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**
3. Remove the columns ID & ZIP
4. Variable “Education” has 3 categories , so create dummy variables
5. Standardization of data – use range method
6. Split the data into train and test datasets
7. **Model Building**

#install.packages("e1071")

library(e1071)

**# Store the independent variables and target variable separately (for easy use)**

* x = subset (train\_bankdata, select = -Personal.Loan) #remove response variable
* y = as.factor (train\_bankdata$Personal.Loan)

**#Build the model on train data**

model = svm(x,y, 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.

1. **Predict on train & test data**
2. **Build the confusion matrix**
3. **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)

names(train\_bankdata)

#**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

**#Predict model "kern\_rbf" (on test data)**

kpred\_rbf<- predict(kern\_rbf,test\_bankdata[-7])

confMatrix <- table(test\_bankdata$Personal.Loan, kpred\_rbf)

acc\_rbf = sum(diag(confMatrix))/sum(confMatrix);acc\_rbf

rec\_rbf = (confMatrix[2,2]/(confMatrix[2,2]+confMatrix[2,1]));rec\_rbf

**#Predict model "kern\_vanilla" (on test data)**

kpred\_vanilla<- predict(kern\_vanilla,test\_bankdata[-7])

confMatrix <- table(test\_bankdata$Personal.Loan, kpred\_vanilla)

acc\_vanilla = sum(diag(confMatrix))/sum(confMatrix);acc\_vanilla

rec\_vanilla = (confMatrix[2,2]/(confMatrix[2,2]+confMatrix[2,1]));rec\_vanilla

**#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)

**#Predict model and calculate errors**

tunedModel <- tuneResult$best.model tunedModelY <- predict(tunedModel, as.matrix(x)) Conf <- table(y, tunedModelY)

# you can now compute the metrics.

**Assignment: SVM model building for regression**

* Given the data BostonHousing.csv, we need to predict the variable ‘medv’, which is the median value of owner-occupied homes in USD in 1000’s.
* Perform required preprocessing steps.
* Split the data into test and train.
* Run a regression using svm. Read the help function to understand how to perform a regression.
* Perform tuning to obtain the best metrics on test data