1. Objective: Build SVM classification model to predict if the customer is likely to accept the personal loan offered by the bank.

2. Another library kernlab for kernel SVM

3. Grid search

|  |  |
| --- | --- |
| **Dataset Details Attribute** | **Description** |
| ID | Customer ID |
| Age | Customer's age in completed years |
| Experience | #years of professional experience |
| Income | Annual income of the customer ($000) |
| ZIPCode | Home Address ZIP code. |
| Family | Family size of the customer |
| CCAvg | Avg. spending on credit cards per month ($000) |
| Education | Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professional |
| Mortgage | Value of house mortgage if any. ($000) |
| Personal Loan | Did this customer accept the personal loan offered in the last campaign? **(Target attribute)** |
| Securities Account | Does the customer have a securities account with the bank? |
| CD Account | Does the customer have a certificate of deposit (CD) account with the bank? |
| Online | Does the customer use internet banking facilities? |
| CreditCard | Does the customer use a credit card issued by UniversalBank? |

**############### Classification using e1071 #############**

**1. Load Data into R**

**2. Data preparation**

a. Remove the columns ID & ZIP

b. Split the data into train and test datasets

c. Variable “Education” has 3 categories, so create dummy variables

d. Standardization of data

**3. Model Building**

#install.packages("e1071")

library(e1071)

**# Store the independent variables and target variable separately**

**#Build the model on train data**

model = svm(x.train,y.train, method = "C-classification", kernel = "linear", cost = 10, gamma = 0.1)

summary(model)

#The "cost" parameter balances the trade-off between having a large margin and

#classifying all points correctly. It is important to choose it well to have

#good generalization.

**4. Predict on train & test data**

**5. Build the confusion matrix**

**6. Compute the error metrics**

Note: Build SVM model by changing the kernel function to “radial” and check if the accuracies are better.

**####### Classification using KSVM #############**

#install.packages("kernlab")

library(kernlab)

#Build model using ksvm with "rbfdot" kernel

kern\_rbf <- ksvm(as.matrix(train\_bankdata[,-7]),train\_bankdata[,7],

type='C-svc',kernel="rbfdot",kpar=list(sigma=(0:1)),

C=10, cross=5)

kern\_rbf

kern\_rbf <- ksvm(as.matrix(train\_bankdata[,-7]),train\_bankdata[,7],

type='C-svc',kernel="rbfdot",kpar="automatic",

C=10, cross=5)

**#Build model using ksvm with "vanilladot" kernel**

kern\_vanilla <- ksvm(as.matrix(train\_bankdata[,-7]),train\_bankdata[,7],

type='C-svc',kernel="vanilladot", C = 10)

kern\_vanilla

**7. Predict on train & test data**

**8. Build the confusion matrix**

**9. Compute the error metrics**

**#Perform a grid search**

tuneResult <- tune(svm, train.x = x, train.y = y, ranges = list(gamma = 10^(-6:-1), cost =

2^(2:3)))

print(tuneResult)

**10. Predict on train & test data**

**11. Build the confusion matrix**

**12. Compute the error metrics**