An Overview of Modeling Process

Yifei Sun, Runze Cui

Contents

Data partition **2**

Data visualization **2**
ggplot 2
featurePlot 3

What is k-Nearest Neighbour? **4**
KNN from scratch 5
Using knn.reg() 6

Model training in `caret` **6**

Model training in `tidymodels` **8**

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

set.seed(2024)
```

The goal of this tutorial is to provide an overview of the modeling process. The functions from the packages `caret` and `tidymodels` 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.

```r
# 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

```r
datSplit <- initial_split(data = dat, prop = 0.8)
trainData <- training(datSplit)
testData <- testing(datSplit)

head(trainData)
```

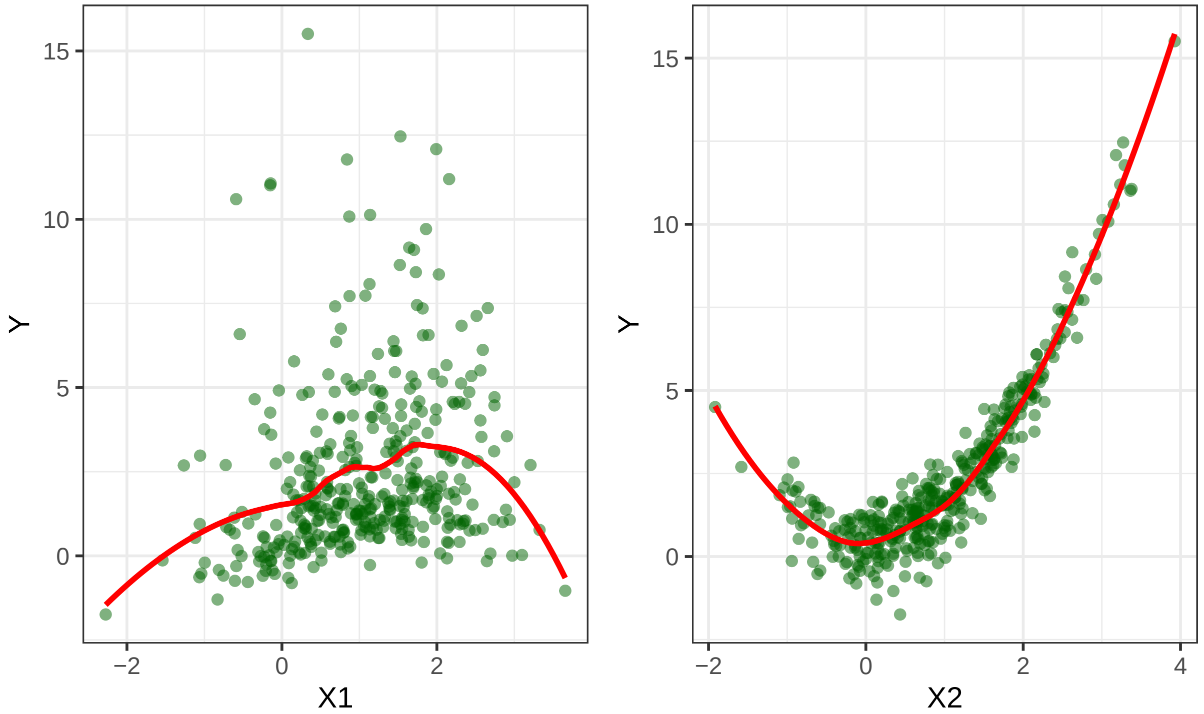
```
##           Y        X1       X2
## 1 10.129156 1.1383772 3.007886
## 2  7.450602 1.7434658 2.448874
## 3  1.692010 1.9881643 1.197497
## 4  2.764881 0.9391763 1.245817
## 5  7.129447 2.5130750 2.622022
## 6  1.563333 0.7352091 1.049990
```

Data visualization

ggplot

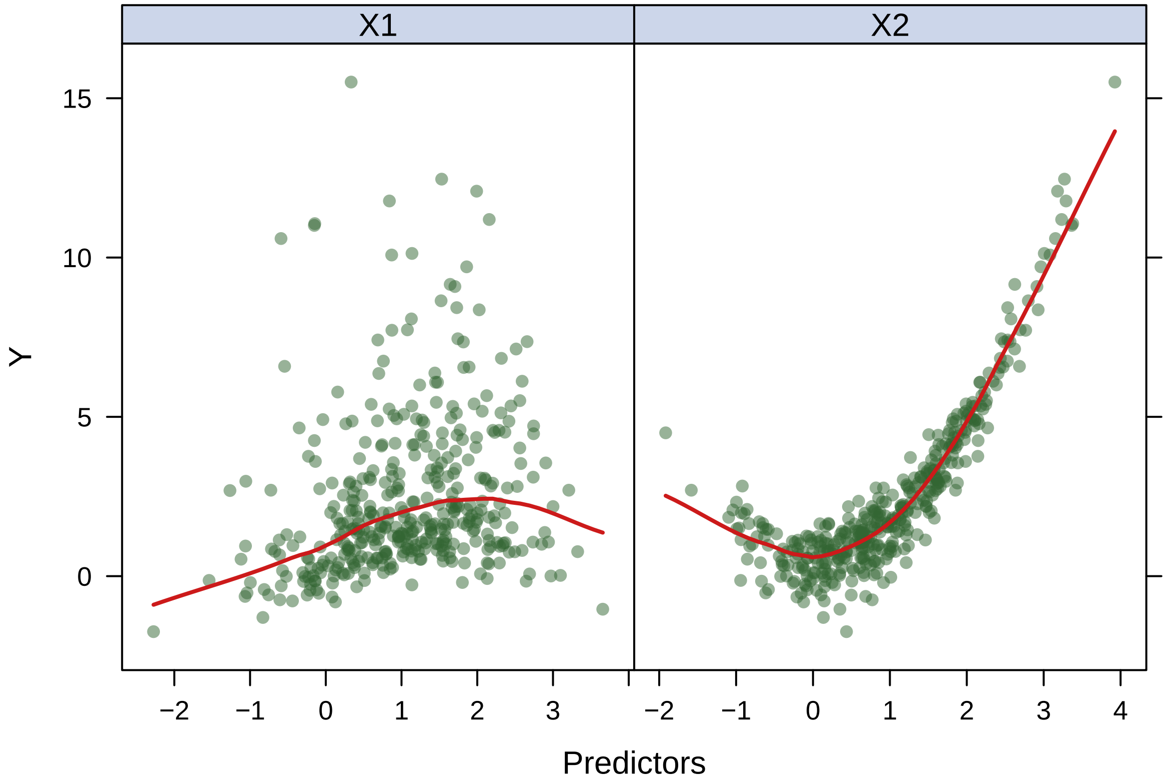
```r
p1 <- ggplot(trainData, aes_string(x = "X1", y = "Y")) +
  geom_point(color = "darkgreen", alpha = 0.5) +
  geom_smooth(method = "loess", span = 0.5, color = "red", se = FALSE) +
  theme_bw() +
  labs(x = "X1", y = "Y")
p2 <- ggplot(trainData, aes_string(x = "X2", y = "Y")) +
```

```r
  geom_point(color = "darkgreen", alpha = 0.5) +
  geom_smooth(method = "loess", span = 0.5, color = "red", se = FALSE) +
  theme_bw() +
  labs(x = "X2", y = "Y")

library(patchwork)
p1 + p2
```



featurePlot

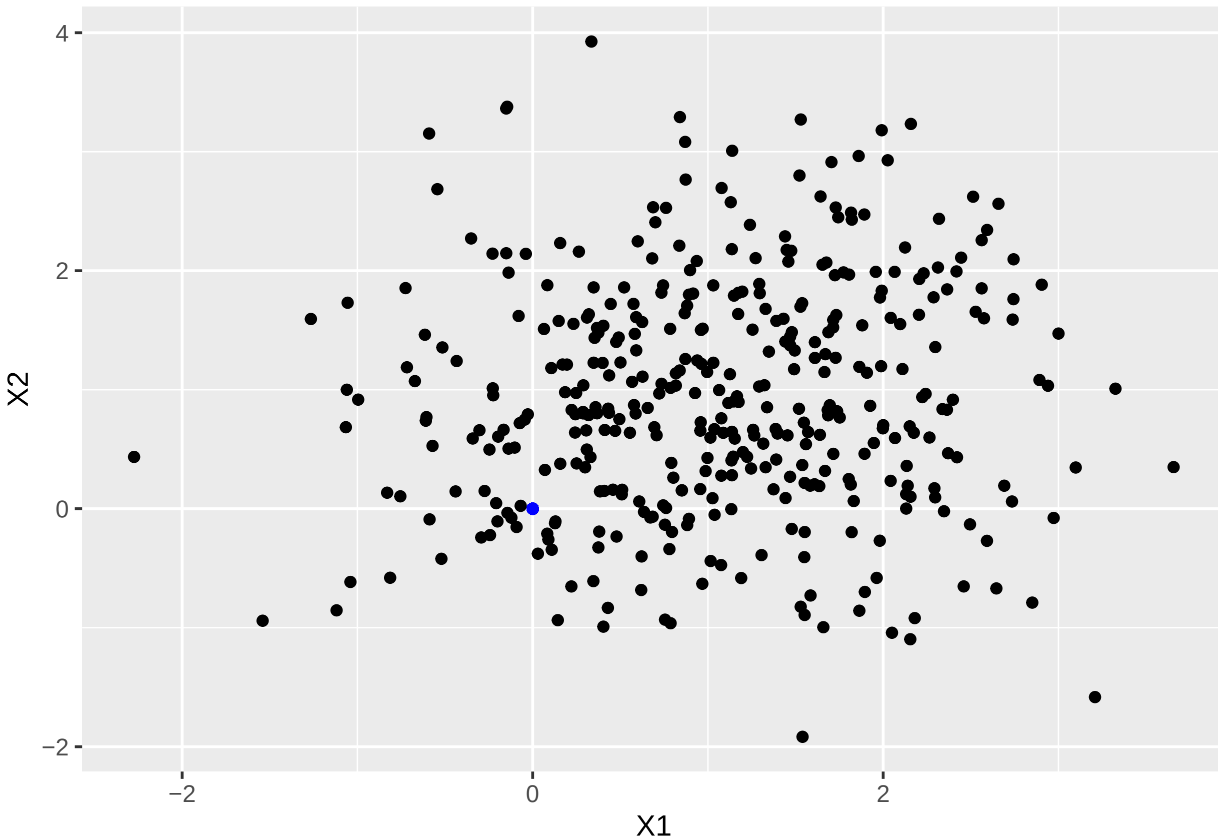
```r
theme1 <- trellis.par.get()
theme1$plot.symbol$col <- rgb(.2, .4, .2, .5)
theme1$plot.symbol$pch <- 16
theme1$plot.line$col <- rgb(.8, .1, .1, 1)
theme1$plot.line$lwd <- 2
theme1$strip.background$col <- rgb(.0, .2, .6, .2)
trellis.par.set(theme1)

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?

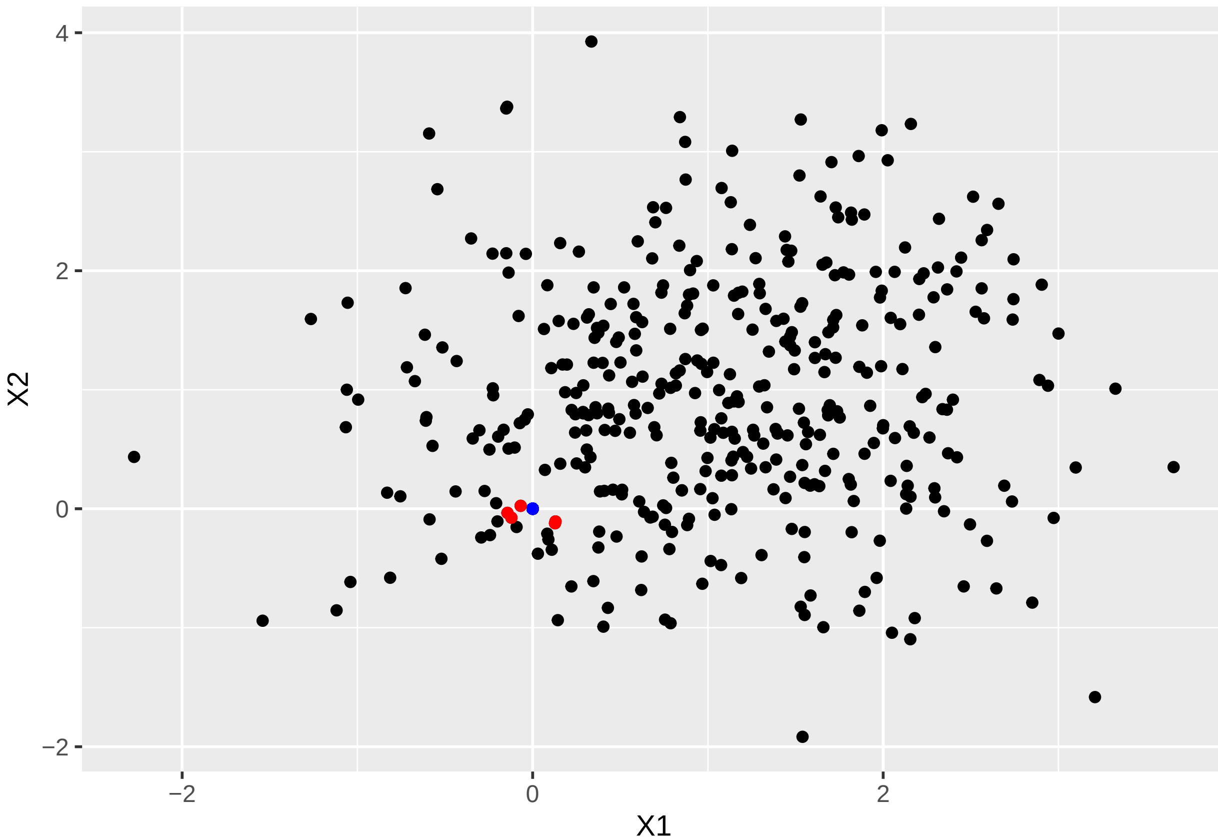
Now let's make prediction for a new data point with `X1 = 0` and `X2 = 0`.

```
# 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
```



KNN from scratch

```
# find the 5 nearest neighbours of (0,0)
dist0 <- sqrt( (trainData[,2] - 0)^2 + (trainData[,3] - 0)^2 ) # calculate the distances
neighbor0 <- doBy::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] -0.216995
```

Using knn.reg()

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

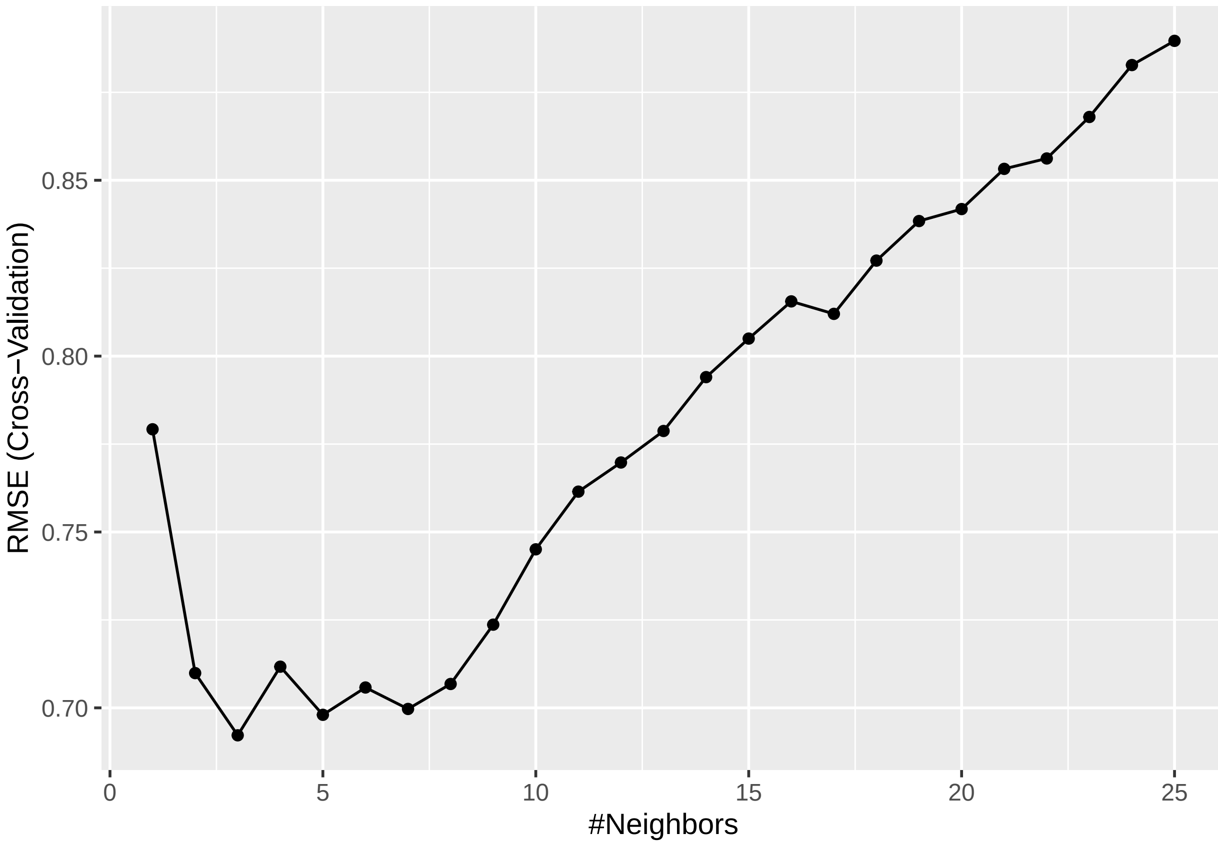
```
## Prediction:
## [1] -0.216995
```

Model training in caret

We consider two candidate models: KNN and linear regression.

```
set.seed(1)
fit.knn <- train(Y ~ .,
                 data = trainData,
                 method = "knn",
                 trControl = trainControl(method = "cv", number = 10), # ten-fold cross-validation
                 tuneGrid = expand.grid(k = seq(from = 1, to = 25, by = 1)))
```

```r
ggplot(fit.knn)
```




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

k = 3 was selected.

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

Which is better?

```r
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 0.5680585 0.5889135 0.6342311 0.6921973 0.7844169 0.9842471    0
## lm  1.0531365 1.1113199 1.4125226 1.4056799 1.6595075 1.8673024    0
```

Evaluating the final model on the test data

```
pred.knn <- predict(fit.knn, newdata = testData)
RMSE(pred.knn, testData[,1])
```

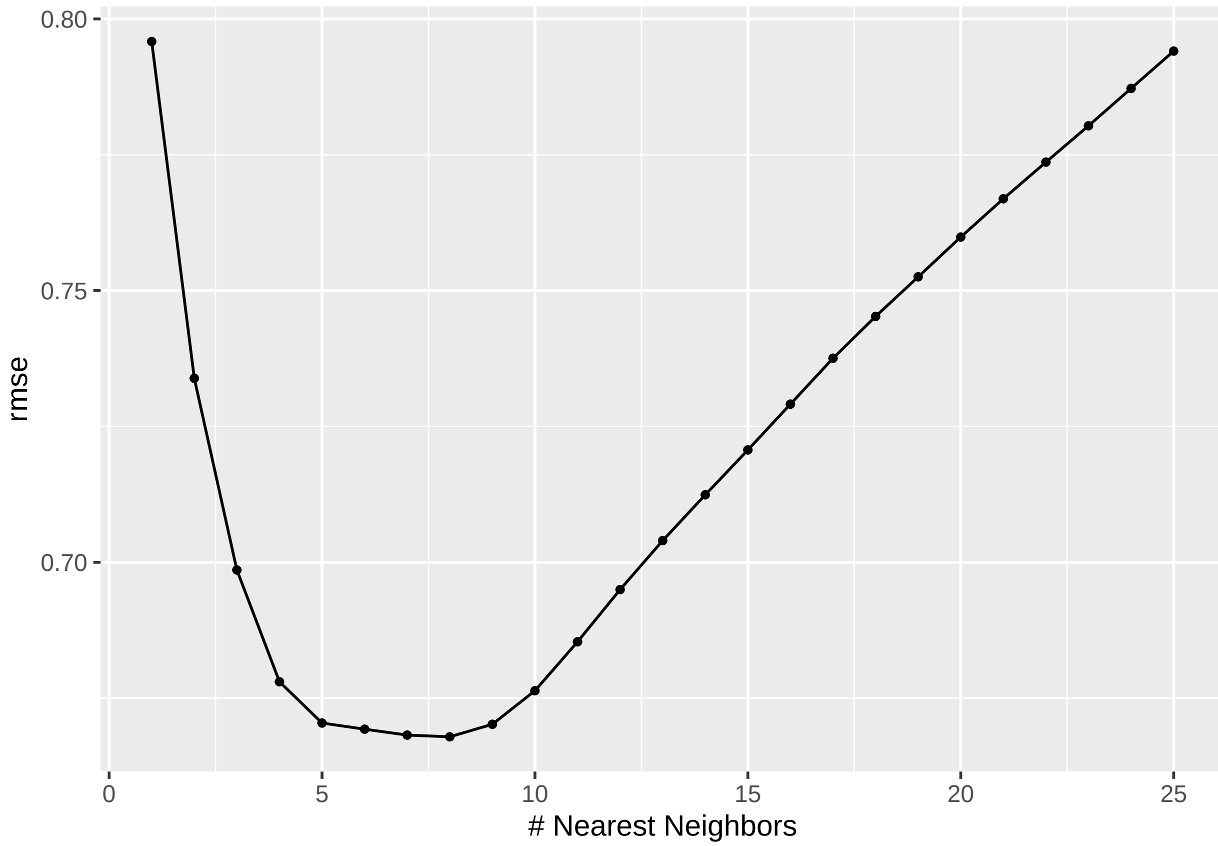
```
## [1] 0.9409212
```

```
# pred.lm <- predict(fit.lm, newdata = testData)
# RMSE(pred.lm, testData[,1])
```

Model training in `tidymodels`

We consider two candidate models: KNN and linear regression.

```
# Model specification for KNN
knn_spec <- nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>%
  set_mode("regression")

set.seed(1)
# Split training data: 10-fold cross-validation
cv_folds <- vfold_cv(trainData, v = 10)

# Set up the workflow
knn_workflow <- workflow() %>%
  add_model(knn_spec) %>%
  add_formula(Y ~ .)

# Specify the grid of k to consider
k_values <- tibble(neighbors = seq(from = 1, to = 25, by = 1))

# Tune the KNN model
tune_knn <- tune_grid(
  knn_workflow,
  resamples = cv_folds,
  grid = k_values
)

# You can autoplot the results to see the performance
autoplot(tune_knn, metric = "rmse")
```




```
# Select the best K value based on a performance metric, e.g., accuracy
best_k <- select_best(tune_knn, metric = "rmse")
```

k = 8 was selected.

```
# Finalize the model with the best K
final_knn_spec <- knn_spec %>%
  update(neighbors = best_k$neighbors)

# Model specification for linear regression
lm_spec <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")
```

Which is better?

```
workflow_set(preproc = list(Y ~ .),
             models = list(lm = lm_spec, knn = final_knn_spec)) %>%
workflow_map(resamples = cv_folds) %>%
  collect_metrics() %>%
  filter(.metric == "rmse") %>%
  select(model, mean)
```

```
## # A tibble: 2 x 2
##   model              mean
##   <chr>             <dbl>
## 1 linear_reg        1.39
## 2 nearest_neighbor 0.668
```

Evaluating the final model on the test data

```
workflow() %>%
  add_model(final_knn_spec) %>%
  add_formula(Y ~ .) %>%
  last_fit(datSplit) %>%
  collect_metrics() %>%
  filter(.metric == "rmse")
```

```
## # A tibble: 1 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       0.971 Preprocessor1_Model1
```

```
# workflow() %>%
#   add_model(lm_spec) %>%
#   add_formula(Y ~ .) %>%
#   last_fit(datSplit) %>%
#   collect_metrics() %>%
#   filter(.metric == "rmse")
```