Linear Discriminant Analysis

# Importing the dataset

dataset = read.csv('file.choose'())

# Splitting the dataset into the Training set and Test set

# install.packages('caTools')

library(caTools)

library(stats)

set.seed(123)

split = sample.split(dataset$left, SplitRatio = 4/5)

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

# Feature Scaling

training\_set[,1:7] = scale(training\_set[,1:7])

test\_set[,1:7] = scale(test\_set[,1:7])

# Applying LDA

library(MASS)

lda = lda(formula = left ~ ., data = training\_set)

training\_set = as.data.frame(predict(lda, training\_set))

training\_set = training\_set[c(4, 1)]

test\_set = as.data.frame(predict(lda, test\_set))

test\_set = test\_set[c(4, 1)]

# Fitting SVM to the Training set

# install.packages('e1071')

library(e1071)

classifier = svm(formula = class ~ .,

data = training\_set,

type = 'C-classification',

kernel = 'linear')

# Predicting the Test set results

y\_pred = predict(classifier, newdata = test\_set[-2])

# Making the Confusion Matrix

cm = table(test\_set[, 2], y\_pred)

> cm

y\_pred

0 1

0 2590 0

1 16 394

# Visualising the Training set results

library(ElemStatLearn)

set = training\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

grid\_set = expand.grid(X1)

colnames(grid\_set) = c('LD1')

y\_grid = predict(classifier, newdata = grid\_set)

plot(set[, -3], main = 'SVM (Training set)',

xlab = 'LD1',

xlim = range(X1))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)

points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'deepskyblue', 'tomato'))

points(set, pch = 21, bg = ifelse(set[, 2] == 1, 'blue3', 'red3'))

# Visualising the Test set results

library(ElemStatLearn)

set = test\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

grid\_set = expand.grid(X1,X2)

colnames(grid\_set) = c('LD1', 'LD1')

y\_grid = predict(classifier, newdata = grid\_set)

plot(set[, -2], main = 'SVM (Test set)',

xlab = 'LD1',ylab = 'LD1',

xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)

points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'deepskyblue', 'tomato'))

points(set, pch = 21, bg = ifelse(set[, 2] == 1, 'blue3', 'red3'))

LOGISTIC REGRESSION

# Data Preprocessing Template

# Importing the dataset

dataset = read.csv('HR\_comma\_sep.csv')

# Encoding the target feature as factor

# Splitting the dataset into the Training set and Test set

install.packages('caTools')

library(caTools)

library(stats)

set.seed(123)

split = sample.split(dataset$left, SplitRatio = 4/5)

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

# Feature Scaling

training\_set[,1:7] = scale(training\_set[,1:7])

test\_set[,1:7] = scale(test\_set[,1:7])

# Fitting Logistic Regression to the Training set

classifier = glm(left ~ .,binomial, training\_set)

> summary(classifier)

Call:

glm(formula = left ~ ., family = binomial, data = training\_set)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.2383 -0.6623 -0.4055 -0.1201 3.0725

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -3.13311 0.17431 -17.974 < 2e-16 \*\*\*

satisfaction\_level -1.02117 0.02718 -37.578 < 2e-16 \*\*\*

last\_evaluation 0.09877 0.02854 3.461 0.000539 \*\*\*

number\_project -0.36859 0.02935 -12.560 < 2e-16 \*\*\*

average\_montly\_hours 0.22306 0.02869 7.775 7.57e-15 \*\*\*

time\_spend\_company 0.40128 0.02538 15.812 < 2e-16 \*\*\*

Work\_accident -0.55631 0.03566 -15.600 < 2e-16 \*\*\*

promotion\_last\_5years -0.19509 0.04010 -4.865 1.15e-06 \*\*\*

Departmenthr 0.25467 0.14964 1.702 0.088776 .

DepartmentIT -0.15068 0.13721 -1.098 0.272129

Departmentmanagement -0.38809 0.17675 -2.196 0.028110 \*

Departmentmarketing 0.12520 0.14760 0.848 0.396304

Departmentproduct\_mng -0.10433 0.14573 -0.716 0.474021

DepartmentRandD -0.39922 0.16040 -2.489 0.012810 \*

Departmentsales -0.00693 0.11568 -0.060 0.952228

Departmentsupport 0.07687 0.12327 0.624 0.532881

Departmenttechnical 0.12263 0.12005 1.021 0.307036

salarylow 1.93374 0.14344 13.481 < 2e-16 \*\*\*

salarymedium 1.38383 0.14418 9.598 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 13172 on 11998 degrees of freedom

Residual deviance: 10272 on 11980 degrees of freedom

AIC: 10310

Number of Fisher Scoring iterations: 5

# Predicting the Test set results

prob\_pred = predict(classifier, type = 'response', newdata = test\_set[-10])

y\_pred = ifelse(prob\_pred > 0.5, 1,0)

# Making the Confusion Matrix

cm = table(test\_set[,10], y\_pred)

> cm

y\_pred

0 1

0 2117 169

1 458 256

# Visualising the Training set results

library(ElemStatLearn)

set = training\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid\_set = expand.grid(X1, X2)

colnames(grid\_set) = c('Age', 'EstimatedSalary')

prob\_set = predict(classifier, type = 'response', newdata = grid\_set)

y\_grid = ifelse(prob\_set > 0.5, 1, 0)

plot(set[, -3],

main = 'Logistic Regression (Training set)',

xlab = 'Age', ylab = 'Estimated Salary',

xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)

points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'springgreen3', 'tomato'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))

# Visualising the Test set results

library(ElemStatLearn)

set = test\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid\_set = expand.grid(X1, X2)

colnames(grid\_set) = c('Age', 'EstimatedSalary')

prob\_set = predict(classifier, type = 'response', newdata = grid\_set)

y\_grid = ifelse(prob\_set > 0.5, 1, 0)

plot(set[, -3],

main = 'Logistic Regression (Test set)',

xlab = 'Age', ylab = 'Estimated Salary',

xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)

points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'springgreen3', 'tomato'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))

XGBOOST USING K-FOLD CROSS VALIDATION

# Importing the dataset

dataset = read.csv('HR\_comma\_sep.csv')

# Encoding the categorical variables as factors

# Encoding the categorical variables as factors

dataset$Department = as.numeric(factor(dataset$Department,

levels = c('sales','accounting','hr','technical','support','management','IT','product\_mng','marketing','RandD'),

labels = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)))

dataset$salary = as.numeric(factor(dataset$salary,

levels = c('low', 'medium','high'),

labels = c(1, 2, 3)))

# Splitting the dataset into the Training set and Test set

# install.packages('caTools')

# Splitting the dataset into the Training set and Test set

install.packages('caTools')

library(caTools)

library(stats)

set.seed(123)

split = sample.split(dataset$left, SplitRatio = 4/5)

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

# Fitting XGBoost to the Training set

install.packages('xgboost')

library(xgboost)

> classifier = xgboost(data = as.matrix(training\_set[-10]), label = training\_set$left, nrounds = 20)

[1] train-rmse:0.363685

[2] train-rmse:0.272141

[3] train-rmse:0.212954

[4] train-rmse:0.176444

[5] train-rmse:0.154598

[6] train-rmse:0.141133

[7] train-rmse:0.133008

[8] train-rmse:0.127959

[9] train-rmse:0.124200

[10] train-rmse:0.120824

[11] train-rmse:0.119458

[12] train-rmse:0.117243

[13] train-rmse:0.115593

[14] train-rmse:0.113856

[15] train-rmse:0.112611

[16] train-rmse:0.110132

[17] train-rmse:0.108937

[18] train-rmse:0.108066

[19] train-rmse:0.107406

[20] train-rmse:0.106758

# Predicting the Test set results

> y\_pred = predict(classifier, newdata = as.matrix(test\_set[-10]))

> y\_pred = (y\_pred >= 0.5)

> cm = table(test\_set[, 10], y\_pred)

> cm

y\_pred

FALSE TRUE

0 2277 9

1. 52 662

> # Applying k-Fold Cross Validation

> #install.packages('caret')

> library(caret)

> folds = createFolds(training\_set$left, k = 10)

> cv = lapply(folds, function(x) {

+ training\_fold = training\_set[-x, ]

+ test\_fold = training\_set[x, ]

+ classifier = xgboost(data = as.matrix(training\_set[-10]), label = training\_set$left, nrounds = 20)

+ y\_pred = predict(classifier, newdata = as.matrix(test\_fold[-10]))

+ y\_pred = (y\_pred >= 0.5)

+ cm = table(test\_fold[, 10], y\_pred)

+ accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])

+ return(accuracy)

+ })

> cv

$Fold01

[1] 0.9841667

$Fold02

[1] 0.9866667

$Fold03

[1] 0.985

$Fold04

[1] 0.9841535

$Fold05

[1] 0.9883333

$Fold06

[1] 0.985

$Fold07

[1] 0.99

$Fold08

[1] 0.9783333

$Fold09

[1] 0.99

$Fold10

[1] 0.9875

> accuracy = mean(as.numeric(cv))

> accuracy

[1] 0.9859153

PCA

> # PCA

>

> # Importing the dataset

> dataset = read.csv('HR\_comma\_sep.csv')

>

>

> # Encoding the categorical variables as factors

dataset$Department = as.numeric(factor(dataset$Department,

levels = c('sales','accounting','hr','technical','support','management','IT','product\_mng','marketing','RandD'),

labels = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)))

dataset$salary = as.numeric(factor(dataset$salary,

levels = c('low', 'medium','high'),

labels = c(1, 2,3)))

# Splitting the dataset into the Training set and Test set

# install.packages('caTools')

library(caTools)

set.seed(123)

split = sample.split(dataset$left, SplitRatio = 0.8)

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

# Feature Scaling

training\_set[,1:7] = scale(training\_set[,1:7])

test\_set[,1:7] = scale(test\_set[,1:7])

# Applying PCA

# install.packages('caret')

library(caret)

# install.packages('e1071')

library(e1071)

pca = preProcess(x = training\_set[-10], method = 'pca', pcaComp = 2)

training\_set = predict(pca, training\_set)

training\_set = training\_set[c(2, 3, 1)]

test\_set = predict(pca, test\_set)

test\_set = test\_set[c(2, 3, 1)]

# Fitting SVM to the Training set

# install.packages('e1071')

library(e1071)

classifier = svm(formula = left ~ .,

data = training\_set,

type = 'C-classification',

kernel = 'linear')

# Predicting the Test set results

y\_pred = predict(classifier, newdata = test\_set[-3])

# Making the Confusion Matrix

cm = table(test\_set[, 3], y\_pred)

cm

y\_pred

0 1

0 2286 0

1 714 0

# Visualising the Training set results

library(ElemStatLearn)

set = training\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid\_set = expand.grid(X1, X2)

colnames(grid\_set) = c('PC1', 'PC2')

y\_grid = predict(classifier, newdata = grid\_set)

plot(set[, -3],

main = 'SVM (Training set)',

xlab = 'PC1', ylab = 'PC2',

xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)

Warning message:

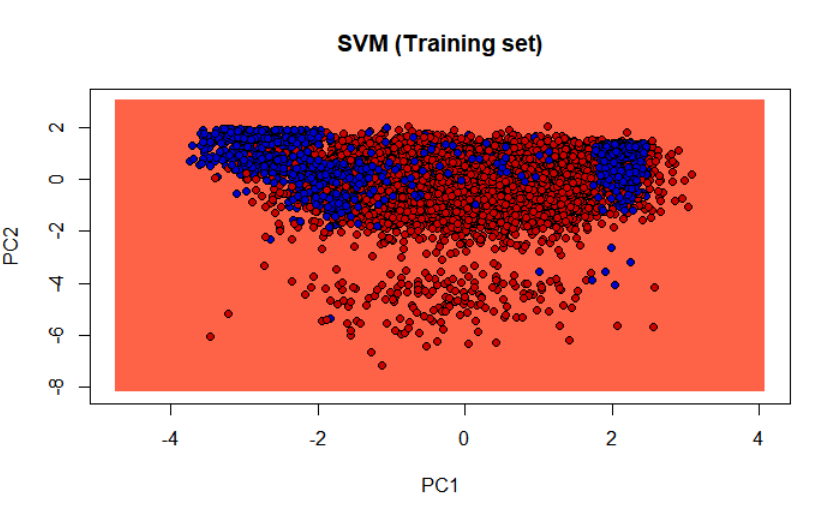
In contour.default(X1, X2, matrix(as.numeric(y\_grid), length(X1), :

all z values are equal

points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'deepskyblue',

'tomato'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'blue3', 'red3'))



> # Visualising the Test set results

> library(ElemStatLearn)

Warning messages:

1: In doTryCatch(return(expr), name, parentenv, handler) :

all z values are equal

2: In doTryCatch(return(expr), name, parentenv, handler) :

all z values are equal

3: In doTryCatch(return(expr), name, parentenv, handler) :

all z values are equal

> set = test\_set

> X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

> X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

> grid\_set = expand.grid(X1, X2)

> colnames(grid\_set) = c('PC1', 'PC2')

> y\_grid = predict(classifier, newdata = grid\_set)

> plot(set[, -3], main = 'SVM (Test set)',

+ xlab = 'PC1', ylab = 'PC2',

+ xlim = range(X1), ylim = range(X2))

> contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)

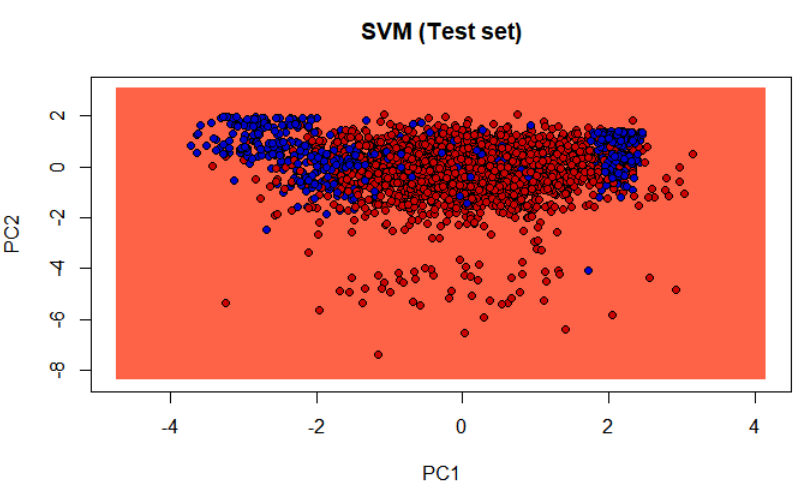
Warning message:

In contour.default(X1, X2, matrix(as.numeric(y\_grid), length(X1), :

all z values are equal

> points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'deepskyblue', 'tomato'))

> points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'blue3', 'red3'))



|  |
| --- |
| #RandomFOREST  library(caTools)  library(stats)  set.seed(123)  dataset = read.csv('file.choose'())  # Encoding the categorical variables as factors  dataset$Department = as.numeric(factor(dataset$Department,  levels = c('sales','accounting','hr','technical','support','management','IT','product\_mng','marketing','RandD'),  labels = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)))  dataset$salary = as.numeric(factor(dataset$salary,  levels = c('low', 'medium','high'),  labels = c(1, 2, 3)))  split = sample.split(dataset$left, SplitRatio = 4/5)  training\_set = subset(dataset, split == TRUE)  test\_set = subset(dataset, split == FALSE)  # Feature Scaling  training\_set[,1:5] = scale(training\_set[,1:5])  test\_set[,1:5] = scale(test\_set[,1:5])  library(e1071)  library(randomForest)  classifier = randomForest(x = training\_set[-10],  y = training\_set$left,  ntree = 100)  y\_pred = predict(classifier, type = 'response',newdata = test\_set[-10], cutoff = 0.5)  pred = ifelse(y\_pred > 0.5, 1,0)  > confusionMatrix(as.factor(test\_set$left), as.factor(pred) )  Confusion Matrix and Statistics  Reference  Prediction 0 1  0 2282 4  1 63 651    Accuracy : 0.9777  95% CI : (0.9717, 0.9827)  No Information Rate : 0.7817  P-Value [Acc > NIR] : < 2.2e-16    Kappa : 0.9366    Mcnemar's Test P-Value : 1.382e-12    Sensitivity : 0.9731  Specificity : 0.9939  Pos Pred Value : 0.9983  Neg Pred Value : 0.9118  Prevalence : 0.7817  Detection Rate : 0.7607  Detection Prevalence : 0.7620  Balanced Accuracy : 0.9835    'Positive' Class : 0 |
|  |
| |  | | --- | | > | |