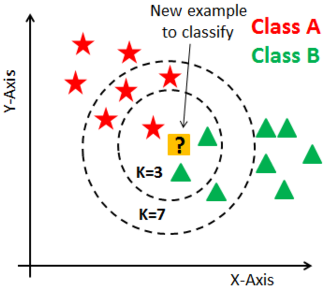
k-Nearest Neighbors

Yang Wang

College of Charleston

k-Nearest Neighbors (kNN)

- Goal: predict Y for given X when X is continuous
- Supervised method
- May be used for both classification and regression
- Decision is made based on the characteristics of the k closest points



k-Nearest Neighbors (kNN)

- Idea: Identify *k* records (neighbors) in the training dataset that are similar to a new record we wish to predict
- How to identify neighbors?
  - Use distance (Euclidean distance is most popular)
  - Euclidean distance between two records $(x_1, x_2, \ldots x_p)$ and $(u_1, u_2, \ldots, u_p)$ is given by $\sqrt{(x_1 - u_1)^2 + (x_2 - u_2)^2 + \cdots + (x_p - u_p)^2}$
  - Note: for a categorical predictor with *m* levels, convert it to *m* dummies
- To equalize the scales that the various predictors may have, usually, predictors are standardized before computation of Euclidean distance

Example: Euclidean distance

- Suppose the scores for midterm1, midterm2 and final for Student A are (80, 78, 85) and the same for Student B are (95, 97, 91)
- Recall: Euclidean distance between two records $(x_1, x_2, \ldots x_p)$ and $(u_1, u_2, \ldots, u_p)$ is given by $\sqrt{(x_1 - u_1)^2 + (x_2 - u_2)^2 + \cdots + (x_p - u_p)^2}$
- The Euclidean distance between Student A and Student B is given by $\sqrt{(80 - 95)^2 + (78 - 97)^2 + (85 - 91)^2} = 26.495$
- Exercise: Suppose the scores for Student C are (77, 80, 81). Find the Euclidean distance between Student A and Student C.

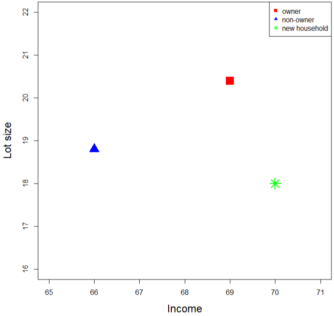
k-Nearest Neighbors (kNN): prediction rule

- Classification problem
  - Find the nearest k neighbors to the record to be classified
  - Record to be classified as a member of the majority class of the k neighbors
- Regression problem
  - Find the nearest k neighbors to the record to be preicted
  - The predicted response value of the new record can be computed by taking the average of the response values of the k neighbors

How to choose the neighbors?

- Consider the dataset called RidingMowers.csv. The response variable Ownership denotes whether a family owns a riding mower or not.The predictors are Income and Lotsize of the associated household.
- Income ranges from $33-$110.1K. LotSize ranges from 14-23.6 sqft
- Two households are considered in the plot. One is owner (Income = 69, LotSize = 20), the other is non-owner (Income = 66, LotSize = 18.4).
- A new household is introduced where Income = 70, LotSize = 18
- Who is the closest neighbor?

- For new household: $Income = 70, LotSize = 18$
- For owner: $Income = 69, LotSize = 20$
- For non-owner: $Income = 66, LotSize = 18.4$
- Euclidean distance:
    - between ”new” and ”owner”: $\sqrt{(70-69)^2+(18-20)^2} = 2.23$
    - between ”new” and ”nonowner”: $\sqrt{(70-66)^2+(18-18.4)^2} = 4.02$



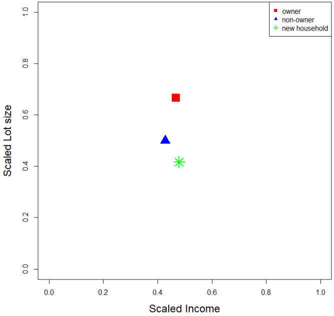
Normalizing and rescaling

- To equalize the scales that the various predictors may have, usually, predictors are standardized before computation of Euclidean distance
- Standardization: subtract mean from each value and then divide by the standard deviation (this tells us the data points are how many standard deviation away from the mean value. R function *scale*)
- Normalization: subtract the minimum value and then divide by the range (this will make all variables to lie between [0,1]. (R function *rescale* from *scales* package)

How to choose the neighbors?

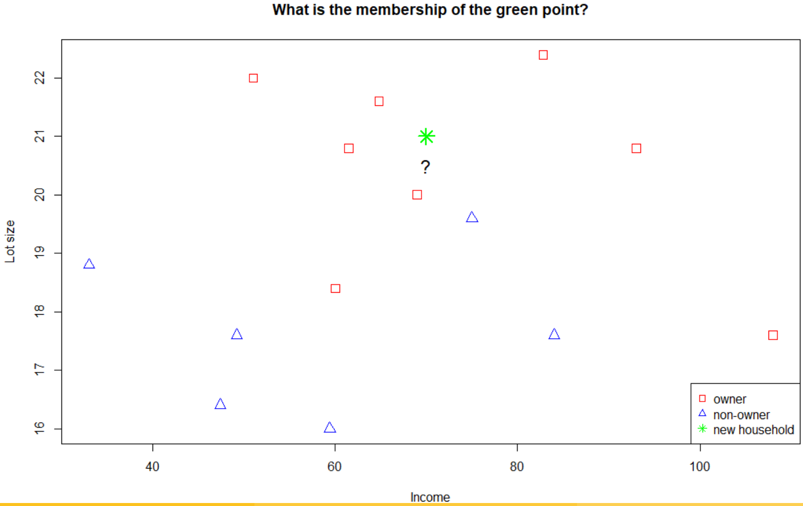
- Calculate distance after rescaling *i.e.* subtract the minimum value and then divide by the range
- For the new observation, use the minimum and maximum value for the training data
- Example: for new observation
  - $ScaledIncome = (70 - 33)/(110.1 - 33) = 0.48$
  - $ScaledLotSize = (18 - 14)/(23.6 - 14) = 0.42$

- For new household: $ScaledIncome = 0.48$, $ScaledLotSize = 0.42$
- For owner: $ScaledIncome = 0.47$, $ScaledLotSize = 0.625$
- For non-owner: $ScaledIncome = 0.43$, $ScaledLotSize = 0.46$
- Euclidean distance:
  - between "new" and "owner": $= 0.21$
  - between "new" and "nonowner": $= 0.07$

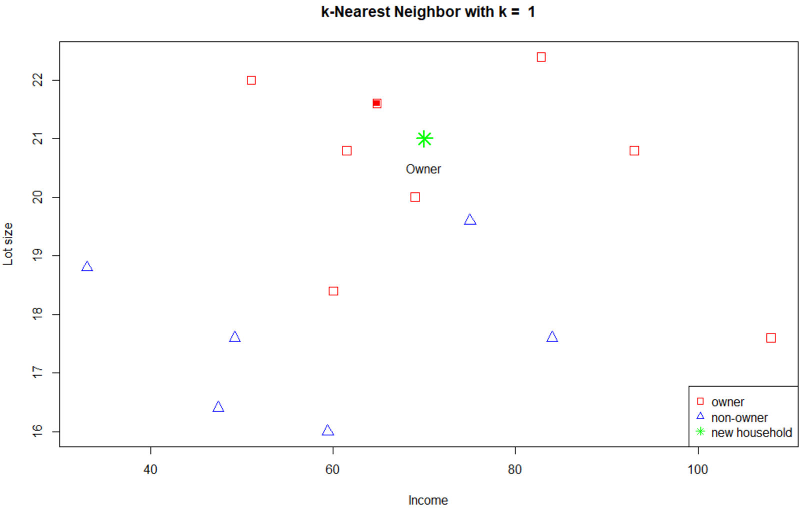


k-Nearest Neighbors (kNN)

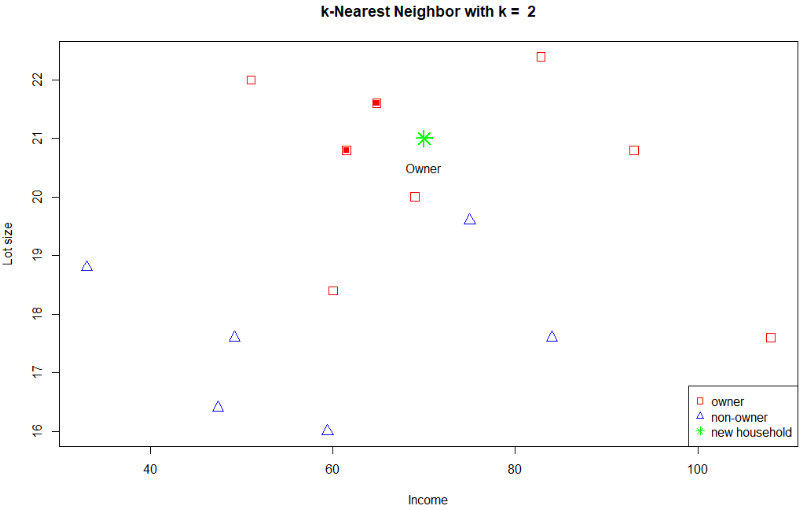
14 households are classified by ownership: 8 owner, 6 nonowner
What is the membership of the new household?



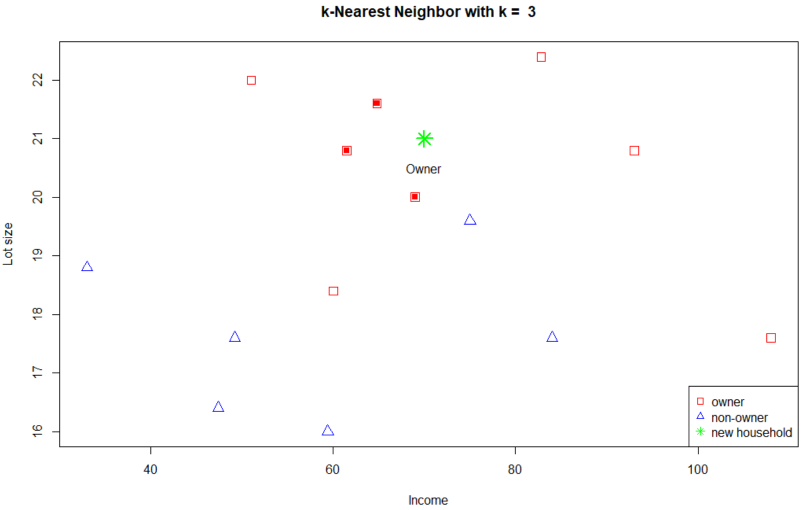
k-Nearest Neighbors (kNN)



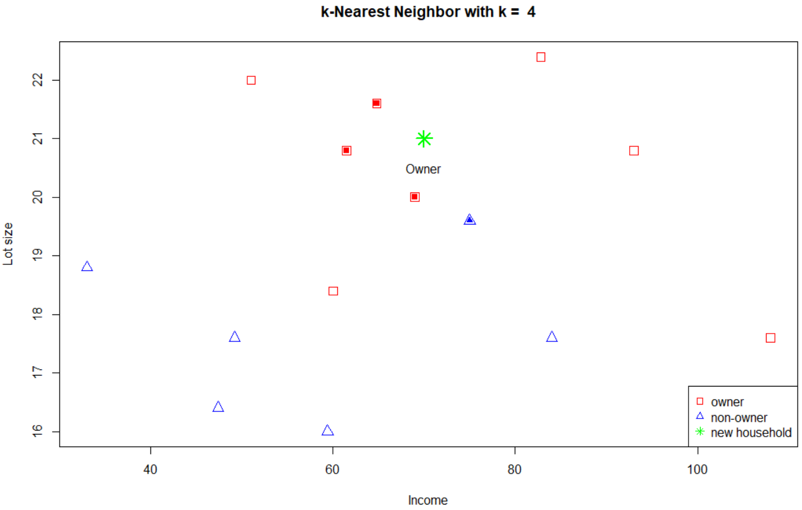
k-Nearest Neighbors (kNN)



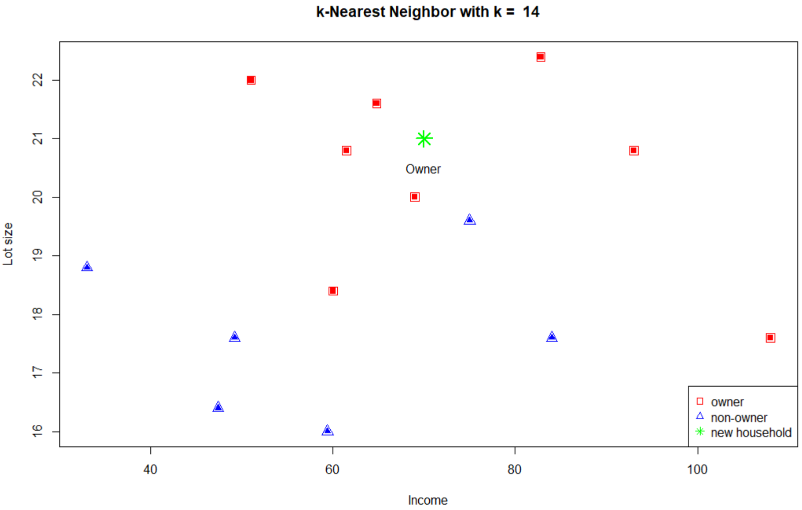
k-Nearest Neighbors (kNN)



k-Nearest Neighbors (kNN)

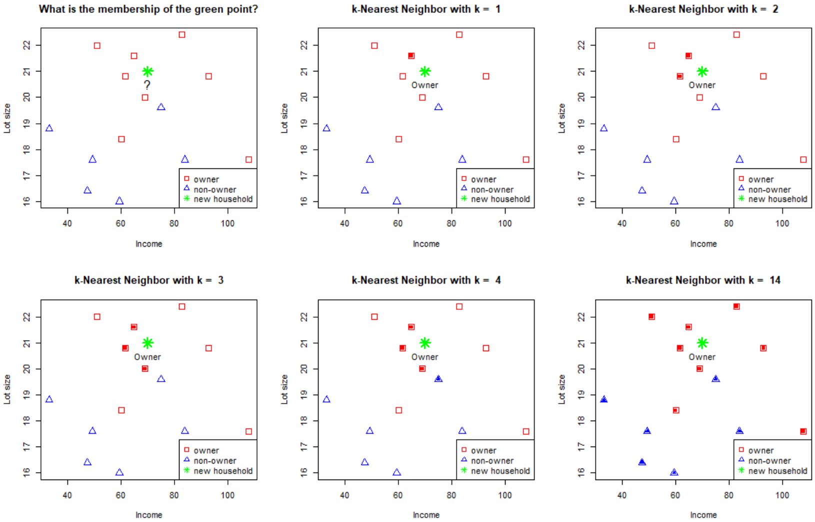


k-Nearest Neighbors (kNN)



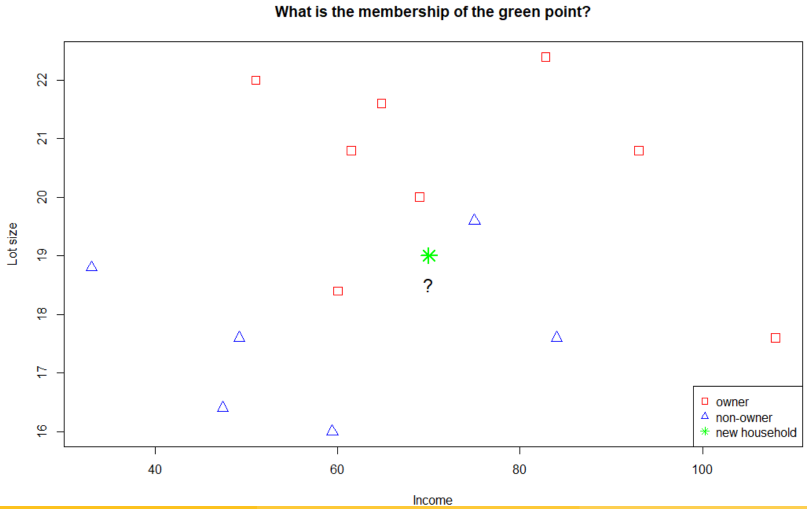
k-Nearest Neighbors (kNN)

14 households are classified by ownership: 8 owner, 6 nonowner

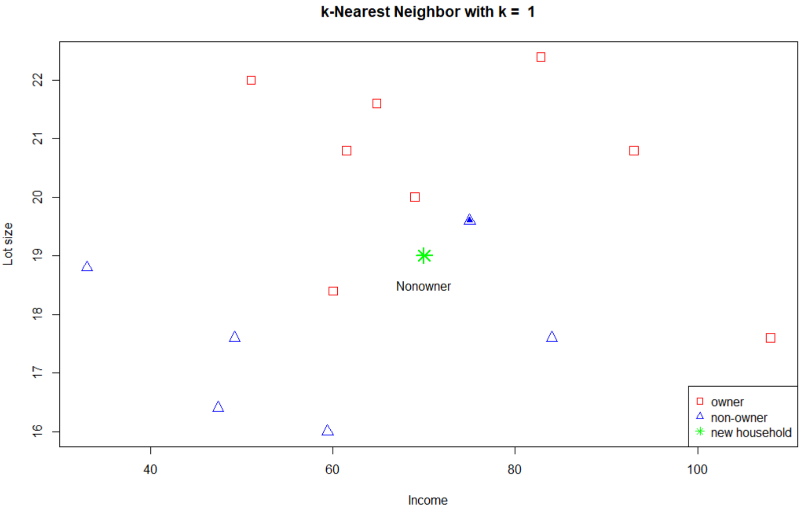


k-Nearest Neighbors (kNN)

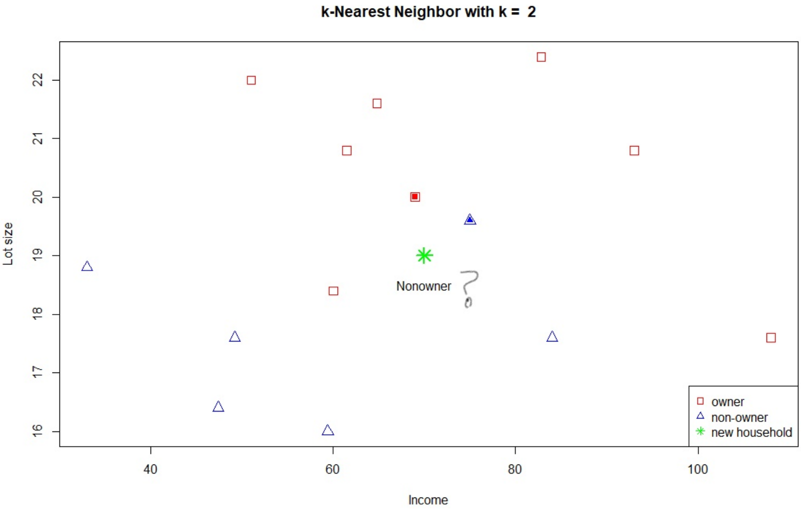
Let's look at another example
What is the membership of the this household?



k-Nearest Neighbors (kNN)



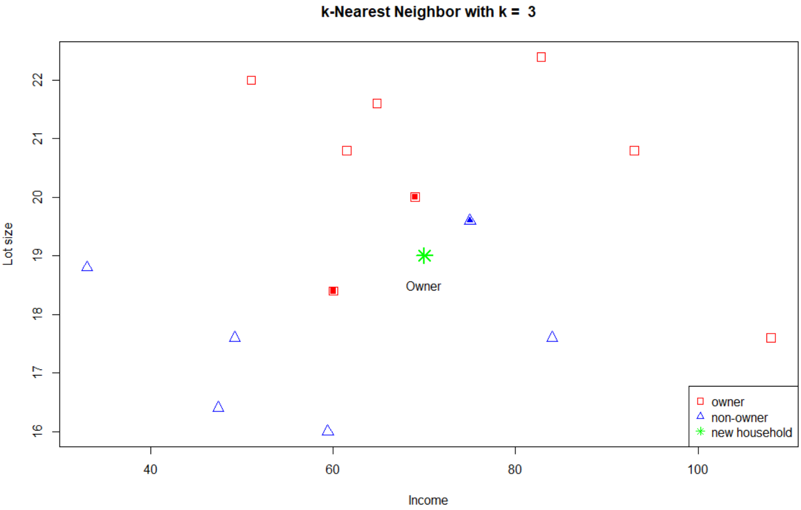
k-Nearest Neighbors (kNN)



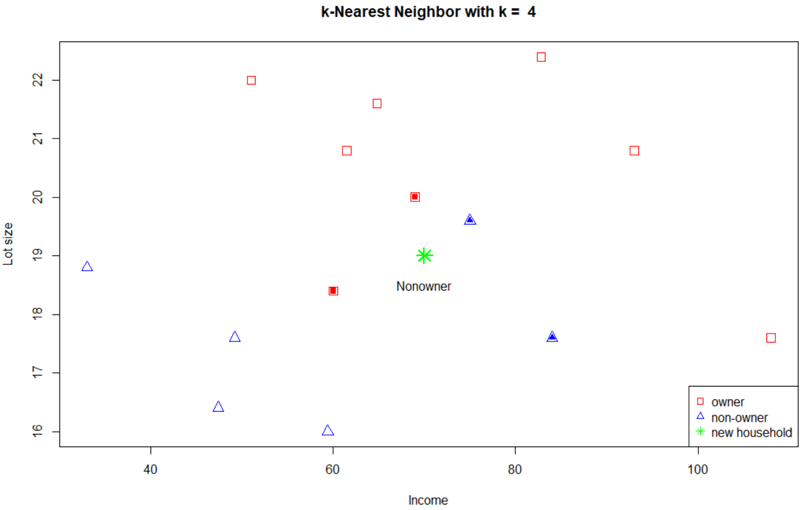
Knn: how to assign membership if there is a tie

- Problem: for $k = 2$, the nearest neighbors are observations 1 and 14 with labels "0" and "1" respectively. What class membership should we assign?
- Solution 1: flip a coin to decide which class membership to assign
- Solution 2: Use $k = n$ to decide membership *i.e* the class with higher representation in training data is to be assigned
- Solution 3: Use $k = 1$ *i.e.* assign membership based on the membership of the single nearest neighbor
- Solution 4: Calculate the total distance for neighbors labeled "0" and same for neighbors labeled "1". Assign membership label for which total distance is shorter.
- Solution 5: Use the next odd number.

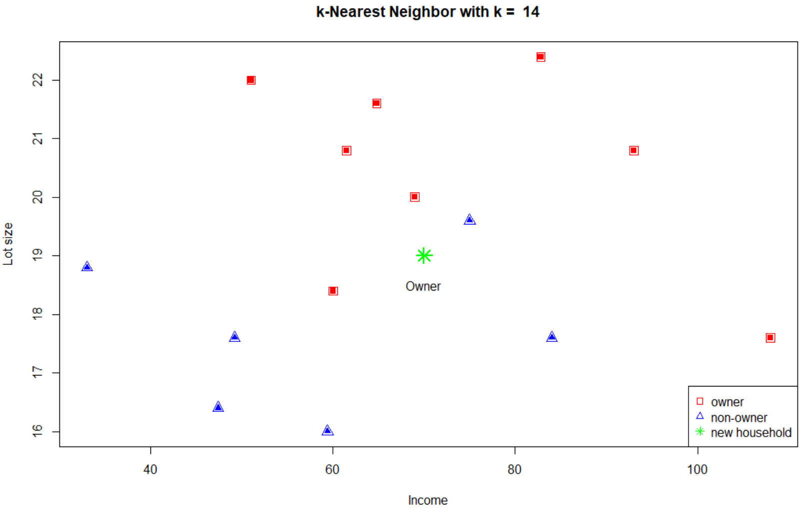
k-Nearest Neighbors (kNN)



k-Nearest Neighbors (kNN)

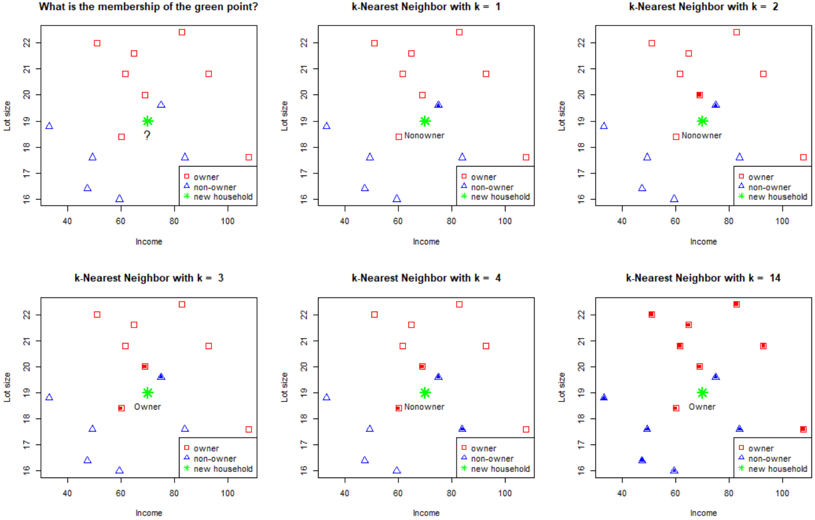


k-Nearest Neighbors (kNN)



k-Nearest Neighbors (kNN)

14 households are classified by ownership: 8 owner, 6 nonowner



k-Nearest Neighbors (kNN): how to choose k?

- Choose the k with the best prediction performance on validation data
  - Use training data to make prediction on validation data
  - Compute error rate on validation data for various choices of k
  - Choose k with minimum error rate
  - Use odd numbers to avoid ties
- Note: for this algorithm the validation set is used as a part of the training process (to set k)
- As a result of the above, it would be ideal to use a test set for evaluating model performance *i.e* the model building process involves the following:
  - partition data into training, validation and test set,
  - use training data for making prediction on validation data
  - use validation data for choosing optimal k
  - use test for evaluating model performance with the chosen k

Example of k-NN for classification problem using R

Use diabetes data with binary response variable.

```
diabetes <- read.csv("E:/Data mining/datasets/diabetes.csv")
diabetes$Outcome <- as.factor(diabetes$Outcome)
levels(diabetes$Outcome) <- c("no", "yes")

## Include the functions required for data partitioning
source("E:/Data mining/Lecture Notes/myfunctions.R")

RNGkind (sample.kind = "Rounding")
set.seed(0)
### call the function for creating 70:20:10 partition
p3 <- partition.3(diabetes, 0.7, 0.2)
training.data <- p3$data.train
validation.data <- p3$data.val
test.data <- p3$data.test

### Rescale the data
training.scaled <- scale(training.data[,-9], center = TRUE, scale = TRUE)
training.scaled.wY <- cbind(training.scaled, training.data[,9])
training.scaled.attr <- attributes(training.scaled)
val.scaled <- scale(validation.data[,-9],
                    center = training.scaled.attr$`scaled:center`,
                    scale = training.scaled.attr$`scaled:scale`)
test.scaled <- scale(test.data[,-9],
                    center = training.scaled.attr$`scaled:center`,
                    scale = training.scaled.attr$`scaled:scale`)
```

Example of k-NN for classification problem using R

Fit Knn on a single new observation.

```
### Fit kNN model on a single new observation with k=5
newObs <- data.frame(Pregnancies = 3, Glucose = 120, BloodPressure = 70,
                     SkinThickness = 20, Insulin = 80, BMI = 30,
                     DiabetesPedigreeFunction = 0.44, Age = 46)
newObs.scaled <- scale(newObs,
                    center = training.scaled.attr$`scaled:center`,
                    scale = training.scaled.attr$`scaled:scale`)
library(FNN)
Knn <- knn(train = training.scaled, test = newObs.scaled,
           cl = training.data[,9], k = 5)
> Knn
[1] no
attr(,"nn.index")
     [,1] [,2] [,3] [,4] [,5]
[1,]  233  332  314  111  455
attr(,"nn.dist")
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 1.125403 1.289751 1.372173 1.399699 1.481678
Levels: no
>
> ## labels of nearest neighbors
> Knn.attr <- attributes(Knn)
> training.data[Knn.attr$nn.index,9]
[1] no  no  yes no  yes
Levels: no yes
```

Example of k-NN for classification problem using R: Understand the output

```
> Knn
[1] no
attr(,"nn.index")
     [,1] [,2] [,3] [,4] [,5]
[1,]  233  332  314  111  455
attr(,"nn.dist")
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 1.125403 1.289751 1.372173 1.399699 1.481678
Levels: no
>
> ## labels of nearest neighbors
> Knn.attr <- attributes(Knn)
> training.data[Knn.attr$nn.index,9]
[1] no  no  yes no  yes
Levels: no yes
```

- With $K = 5$, the nearest neighbors are observations $233, 332, 314, 111, 455$ (see attr(,"nn.index") for this).
- The distance from the neighbors are $1.125403, 1.289751, 1.372173, 1.399699, 1.481678$ respectively (see attr(,"nn.dist") for this).
- The labels for the nearest neighbors are *no*, *no*, *yes*, *no*, *yes* (get it from training.data[Knn.attr$nn.index,9]).
- By the majority rule assign class membership *no* to the new observation.
- Suppose we want to infer the outcome for $k = 3$ from this result. If $k = 3$, the nearest neighbors are observations $233, 332, 314$ with labels *no*, *no*, *yes*. By the majority rule assign class membership *no* to the new observation.

Example of k-NN for classification problem using R

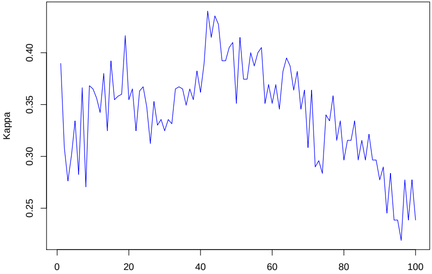
Note: The rule of majority is effectively same as using a 0.5 cut-off value. If different cutoff is needed, calculate the proportion of ”yes” labels among the *k* nearest neighbors and use a cutoff on that.

```
### Using different cutoff values
# First we need to collect the labels of the nearest neighbors
k.labels <- training.data[Knn.attr$nn.index,9]
K <- 5
cutoff <- 0.3 # if proportion of occurrences of class "yes" > cutoff then predicted label = "yes"
pred <- ifelse((sum(k.labels == "yes")/K) >= cutoff, "yes", "no")
> pred
[1] "yes"
```

Example of k-NN for classification problem using R

Use validation data to find optimal k. Kappa is maximum for k=44. Hence k=44 is optimal value.

```
### fit k-nn model for k = 1, ..., 100
K <- 100
kappa <- rep(0, K)
for (kk in 1:K){
  Knn <- knn(train = training.scaled, test = val.scaled,
             cl = training.data[,9], k = kk)
  c <- confusionMatrix(as.factor(Knn), as.factor(validation.data[,9]),
                       positive = "yes")
  kappa[kk] <- c$overall["Kappa"]
  cat("K", kk, "Kappa", kappa[kk], "\n")
}
# create a plot for k vs
plot(c(1:K), kappa, xlab                                         ")
```



Example of k-NN for classification problem using R

Get final model and evaluate on test data

```
### fit k-nn model on test data with k=44
training.data.all <- rbind(training.data, validation.data)
training.data.scaled.all <- rbind(training.scaled, val.scaled)
Knn <- knn(train = training.data.scaled.all, test = test.scaled,
           cl = training.data.all[,9], k = which.max(kappa))
> confusionMatrix(as.factor(Knn), as.factor(test.data[,9]),
                positive = "yes")
Confusion Matrix and Statistics

          Reference
Prediction no yes
       no  48  10
       yes  4  14

               Accuracy : 0.8158
                 95% CI : (0.7103, 0.8955)
    No Information Rate : 0.6842
    P-Value [Acc > NIR] : 0.007432
                  Kappa : 0.543
 Mcnemar's Test P-Value : 0.181449
            Sensitivity : 0.5833
            Specificity : 0.9231
         Pos Pred Value : 0.7778
         Neg Pred Value : 0.8276
             Prevalence : 0.3158
         Detection Rate : 0.1842
   Detection Prevalence : 0.2368
      Balanced Accuracy : 0.7532
       'Positive' Class : yes
```

Example of k-NN for classification problem using R

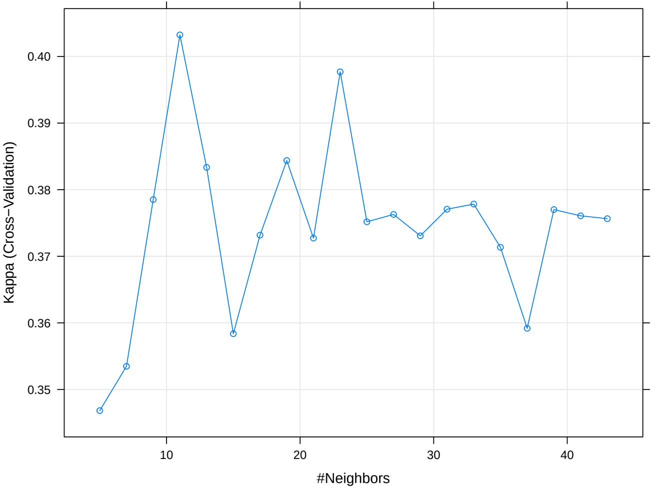
Using cross-validation approach

```
## K-fold Cross Validation
# value of K equal to 10
set.seed(0)
train_control <- trainControl(method = "cv",
                              number = 10)
training.data.all <- rbind(training.data, validation.data)
# Fit K-fold CV model
Knn_kcv <- train(Outcome ~ ., data = training.data.all, method = "knn",
                 trControl = train_control, preProcess = c("center","scale"),
                 tuneLength = 20, metric = "Kappa")
> Knn_kcv$finalModel
11-nearest neighbor model
Training set outcome distribution:

 no yes
448 244
### fit k-nn model on test data with k=11
training.data.scaled.all <- rbind(training.scaled, val.scaled)
Knn <- knn(train = training.data.scaled.all, test = test.scaled,
           cl = training.data.all[,9], k = 11)
confusionMatrix(as.factor(Knn), as.factor(test.data[,9]),
                positive = "yes")
```

Example of k-NN for classification problem using R

Using cross-validation approach



Practice assignment: k-NN for regression problem using R

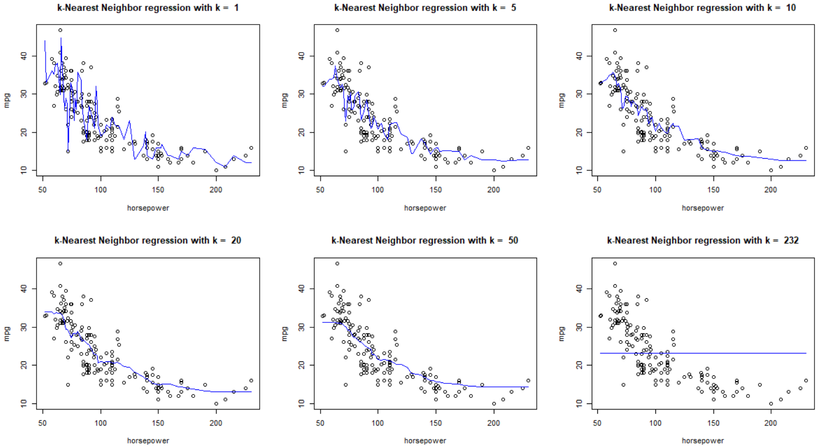
- Load autompg data.
- Partition data into training, validation and test.
- Standardize data using scale() function.
- Use knn.reg from package FNN to fit knn model to make prediction on validation data using k=5.
- Find an optimal value of *k* using cross validation approach.
- Evaluate the performance of the chosen model on test data

k-NN for categorical predictors

- For categorical data we cannot calculate Euclidean distance between them.
- create dummy variables for a categorical variable.

k-Nearest Neighbors (kNN Regression)

- If k is too low we may be fitting to the noise
- If k is too high the local structure of the data may be ignored
- Note: k-NN with $k = n$ yeilds a fit that equals $\bar{y}$
- Models below are fitted on Auto.csv training data with 232 records



Advantages kNN

- Simplicity
- Nonparametric: no assumption has been made regarding the form of relationship between response and predictors

Shortcomings of kNN

- Curse of dimensionality: the number of records required in the training set to qualify as large increases exponentially with the number of predictors p.
- Computationally expensive: for each new prediction, the distance between the new record and all training data points need to be calculated at the time of prediction (lazy learner).
  - Remedy 1: dimension reduction: work in a reduced dimension space (use principal component analysis)
  - Remedy 2: Locality Sensitive Hashtag (LSH)

- Create random hyper-planes $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k$ that slice the space into 2^k regions.
- Compare the new observation (test) only to the training points in the same region.
- Caution: it is possible that the nearest neighbors belong to a different region.
- Remedy: repeat the process many times