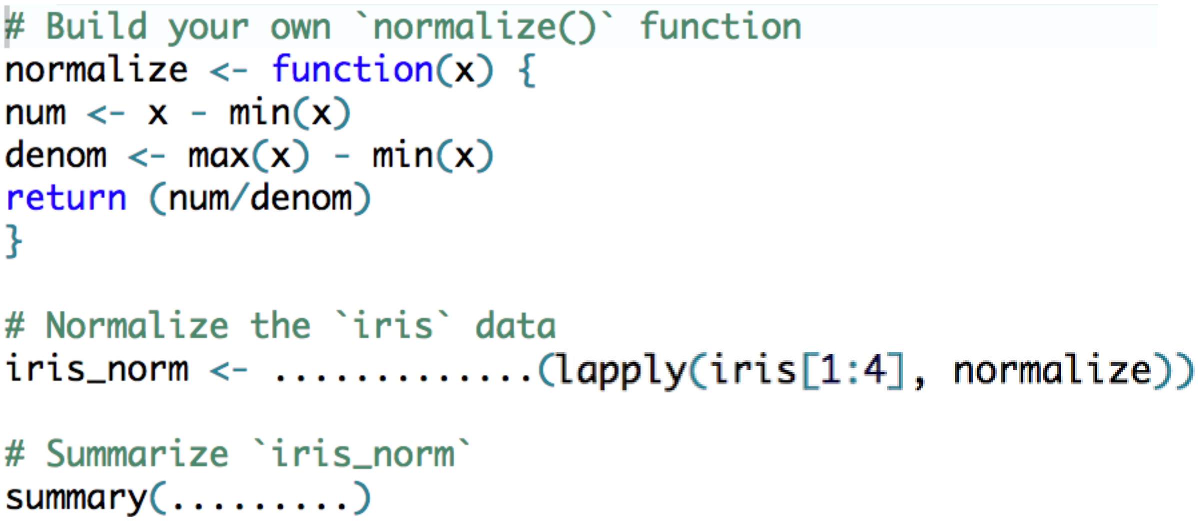
Nearest Neighbor Models in R

Normalization

- Use ‘iris’ data

  > head(iris)

- Normalization makes it easier for the kNN algorithm to learn
  - Guess ‘……….’



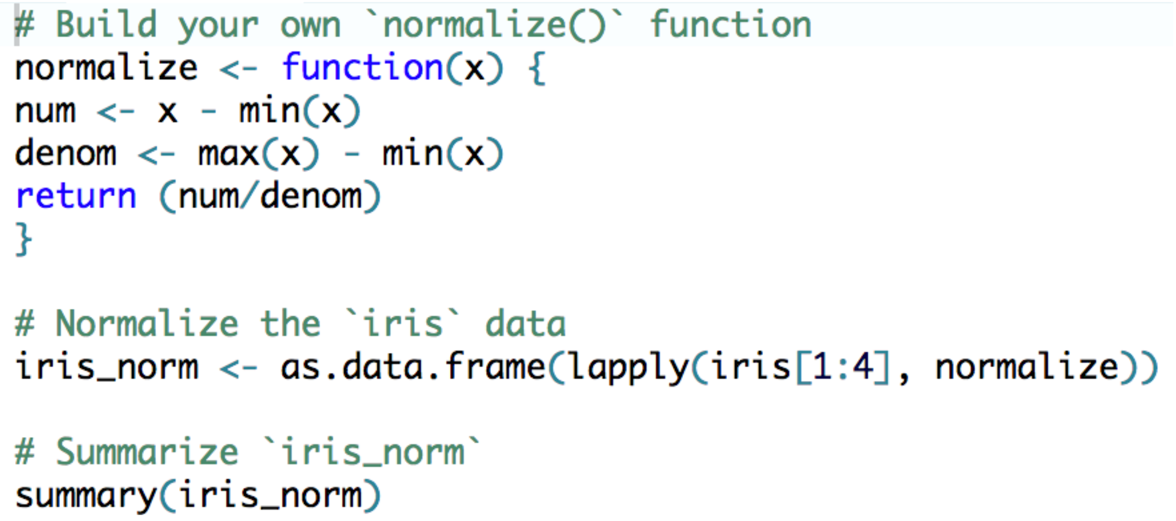
```
# Build your own `normalize()` function
normalize <- function(x) {
num <- x - min(x)
denom <- max(x) - min(x)
return (num/denom)
}

# Normalize the `iris` data
iris_norm <- .............(lapply(iris[1:4], normalize))

# Summarize `iris_norm`
summary(.........)
```

Normalization

- Use ‘iris’ data

  > head(iris)

- Normalization makes it easier for the kNN algorithm to learn
  - Guess ‘……….’

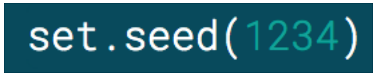


```
# Build your own `normalize()` function
normalize <- function(x) {
num <- x - min(x)
denom <- max(x) - min(x)
return (num/denom)
}

# Normalize the `iris` data
iris_norm <- as.data.frame(lapply(iris[1:4], normalize))

# Summarize `iris_norm`
summary(iris_norm)
```

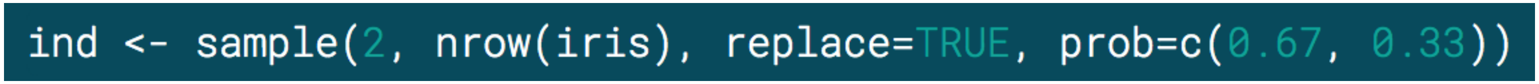
Training and test sets

- To assess your model's performance later, divide the data set into two parts: a training set and a test set
  - The first is used to train the system, while the second is used to evaluate the learned or trained system
  - The division of your data set into a test and a training sets is disjoint
  - The most common splitting choice is to take 2/3 of your original data set as the training set, while the 1/3 that remains will compose the test set

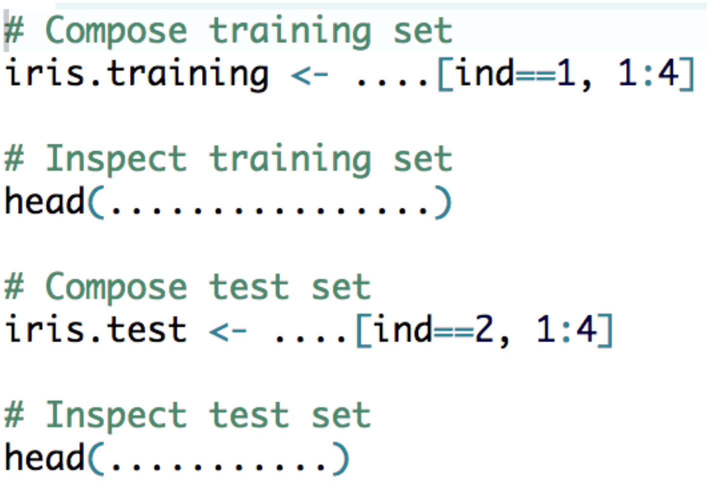
Training and test sets



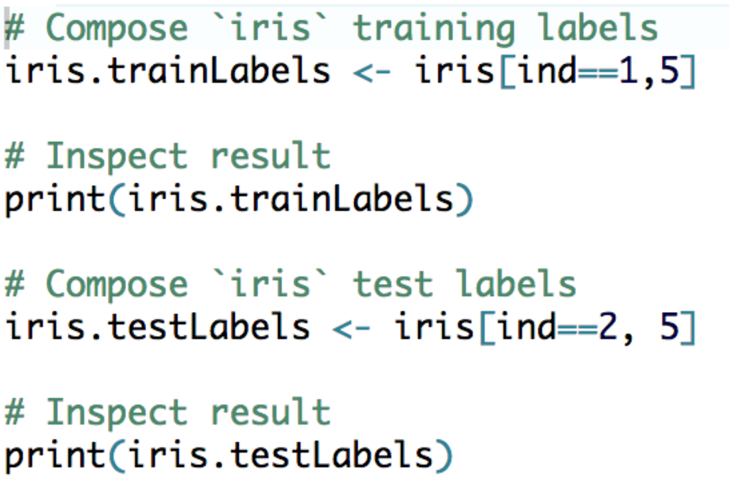
```
set.seed(1234)
```



```
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.67, 0.33))
```

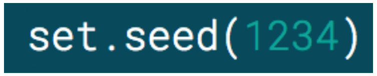


```
# Compose training set
iris.training <- ....[ind==1, 1:4]

# Inspect training set
head(...............)

# Compose test set
iris.test <- ....[ind==2, 1:4]

# Inspect test set
head(..........)
```

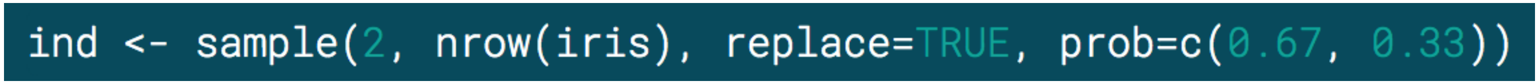


```
# Compose `iris` training labels
iris.trainLabels <- iris[ind==1,5]

# Inspect result
print(iris.trainLabels)

# Compose `iris` test labels
iris.testLabels <- iris[ind==2, 5]

# Inspect result
print(iris.testLabels)
```

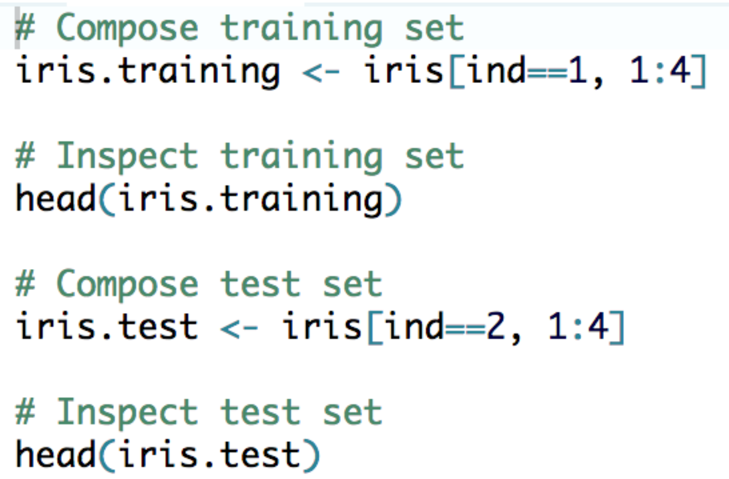
Training and test sets



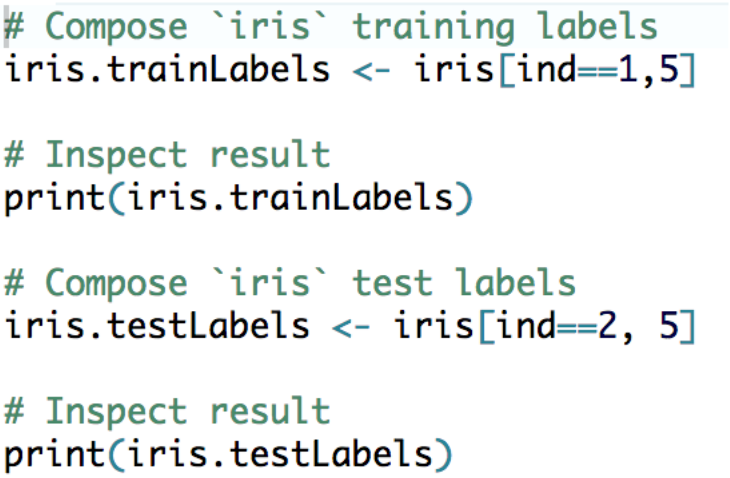
```
set.seed(1234)
```



```
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.67, 0.33))
```



```
# Compose training set
iris.training <- iris[ind==1, 1:4]

# Inspect training set
head(iris.training)

# Compose test set
iris.test <- iris[ind==2, 1:4]

# Inspect test set
head(iris.test)
```

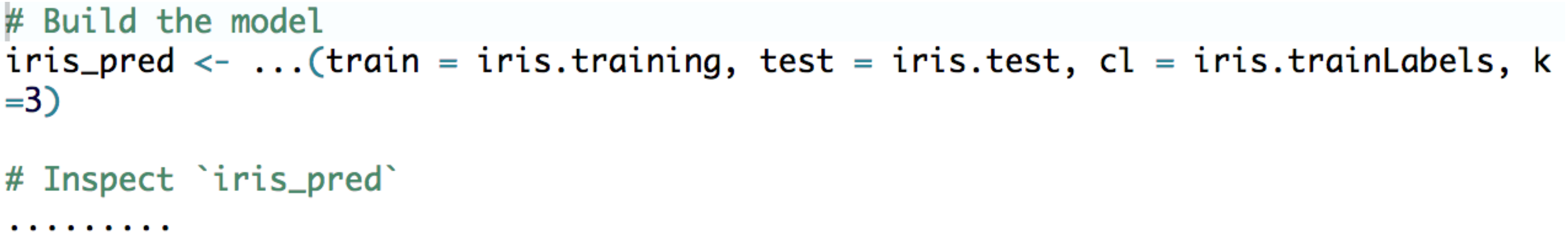


```
# Compose `iris` training labels
iris.trainLabels <- iris[ind==1,5]

# Inspect result
print(iris.trainLabels)

# Compose `iris` test labels
iris.testLabels <- iris[ind==2, 5]

# Inspect result
print(iris.testLabels)
```

Actual *k*-NN Model

- Build your classifier using knn() function

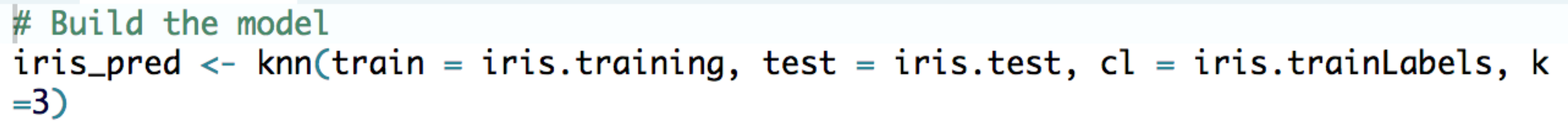
>library(class) #contains knn function



```
# Build the model
iris_pred <- ...(train = iris.training, test = iris.test, cl = iris.trainLabels, k
=3)

# Inspect `iris_pred`
.........
```

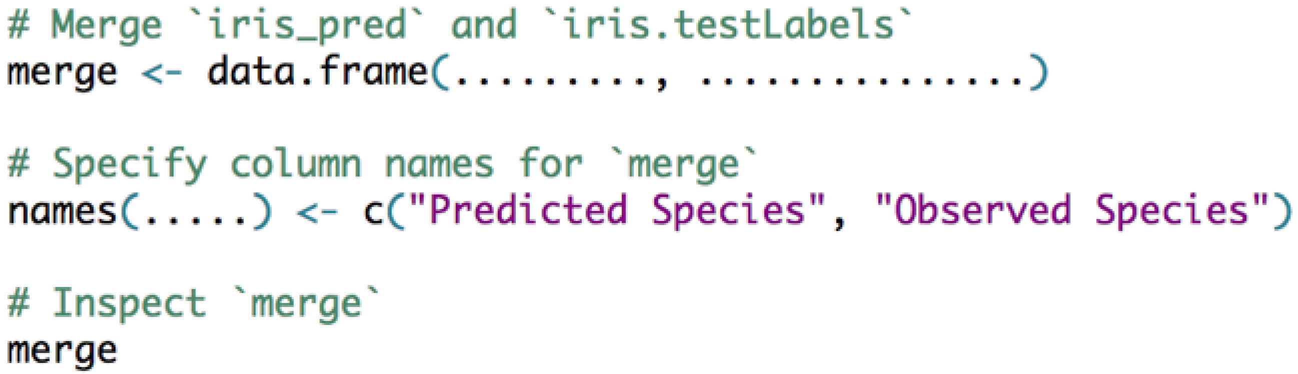
Actual *k*-NN Model

- Build your classifier using knn() function



```
# Build the model
iris_pred <- knn(train = iris.training, test = iris.test, cl = iris.trainLabels, k
=3)

# Inspect `iris_pred`
iris_pred
```

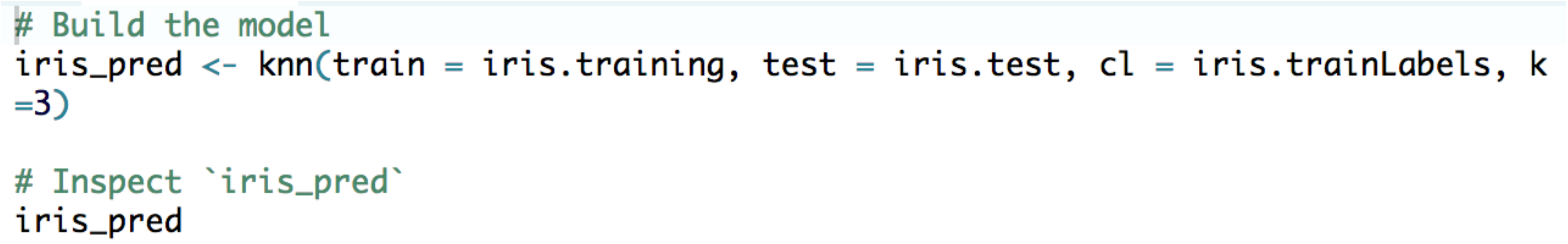
- Evaluation



```
# Merge `iris_pred` and `iris.testLabels`
merge <- data.frame(........., ...............)

# Specify column names for `merge`
names(.....) <- c("Predicted Species", "Observed Species")

# Inspect `merge`
merge
```

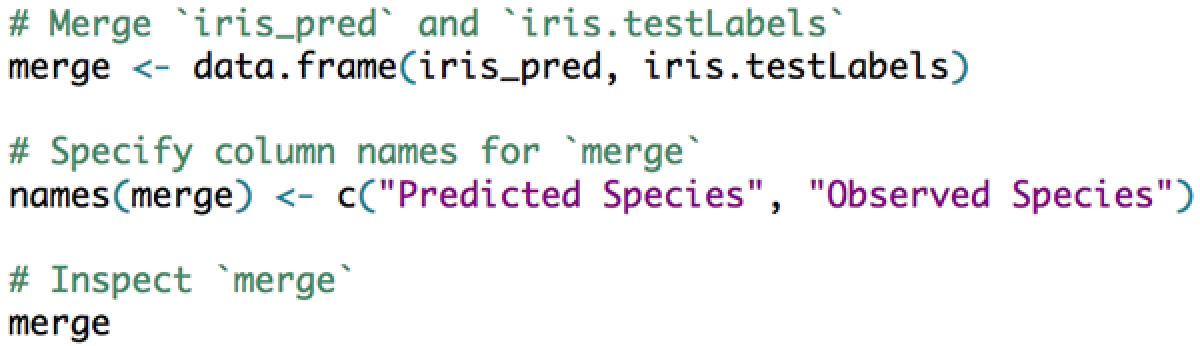
Actual *k*-NN Model

- Build your classifier using knn() function



```
# Build the model
iris_pred <- knn(train = iris.training, test = iris.test, cl = iris.trainLabels, k
=3)

# Inspect `iris_pred`
iris_pred
```

- Evaluation

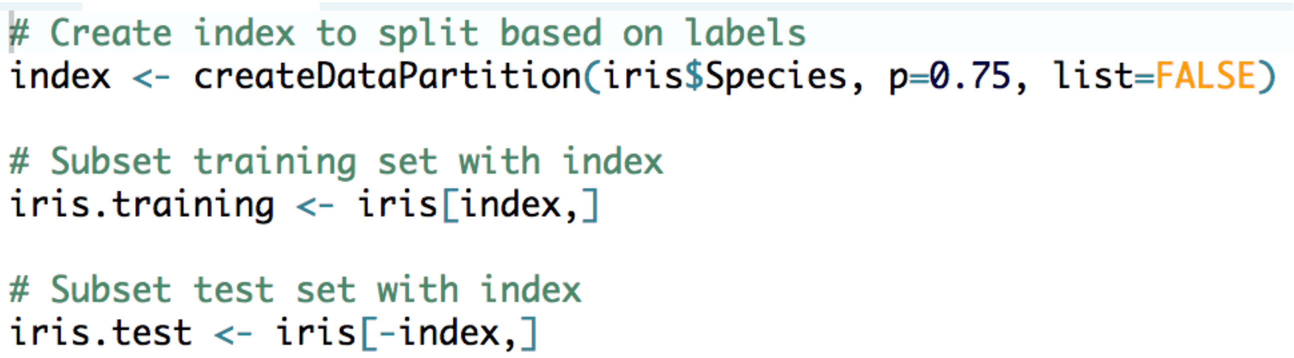


```
# Merge `iris_pred` and `iris.testLabels`
merge <- data.frame(iris_pred, iris.testLabels)

# Specify column names for `merge`
names(merge) <- c("Predicted Species", "Observed Species")

# Inspect `merge`
merge
```

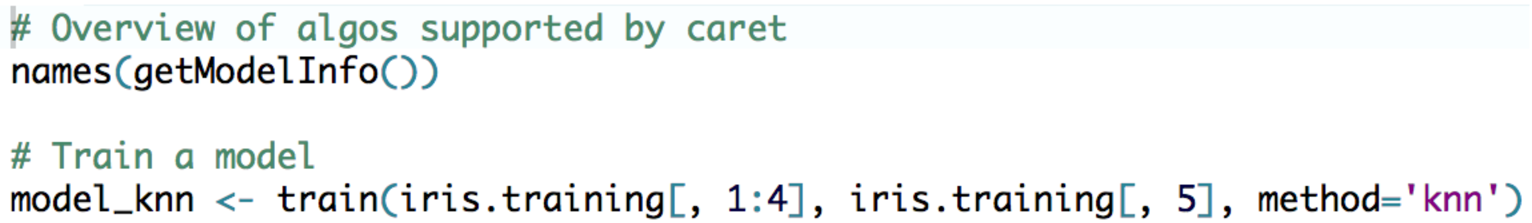
k-NN using caret package

- Simple for classification and regression training
- Use library ‘caret’ and train models

  > library(caret)



```r
# Create index to split based on labels
index <- createDataPartition(iris$Species, p=0.75, list=FALSE)

# Subset training set with index
iris.training <- iris[index,]

# Subset test set with index
iris.test <- iris[-index,]
```

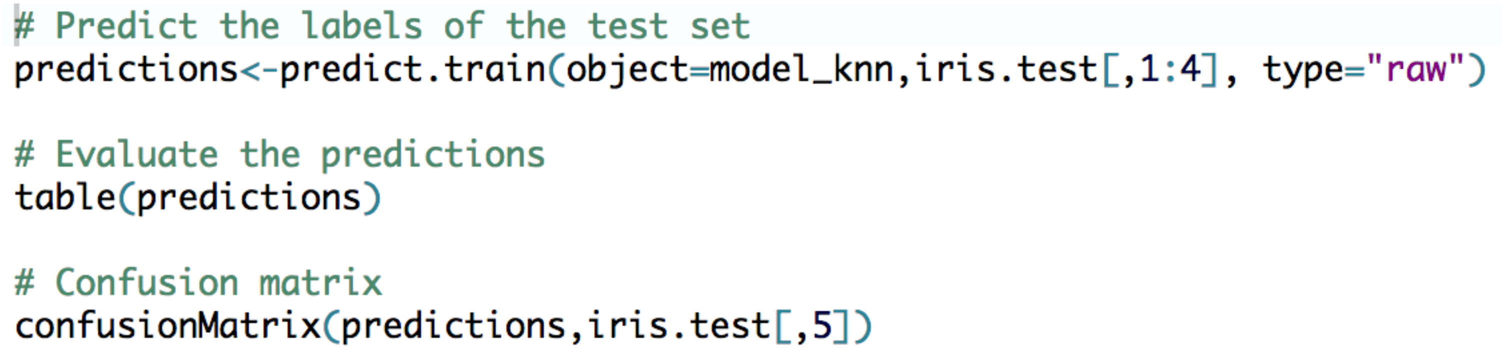


```r
# Overview of algos supported by caret
names(getModelInfo())

# Train a model
model_knn <- train(iris.training[, 1:4], iris.training[, 5], method='knn')
```

Change argument method for making other models

k-NN using caret package

- Predict the labels of the test set



```
# Predict the labels of the test set
predictions<-predict.train(object=model_knn,iris.test[,1:4], type="raw")

# Evaluate the predictions
table(predictions)

# Confusion matrix
confusionMatrix(predictions,iris.test[,5])
```