# SCIM 323

# *k*-Nearest Neightbors (*k*-NN)

# *k*-NN Classifier (Categorical Outcome)

The idea in k-nearest neighbor methods is to identify *k* records in the training dataset that are similar to the new record that we wish to classify.

We then use there similar (neighboring ) records to classify the new record into a class, assigning the new record to the predominant class among these neighbors.

# Determining neighbors

The nearest neighbor is determined by the nearest distance between records.

The most popular measure of distance is the Euclidean distance.

The Euclidean distance between the two records ($x_1$, $x_2$, …, $x_p$) and ($u_1$, $u_2$, …, $u_p$) is

$$\sqrt{(x_1-u_1)^2+(x_2-u_2)^2+\ldots+(x_p-u_p)^2}.$$

# Classification Rules

1. Find the nearest *k* neighbors to the records to be classified.

2. Use a majority decision rules to classify the record, where the record is classified as a member of the majority class of the *k* neighbors.

## Example: Riding Mowers

A riding-mower manufacturer would like to find a way of classifying families in a city into those likely to purchase a riding mower and those not likely to buy one.

y = ownership of riding mower (owner, nonowner)

x1 = income ($1000)

x2 = lot size (1000 ft$^2$)

# Example: Riding Mowers

| Household number | Income ($000s) | Lot Size (000s ft2) | Ownership of Riding Mower |
|---|---|---|---|
| 1 | 60.0 | 18.4 | Owner |
| 2 | 85.5 | 16.8 | Owner |
| 3 | 64.8 | 21.6 | Owner |
| 4 | 61.5 | 20.8 | Owner |
| 5 | 87.0 | 23.6 | Owner |
| 6 | 110.1 | 19.2 | Owner |
| 7 | 108.0 | 17.6 | Owner |
| 8 | 82.8 | 22.4 | Owner |
| 9 | 69.0 | 20.0 | Owner |
| 10 | 93.0 | 20.8 | Owner |
| 11 | 51.0 | 22.0 | Owner |
| 12 | 81.0 | 20.0 | Owner |
| 13 | 75.0 | 19.6 | Nonowner |
| 14 | 52.8 | 20.8 | Nonowner |
| 15 | 64.8 | 17.2 | Nonowner |
| 16 | 43.2 | 20.4 | Nonowner |
| 17 | 84.0 | 17.6 | Nonowner |
| 18 | 49.2 | 17.6 | Nonowner |
| 19 | 59.4 | 16.0 | Nonowner |
| 20 | 66.0 | 18.4 | Nonowner |
| 21 | 47.4 | 16.4 | Nonowner |
| 22 | 33.0 | 18.8 | Nonowner |
| 23 | 51.0 | 14.0 | Nonowner |
| 24 | 63.0 | 14.8 | Nonowner |

## Classify the new data point

Suppose the new household has \$60,000 income and lot size 20,000 ft$^2$. Should it be classified to owner or nonowner class?

Before calculating the distance, there is one step we need to do first. What is that step?

# Standardized Data

| Household number | Income ($000s) | Lot Size (000s ft2) | Ownership of Riding Mower |
|---|---|---|---|
| 1 | -0.42 | -0.25 | Owner |
| 2 | 0.89 | -0.92 | Owner |
| 3 | -0.17 | 1.09 | Owner |
| 4 | -0.34 | 0.76 | Owner |
| 5 | 0.97 | 1.93 | Owner |
| 6 | 2.16 | 0.09 | Owner |
| 7 | 2.05 | -0.58 | Owner |
| 8 | 0.76 | 1.43 | Owner |
| 9 | 0.05 | 0.42 | Owner |
| 10 | 1.28 | 0.76 | Owner |
| 11 | -0.88 | 1.26 | Owner |
| 12 | 0.66 | 0.42 | Owner |
| 13 | 0.35 | 0.25 | Nonowner |
| 14 | -0.79 | 0.76 | Nonowner |
| 15 | -0.17 | -0.75 | Nonowner |
| 16 | -1.28 | 0.59 | Nonowner |
| 17 | 0.82 | -0.58 | Nonowner |
| 18 | -0.97 | -0.58 | Nonowner |
| 19 | -0.45 | -1.25 | Nonowner |
| 20 | -0.11 | -0.25 | Nonowner |
| 21 | -1.06 | -1.09 | Nonowner |
| 22 | -1.80 | -0.08 | Nonowner |
| 23 | -0.88 | -2.09 | Nonowner |
| 24 | -0.26 | -1.76 | Nonowner |

## Classify the new data point

The new household has $60,000 income and lot size 20,000 ft$^2$.

The standardized income is -0.42 and the standardized lot size is  0.42.

# The Euclidean Distance

| Household number | Income ($000s) | Lot Size (000s ft2) | Distance | Ownership of Riding Mower |
|---|---|---|---|---|
| 1 | -0.42 | -0.25 | 0.67 | Owner |
| 2 | 0.89 | -0.92 | 1.88 | Owner |
| 3 | -0.17 | 1.09 | 0.71 | Owner |
| 4 | -0.34 | 0.76 | 0.34 | Owner |
| 5 | 0.97 | 1.93 | 2.05 | Owner |
| 6 | 2.16 | 0.09 | 2.60 | Owner |
| 7 | 2.05 | -0.58 | 2.66 | Owner |
| 8 | 0.76 | 1.43 | 1.54 | Owner |
| 9 | 0.05 | 0.42 | 0.46 | Owner |
| 10 | 1.28 | 0.76 | 1.73 | Owner |
| 11 | -0.88 | 1.26 | 0.96 | Owner |
| 12 | 0.66 | 0.42 | 1.08 | Owner |
| 13 | 0.35 | 0.25 | 0.79 | Nonowner |
| 14 | -0.79 | 0.76 | 0.50 | Nonowner |
| 15 | -0.17 | -0.75 | 1.20 | Nonowner |
| 16 | -1.28 | 0.59 | 0.88 | Nonowner |
| 17 | 0.82 | -0.58 | 1.59 | Nonowner |
| 18 | -0.97 | -0.58 | 1.15 | Nonowner |
| 19 | -0.45 | -1.25 | 1.68 | Nonowner |
| 20 | -0.11 | -0.25 | 0.74 | Nonowner |
| 21 | -1.06 | -1.09 | 1.64 | Nonowner |
| 22 | -1.80 | -0.08 | 1.48 | Nonowner |
| 23 | -0.88 | -2.09 | 2.56 | Nonowner |
| 24 | -0.26 | -1.76 | 2.18 | Nonowner |

# Classification

If *k* = 1, what class should we assign to this new data point?

# The Euclidean Distance

| Household number | Income ($000s) | Lot Size (000s ft2) | Distance | Ownership of Riding Mower |
|---|---|---|---|---|
| 1 | -0.42 | -0.25 | 0.67 | Owner |
| 2 | 0.89 | -0.92 | 1.88 | Owner |
| 3 | -0.17 | 1.09 | 0.71 | Owner |
| 4 | -0.34 | 0.76 | 0.34 | Owner |
| 5 | 0.97 | 1.93 | 2.05 | Owner |
| 6 | 2.16 | 0.09 | 2.60 | Owner |
| 7 | 2.05 | -0.58 | 2.66 | Owner |
| 8 | 0.76 | 1.43 | 1.54 | Owner |
| 9 | 0.05 | 0.42 | 0.46 | Owner |
| 10 | 1.28 | 0.76 | 1.73 | Owner |
| 11 | -0.88 | 1.26 | 0.96 | Owner |
| 12 | 0.66 | 0.42 | 1.08 | Owner |
| 13 | 0.35 | 0.25 | 0.79 | Nonowner |
| 14 | -0.79 | 0.76 | 0.50 | Nonowner |
| 15 | -0.17 | -0.75 | 1.20 | Nonowner |
| 16 | -1.28 | 0.59 | 0.88 | Nonowner |
| 17 | 0.82 | -0.58 | 1.59 | Nonowner |
| 18 | -0.97 | -0.58 | 1.15 | Nonowner |
| 19 | -0.45 | -1.25 | 1.68 | Nonowner |
| 20 | -0.11 | -0.25 | 0.74 | Nonowner |
| 21 | -1.06 | -1.09 | 1.64 | Nonowner |
| 22 | -1.80 | -0.08 | 1.48 | Nonowner |
| 23 | -0.88 | -2.09 | 2.56 | Nonowner |
| 24 | -0.26 | -1.76 | 2.18 | Nonowner |

```
> install.packages("ipred")
> library(ipred)
> library(mlbench)
> library(klaR)

> mower <- read.csv("C:/MA 299/R/mower.csv")
> mower[,2:3]=scale(mower[,2:3])#standardized column 2 and column 3
> new <- mower[25,]#define new data
> train <- mower[1:24,]#define train set

> KNN=ipredknn(ownership~income+lotsize,data=train,k=1)

> result=predict(KNN,new,"class")
> result
```

Result from R $\longrightarrow$

```
[1] Owner
attr(,"prob")
[1] 1
Levels:  Nonowner Owner
```

# Classification

If *k* = 3, what class should we assign to this new data point?

## The Euclidean Distance

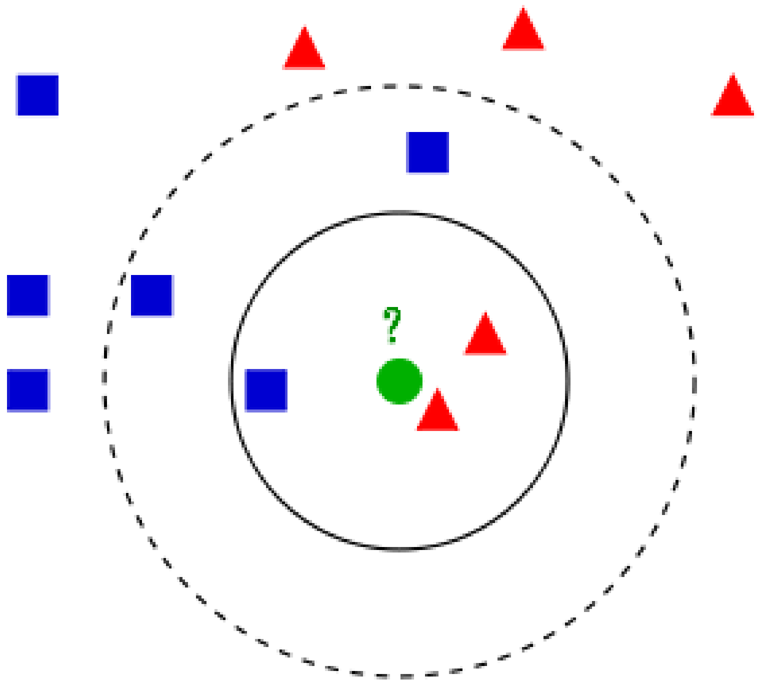
| Household number | Income ($000s) | Lot Size (000s ft2) | Distance | Ownership of Riding Mower |
|---|---|---|---|---|
| 1 | -0.42 | -0.25 | 0.67 | Owner |
| 2 | 0.89 | -0.92 | 1.88 | Owner |
| 3 | -0.17 | 1.09 | 0.71 | Owner |
| 4 | -0.34 | 0.76 | 0.34 | Owner |
| 5 | 0.97 | 1.93 | 2.05 | Owner |
| 6 | 2.16 | 0.09 | 2.60 | Owner |
| 7 | 2.05 | -0.58 | 2.66 | Owner |
| 8 | 0.76 | 1.43 | 1.54 | Owner |
| 9 | 0.05 | 0.42 | 0.46 | Owner |
| 10 | 1.28 | 0.76 | 1.73 | Owner |
| 11 | -0.88 | 1.26 | 0.96 | Owner |
| 12 | 0.66 | 0.42 | 1.08 | Owner |
| 13 | 0.35 | 0.25 | 0.79 | Nonowner |
| 14 | -0.79 | 0.76 | 0.50 | Nonowner |
| 15 | -0.17 | -0.75 | 1.20 | Nonowner |
| 16 | -1.28 | 0.59 | 0.88 | Nonowner |
| 17 | 0.82 | -0.58 | 1.59 | Nonowner |
| 18 | -0.97 | -0.58 | 1.15 | Nonowner |
| 19 | -0.45 | -1.25 | 1.68 | Nonowner |
| 20 | -0.11 | -0.25 | 0.74 | Nonowner |
| 21 | -1.06 | -1.09 | 1.64 | Nonowner |
| 22 | -1.80 | -0.08 | 1.48 | Nonowner |
| 23 | -0.88 | -2.09 | 2.56 | Nonowner |
| 24 | -0.26 | -1.76 | 2.18 | Nonowner |

```
> install.packages("ipred")
> library(ipred)
> library(mlbench)
> library(klaR)

> mower <- read.csv("C:/MA 299/R/mower.csv")
> mower[,2:3]=scale(mower[,2:3])#standardized column 2 and column 3
> new <- mower[25,]#define new data
> train <- mower[1:24,]#define train set

> KNN=ipredknn(ownership~income+lotsize,data=train,k=3)

> result=predict(KNN,new,"class")
> result
```

Result from R →

```
[1] Owner
attr(,"prob")
[1] 0.6666667
Levels:  Nonowner Owner
```

## Example: Effect of *k*



Source: http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

# Choose *k*

If we choose *k* too low, we may be fitting to the noise of the data.

If we choose *k* too high, we will miss out on the method’s ability to capture the local structure in the data.

If k = n, we simply assign all records to the majority class in the training data.

Typically, values of k falls in the range of 1-20.

The odd number is normally chosen to avoid ties.

## Common Methods to Choose *k*

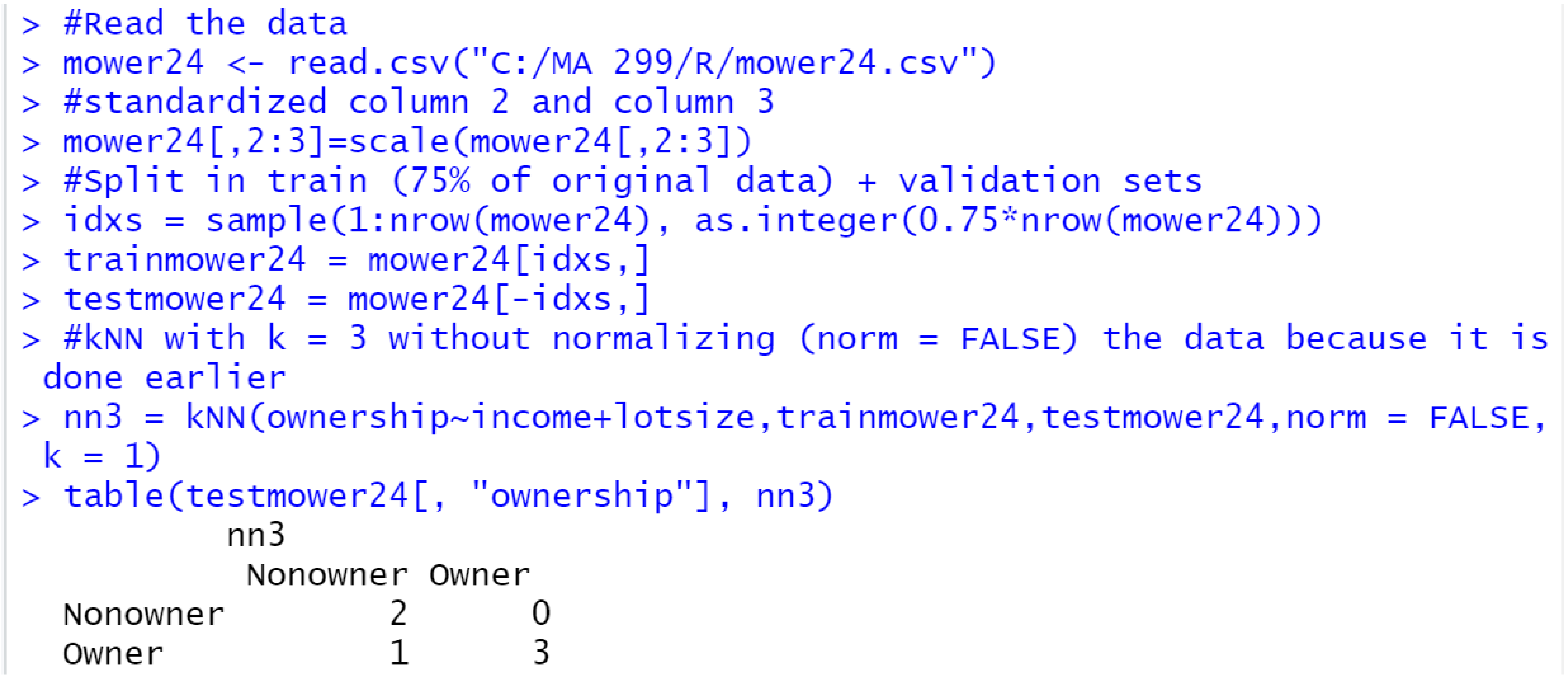
1. We partition the data into training data and validation data.

- For example, 18 data points for the training data and 6 data points for  the validation data.
- Use the training data to classify the records in the validation data, then compute the error rates for various choices of *k*.

2. Perform k-fold cross validation and record the error rate for various choices of *k*.

## Train vs Validation Sets

```
#Cross validation (train and validation sets)
> install.packages("DMwR")
#For the newer version of R, you may need to install DMwR2 instead.
> library(DMwR)
#Read the data
> mower24 <- read.csv("C:/MA 299/R/mower24.csv")
#standardized column 2 and column 3
> mower24[,2:3]=scale(mower24[,2:3])
#Split in train (75% of original data) + validation sets
> idxs = sample(1:nrow(mower24), as.integer(0.75*nrow(mower24)))
> trainmower24 = mower24[idxs,]
> testmower24 = mower24[-idxs,]
#kNN with k = 3 without normalizing the data because it is done earlier
> nn3 = kNN(ownership~income+lotsize,trainmower24,testmower24,norm = FALSE, k = 3)
> table(testmower24[, "ownership"], nn3)
```



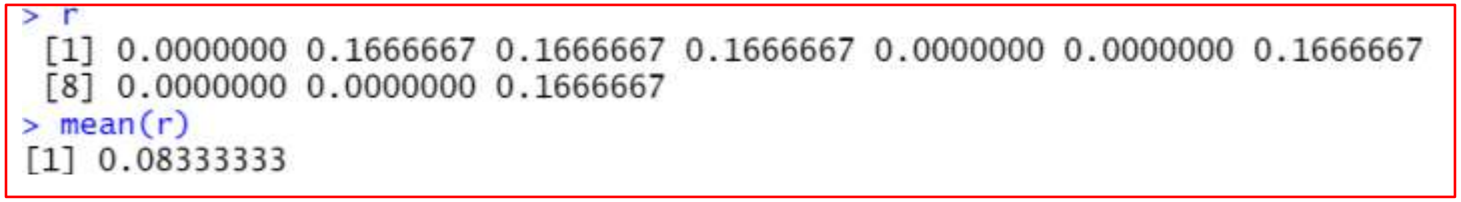
```
> #Read the data
> mower24 <- read.csv("C:/MA 299/R/mower24.csv")
> #standardized column 2 and column 3
> mower24[,2:3]=scale(mower24[,2:3])
> #Split in train (75% of original data) + validation sets
> idxs = sample(1:nrow(mower24), as.integer(0.75*nrow(mower24)))
> trainmower24 = mower24[idxs,]
> testmower24 = mower24[-idxs,]
> #kNN with k = 3 without normalizing (norm = FALSE) the data because it is
 done earlier
> nn3 = kNN(ownership~income+lotsize,trainmower24,testmower24,norm = FALSE,
 k = 1)
> table(testmower24[, "ownership"], nn3)
          nn3
           Nonowner Owner
  Nonowner        2     0
  Owner           1     3
```

```
> idxs = sample(1:nrow(mower24), as.integer(0.75*nrow(mower24)))
> trainmower24 = mower24[idxs,]
> testmower24 = mower24[-idxs,]
> #kNN with k = 3 without normalizing (norm = FALSE) the data because it is
 done earlier
> nn3 = kNN(ownership~income+lotsize,trainmower24,testmower24,norm = FALSE,
 k = 1)
> table(testmower24[, "ownership"], nn3)
          nn3
           Nonowner Owner
  Nonowner        2     0
  Owner           0     4
```

```
> idxs = sample(1:nrow(mower24), as.integer(0.75*nrow(mower24)))
> trainmower24 = mower24[idxs,]
> testmower24 = mower24[-idxs,]
> #kNN with k = 3 without normalizing (norm = FALSE) the data because it is
 done earlier
> nn3 = kNN(ownership~income+lotsize,trainmower24,testmower24,norm = FALSE,
 k = 1)
> table(testmower24[, "ownership"], nn3)
          nn3
           Nonowner Owner
  Nonowner        3     0
  Owner           0     3
```

# Train vs Validation Sets (10 runs)

```
mower24[,2:3]=scale(mower24[,2:3])
r<-rnorm(10)
for (i in 1:10) {
  idxs = sample(1:nrow(mower24), as.integer(0.75*nrow(mower24)))
  trainmower24 = mower24[idxs,]
  testmower24 = mower24[-idxs,]
  nn3 = kNN(ownership~income+lotsize,trainmower24,testmower24,norm = FALSE, k =
1)
  Table = table(testmower24[, "ownership"], nn3)
  a = Table[1,1]
  b = Table[1,2]
  c = Table[2,1]
  d = Table[2,2]
  rate = (b+c)/(a+b+c+d)
  r[i]=rate}
r
mean(r)
```



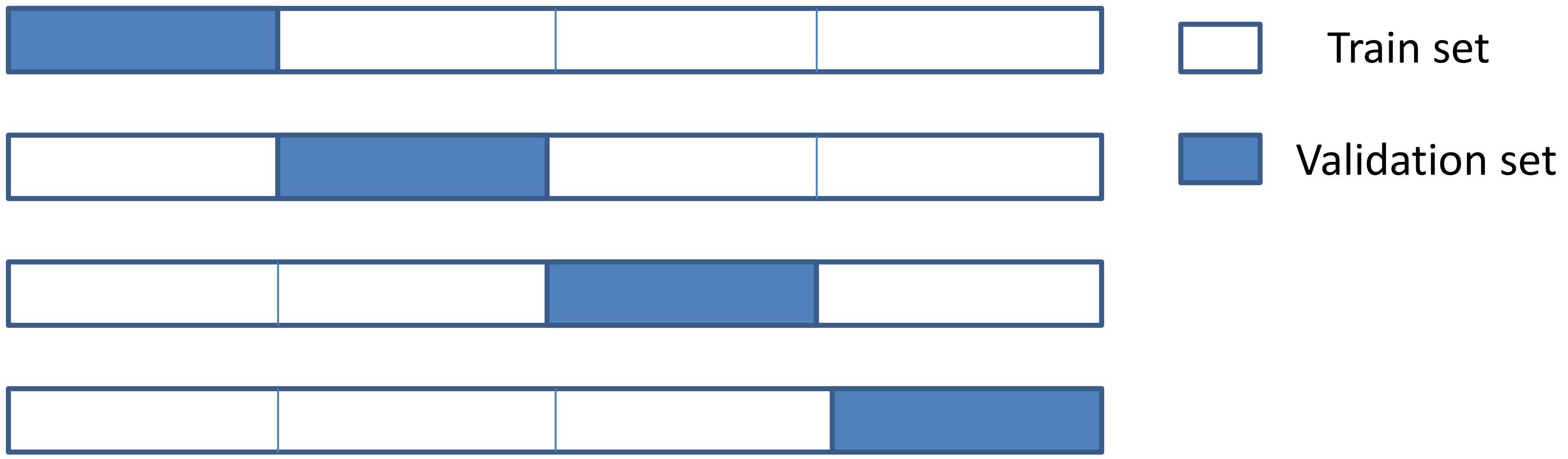
```
> r
 [1] 0.0000000 0.1666667 0.1666667 0.1666667 0.0000000 0.0000000 0.1666667
 [8] 0.0000000 0.0000000 0.1666667
> mean(r)
[1] 0.08333333
```

## Result Comparisons with 10 runs

| k | Income | Lot size | Income + Lot size |
|---|---|---|---|
| 1 | 0.017 | 0.067 | 0.050 |
| 3 | 0.017 | 0.033 | 0.050 |
| 5 | 0.067 | 0.067 | 0.083 |
| 7 | 0.100 | 0.100 | 0.033 |
| 9 | 0.050 | 0.033 | 0.050 |

# K-fold Cross Validation

Step 1: Divide the data into k roughly equal parts.



Step 2: For each k = 1, 2,…, K, fit the model with the training part and compute its error in the validation part

Step 3: Find the average error rate from all K folds.

## K-fold Cross Validation

The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing and each observation is used for validation exactly once

# k-fold

```
#k-fold
#Note: You need to install Java in your computer as well.
#Go to http://www.java.com/en/download/.
> install.packages("rJava")
> install.packages("RWekajars")
> install.packages("RWeka")
> library(rJava)
> library(RWeka)
#Read the data
> mower24 <- read.csv("C:/MA 299/R/mower24.csv")
#standardized column 2 and column 3
> mower24[,2:3]=scale(mower24[,2:3])
#k-fold cross validation for kNN
> classifier = IBk(ownership~income+lotsize,data=mower24,control
= Weka_control(K = 3, X = TRUE)) # number of neighbors is 3
> evaluate_Weka_classifier(classifier, numFolds = 24) # number of fold is 24
```

```
=== 24 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances            20          83.3333 %
Incorrectly Classified Instances           4          16.6667 %
Kappa statistic                        0.6667
Mean absolute error                    0.3248
Root mean squared error                0.4237
Relative absolute error               62.4638 %
Root relative squared error           81.4713 %
Coverage of cases (0.95 level)        91.6667 %
Mean rel. region size (0.95 level)    81.25   %
Total Number of Instances                 24

=== Confusion Matrix ===

  a  b   <-- classified as
 10  2 |  a = Nonowner
  2 10 |  b = Owner
```

For a 24-fold cross validation

| Value of k | % Misclassification rate |
|---|---|
| 1 | 37.50 |
| 2 | 37.50 |
| 3 | 16.67 |
| 4 | 16.67 |
| 5 | 16.67 |
| 6 | 16.67 |
| 7 | 16.67 |
| 8 | 16.67 |
| 9 | 16.67 |
| 10 | 16.67 |
| 11 | 20.83 |
| 12 | 20.83 |
| 13 | 20.83 |
| 14 | 20.83 |
| 15 | 20.83 |
| 16 | 20.83 |
| 17 | 20.83 |
| 18 | 20.83 |
| 19 | 20.83 |
| 20 | 20.83 |
| 21 | 20.83 |
| 22 | 20.83 |
| 23 | 20.83 |

# Example: BOSTON.HOUSING.KNN

**This dataset contains information on neighborhoods in Boston.**

**Variable Information:**

|  |  |
|---|---|
| CRIM: | per capita crime rate by town |
| ZN: | proportion of residential land zoned for lots over 25,000 sq.ft. |
| INDUS: | proportion of non-retail business acres per town |
| NOX: | nitric oxides concentration (parts per 10 million) |
| RM: | average number of rooms per dwelling |
| AGE: | proportion of owner-occupied units built prior to 1940 |
| DIS: | weighted distances to five Boston employment centres |
| RAD: | index of accessibility to radial highways |
| TAX: | full-value property-tax rate per $10,000 |
| PTRATIO: | pupil-teacher ratio by town |
| LSTAT: | % lower status of the population |
| CAT.MEDV: | Is median value of owner-occupied homes in tract above $30,000 (CAT.MEDV = 1) or not (CAT.MEDV = 0) |

Note: Compare to Boston.housing.csv, this dataset does not contain variables ‘chas’, ‘black’, and ‘medv’.

| CRIM | ZN | INDUS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | LSTAT | CAT.MEDV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.00632 | 18 | 2.31 | 0.538 | 6.575 | 65.2 | 4.09 | 1 | 296 | 15.3 | 4.98 | Low |
| 0.02731 | 0 | 7.07 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 9.14 | Low |
| 0.02729 | 0 | 7.07 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 4.03 | High |
| 0.03237 | 0 | 2.18 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 2.94 | High |
| 0.06905 | 0 | 2.18 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 5.33 | High |
| 0.02985 | 0 | 2.18 | 0.458 | 6.43 | 58.7 | 6.0622 | 3 | 222 | 18.7 | 5.21 | Low |
| 0.08829 | 12.5 | 7.87 | 0.524 | 6.012 | 66.6 | 5.5605 | 5 | 311 | 15.2 | 12.43 | Low |
| 0.14455 | 12.5 | 7.87 | 0.524 | 6.172 | 96.1 | 5.9505 | 5 | 311 | 15.2 | 19.15 | Low |
| 0.21124 | 12.5 | 7.87 | 0.524 | 5.631 | 100 | 6.0821 | 5 | 311 | 15.2 | 29.93 | Low |
| 0.17004 | 12.5 | 7.87 | 0.524 | 6.004 | 85.9 | 6.5921 | 5 | 311 | 15.2 | 17.1 | Low |
| 0.22489 | 12.5 | 7.87 | 0.524 | 6.377 | 94.3 | 6.3467 | 5 | 311 | 15.2 | 20.45 | Low |
| 0.1174 | | | | | | | | | | | |

## Experiment with 1000 runs for three models

```
#Model 1: Full model with all predictors
library(DMwR)
Boston <- read.csv("C:/MA 299/R/BOSTON.HOUSING.KNN.csv")
Boston[,1:11]=scale(Boston[,1:11])
r<-rnorm(1000)
for (i in 1:1000) {
  idxs = sample(1:nrow(Boston), as.integer(0.75*nrow(Boston)))
  trainBoston = Boston[idxs,]
  testBoston = Boston[-idxs,]
  nn3 = kNN(CAT.MEDV~CRIM+ZN+INDUS+NOX+RM+AGE+DIS+RAD+TAX+PTRATIO+LSTAT,
           trainBoston,testBoston,norm = FALSE, k = 1)
  Table = table(testBoston[, "CAT.MEDV"], nn3)
  a = Table[1,1]
  b = Table[1,2]
  c = Table[2,1]
  d = Table[2,2]
  rate = (b+c)/(a+b+c+d)
  r[i]=rate}
mean(r)
```

## Experiment with 1000 runs for three models

```
#Model 2: Model without Age
library(DMwR)
Boston <- read.csv("C:/MA 299/R/BOSTON.HOUSING.KNN.csv")
Boston[,1:11]=scale(Boston[,1:11])
r<-rnorm(1000)
for (i in 1:1000) {
 idxs = sample(1:nrow(Boston), as.integer(0.75*nrow(Boston)))
 trainBoston = Boston[idxs,]
 testBoston = Boston[-idxs,]
 nn3 = kNN(CAT.MEDV~CRIM+ZN+INDUS+NOX+RM+DIS+RAD+TAX+PTRATIO+LSTAT,
          trainBoston,testBoston,norm = FALSE, k = 1)
 Table = table(testBoston[, "CAT.MEDV"], nn3)
 a = Table[1,1]
 b = Table[1,2]
 c = Table[2,1]
 d = Table[2,2]
 rate = (b+c)/(a+b+c+d)
 r[i]=rate}
mean(r)
```

## Experiment with 1000 runs for three models

```
#Model 3: Model without Age, Indus
library(DMwR)
Boston <- read.csv("C:/MA 299/R/BOSTON.HOUSING.KNN.csv")
Boston[,1:11]=scale(Boston[,1:11])
r<-rnorm(1000)
for (i in 1:1000) {
  idxs = sample(1:nrow(Boston), as.integer(0.75*nrow(Boston)))
  trainBoston = Boston[idxs,]
  testBoston = Boston[-idxs,]
  nn3 = kNN(CAT.MEDV~CRIM+ZN+NOX+RM+DIS+RAD+TAX+PTRATIO+LSTAT,
           trainBoston,testBoston,norm = FALSE, k = 1)
  Table = table(testBoston[, "CAT.MEDV"], nn3)
  a = Table[1,1]
  b = Table[1,2]
  c = Table[2,1]
  d = Table[2,2]
  rate = (b+c)/(a+b+c+d)
  r[i]=rate}
mean(r)
```

## Result Comparison

| k | Full Model | Exclude Age | Exclude Age and Indus |
|---|---|---|---|
| 1 | 0.0604 | 0.0605 | 0.0597 |
| 3 | 0.0638 | 0.0633 | 0.0636 |
| 5 | 0.0613 | 0.0625 | 0.0627 |
| 7 | 0.0674 | 0.0676 | 0.0678 |
| 9 | 0.0705 | 0.0708 | 0.0720 |

## Notes for kNN

1. The default of the cutoff value is 0.5 but it can be set differently.
2. It can be used to classify the response with more than two classes.
3. It can be applied with a numerical response.

- The first step of determining neighbors by computing distances remains unchanged.
- The second step is modified such that we take the average response value of the k-nearest neighbors to determine the prediction.
- The best k can be determined by the other measure besides the misclassification rate.

# Notes for kNN

4. With categorical predictors, some practitioners use one hot coding.

| Location | Price |
|---|---|
| North | 10,090 |
| East | 21,300 |
| West | 10,500 |
| South | 15,000 |

| North | East | West | South | Price |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 10,090 |
| 0 | 1 | 0 | 0 | 21,300 |
| 0 | 0 | 1 | 0 | 10,500 |
| 0 | 0 | 0 | 1 | 15,000 |

## Advantages of k-NN algorithms

- It is simple.
- It does not require parametric assumptions.
- It performs well with a large enough training set.

## Shortcomings of k-NN algorithms

- The time to find the nearest neighbors in a large training set can be prohibitive.

- The number of records required in the training set to qualify as large increases exponentially with the number of predictors p.