Learning outcomes:

After solving these exercises, you should be able to understand the following:

1. Read and preprocess the data.
2. Build the classification and regression models using SVM algorithm.
3. Hyper parameter tuning using grid search
4. Evaluate the model performance

Dataset schema:

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?
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?

Steps:

- Build a SVM classification model to predict whether the customer is going to default on loan or not.

1. Load the following libraries and read "UniversalBank.csv" file into R data frame.

```
# Load required libraries
library(vegan)
library(dummies)
library(e1071)

attr = c('id', 'age', 'exp', 'inc', 'zip', 'family',
         'ccavg', 'edu', 'mortgage', 'loan',
         'securities', 'cd', 'online', 'cc')

# Read the data using csv file
data = read.csv(file = "UniversalBank.csv",
                header = TRUE, col.names = attr)
```

2. Understand the *structure* and summary of the data using *str* and summary R commands

3. Remove the following unused attributes from the data frame.

```
drop_Attr = c("id", "zip", "exp")
attr = setdiff(attr, drop_Attr)
data = data[, attr]
```

4. Using domain knowledge separate categorical and numeric attributes. Convert them into appropriate type.

- To numeric using as.numeric()
- To categorical using as.factor()

Hint: Try using “sapply” function.

```
cat_Data <- data.frame(sapply(data[,cat_Attr], as.factor))
num_Data <- data.frame(sapply(data[,num_Attr], as.numeric))
```

5. R SVM function only accepts numeric attributes, so convert all categorical attributes to numeric.

Note: Target attribute type should be as it is.

Case 1: If the categorical attribute is having 0 and 1 as it levels, then directly convert it in to numeric using as.numeric()

Case 2: Otherwise, Convert all categorical and ordinal attributes to numeric using dummy function.

- E.g. convert "Education" categorical attribute to numeric using dummy function in dummies R library
- Drop actual Education attribute from original data set
- Add created dummy Education variables to original data set

6. Standardize the independent numeric variables using decostand function in vegan R library
Note: To standardize the data using 'Range' method

7. Recombine all the attributes using cbind.

8. Separate the data into train, test and eval.

```
# Divide the data into test and train
set.seed(123)

train_RowIDs = sample(1:nrow(cla_Data), nrow(cla_Data)*0.6)
train_Data = cla_Data[train_RowIDs,]
test_Data = cla_Data[-train_RowIDs,]
```

9. Check the distribution of train, test and eval data w.r.t target attribute.

```
table(cla_Data$loan)
table(train_Data$loan)
table(test_Data$loan)
```

10. Build the SVM models:

```
# Build best SVM model
model = svm(x = train_Data[,ind_Attr],
            y = train_Data$loan,
            type = "C-classification",
            kernel = "linear", cost = 10, gamma = 0.1)
```

11. Look at the model summary:

```
summary(model)

plot(cmdscale(dist(train_Data[,ind_Attr])),
     col = as.integer(train_Data$loan),
     pch = c("o","+")[1:nrow(train_Data) %in% model$index + 1])
```

12.Predict the values on train data and build the confusion matrix.

```
# Predict on train data
pred_Train  =  predict(model, train_Data[,ind_Attr])

# Build confusion matrix and find accuracy
cm_Train = table(train_Data$loan, pred_Train)
accu_Train= sum(diag(cm_Train))/sum(cm_Train)
```

13.Predict the values on test data and build the confusion matrix.

```
# Predict on test data
pred_Test = predict(model, test_Data[,ind_Attr])

# Build confusion matrix and find accuracy
cm_Test = table(test_Data$loan, pred_Test)
accu_Test= sum(diag(cm_Test))/sum(cm_Test)
rm(pred_Test, cm_Test)
```

14.Tuning your support vector machine model:

- 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 this is using grid search ,where we train a lot of models for different combinations of epsilon and cost and choose the best one.

```
tuneResult <- tune(svm,train_Data[,ind_Attr],train_Data$loan,
                   ranges = list(gamma = 10^(-6:-1), cost = 2^(2:3)))
print(tuneResult)

tunedModel <- tuneResult$best.model
tunedModelY <- predict(tunedModel, as.matrix(train_Data[,ind_Attr]))
Conf <- table(train_Data$loan, tunedModelY)
```

- Now calculate the error metrics

→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.

1. Perform required preprocessing steps.
2. Split the data into test and train.
3. Run a regression using svm. Read the help function to understand how to perform a regression.

```
model = svm(x = train_Data[,1:13],
            y = train_Data[,14],
            type = "nu-regression",
            kernel = "linear", cost = 1e-7)
```

4. Perform tuning to obtain the best metrics on test data