Introduction to Data Science

DSA1101

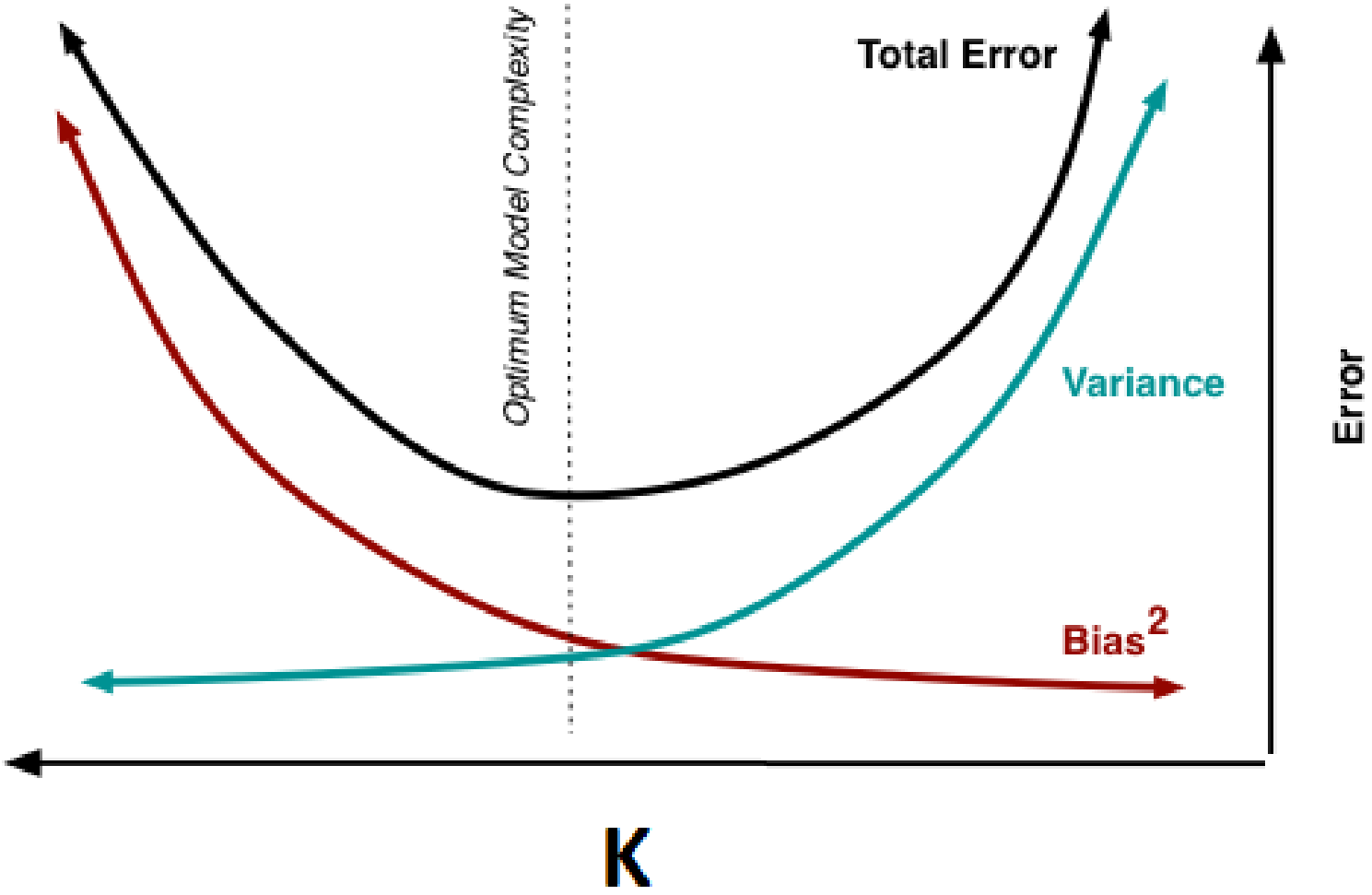
Semester 1, 2018/2019
Week 6

n-fold cross validation in R

n-fold cross validation in R

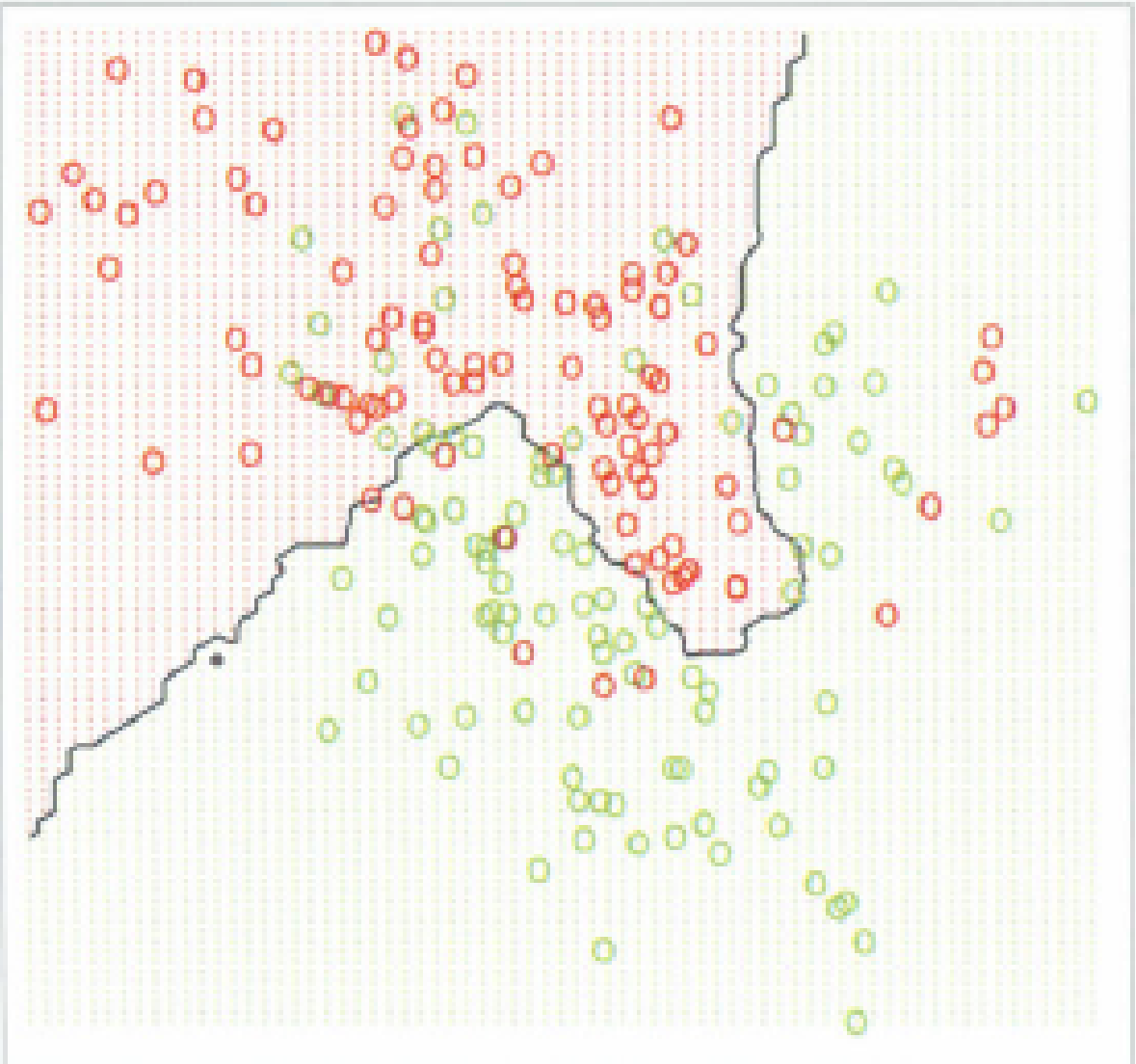
- This week, we will continue to explore n-fold cross validation in R
- We will also examine the R code in more detail

n-fold cross validation in R



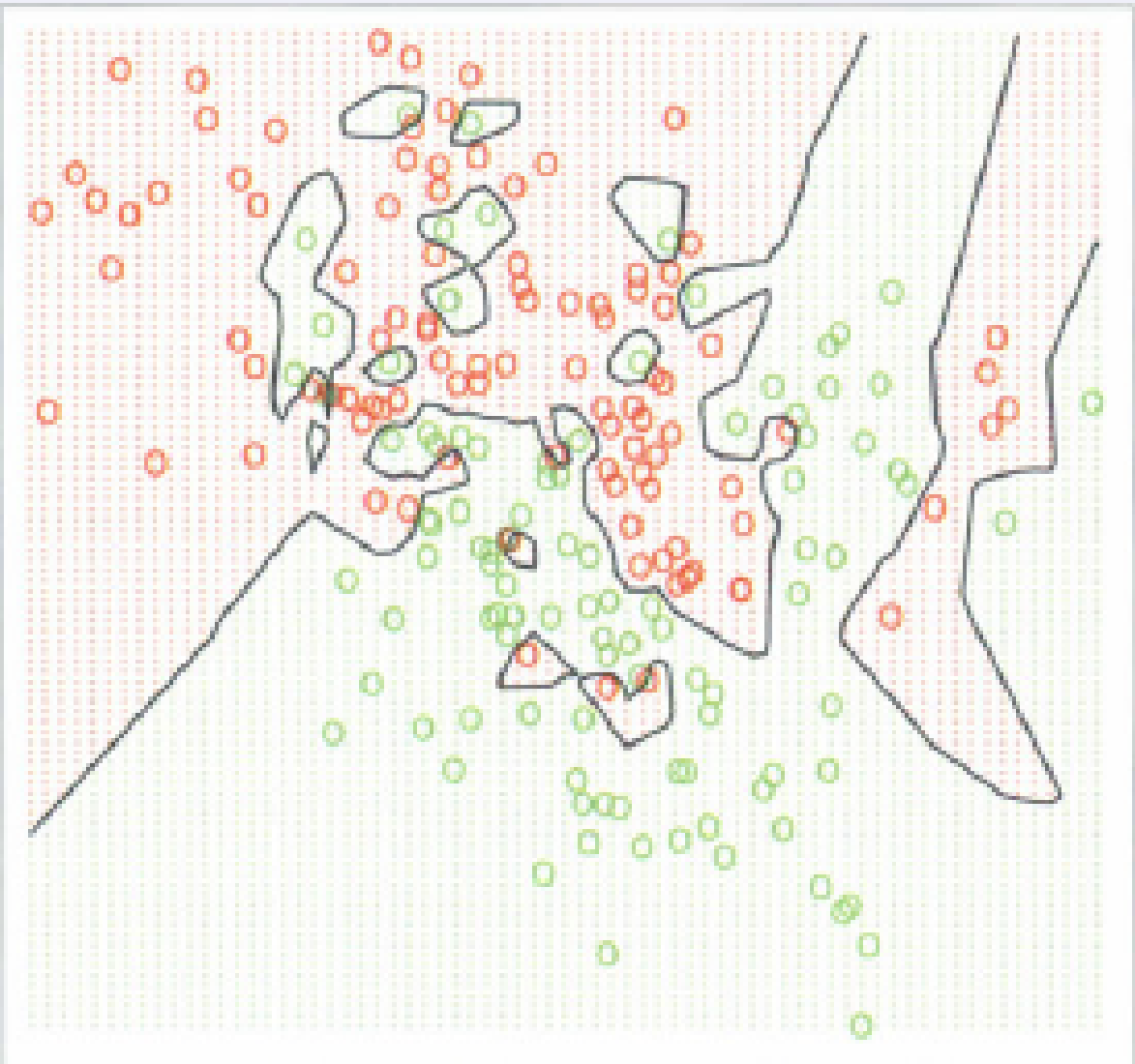
Bias-variance tradeoff. Source: *http://scott.fortmann-roe.com*

n-fold cross validation in R



Prediction by majority vote with 15 nearest neighbors. Source: *The Elements of Statistical Learning*, Hastie et al.

n-fold cross validation in R

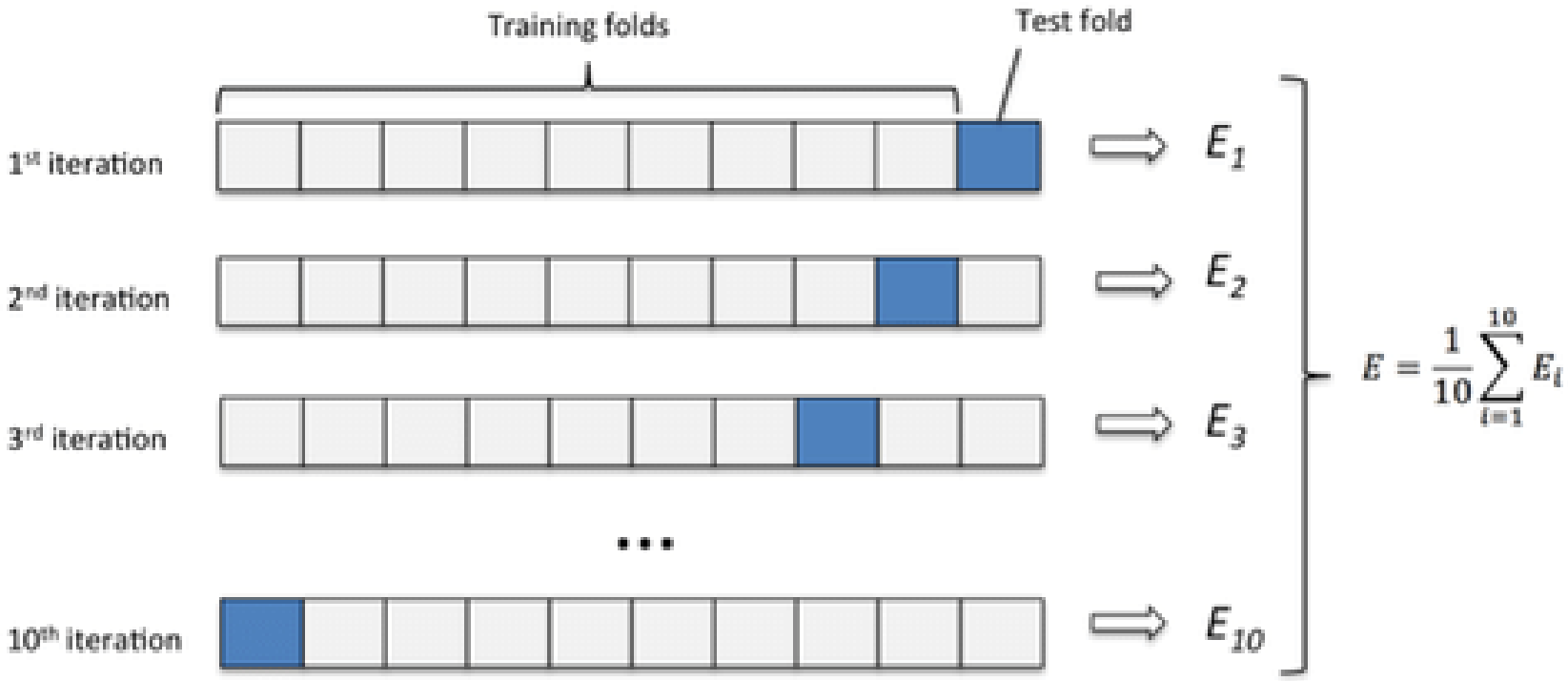


Prediction by majority vote with one nearest neighbor. Source: *The Elements of Statistical Learning*, Hastie et al.

n-fold cross validation in R

- The entire dataset is randomly split into N datasets of approximately equal size.
- N-1 of these datasets are treated as the training dataset, while the remaining one is the test dataset. A measure of the model error is obtained.
- This process is repeated across the various combinations of N datasets taken $N - 1$ at a time.
- The observed N models errors are averaged across the N folds.

Diagnostics of Classifiers



n-fold cross validation in R

- We will illustrate how to implement n-fold cross-validation in R to evaluate the performance of the k-nearest neighbor classifier
- In particular, we will attempt to estimate the optimal value of k (or the optimal model complexity) that will give the best classification performance

n-fold cross validation in R

- We will illustrate n-fold cross validation using the famous Iris Flower Dataset which was first introduced in 1936 by the famous statistician Ronald Fisher and consists of 50 observations from each of three species of Iris (*Iris setosa*, *Iris virginica* and *Iris versicolor*).
- Four features (x) were measured from each sample: the length and the width of the sepals and petals.

n-fold cross validation in R

Source: http://suruchifialoke.com

n-fold cross validation in R

- The dataset has been posted to IVLE as 'iris.csv'
- Also available at `https://archive.ics.uci.edu/ml/datasets/Iris`
- 150 data points with the following attribute information:

(1) sepal length in cm

(2) sepal width in cm

(3) petal length in cm

(4) petal width in cm

(5) class: *Iris Setosa*, *Iris Versicolour* or *Iris Virginica*

n-fold cross validation in R

- Read in the dataset. Note that variable names are not found in the CSV file

```
> iris= read.csv("iris.csv",header=FALSE)
> head(iris)
   V1  V2  V3  V4          V5
1 5.1 3.5 1.4 0.2 Iris-setosa
2 4.9 3.0 1.4 0.2 Iris-setosa
3 4.7 3.2 1.3 0.2 Iris-setosa
4 4.6 3.1 1.5 0.2 Iris-setosa
5 5.0 3.6 1.4 0.2 Iris-setosa
6 5.4 3.9 1.7 0.4 Iris-setosa
```

n-fold cross validation in R

- We can rename the attributes more meaningfully
-

```
> names(iris)
[1] "V1" "V2" "V3" "V4" "V5"
> names(iris)= c("X1","X2","X3","X4","Y")
> names(iris)
[1] "X1" "X2" "X3" "X4" "Y"
> head(iris)
   X1  X2  X3  X4           Y
1 5.1 3.5 1.4 0.2 Iris-setosa
2 4.9 3.0 1.4 0.2 Iris-setosa
3 4.7 3.2 1.3 0.2 Iris-setosa
4 4.6 3.1 1.5 0.2 Iris-setosa
5 5.0 3.6 1.4 0.2 Iris-setosa
6 5.4 3.9 1.7 0.4 Iris-setosa
```

n-fold cross validation in R

- Using the same dataset as both the training and testing data can give misleading results
- For example, when $k = 1$, we use each single data point to predict itself:
-

```
> set.seed(1)
> library(class)
> X=iris[,c("X1","X2","X3","X4")]
> Y=iris[,c("Y")]
> pred <- knn(train=X, test=X, cl=Y, k=1)
> mean(Y != pred)
[1] 0
> mean(Y == pred)
[1] 1
```

- Therefore there is a need for more principled methods to evaluate classifier performance, e.g. n-fold cross validation

n-fold cross validation in R

- Note that the code `mean(Y!=pred)` computes the proportion of data points which label is predicted incorrectly
- A small example:
-

```
> Predicted= c(1,1,0,0,1,1)
> Actual= c(1,1,1,0,0,1)
> Predicted != Actual
[1] FALSE FALSE  TRUE FALSE  TRUE FALSE
> mean(Predicted != Actual)
[1] 0.3333333
> Predicted == Actual
[1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE
> mean(Predicted == Actual)
[1] 0.6666667
```

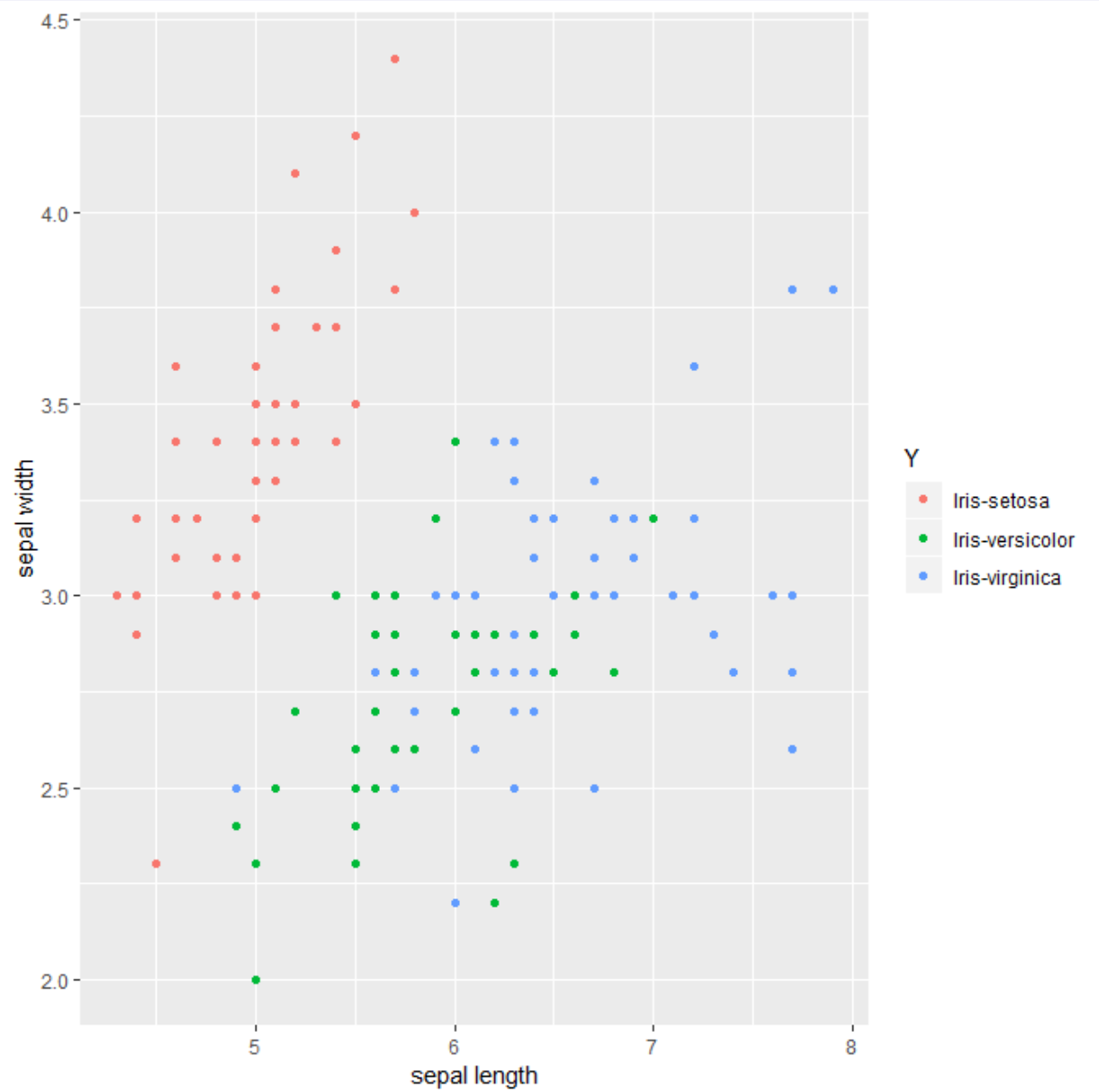
- Similarly, the code `mean(Y==pred)` computes the proportion of data points which label is predicted correctly

n-fold cross validation in R

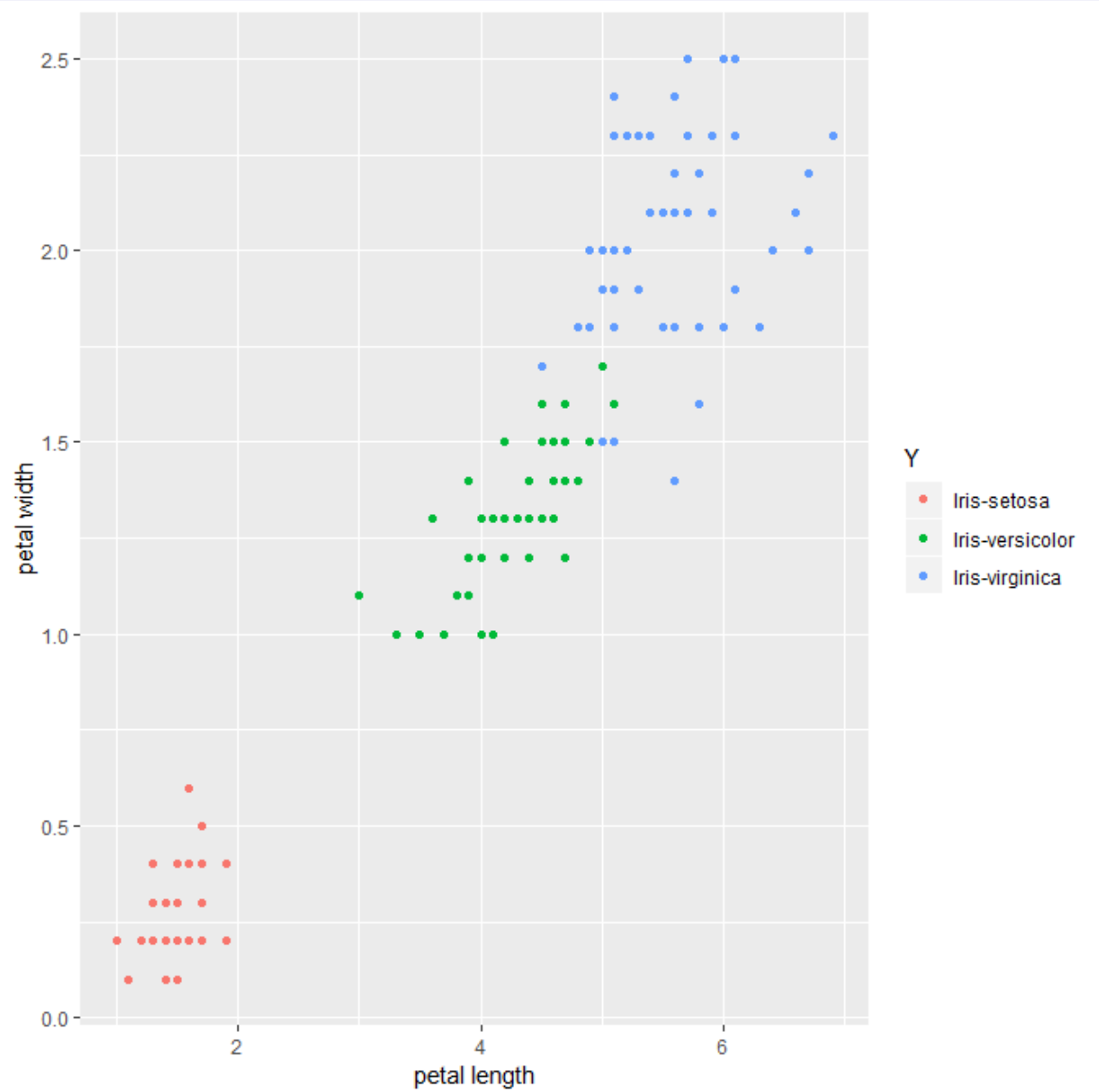
- Before running *n*-fold cross validation, we can try to visualize the label and feature patterns

```
library(ggplot2)
library(magrittr)
# sepal width vs. sepal length
 ggplot(aes(x=X[,1], y=X[,2], color=Y)) +
 geom_point()+
 labs(x = "sepal length")+  labs(y = "sepal width"
     )
# petal width vs. petal length
 ggplot(aes(x=X[,3], y=X[,4], color=Y)) +
 geom_point()+
 labs(x = "petal length")+  labs(y = "petal width"
     )
```

n-fold cross validation in R



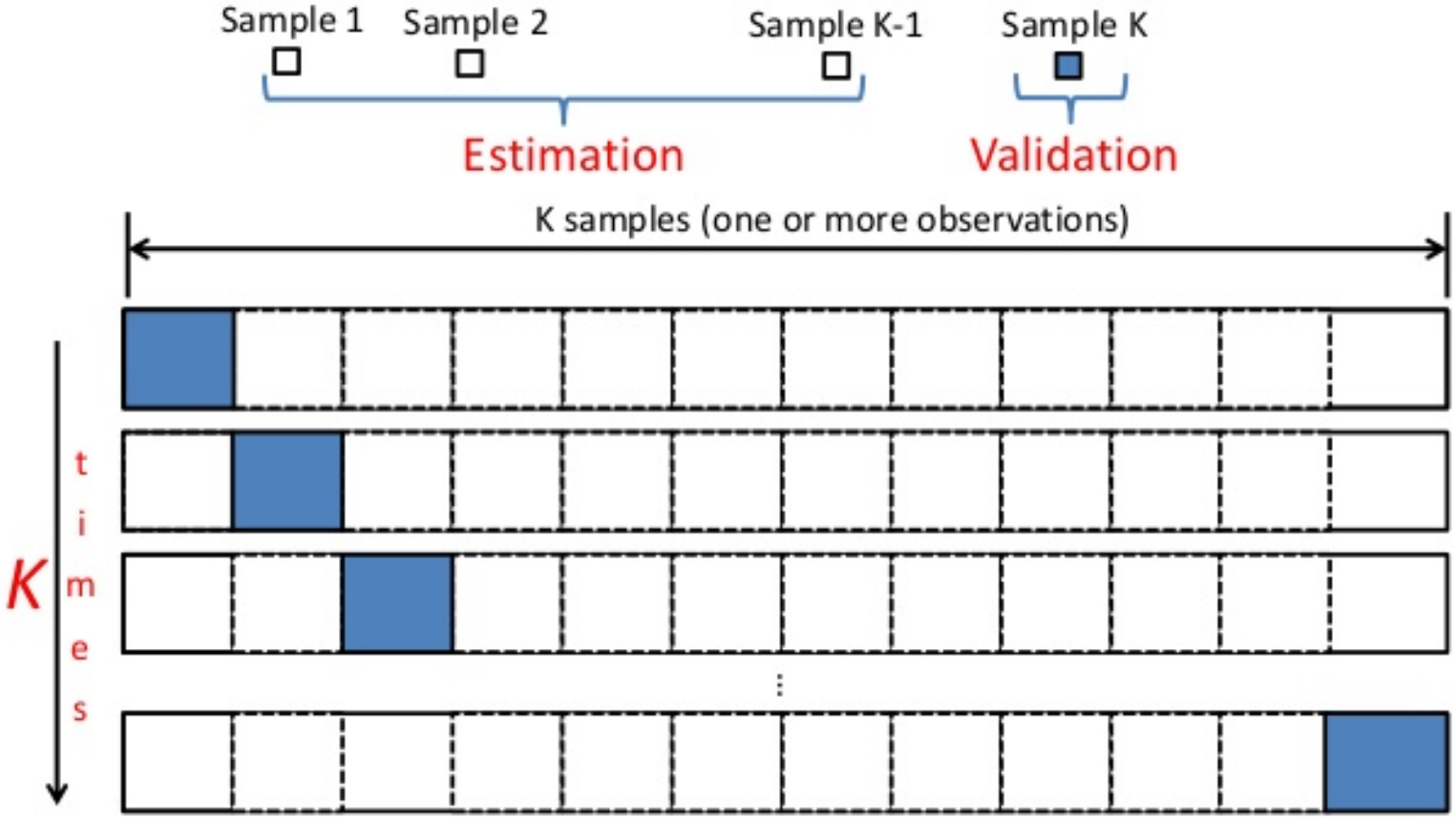
n-fold cross validation in R



n-fold cross validation in R

Cross-validation: How it works?

- K-fold cross-validation:



n-fold cross validation in R

- We will perform 10-fold cross validation; first randomly split the 150 data points into 10 sets:

```
> n_folds=10
> folds_i <- sample(rep(1:n_folds, length.out =
    150))
>  table(folds_i)
folds_i
 1  2  3  4  5  6  7  8  9 10
15 15 15 15 15 15 15 15 15 15
```

n-fold cross validation in R

- We illustrate how the 150 data points is randomly divided into 10 sets using the previous code
- A simple example involves 3-fold, 12 data points
-

```
> n_folds=3
> Number_of_datapoints=12
> rep(1:n_folds,length.out = Number_of_datapoints)
 [1] 1 2 3 1 2 3 1 2 3 1 2 3
> n_folds=3
> Number_of_datapoints=12
> index=rep(1:n_folds,length.out = Number_of_
    datapoints)
> sample(index)
 [1] 3 2 2 1 1 1 2 3 1 3 2 3
```

n-fold cross validation in R

- In this case, `sample(index)` generates a random permutation of the elements of `index`

```
> n_folds=3
> Number_of_datapoints=12
> rep(1:n_folds,length.out = Number_of_datapoints)
 [1] 1 2 3 1 2 3 1 2 3 1 2 3
> n_folds=3
> Number_of_datapoints=12
> index=rep(1:n_folds,length.out = Number_of_
    datapoints)
> sample(index)
 [1] 3 2 2 1 1 1 2 3 1 3 2 3
```

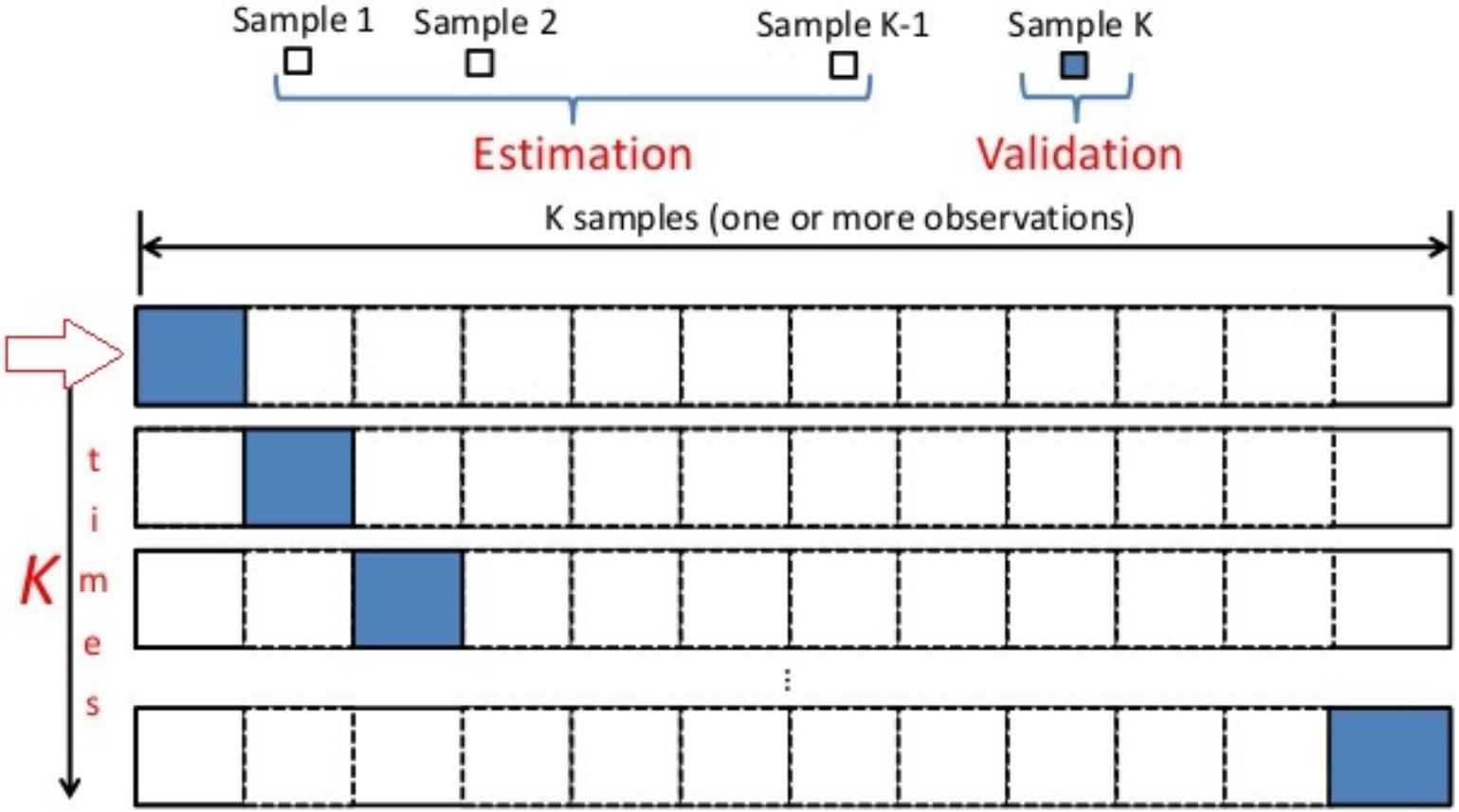
n-fold cross validation in R

- Now, we are ready to run n-fold cross validation in R
- We will first set $k = 1$ for the k-nearest neighbors classifier

n-fold cross validation in R

Cross-validation: How it works?

- K-fold cross-validation:



n-fold cross validation in R

- We start with $k = 1$, and observe first iteration: using the first dataset as our test data and the remaining 9 datasets as training data

```
> test_j <- which(folds_j == 1)
> pred <- knn(train=X[ -test_j, ], test=X[test_j,
    ], cl=Y[-test_j ], k=1)
> mean(Y[test_j] != pred)
[1] 0.06666667
> mean(Y[test_j] == pred)
[1] 0.9333333
```

n-fold cross validation in R

- Note that the command `test_j<-which(folds_j==1)` generates the indices corresponding to data points in the first data set:

```
> folds_j[1:20]
 [1]  4  9  8  4  7  7  1 10  1  1  6  3
[13]  8  1  7  9  5  7  4  3
> which(folds_j == 1)
 [1]   7   9  10  14  22  23  26  37  43
[10]  98 107 118 130 132 146
```

n-fold cross validation in R

- The command `X[-test_j,]` removes the data points from first data set from the feature dataframe, keeping the data points from the remaining nine data sets
- Similarly, the command `Y[-test_j]` removes the data points from first data set from the label vector, keeping the data points from the remaining nine data sets
- The command `X[test_j,]` keeps the data points only from first data set from the feature dataframe

n-fold cross validation in R

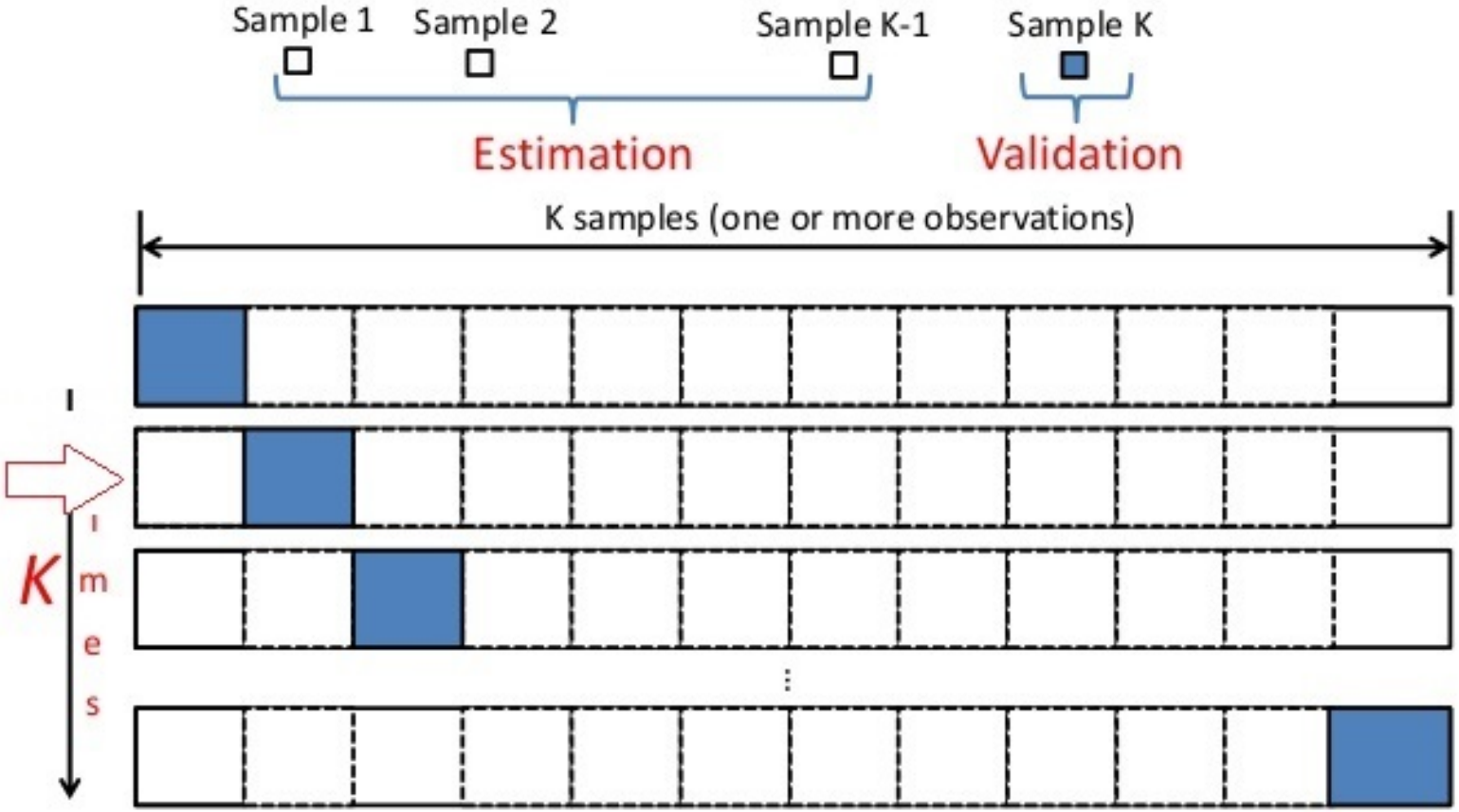
- Therefore, the following command takes only the first data set as the test data, and the remaining nin data sets as the training data

```
> pred <- knn(train=X[ -test_j, ], test=X[test_j,
    ], cl=Y[-test_j ], k=1)
```

n-fold cross validation in R

Cross-validation: How it works?

- K-fold cross-validation:



n-fold cross validation in R

- Now we perform the second iteration: using the second dataset as our test data and the remaining 9 datasets as training data

```
> ## second fold of the iteration
> test_j <- which(folds_j == 2)
> pred <- knn(train=X[ -test_j, ], test=X[test_j,
    ], cl=Y[-test_j ], k=1)
> mean(Y[test_j] != pred)
[1] 0.06666667
> mean(Y[test_j] == pred)
[1] 0.9333333
```

n-fold cross validation in R

- We need to run the same code for $j = 3, 4, 5, ..., 10$.
- The `for` loop in R allows us to write compact code for this

```
> for (j in 1:10) {
+   print(j)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
```

n-fold cross validation in R

- Putting it altogether in a `for` loop in R:

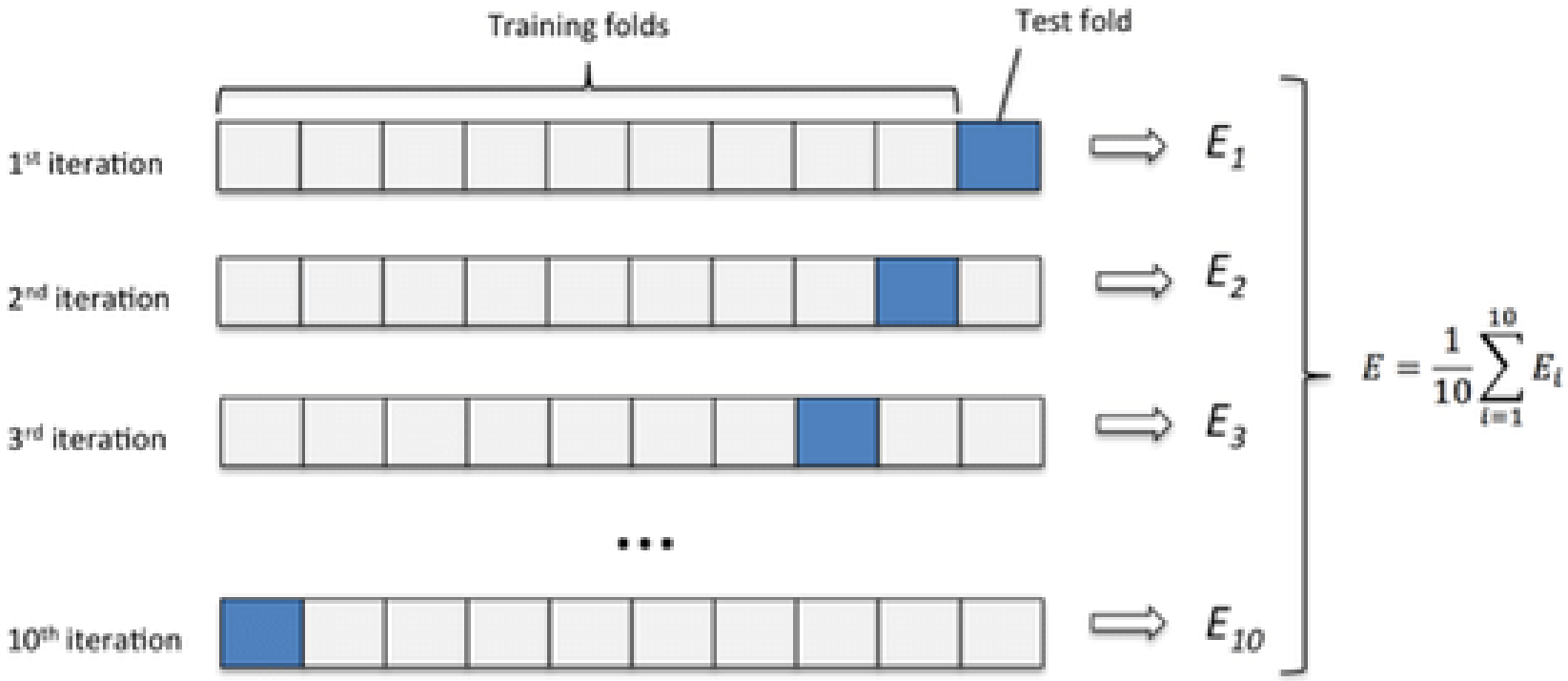
```
> err=numeric(10)
> acc=numeric(10)
>
> for (j in 1:10) {
+
+ test_j <- which(folds_j == j)
+ pred <- knn(train=X[ -test_j, ], test=X[test_j,
    ], cl=Y[-test_j ], k=1)
+
+ err[j]=mean(Y[test_j] != pred)
+ acc[j]=mean(Y[test_j] == pred)
+ }
>
> err
 [1] 0.06666667 0.06666667 0.06666667
 [4] 0.06666667 0.00000000 0.06666667
 [7] 0.06666667 0.00000000 0.00000000
[10] 0.00000000
```

n-fold cross validation in R

- Here, `err=numeric(10)` declares a numeric vector of length 10
- Therefore, the error rate from each iteration of the 10-fold cross-validation can be saved to the vector `err`
- Similarly, the accuracy from each iteration of the 10-fold cross-validation can be saved to the vector `acc`

```
> test=numeric(10)
> test
 [1] 0 0 0 0 0 0 0 0 0 0
```

Diagnostics of Classifiers



n-fold cross validation in R

- For the k-nearest neighbor classifier with $k = 1$, the final estimated error rate is the average of the ten error rates across the 10-fold iterations.
- Similarly, the final estimated accuracy is the average of the ten accuracies across the 10-fold iterations.

```
> error=mean(err)
> accur=mean(acc)
> error
[1] 0.04
> accur
[1] 0.96
```

n-fold cross validation in R

- However, to plot the accuracy (or error rate) against different values of k, we need to repeat the 10-fold cross validation procedure at different values of k
- This suggest another `for` loop structure, indexed by k
- In this example, we will perform 10-fold cross validation for $k = 1, 2, ..., 50$.

n-fold cross validation in R

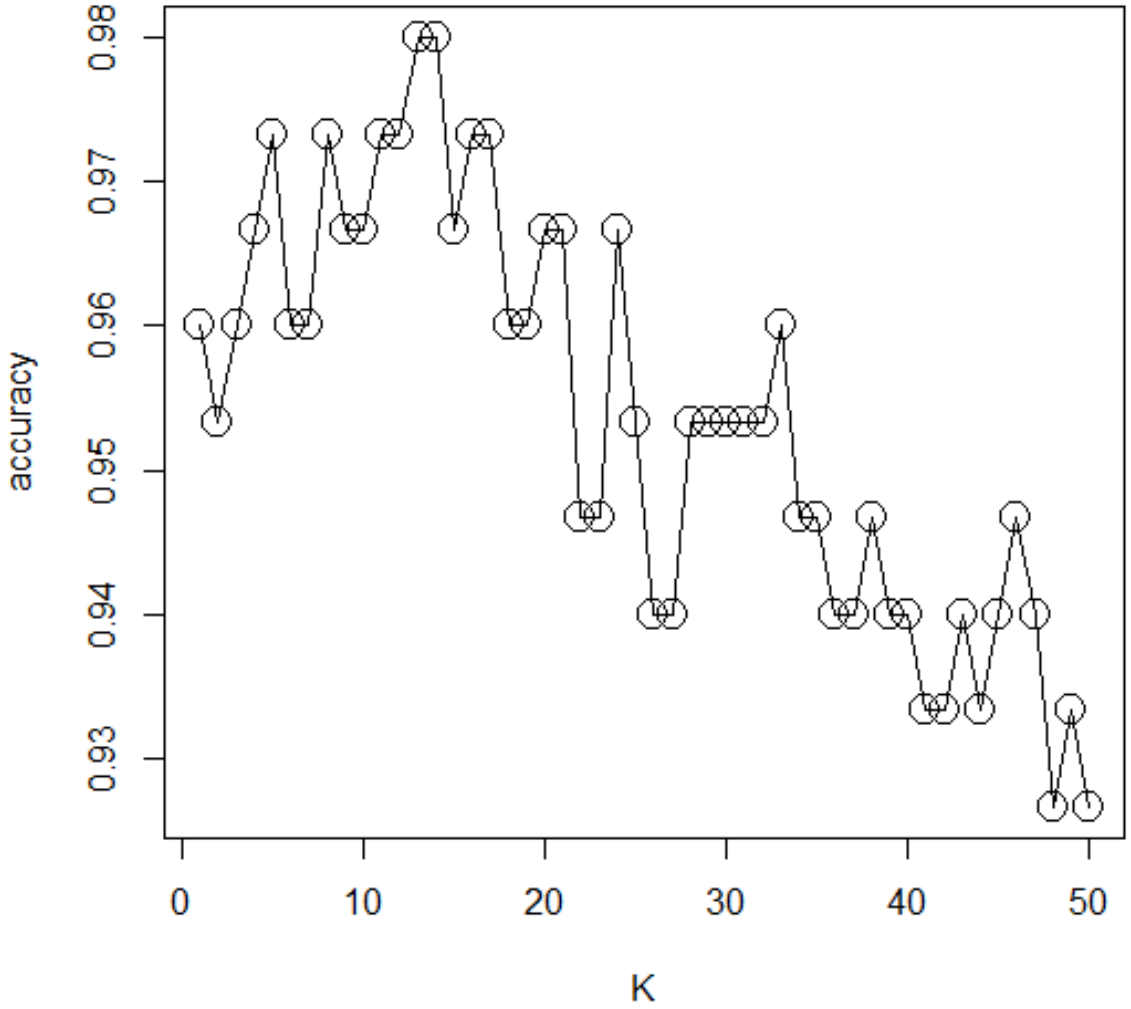
```
error_differentK=numeric(50)
accur_differentK=numeric(50)

for (K in 1:50) {

error=numeric(10)
accur=numeric(10)
for (j in 1:10) {
  test_j <- which(folds_j == j)
  pred <- knn(train=X[ -test_j, ], test=X[test_j,
      ], cl=Y[-test_j ], k=K)
  error[j]=mean(Y[test_j] != pred)
  accur[j]=mean(Y[test_j] == pred)
}
error_differentK[K]=mean(error)
accur_differentK[K]=mean(accur)
}
```

n-fold cross validation in R

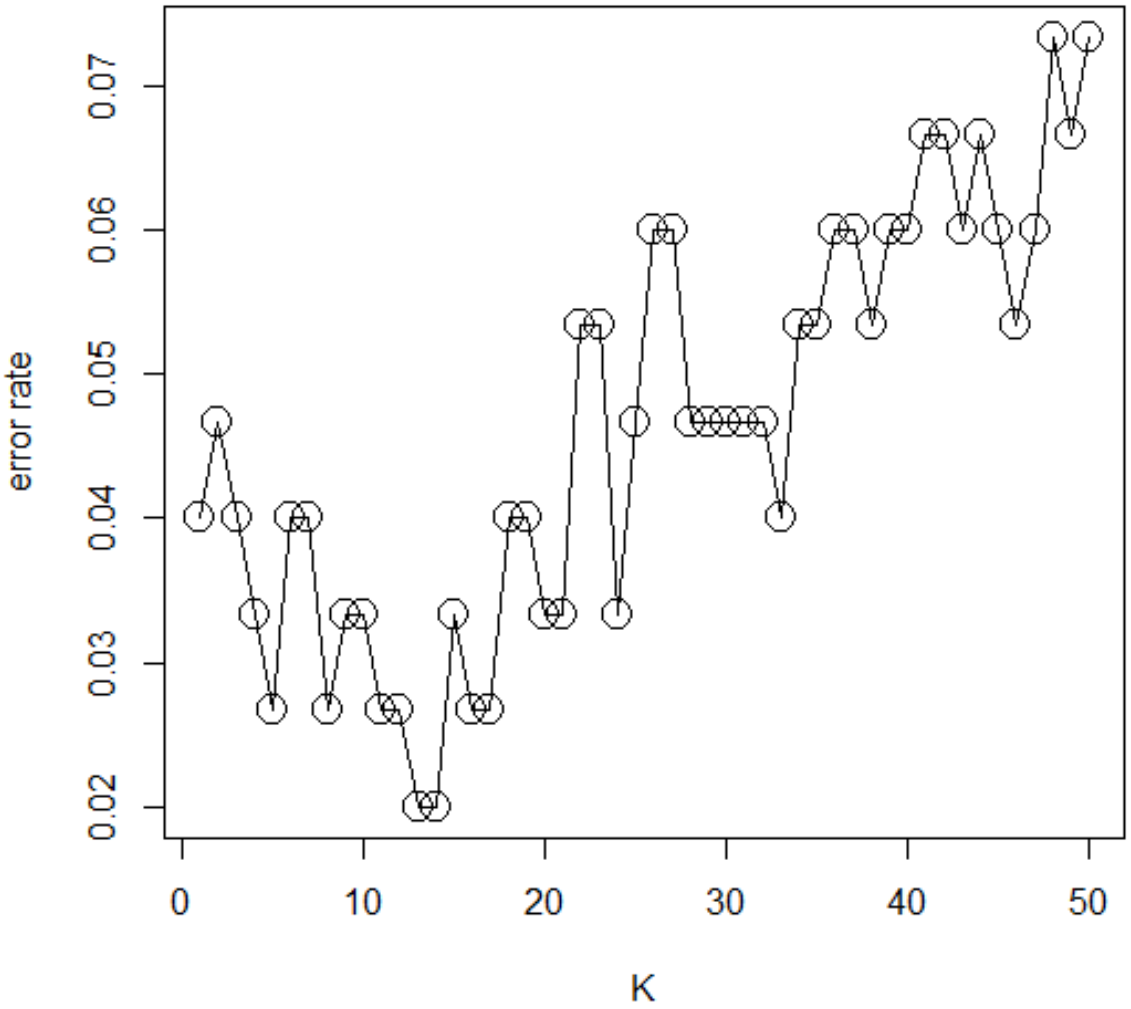
- Then, we can plot the accuracy (or error rate) against different values of k, to determine the optimal k.
-

```
plot(1:50, accur_differentK,type="o",ylab="
    accuracy",xlab="K",cex.axis=1,cex=2)
plot(1:50, error_differentK,type="o",ylab="error
    rate",xlab="K",cex.axis=1,cex=2)
```

n-fold cross validation in R



n-fold cross validation in R



n-fold cross validation in R

- Recall the confusion matrix

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

n-fold cross validation in R

- Recall the confusion matrix

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

- accuracy $= \frac{TP+TN}{TP+FN+FP+TN}$
- error rate $= \frac{TP+TN}{TP+FN+FP+TN}$
- Therefore accuracy $= 1 -$ error rate

n-fold cross validation in R

- To illustrate the *n*-fold cross validation procedure, we have written our own code in R to implement it
- There are many data analytics packages in R that offers ready-made solutions which make our jobs much easier
- We will use one such package, 'caret', as an example
-

```
library(caret)
```

n-fold cross validation in R

- We will perform 10-fold cross-validation for the *k*-nearest neighbor classifier:

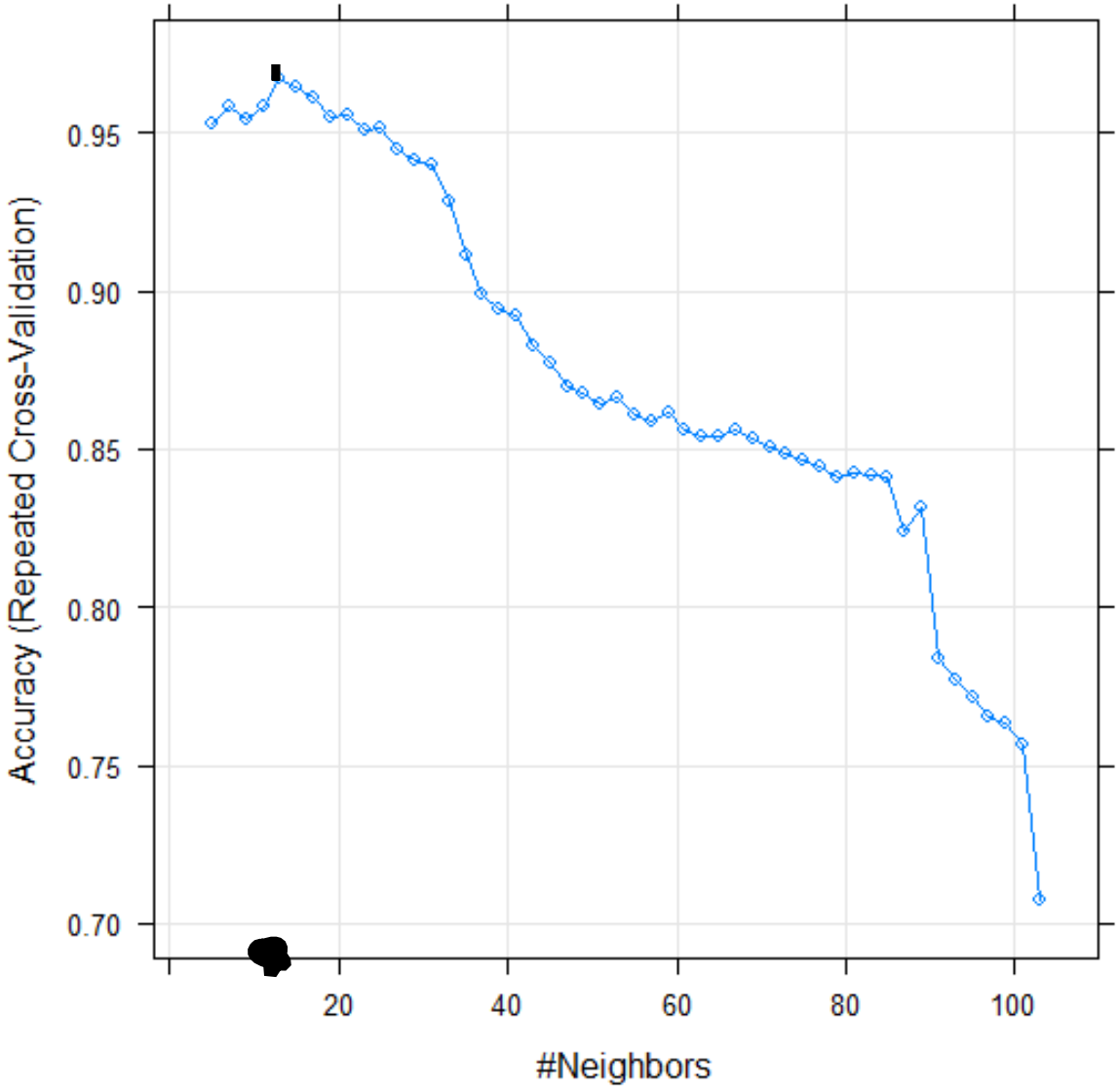
```
fitControl <- trainControl(## 10-fold CV
                           method = "repeatedcv",
                           number = 10,
                           ## repeated ten times
                           repeats = 10)
set.seed(2)
knnFit1 <- train(Y ~ ., data = iris,
                 method = "knn",
                 trControl = fitControl,
                 preProcess = c("center", "scale")
                     ,
                 tuneLength=50)
```

n-fold cross validation in R

- We will perform 10-fold cross-validation for the k-nearest neighbor classifier:

```
> knnFit1
k-Nearest Neighbors
    ...
Resampling results across tuning parameters:

  k     Accuracy   Kappa
    5   0.9526667  0.929
    7   0.9586667  0.938
    9   0.9540000  0.931
    ...
Accuracy was used to select the optimal model
    using the largest value.
The final value used for the model was k = 13.

plot(knnFit1)
```

n-fold cross validation in R



n-fold cross validation in R: Wine recognition example

- Wines were grown in the same region in Italy but derived from 3 different cultivars.
- The task is to predict wine origin based on 13 attributes having continuous values

n-fold cross validation in R: Wine recognition example

- The 13 features (X) of the dataset are:

1. Alcohol
2. Malic acid
3. Ash
4. Alkalinity of ash
5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavonoids phenols
9. Proanthocyanins
10. Color intensity
11. Hue
12. OD280/OD315 of diluted wines
13. Proline

n-fold cross validation in R: Wine recognition example

- The data set is available from `https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data`
- Task is to predict label Y of origin: 1,2 or 3
-

```
dataurl <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
download.file(url = dataurl, destfile = "wine.data")
wine_df <- read.csv("wine.data", header = FALSE)
```

n-fold cross validation in R: Wine recognition example

```
> head(wine_df)
  V1    V2   V3   V4   V5  V6   V7   V8
1  1 14.23 1.71 2.43 15.6 127 2.80 3.06
2  1 13.20 1.78 2.14 11.2 100 2.65 2.76
3  1 13.16 2.36 2.67 18.6 101 2.80 3.24
4  1 14.37 1.95 2.50 16.8 113 3.85 3.49
5  1 13.24 2.59 2.87 21.0 118 2.80 2.69
6  1 14.20 1.76 2.45 15.2 112 3.27 3.39
    V9  V10  V11  V12  V13  V14
1 0.28 2.29 5.64 1.04 3.92 1065
2 0.26 1.28 4.38 1.05 3.40 1050
3 0.30 2.81 5.68 1.03 3.17 1185
4 0.24 2.18 7.80 0.86 3.45 1480
5 0.39 1.82 4.32 1.04 2.93  735
6 0.34 1.97 6.75 1.05 2.85 1450
```

- First column 'V1' contains the wine origin labels

n-fold cross validation in R: Wine recognition example

- We will perform 10-fold cross validation for the k-nearest neighbors classifier on the wine data set:

```
wine_df$V1=factor(wine_df$V1)
trctrl <- trainControl(method = "repeatedcv",
    number = 10, repeats = 10)
set.seed(3)
knn_fit <- train(V1 ~., data = wine_df, method = "
    knn",
 trControl=trctrl,
 preProcess = c("center", "scale"),
 tuneLength = 20)
 plot(knn_fit)
```

n-fold cross validation in R: Wine recognition example

- We will perform 10-fold cross validation for the k-nearest neighbors classifier on the wine data set:

```
> knn_fit
k-Nearest Neighbors

178 samples
 13 predictor
  3 classes: '1', '2', '3'
 ...
  k   Accuracy   Kappa
   5  0.9718920  0.9577857
   7  0.9689800  0.9534599
   9  0.9713657  0.9570151
  11  0.9657086  0.9486180
 ...
 Accuracy was used to select the optimal model
     using
 the largest value.
The final value used for the model was k = 35.
```

n-fold cross validation in R