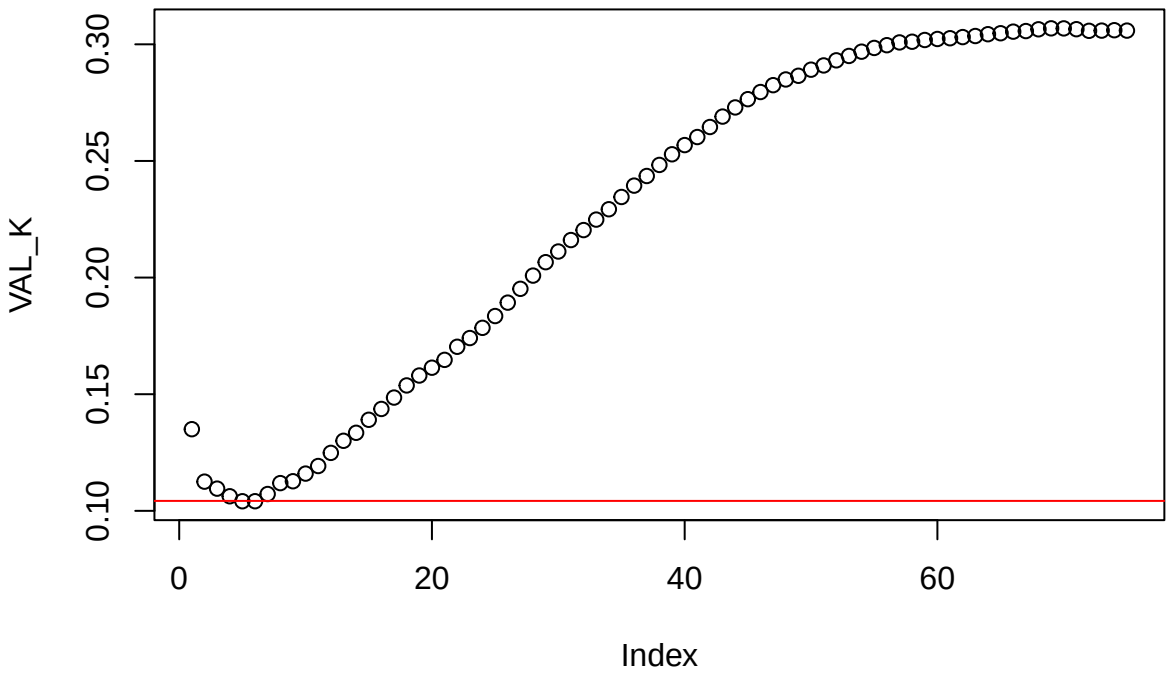
```
# overall

TEST_RMSE
```

```
##  [1] 0.11238640 0.10363781 0.11900934 0.09641749 0.06929462 0.12274918
##  [7] 0.12160159 0.08702347 0.12211825 0.08840863
```

```
mean(TEST_RMSE)
```

```
## [1] 0.1042647
```

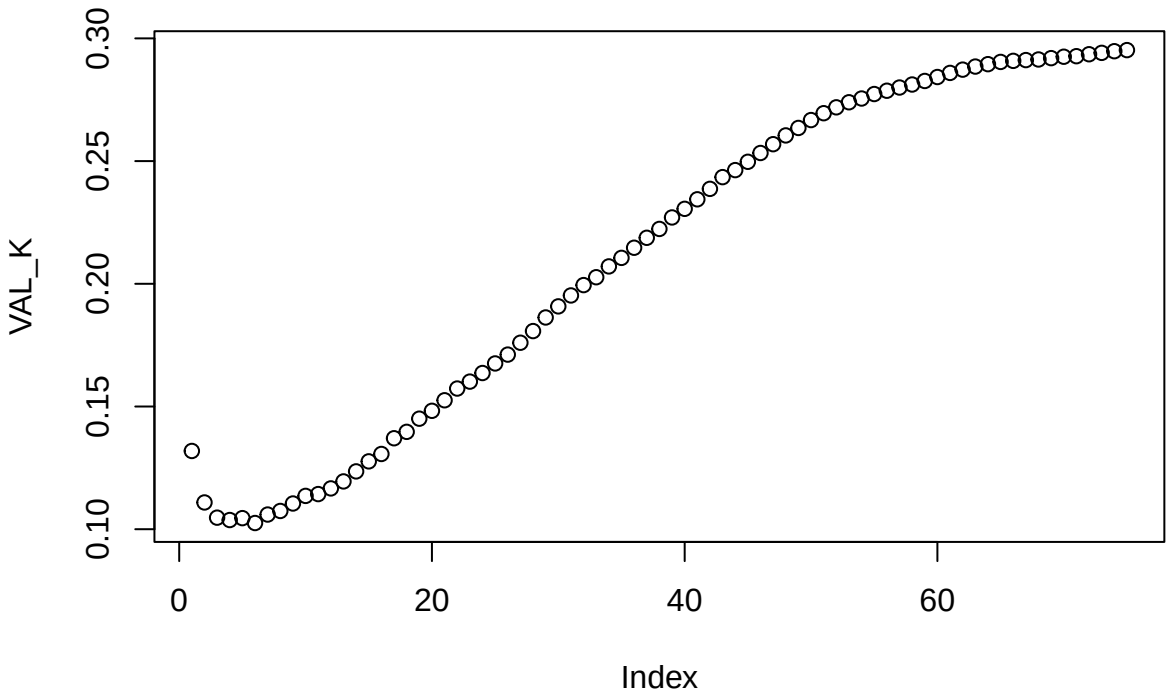
```
plot(VAL_K)
abline(h=mean(TEST_RMSE),col='red')
```



How do we build the final model for prediction? Basically pull out the inner loop

```
tv_flds = createFolds(1:nrow(df),k=10) # use all df

K_seq = seq(1,75)
VAL_MTX = array(NA,c(length(tv_flds),length(K_seq)))

for(j in 1:length(tv_flds)){

    val_idx = tv_flds[[j]]
    train_idx = unlist(tv_flds[-j])
    train_df = df[train_idx,]
    val_df = df[val_idx,]

    for(k in K_seq){
        knn_mod = knnreg(y~.,data=train_df,k=k)
        val_preds = predict(knn_mod,val_df)
        VAL_MTX[j,k] = sqrt(mean((val_df$y-val_preds)^2))
    }
}

VAL_K = apply(VAL_MTX,2,mean)
K_hat = K_seq[which.min(VAL_K)]

# fit with all the data
knn_mod = knnreg(y~.,data=df,k=K_hat)
```

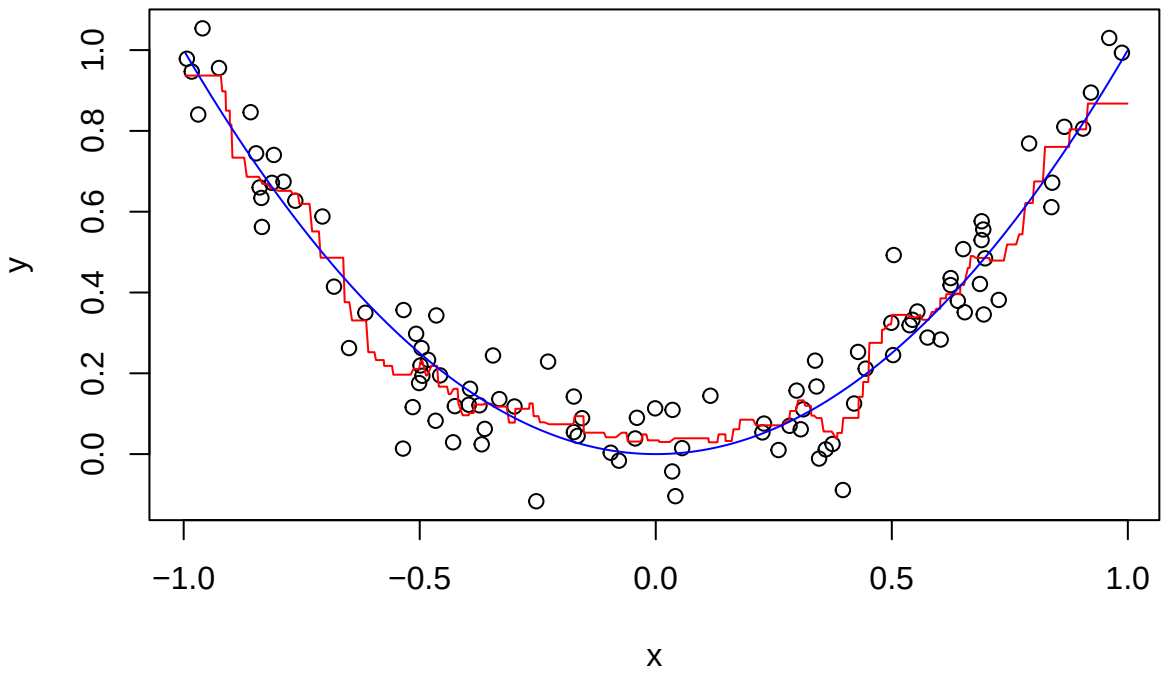
```
plot(VAL_K)
```




```
K_hat
```

```
## [1] 6
```

```
plot(x,y)
xe = data.frame(x=sort(runif(1000,-1,1)))
lines(xe$x,predict(knn_mod,xe),col='red')
lines(xe$x,xe$x^2,col='blue')
```


y
0.0
0.2
0.4
0.6
0.8
1.0
−1.0
−0.5
0.0
0.5
1.0
x