An Overview of Modeling Process

Yifei Sun

Contents

```
library(caret)
library(FNN) # knn.reg()
library(doBy) # which.minn()

set.seed(2022)
```

The goal of this tutorial is to provide an overview of the modeling process. The functions from the package `caret` will be discussed in details in our future lectures.

You can generate a simulated training dataset or use an existing dataset. For illustration, we use a simulated dataset with two predictors.

```
# Data generating - you can replace this with your own function
genData <- function(N)
{
  X1 <- rnorm(N, mean = 1)
  X2 <- rnorm(N, mean = 1)
  eps <- rnorm(N, sd = .5)
  Y <- sin(X1) + (X2)^2 + eps
  # Y <- X1 + X2 + eps
  data.frame(Y = Y, X1 = X1, X2 = X2)
}

dat <- genData(500)
```

Data partition

```
indexTrain <- createDataPartition(y = dat$Y, p = 0.8, list = FALSE)
trainData <- dat[indexTrain, ]
testData <- dat[-indexTrain, ]

head(trainData)
```

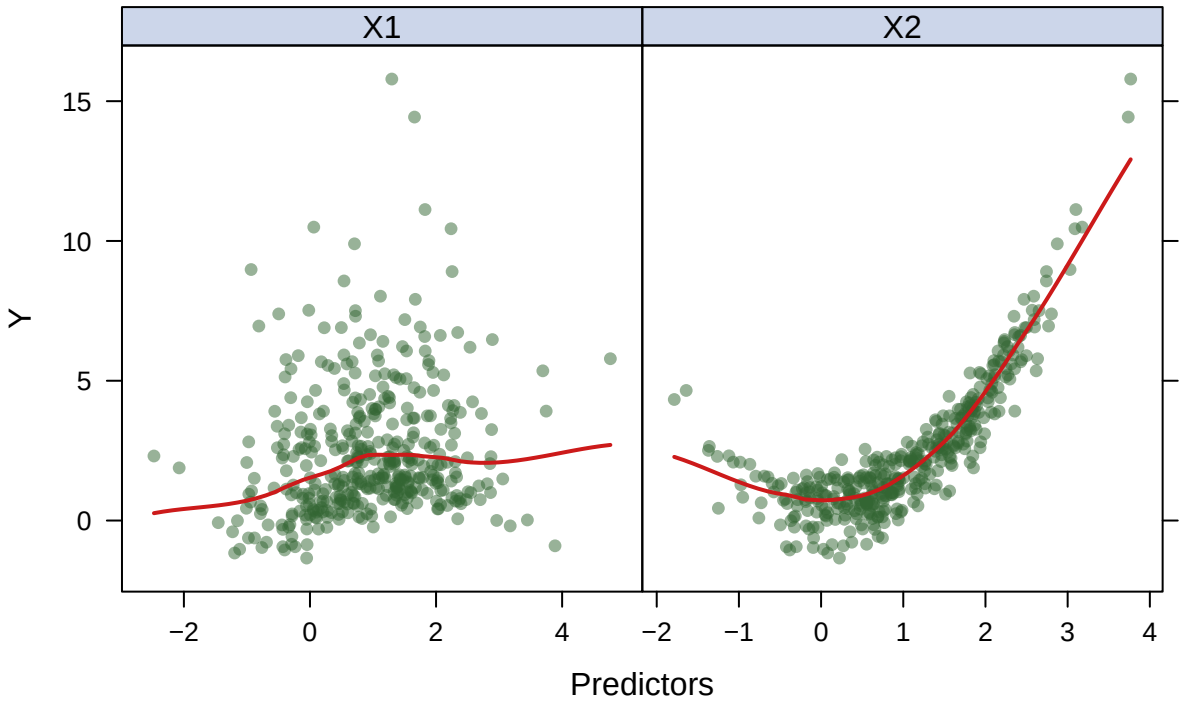
```
##            Y         X1          X2
## 2  2.5453757 -0.1733458  1.51256677
## 3  0.3007461  0.1025146 -0.05365794
## 4 -0.1575650 -0.4445014  0.48895504
## 5  5.6818890  0.6689864  2.19138619
## 8  0.1316369  1.2779547  0.74466035
## 9  6.9223616  1.7494859  2.59817771
```

Data visualization

The function `featurePlot()` in `caret` is a wrapper for different lattice plots to visualize multivariate data. The various graphical parameters (color, line type, background, etc) that control the look of Trellis displays are highly customizable. You can explore `trellis.par.set()` after class.

```
theme1 <- trellis.par.get()
theme1$plot.symbol$col <- rgb(.2, .4, .2, .5)
theme1$plot.symbol$pch <- 16
theme1$plot.line$col <- rgb(.8, .1, .1, 1)
theme1$plot.line$lwd <- 2
theme1$strip.background$col <- rgb(.0, .2, .6, .2)
trellis.par.set(theme1)

featurePlot(x = trainData[ ,2:3],
            y = trainData[ ,1],
            plot = "scatter",
            span = .5,
            labels = c("Predictors","Y"),
            type = c("p", "smooth"),
            layout = c(2, 1))
```

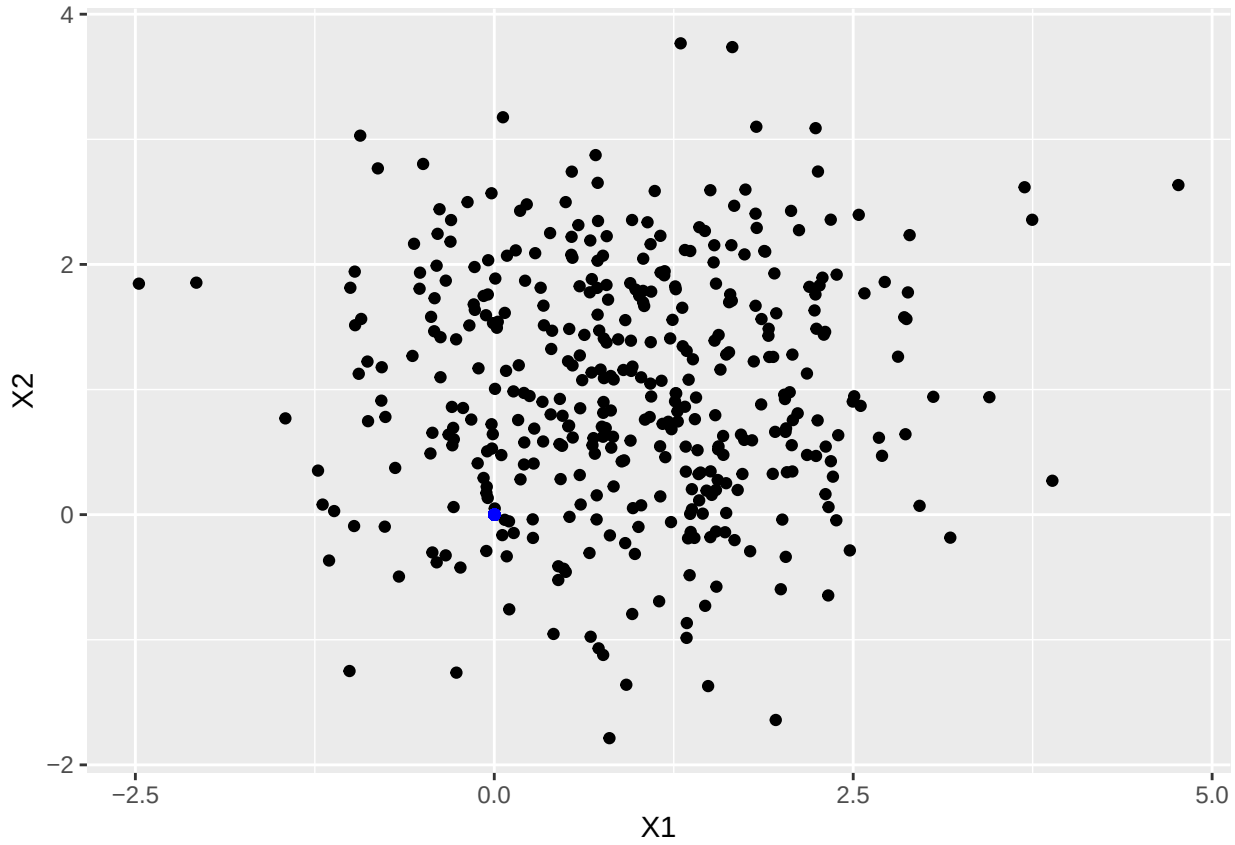


What is k-Nearest Neighbour?

Now let's make prediction for a new data point with `X1 = 0` and `X2 = 0`.

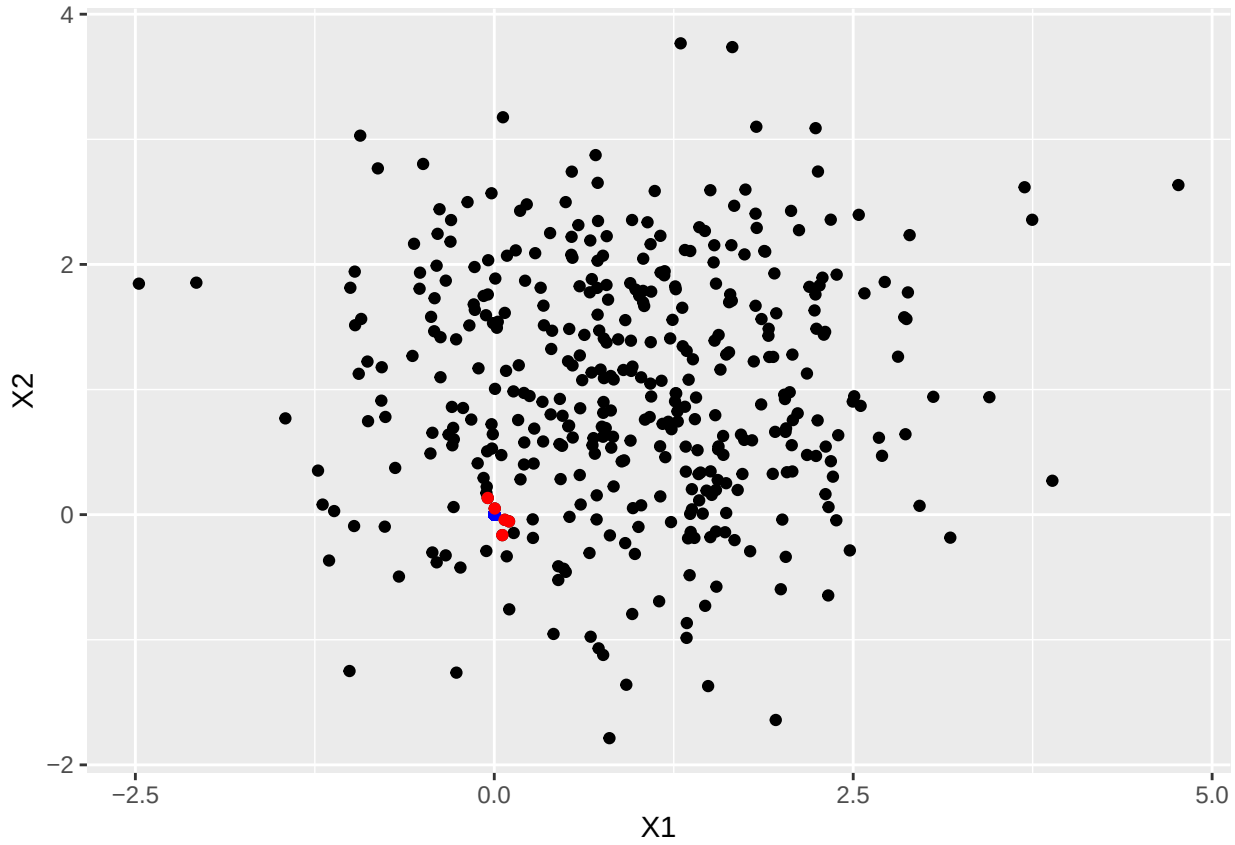
```
# scatter plot of X2 vs. X
p <- ggplot(trainData, aes(x = X1, y = X2)) + geom_point() +
  geom_point(aes(x = 0, y = 0), colour="blue")

p
```




```
# find the 5 nearest neighbours of (0,0)
dist0 <- sqrt( (trainData[,2] - 0)^2 + (trainData[,3] - 0)^2 ) # calculate the distances
neighbor0 <- which.minn(dist0, n = 5) # indices of the 5 smallest distances

# visualize the neighbours
p + geom_point(data = trainData[neighbor0, ],
               colour = "red")
```




```
# calculate the mean outcome of the nearest neighbours as the predicted outcome
mean(trainData[neighbor0,1])
```

```
## [1] 0.2179523
```

```
# Using the knn.reg() function
knn.reg(train = trainData[,2:3],
        test = c(0,0),
        y = trainData[,1],
        k = 5)
```

```
## Prediction:
## [1] 0.2179523
```

Model training

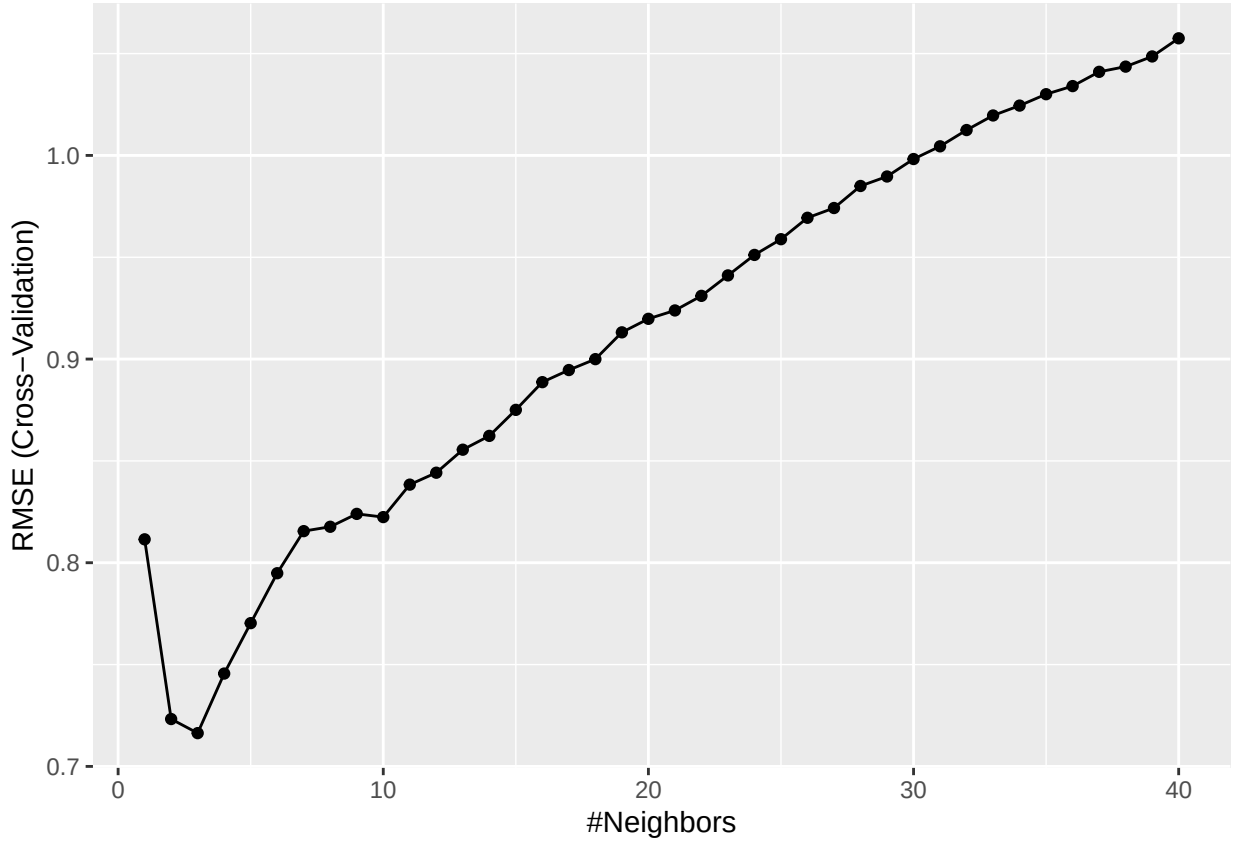
We consider two candidate models: KNN and linear regression.

```
kGrid <- expand.grid(k = seq(from = 1, to = 40, by = 1))

set.seed(1)
fit.knn <- train(Y ~ .,
                 data = trainData,
```

```
                  method = "knn",
                  trControl = trainControl(method = "cv", number = 10), # ten-fold cross-validation
                  tuneGrid = kGrid)

ggplot(fit.knn)
```




```
# plot(fit.knn)
```

The kNN approach (k = 3) was selected as the final model.

```
set.seed(1)
fit.lm <- train(Y ~ .,
                data = trainData,
                method = "lm",
                trControl = trainControl(method = "cv", number = 10))
```

Which is better?

```
rs <- resamples(list(knn = fit.knn, lm = fit.lm))

summary(rs, metric = "RMSE")
```

```
##
## Call:
```

```
## summary.resamples(object = rs, metric = "RMSE")
##
## Models: knn, lm
## Number of resamples: 10
##
## RMSE
##         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## knn 0.480682 0.6578625 0.7552313 0.7163339 0.8057817 0.8266776    0
## lm  1.007314 1.1506569 1.4464566 1.4183897 1.5669518 1.8574761    0
```

Evaluating the model on the test data

```
pred.knn <- predict(fit.knn, newdata = testData)
pred.lm <- predict(fit.lm, newdata = testData)

RMSE(pred.knn, testData[,1])
```

```
## [1] 1.182837
```

```
RMSE(pred.lm, testData[,1])
```

```
## [1] 1.734175
```