Applying Support Vector Machine (SVM) Algorithm on US Census Data for customer segmentation

Code

Hide

```
library(e1071)
# Importing the dataset
dataset=read.csv('Cencus Income Data.csv')
dataset=data.frame(dataset)
dim(dataset)
```

```
[1] 16281    15
```

Hide

```
str(dataset)
```

```
'data.frame':   16281 obs. of  15 variables:
 $ age           : int  25 38 28 44 18 34 29 63 24 55 ...
 $ workclass     : Factor w/ 9 levels " ?"," Federal-gov",..: 5 5 3 5 1 5 1 7 5 5 ...
 $ fnlwgt        : int  226802 89814 336951 160323 103497 198693 227026 104626 369667 104996 ...
 $ education     : Factor w/ 16 levels " 10th"," 11th",..: 2 12 8 16 16 1 12 15 16 6 ...
 $ education.num : int  7 9 12 10 10 6 9 15 10 4 ...
 $ marital.status: Factor w/ 7 levels " Divorced"," Married-AF-spouse",..: 5 3 3 3 5 5 5 3 5 3
...
 $ occupation    : Factor w/ 15 levels " ?"," Adm-clerical",..: 8 6 12 8 1 9 1 11 9 4 ...
 $ relationship  : Factor w/ 6 levels " Husband"," Not-in-family",..: 4 1 1 1 4 2 5 1 5 1 ...
 $ race          : Factor w/ 5 levels " Amer-Indian-Eskimo",..: 3 5 5 3 5 5 3 5 5 5 ...
 $ sex           : Factor w/ 2 levels " Female"," Male": 2 2 2 2 1 2 2 2 1 2 ...
 $ capital.gain  : int  0 0 0 7688 0 0 0 3103 0 0 ...
 $ capital.loss  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week: int  40 50 40 40 30 30 40 32 40 10 ...
 $ native.country: Factor w/ 41 levels " ?"," Cambodia",..: 39 39 39 39 39 39 39 39 39 39 ...
 $ Income        : Factor w/ 2 levels " <=50K"," >50K": 1 1 2 2 1 1 1 2 1 1 ...
```

Hide

```
#Checking for missing data
d3=dataset1
for(i in 1:ncol(d3))
   {
    print(colnames(d3[i]))
    print(sum(is.na(d3[i])))
   }
```

```
[1] "age"
[1] 0
[1] "workclass"
[1] 0
[1] "fnlwgt"
[1] 0
[1] "education.num"
[1] 0
[1] "marital.status"
[1] 0
[1] "occupation"
[1] 0
[1] "relationship"
[1] 0
[1] "race"
[1] 0
[1] "sex"
[1] 0
[1] "capital.gain"
[1] 0
[1] "capital.loss"
[1] 0
[1] "hours.per.week"
[1] 0
[1] "native.country"
[1] 0
[1] "Income"
[1] 0
```

Hide

```
# Removing Missing Data in the form of "?"
dataset = dataset[dataset$workclass!= " ?",]
dim(dataset)
```

```
[1] 15318    15
```

Hide

```
dataset = dataset[dataset$occupation != " ?",]
dim(dataset)
```

```
[1] 15315    15
```

Hide

```
dataset = dataset[dataset$native.country != " ?",]
dim(dataset)
```

```
[1] 15060    15
```

Hide

```
# Dropping the Education in favor of substitute data
dataset=dataset[-4]
# Income variable set as factor for classification
dataset$Income = ifelse(dataset$Income == " >50K",1,0)
str(dataset$Income)
```

```
 num [1:15060] 0 0 1 1 0 1 0 0 1 0 ...
```

Hide

```
dataset$Income = as.factor(dataset$Income)
str(dataset$Income)
```

```
 Factor w/ 2 levels "0","1": 1 1 2 2 1 2 1 1 2 1 ...
```

Hide

```
# Defining the categorical and Numeric Input Data
dataset$age = as.numeric(dataset$age)
dataset$workclass = as.factor(dataset$workclass)
dataset$fnlwgt = as.numeric(dataset$fnlwgt)
dataset$education.num = as.factor(dataset$education.num)
dataset$marital.status = as.factor(dataset$marital.status)
dataset$occupation = as.factor(dataset$occupation)
dataset$relationship = as.factor(dataset$relationship)
dataset$race = as.factor(dataset$race)
dataset$sex = as.factor(dataset$sex)
dataset$capital.gain = as.numeric(dataset$capital.gain)
dataset$capital.loss = as.numeric(dataset$capital.loss)
dataset$hours.per.week = as.numeric(dataset$hours.per.week)
dataset$native.country = as.factor(dataset$native.country)
```

Hide

```
dataset1 = dataset
# Exploring the data set components
str(dataset1)
```

```
'data.frame':	15060 obs. of  14 variables:
 $ age           : num  25 38 28 44 34 63 24 55 65 36 ...
 $ workclass     : Factor w/ 9 levels " ?"," Federal-gov",..: 5 5 3 5 5 7 5 5 5 2 ...
 $ fnlwgt        : num  226802 89814 336951 160323 198693 ...
 $ education.num : Factor w/ 16 levels "1","2","3","4",..: 7 9 12 10 6 15 10 4 9 13 ...
 $ marital.status: Factor w/ 7 levels " Divorced"," Married-AF-spouse",..: 5 3 3 3 5 3 5 3 3 3
...
 $ occupation    : Factor w/ 15 levels " ?"," Adm-clerical",..: 8 6 12 8 9 11 9 4 8 2 ...
 $ relationship  : Factor w/ 6 levels " Husband"," Not-in-family",..: 4 1 1 1 2 1 5 1 1 1 ...
 $ race          : Factor w/ 5 levels " Amer-Indian-Eskimo",..: 3 5 5 3 5 5 5 5 5 5 ...
 $ sex           : Factor w/ 2 levels " Female"," Male": 2 2 2 2 2 2 1 2 2 2 ...
 $ capital.gain  : num  0 0 0 7688 0 ...
 $ capital.loss  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week: num  40 50 40 40 30 32 40 10 40 40 ...
 $ native.country: Factor w/ 41 levels " ?"," Cambodia",..: 39 39 39 39 39 39 39 39 39 39 ...
 $ Income        : Factor w/ 2 levels "0","1": 1 1 2 2 1 2 1 1 2 1 ...
```

Hide

```
dim(dataset1)
```

```
[1] 15060    14
```

Hide

```
# Splitting the dataset into the Training set and Test set
library(caTools)
set.seed(123)
split = sample.split(dataset1$Income, SplitRatio = 0.8)
training_set = subset(dataset1, split == TRUE)
test_set = subset(dataset1, split == FALSE)
# Feature Scaling (Normalization and dropping the predicted variable)
training_set[,-c(2,4,5,6,7,8,9,13,14)] = scale(training_set[-c(2,4,5,6,7,8,9,13,14)])
test_set[-c(2,4,5,6,7,8,9,13,14)] = scale(test_set[-c(2,4,5,6,7,8,9,13,14)])
# Applying Kernel SVM Model on the Training set
library(e1071)
classifier = svm(formula = Income ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'radial')
summary(classifier)
```

```
Call:
svm(formula = Income ~ ., data = training_set, type = "C-classification", 
    kernel = "radial")


Parameters:
   SVM-Type:  C-classification 
 SVM-Kernel:  radial 
       cost:  1 
      gamma:  0.01010101 

Number of Support Vectors:  4701

 ( 2360 2341 )


Number of Classes:  2 

Levels: 
 0 1
```

Hide

```
# Predicting the Test set results
predict_val = predict(classifier, newdata = test_set[-14])
# Confusion Matrix
cm = table(test_set[, 14], predict_val)
print(cm)
```

```
   predict_val
       0    1
  0 2159  113
  1  315  425
```

Hide

```
# Evaluating Model Accuracy on test data set using Confusion Matrix
Model_Accuracy=(cm[1,1] + cm[2,2])/ (cm[1,1] + cm[1,2] + cm[2,1] + cm[2,2])
print("Model Accuracy is")
```

```
[1] "Model Accuracy is"
```

Hide

```
print(Model_Accuracy)
```

```
[1] 0.8579017
```