Learning outcomes:

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 KSVM for kernel SVMs.
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?

1. **Load Data into R:**
2. **Data preparation**
   a. **to remove the columns ID & ZIP**
   b. **Convert categorical attribute “Education” to numeric**
   c. **Standardization of Data**
   d. **Split the data into train and test datasets**
3. **Model Building**

 # Classification using SVM

```
install.packages("e1071")
library(e1071)
```

#Building the model on train data

```
x = subset(train_bankdata, select = -Personal.Loan) #remove response
variable y  =  as.factor(train_bankdata$Personal.Loan)
model  =  svm(x,y, method = "C-classification", kernel = "linear", cost = 10, gamma = 0.1)
summary(model)
```

```
Call:
svm.default(x = x, y = y, kernel = "linear",
    gamma = 0.1, cost = 10, method = "C-classification")

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.1

Number of Support Vectors:  332

 ( 169 163 )

Number of Classes:  2

Levels:
 0 1
```

4. Applying the model on train data & test data and predict

5. Build the confusion matrix

6. Compute the error metrics

II. Run the SVM model with kernel = "radial" and check if there is any improvement in the results

Note:

The svm() function in e1071 provides a rigid interface to libsvm along with visualization and parameter tuning methods. Package kernlab features a variety of kernel-based methods and includes a SVM method based on the optimizers used in libsvm and bsvm (Hsu and Lin
2002c). It aims to provide a flexible and extensible SVM implementation. It also takes advantage of the inherent modularity of kernel-based methods, aiming to allow the user to switch between kernels on an existing algorithm and even create and use own kernel functions for the various kernel methods provided in the package.

```
###############KSVM##############
library(kernlab)
names(train_bankdata)
```

```
kernmodel <- ksvm(as.matrix(train_bankdata[,-
7]),train_bankdata[,7],type='C-
svc',kernel="rbfdot",kpar=list(sigma=1),C=10)
kpred<- predict(kernmodel,test_bankdata[-7])
kRMSE<- rmse(test_bankdata[,7], kpred)
```

In order to improve the performance of the support vector machine model we will need to select the best parameters for the model.
the default epsilon = 0.1 and c = 10. We can change it to avoid overfitting. # The process of choosing these parameters is called hyper parameter optimization, or model selection.
The standard way of doing it is by doing a grid search. It means we will train a lot of models

for the different couples of ϵϵ and cost, and choose the best one. #Grid Search/Hyper-parameter tuning

```
# 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)

tunedModel <- tuneResult$best.model
tunedModelY <- predict(tunedModel, as.matrix(x))
Conf <- table(y, tunedModelY)
# you can now compute the metrics.
```

References:

http://eeecon.uibk.ac.at/~zeileis/papers/Ensemble-2005.pdf
https://escience.rpi.edu/data/DA/svmbasic_notes.pdf