KNNSVM_classifications

Jeffrey Hunt, Anmoldeep Sandhu

2022-10-06

Part 1

Classification in machine learning is a process of categorizing a given set of data into classes. For example trying to determine if an email is spam or not, or determining if a person makes more or less than 50,000 a year. Prediction is when we make predictions on an unknown piece of data. For example what kind of sickness does a person have based on certain health factors. The strengths of classifications is that they are great to get quick results. So for the KNN a great strength is that it is simple to understand, fast, and efficient. The weaknesses on classification however is that it has a difficult time with very small data sets and is not able to predict the correct values. A weakness for the KNN algorithm is the same if we choose to small amount a number for K it won't predict right, if we choose to big a number it will classify everything the same. Prediction in classifications are a great way to predict data. A drawback however is that we need lots of good data to make the correct predictions and this is can be hard to come by.

KNN Classifier

Problem Statement

Study the Adult Census Data and build a machine learning model to predict whether a person makes above or below $50,000 based on the input data received.

install packages

```
library(e1071)
library(caTools)
library(class)
library(caret)
```

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

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

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caTools)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(ggplot2)
```

Load Data

```
data <- read.csv("adultc.csv")
data.df <- data.frame(data)

colnames(data.df) <- c('age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marriage', 'occupat

head(data.df)
```

```
##   age        workclass fnlwgt  education education-num             marriage
## 1  50 Self-emp-not-inc  83311  Bachelors            13   Married-civ-spouse
## 2  38          Private 215646    HS-grad             9             Divorced
## 3  53          Private 234721       11th             7   Married-civ-spouse
## 4  28          Private 338409  Bachelors            13   Married-civ-spouse
## 5  37          Private 284582    Masters            14   Married-civ-spouse
## 6  49          Private 160187        9th             5 Married-spouse-absent
##          occupation  relationship  race    sex capital-gain capital-loss
## 1   Exec-managerial       Husband White   Male            0            0
## 2 Handlers-cleaners Not-in-family White   Male            0            0
## 3 Handlers-cleaners       Husband Black   Male            0            0
## 4    Prof-specialty          Wife Black Female            0            0
## 5   Exec-managerial          Wife White Female            0            0
## 6     Other-service Not-in-family Black Female            0            0
##   hours-per-week       country fifty
## 1             13 United-States <=50K
## 2             40 United-States <=50K
## 3             40 United-States <=50K
## 4             40          Cuba <=50K
## 5             40 United-States <=50K
## 6             16       Jamaica <=50K
```

Data Preprocessing

```
# Change all of word values into integers so we are able to perform KNN on them
data.df$workclass <- as.numeric(factor(data.df$workclass))
data.df$education <- as.numeric(factor(data.df$education))
data.df$marriage <- as.numeric(factor(data.df$marriage))
data.df$occupation <- as.numeric(factor(data.df$occupation))
data.df$relationship <- as.numeric(factor(data.df$relationship))
data.df$race <- as.numeric(factor(data.df$race))
data.df$sex <- as.numeric(factor(data.df$sex))
data.df$country <- as.numeric(factor(data.df$country))
data.df$fifty <- as.numeric(factor(data.df$fifty))

#subset the data
# We removed capital gain and capital loss because there is a lot of 0's that may skew the data
data.df.subset <- data.df[c("age", "workclass", "education", "education-num", "marriage", "occupation",

tail(data.df.subset)
```

```
##       age workclass education education-num marriage occupation relationship
## 32555  22         5        16            10        5         12            2
## 32556  27         5         8            12        3         14            6
## 32557  40         5        12             9        3          8            1
## 32558  58         5        12             9        7          2            5
## 32559  22         5        12             9        5          2            4
## 32560  52         6        12             9        3          5            6
##       race sex hours-per-week country fifty
## 32555    5   2             40      40     1
## 32556    5   1             38      40     1
## 32557    5   2             40      40     2
## 32558    5   1             40      40     1
## 32559    5   2             20      40     1
## 32560    5   1             40      40     2
```

Data Normalization

```
nor <- function(x) {
  (x-min(x))/ (max(x)-min(x))
}

fifty_norm <- as.data.frame(lapply(data.df.subset[1:11], nor))

head(fifty_norm)
```

```
##         age workclass  education education.num  marriage occupation
## 1 0.4520548      0.75 0.60000000     0.8000000 0.3333333  0.2857143
## 2 0.2876712      0.50 0.73333333     0.5333333 0.0000000  0.4285714
## 3 0.4931507      0.50 0.06666667     0.4000000 0.3333333  0.4285714
## 4 0.1506849      0.50 0.60000000     0.8000000 0.3333333  0.7142857
## 5 0.2739726      0.50 0.80000000     0.8666667 0.3333333  0.2857143
```

```
## 6 0.4383562      0.50 0.40000000     0.2666667 0.5000000  0.5714286
##   relationship race sex hours.per.week   country
## 1          0.0  1.0   1      0.1224490 0.9512195
## 2          0.2  1.0   1      0.3979592 0.9512195
## 3          0.0  0.5   1      0.3979592 0.9512195
## 4          1.0  0.5   0      0.3979592 0.1219512
## 5          1.0  1.0   0      0.3979592 0.9512195
## 6          0.2  0.5   0      0.1530612 0.5609756
```

Split into training and testing sets

```
set.seed(123)

dat.d <- sample(1:nrow(fifty_norm), size = nrow(fifty_norm)*0.7, replace = FALSE)

train.fifty <- data.df.subset[dat.d,]
test.fifty <- data.df.subset[-dat.d,]


#Build a separate data frame for the 'fifty' column which is our target
train.fifty_labels <- data.df.subset[dat.d,12]
test.fifty_labels <-  data.df.subset[-dat.d,12]
```

Build Model

```
# use KNN Algorithm
NROW(train.fifty_labels)
```

```
## [1] 22792
```

```
kVal <- sqrt(22792) # A common practice for K value is the sqrt of the total values

model <- knn(train=train.fifty, test=test.fifty, cl=train.fifty_labels, k=kVal, use.all = FALSE)
```

Compute Accuracy, Sensivity, and Specificity

```
confusionMatrix(table(model,test.fifty_labels))
```

```
## Confusion Matrix and Statistics
##
##      test.fifty_labels
## model    1    2
##     1 6901 1328
##     2  509 1030
##
##                Accuracy : 0.8119
##                  95% CI : (0.804, 0.8196)
```

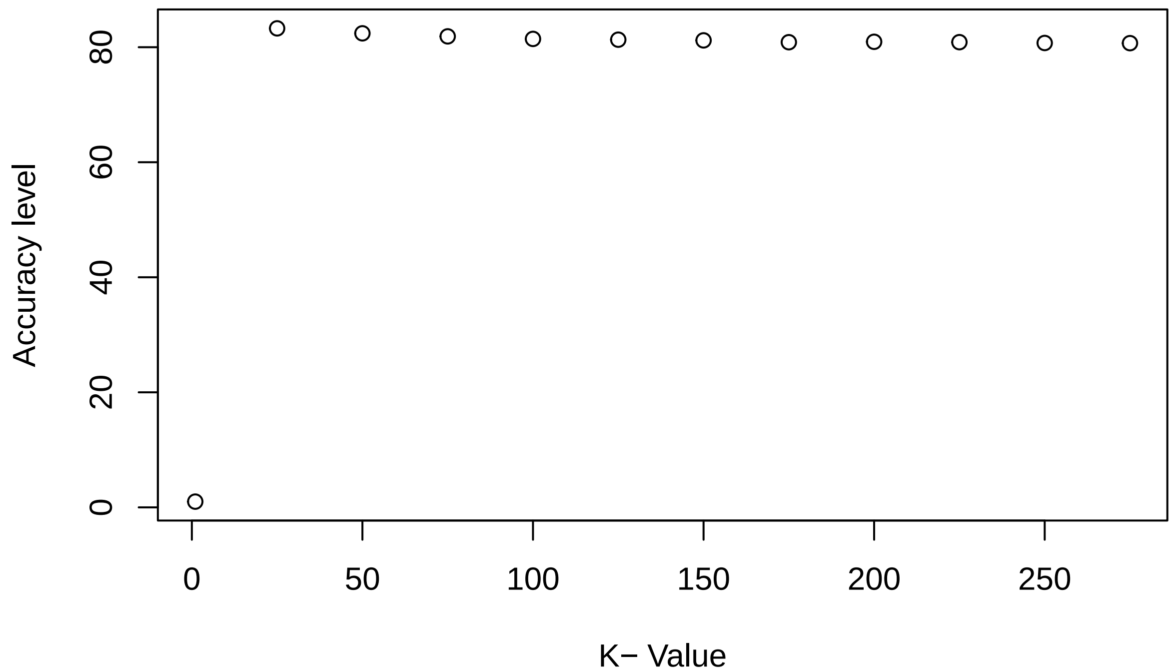
```
##      No Information Rate : 0.7586
##      P-Value [Acc > NIR] : < 2.2e-16
##
##                    Kappa : 0.4176
##
##   Mcnemar's Test P-Value : < 2.2e-16
##
##              Sensitivity : 0.9313
##              Specificity : 0.4368
##           Pos Pred Value : 0.8386
##           Neg Pred Value : 0.6693
##               Prevalence : 0.7586
##           Detection Rate : 0.7065
##     Detection Prevalence : 0.8424
##        Balanced Accuracy : 0.6841
##
##         'Positive' Class : 1
##
```

Compute the Accuracy for multiple K values

```r
k <- c(25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275)

k.optm <- 1

for (i in k) {
  model <- knn(train=train.fifty, test=test.fifty, cl=train.fifty_labels, k=i, use.all = FALSE)
  k.optm[i] <- 100 * sum(test.fifty_labels == model)/NROW(test.fifty_labels)

}
```

Plot the accuracy

```r
#Accuracy plot
plot(k.optm, xlab="K- Value",ylab="Accuracy level")
```



SVM Classifier

Split the Dataset into testing and training

```
set.seed(123)
split <- sample.split(data.df.subset$fifty, SplitRatio = 0.80)

training_set <- subset(data.df.subset, split ==TRUE)
test_set <- subset(data.df.subset, split == FALSE)


# Feature Scaling
training_set[-12] = scale(training_set[-12])
test_set[-12] = scale(test_set[-12])

head(test_set[-12])
```

```
##            age  workclass  education education-num   marriage  occupation
## 4  -0.7717286 0.06818962 -0.3210198     1.1202847 -0.4308374  0.79387363
## 5  -0.1199883 0.06818962  0.4472569     1.5035214 -0.4308374 -0.61816399
## 7   0.9662454 1.43449982  0.1911647    -0.4126621 -0.4308374 -0.61816399
## 13 -0.4820662 0.06818962 -0.8332043     0.7370480  0.9026290  1.26455283
## 17 -0.4820662 0.06818962  0.1911647    -0.4126621  0.9026290  0.08785482
```

```
## 24  1.4731544 0.06818962  0.1911647    -0.4126621 -1.7643039  1.49989244
##    relationship       race        sex hours-per-week    country
## 4     2.2257461 -1.9883231 -1.4076152    -0.01126936 -3.9759381
## 5     2.2257461  0.3920354 -1.4076152    -0.01126936  0.2953942
## 7    -0.9024230  0.3920354  0.7103123     0.39373090  0.2953942
## 13   -0.2767892 -1.9883231  0.7103123     0.79873116  0.2953942
## 17    1.6001123  0.3920354  0.7103123    -0.01126936  0.2953942
## 24    1.6001123  0.3920354 -1.4076152    -0.01126936  0.2953942
```

Buld the Model

```r
classifier <- svm(formula = fifty~.,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')

#names(classifier)
```

Predict the Test Set

```r
y_pred <- predict(classifier, newdata = test_set[-12])
```

Make Confusion Matrix (Compute accuracy, sensitivity, and specificity)

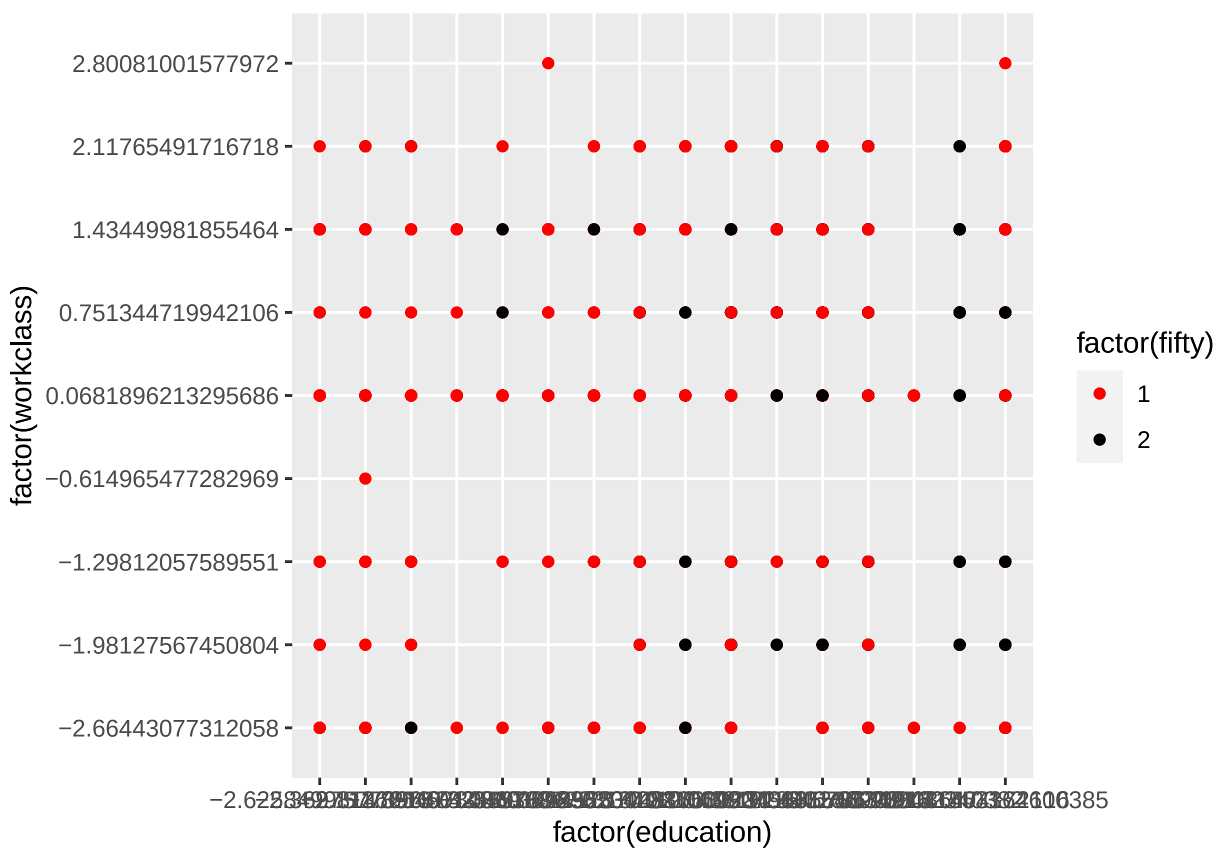
```r
confusionMatrix(table(test_set[, 12], y_pred))
```

```
## Confusion Matrix and Statistics
##
##    y_pred
##        1    2
##   1 4645  299
##   2  947  621
##
##                Accuracy : 0.8087
##                  95% CI : (0.7989, 0.8182)
##     No Information Rate : 0.8587
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3907
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8307
##             Specificity : 0.6750
##          Pos Pred Value : 0.9395
##          Neg Pred Value : 0.3960
##              Prevalence : 0.8587
##          Detection Rate : 0.7133
```

```
##    Detection Prevalence : 0.7592
##       Balanced Accuracy : 0.7528
##
##        'Positive' Class : 1
##
```

Visualize the SVM model

```
#build scatter plot of training dataset
scatter_plot <- ggplot(data = test_set, aes(x = factor(education), y = factor(workclass), color = facto
    geom_point() +
    scale_color_manual(values = c("red", "black"))

#add plot layer marking out the support vectors
scatter_plot
```



Use of the ROC curve and Meaning of Area under ROC curve

The Receiver Operating Characteristic (ROC) curve is a visual representation of how well your classification model works. The ROC curve is calculated by plotting the rate of true Positives vs the rate of False Positives. True Positives are all the values that were predicted right, False Positives are all the values that were wrong. When we plot the ROC curve we need to calcuate the True Positive rate and False Positive Rate for every threshold. For each one is we plot the FPR in the x-axis and the TPR in the Y-axis.

The area covered below the line is the “Area Under the Curve (AUC)”. This is used to evaluate the performance of a classification model. The higher that the AUC is the better model is at distinguishing between classes. Therefore in the ideal world we want to see our line cover most of the upper left of the graph.

Comparison

With comparing the KNN model with the SVM, we found that for the KNN model we got the Balanced accuracy of 68% and for KNN model we got the accuracy of 81% and for the SVM model we got the accuracy of 80%.So, we came to conclusion that both the KNN classifier and SVM classifier is same accurate classifier because both have same accuracy percentage, there is not much differen in between them.The main difference in KNN and SVM classification testing data points is that the SVM utilizes a mathematical function to create the hyperplanes and segregate categorized data points, KNN considers the proximity of its closest k neighbors.