KNN CLASSIFIER- CARET

Sruthi Kizhakathra

July 6, 2018

KNN

PROPERTIES:

Non parametric classifier Good for small to med size datasets Simple to implement and ease of undersatandability Closer to Bayes classifier, no need to understand underlying distribution or structure of data No retraining required if new training patetrn is added to existing training set. Choice of distance metric

Sensitive to outliers Not scalable for larger datasets; time complexity. For every test data, entire distance cal needs to be done from scratch again

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

The data set used constitutes data derived from 3 types of wines.

```r
dataurl <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
download.file(url = dataurl, destfile = "wine.data")
wine_df <- read.csv("wine.data", header = FALSE)
```

The dataset involves 13 attributes describing characteristics of wine and one attribute describing the class label.

```r
str(wine_df)
```

```
## 'data.frame':    178 obs. of  14 variables:
##  $ V1 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V2 : num  14.2 13.2 13.2 14.4 13.2 ...
##  $ V3 : num  1.71 1.78 2.36 1.95 2.59 1.76 1.87 2.15 1.64 1.35 ...
##  $ V4 : num  2.43 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 ...
##  $ V5 : num  15.6 11.2 18.6 16.8 21 15.2 14.6 17.6 14 16 ...
##  $ V6 : int  127 100 101 113 118 112 96 121 97 98 ...
##  $ V7 : num  2.8 2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2.98 ...
##  $ V8 : num  3.06 2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15 ...
##  $ V9 : num  0.28 0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.29 0.22 ...
##  $ V10: num  2.29 1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85 ...
##  $ V11: num  5.64 4.38 5.68 7.8 4.32 6.75 5.25 5.05 5.2 7.22 ...
##  $ V12: num  1.04 1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01 ...
##  $ V13: num  3.92 3.4 3.17 3.45 2.93 2.85 3.58 3.58 2.85 3.55 ...
##  $ V14: int  1065 1050 1185 1480 735 1450 1290 1295 1045 1045 ...
```

Coverting target/label variable into factor. Also, dataset is clean with no missing values.

```r
wine_df$V1 <- factor(wine_df$V1)
summary(wine_df)
```

```
##  V1           V2              V3              V4              V5
##  1:59   Min.   :11.03   Min.   :0.740   Min.   :1.360   Min.   :10.60
##  2:71   1st Qu.:12.36   1st Qu.:1.603   1st Qu.:2.210   1st Qu.:17.20
##  3:48   Median :13.05   Median :1.865   Median :2.360   Median :19.50
##         Mean   :13.00   Mean   :2.336   Mean   :2.367   Mean   :19.49
##         3rd Qu.:13.68   3rd Qu.:3.083   3rd Qu.:2.558   3rd Qu.:21.50
##         Max.   :14.83   Max.   :5.800   Max.   :3.230   Max.   :30.00
##        V6              V7              V8              V9
##  Min.   : 70.00   Min.   :0.980   Min.   :0.340   Min.   :0.1300
##  1st Qu.: 88.00   1st Qu.:1.742   1st Qu.:1.205   1st Qu.:0.2700
##  Median : 98.00   Median :2.355   Median :2.135   Median :0.3400
##  Mean   : 99.74   Mean   :2.295   Mean   :2.029   Mean   :0.3619
##  3rd Qu.:107.00   3rd Qu.:2.800   3rd Qu.:2.875   3rd Qu.:0.4375
##  Max.   :162.00   Max.   :3.880   Max.   :5.080   Max.   :0.6600
##       V10             V11              V12              V13
##  Min.   :0.410   Min.   : 1.280   Min.   :0.4800   Min.   :1.270
##  1st Qu.:1.250   1st Qu.: 3.220   1st Qu.:0.7825   1st Qu.:1.938
##  Median :1.555   Median : 4.690   Median :0.9650   Median :2.780
##  Mean   :1.591   Mean   : 5.058   Mean   :0.9574   Mean   :2.612
##  3rd Qu.:1.950   3rd Qu.: 6.200   3rd Qu.:1.1200   3rd Qu.:3.170
##  Max.   :3.580   Max.   :13.000   Max.   :1.7100   Max.   :4.000
##       V14
##  Min.   : 278.0
##  1st Qu.: 500.5
##  Median : 673.5
##  Mean   : 746.9
##  3rd Qu.: 985.0
##  Max.   :1680.0
```

Lets split data for test train split

```
set.seed(2033)
intrain <- createDataPartition(wine_df$V1, p=0.7,list=FALSE)
training <- wine_df[intrain,]
testing <- wine_df[-intrain,]
```

```
dim(training); dim(testing)
```

```
## [1] 126  14
```

```
## [1] 52 14
```

Trainning KNN

```
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(3333)
knn_fit <- train(V1 ~., data = training, method = "knn",
 trControl=trctrl,
 preProcess = c("center", "scale"),
 tuneLength = 10)

knn_fit
```

```
## k-Nearest Neighbors
##
## 126 samples
##  13 predictor
##   3 classes: '1', '2', '3'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 113, 113, 114, 113, 112, 114, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.9623932  0.9436795
##    7  0.9544872  0.9316619
##    9  0.9495726  0.9246369
##   11  0.9440171  0.9164910
##   13  0.9576923  0.9368928
##   15  0.9519231  0.9280726
##   17  0.9632173  0.9450995
##   19  0.9632173  0.9450995
##   21  0.9578755  0.9370980
##   23  0.9578755  0.9370980
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 19.
```

The model performs best at n=19 and lets do prediction for test data set.

```
test <- predict(knn_fit,testing)
```

The model exhibits 96.15 % accuracy.

```
confusionMatrix(test,testing$V1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 17  1  0
##          2  0 19  0
##          3  0  1 14
##
## Overall Statistics
##
##                Accuracy : 0.9615
##                  95% CI : (0.8679, 0.9953)
##     No Information Rate : 0.4038
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9419
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   0.9048   1.0000
## Specificity            0.9714   1.0000   0.9737
## Pos Pred Value         0.9444   1.0000   0.9333
## Neg Pred Value         1.0000   0.9394   1.0000
## Prevalence             0.3269   0.4038   0.2692
## Detection Rate         0.3269   0.3654   0.2692
## Detection Prevalence   0.3462   0.3654   0.2885
## Balanced Accuracy      0.9857   0.9524   0.9868
```