# K-Nearest neighbors

Chapter 7-Part II Regression Models

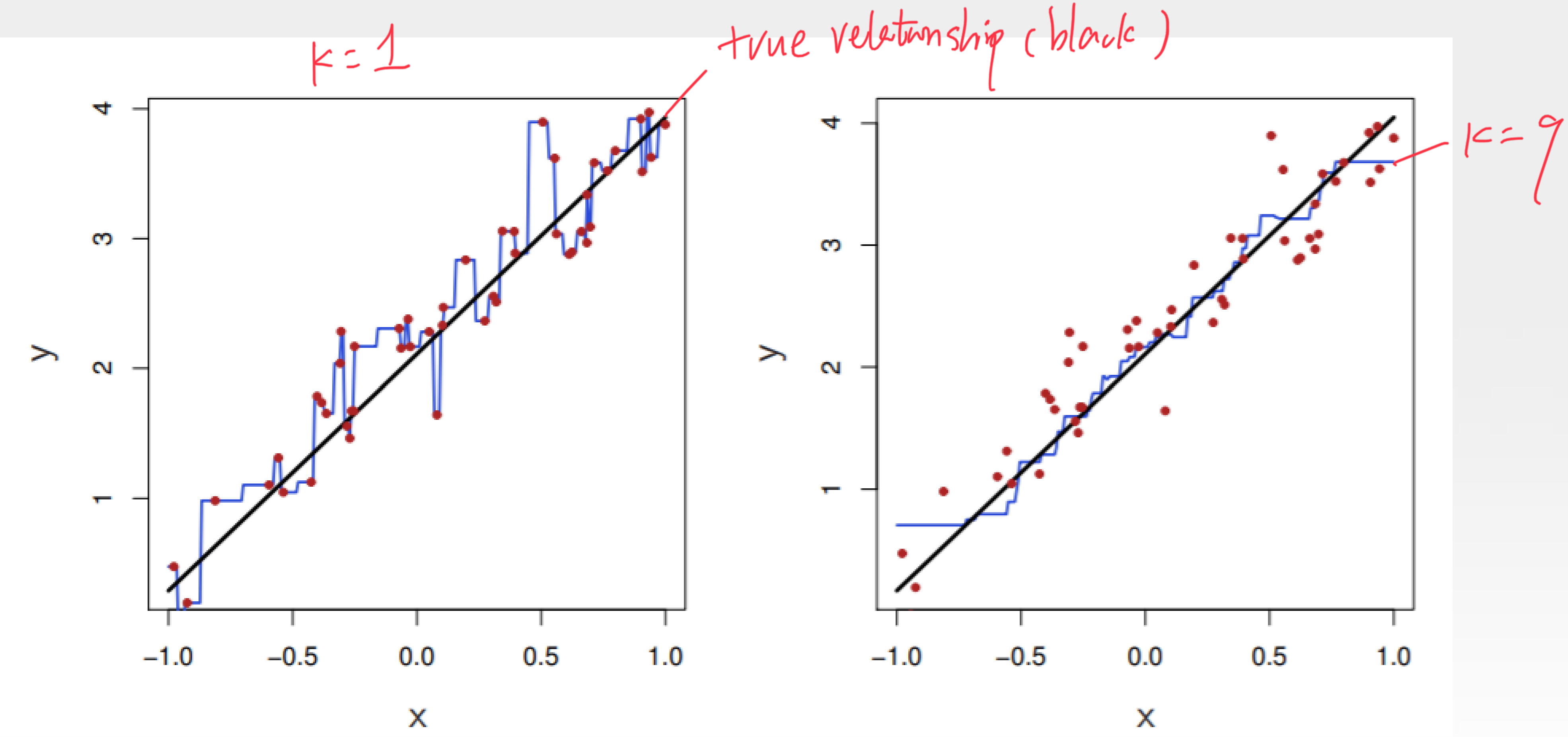
# K-Nearest Neighbors (KNN) regression

- KNN regression is one of the *simplest and best-known nonparametric* methods using the K-closest samples from the training set.
- Given a value for *K* and a prediction point $x_0$, KNN regression first identifies the *K* training observations that are closest to $x_0$, represented by $N_0$.
- It then estimates the nonparametric function $f(x_0)$ using the average of all the training responses in $N_0$, that is

$$\hat{f}(x_0) = \frac{1}{K} \sum_{\underline{x_i \epsilon N_0}} y_i$$

- In general, the optimal value for *K* depends on the *bias-variance tradeoff*. A small value of *K* provides the most flexible fit, which will have low bias but high variance (i.e., over-fit).

# KNN fits for K = 1 and K = 9 one-dimensional data



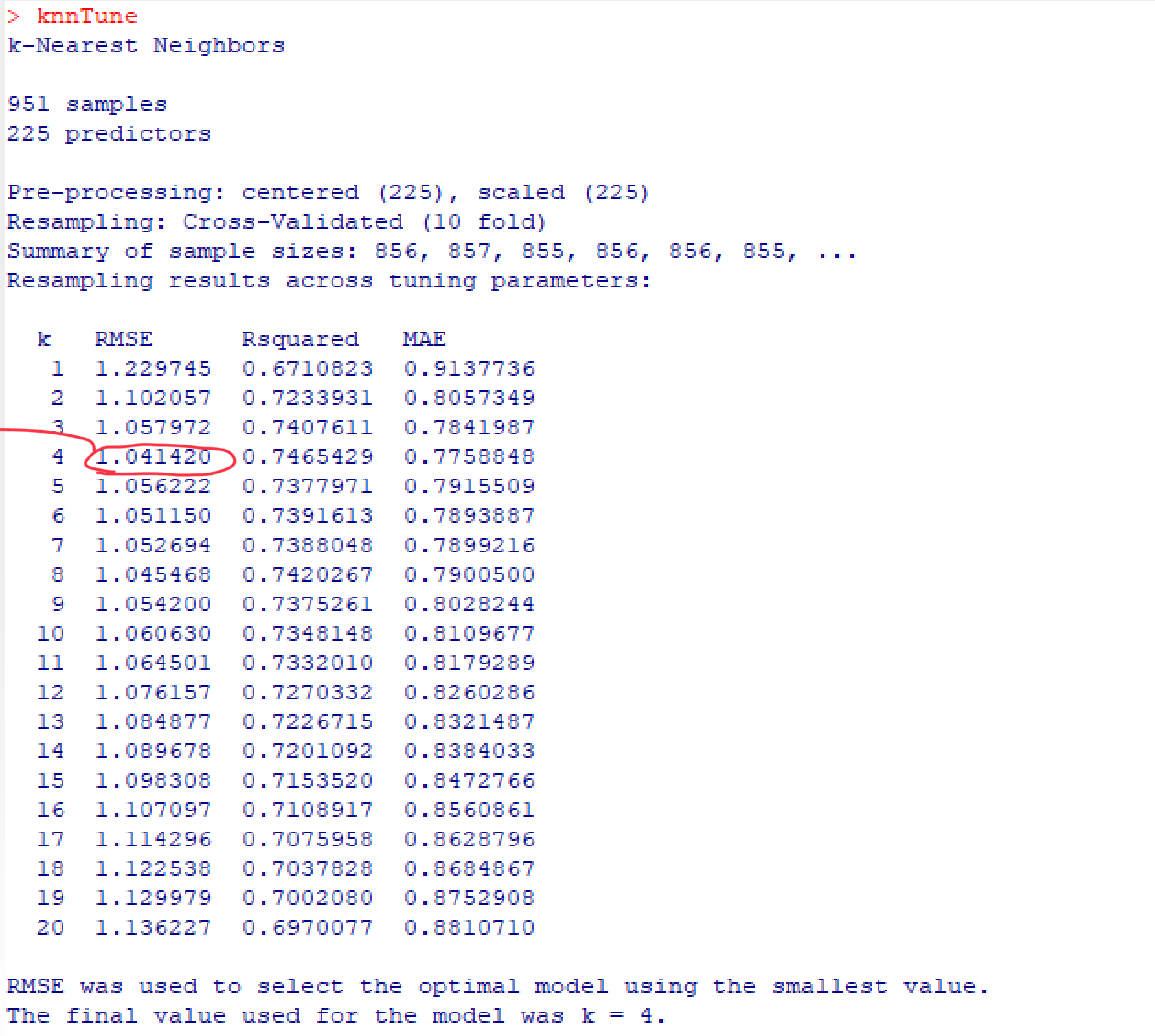
- The tuning parameter K can be determined by *resampling*.

# K-Nearest Neighbors

```
### K-Nearest Neighbors
### First we remove near-zero variance predictors
knnDescr <- solTrainXtrans[, -nearZeroVar(solTrainXtrans)]
ptm <- proc.time() # takes 39.86 seconds to run
set.seed(100)
knnTune <- train(x = knnDescr, y = solTrainY,
            method = "knn",
            preProc = c("center", "scale"),
            tuneGrid = data.frame(k = 1:20),
            trControl = ctrl)

knnTune
proc.time() - ptm

plot(knnTune)

testResults$Knn <- predict(knnTune, solTestXtrans[, names(knnDescr)])
```

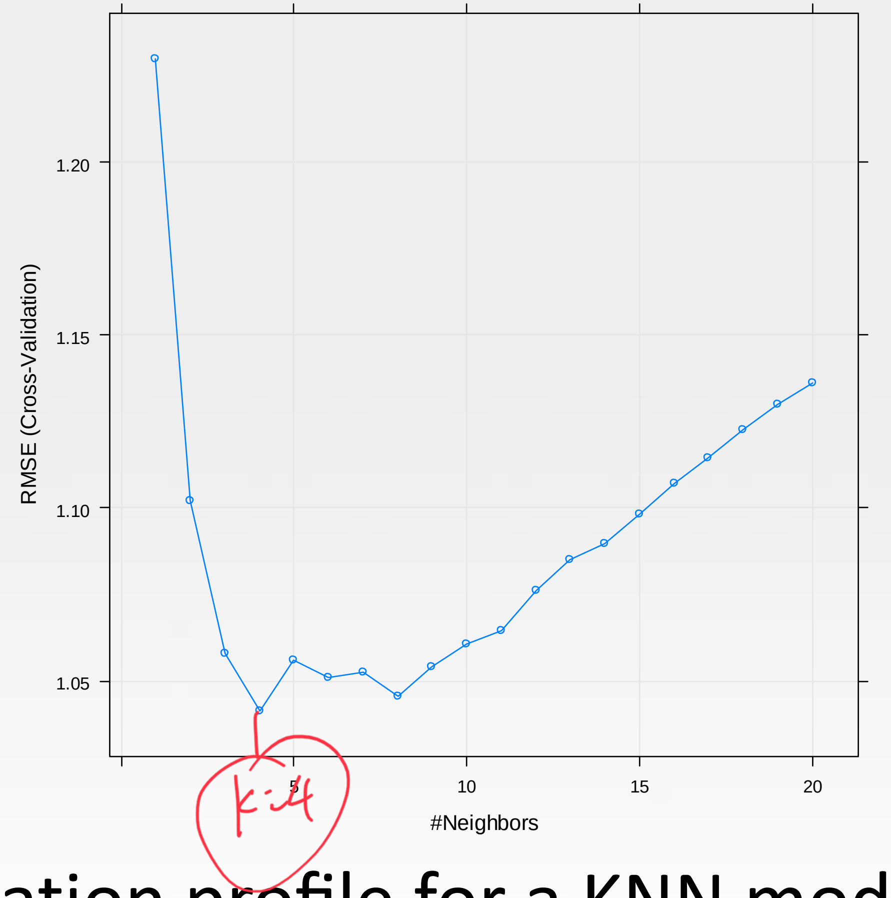
# The tuning parameter K

The smallest RMSE indicates that the optimal value of k = 4



```
> knnTune
k-Nearest Neighbors

951 samples
225 predictors

Pre-processing: centered (225), scaled (225)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 856, 857, 855, 856, 856, 855, ...
Resampling results across tuning parameters:

  k   RMSE      Rsquared   MAE
   1  1.229745  0.6710823  0.9137736
   2  1.102057  0.7233931  0.8057349
   3  1.057972  0.7407611  0.7841987
   4  1.041420  0.7465429  0.7758848
   5  1.056222  0.7377971  0.7915509
   6  1.051150  0.7391613  0.7893887
   7  1.052694  0.7388048  0.7899216
   8  1.045468  0.7420267  0.7900500
   9  1.054200  0.7375261  0.8028244
  10  1.060630  0.7348148  0.8109677
  11  1.064501  0.7332010  0.8179289
  12  1.076157  0.7270332  0.8260286
  13  1.084877  0.7226715  0.8321487
  14  1.089678  0.7201092  0.8384033
  15  1.098308  0.7153520  0.8472766
  16  1.107097  0.7108917  0.8560861
  17  1.114296  0.7075958  0.8628796
  18  1.122538  0.7037828  0.8684867
  19  1.129979  0.7002080  0.8752908
  20  1.136227  0.6970077  0.8810710

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 4.
```

# The tuning parameter K



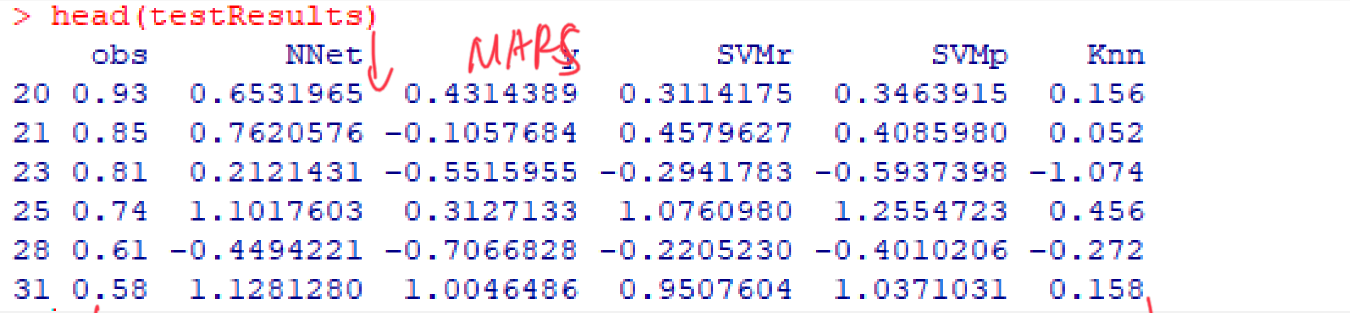
- The RMSE cross-validation profile for a KNN model applied to the solubility data. *The optimal number of neighbors is 4*

# Linear regression vs. KNN

- The linear regression outperforms KNN if the true form of *f* is close to linear.
- KNN outperforms the linear regression if the true form of *f* is strongly non-linear.
- In higher dimensions, say, $p \geq 4$, KNN often performs worse than linear regression due to curse of dimensionality of KNN.
- **A general rule**: Parametric methods will tend to outperform nonparametric approaches when there is a small number of observations per predictor.

# Predicted values based on different models

```
#print out the predicted values based on different models
head(testResults)
```



```
> head(testResults)
    obs        NNet          y       SVMr       SVMp    Knn
20 0.93  0.6531965  0.4314389  0.3114175  0.3463915  0.156
21 0.85  0.7620576 -0.1057684  0.4579627  0.4085980  0.052
23 0.81  0.2121431 -0.5515955 -0.2941783 -0.5937398 -1.074
25 0.74  1.1017603  0.3127133  1.0760980  1.2554723  0.456
28 0.61 -0.4494221 -0.7066828 -0.2205230 -0.4010206 -0.272
31 0.58  1.1281280  1.0046486  0.9507604  1.0371031  0.158
```

MARS

true value of y from the test data

wrost for prediction . $p >> 4$

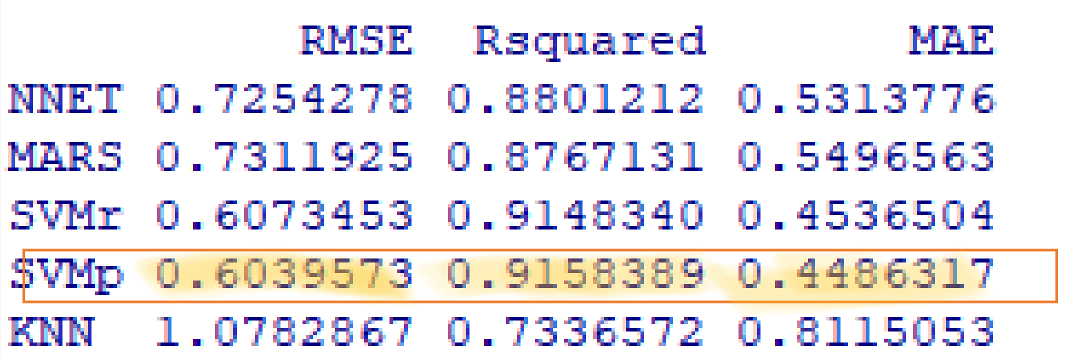
# Performance comparison of nonlinear models

```
#Performance of nonlinear models
set.seed(100)
Nnet.pred = predict(nnetTune, solTestXtrans)
MARS.pred <- predict(marsTune, solTestXtrans)
SVMr.pred <- predict(svmRTune, solTestXtrans)
SVMp.pred <- predict(svmPTune, solTestXtrans)
Knn.pred <- predict(knnTune, solTestXtrans[, names(knnDescr)])

data.frame(rbind(NNET=postResample(pred=Nnet.pred,obs = solTestY),
                 MARS=postResample(pred=MARS.pred ,obs = solTestY),
                 SVMr=postResample(pred=SVMr.pred,obs = solTestY),
                 SVMp=postResample(pred=SVMp.pred,obs = solTestY),
                 KNN=postResample(pred=Knn.pred,obs = solTestY) ))
```

remove columns corresponding near-zero variance.

Specify name of each row

# Performance comparison of nonlinear models



```
          RMSE  Rsquared       MAE
NNET 0.7254278 0.8801212 0.5313776
MARS 0.7311925 0.8767131 0.5496563
SVMr 0.6073453 0.9148340 0.4536504
SVMp 0.6039573 0.9158389 0.4486317
KNN  1.0782867 0.7336572 0.8115053
```

SVM with poly. provides the best result of prediction.

# Introduction to R

Exercise 4