**HOA 732**

**MID TERM PROJECT**

**Kajal Dhimmar**

**2023-07-12**

**Professor: Dr. Ashok Singh**

**Assignment:**

1. **The datafile All Greens Sales Data.csv shows the following variables: The data (X1, X2, X3, X4, X5, X6) are for each franchise store. Y= annual net sales/$1000 X1 = number sq. ft./1000 X2 = inventory/$1000 X3 = amount spent on advertising/$1000 X4 = size of sales district/1000 families X5 = number of competing stores in district.**
2. Fit an MLR model to Y as a function of the potential predictors in the datafile.
3. Fit an SVM model to Y as a function of the potential predictors in the datafile.
4. Compare the two models.
5. **The datafile CBC.csv shows the following variables: Seq# ID# Gender M R F FirstPurch ChildBks YouthBks CookBks DoltYBks RefBks ArtBks GeogBks ItalCook ItalHAtlas ItalArt Florence.**
6. Fit a logistic regression model for Florence as a function of the potential predictors in the datafile CBC.csv
7. Fit an SVM model for Florence as a function of the potential predictors in the datafile CBC.csv data.
8. Compare the two models.

PRF <- function(CM)  
{  
Precision1 <- CM[2,2]/(CM[2,1]+CM[2,2])  
Recall1 <- CM[2,2]/(CM[1,2]+CM[2,2])  
F1.1 <- 2/((1/Recall1)+(1/Precision1))  
Precision0 <- CM[1,1]/(CM[1,1]+CM[1,2])  
Recall0 <- CM[1,1]/(CM[1,1]+CM[2,1])  
  
F1.0 <- 2/((1/Recall0)+(1/Precision0))  
temp <- c(Precision1, Recall1, F1.1, Precision0, Recall0, F1.0)  
  
#df <- as.data.frame(temp)  
#colnames(df) <- cbind.data.frame("Prec1", "Recall1, "F1.1", "Prec0", "Recall0, "F1.0")  
return(temp)  
}

PRF1 <- function(CM)  
{  
Precision1 <- CM[2,2]/(CM[2,1]+CM[2,2])  
Recall1 <- CM[2,2]/(CM[1,2]+CM[2,2])  
F11 <- 2/((1/Recall1)+(1/Precision1))  
Precision0 <- CM[1,1]/(CM[1,1]+CM[1,2])  
Recall0 <- CM[1,1]/(CM[1,1]+CM[2,1])  
F10 <- 2/((1/Recall0)+(1/Precision0))  
result <- c(Precision1, Recall1, F11, Precision0, Recall0, F10)  
names(result) <- c("Prec.1", "Rec.1", "F1.1", "Prec.0", "Rec.0", "F1.0")  
result  
}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\* Problem : 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**(a) Fit an MLR model to Y as a function of the potential predictors in the datafile.** +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

D <- read.csv("All Greens Sales Data.csv")  
dim(D)

## [1] 27 6

names(D)

## [1] "Y" "x1" "x2" "x3" "x4" "x5"

head(D)

## Y x1 x2 x3 x4 x5  
## 1 231 3.0 294 8.2 8.2 11  
## 2 156 2.2 232 6.9 4.1 12  
## 3 10 0.5 149 3.0 4.3 15  
## 4 519 5.5 600 12.0 16.1 1  
## 5 437 4.4 567 10.6 14.1 5  
## 6 487 4.8 571 11.8 12.7 4

summary(D)

## Y x1 x2 x3   
## Min. : 0.5 Min. :0.500 Min. :102.0 Min. : 2.50   
## 1st Qu.: 98.5 1st Qu.:1.400 1st Qu.:204.0 1st Qu.: 4.80   
## Median :341.0 Median :3.500 Median :382.0 Median : 8.10   
## Mean :286.6 Mean :3.326 Mean :387.5 Mean : 8.10   
## 3rd Qu.:450.5 3rd Qu.:4.750 3rd Qu.:551.0 3rd Qu.:10.95   
## Max. :570.0 Max. :8.600 Max. :788.0 Max. :17.40   
## x4 x5   
## Min. : 1.600 Min. : 0.000   
## 1st Qu.: 4.500 1st Qu.: 4.000   
## Median :11.300 Median : 8.000   
## Mean : 9.693 Mean : 7.741   
## 3rd Qu.:14.050 3rd Qu.:12.000   
## Max. :16.300 Max. :15.000

str(D)

## 'data.frame': 27 obs. of 6 variables:  
## $ Y : num 231 156 10 519 437 487 299 195 20 68 ...  
## $ x1: num 3 2.2 0.5 5.5 4.4 ...  
## $ x2: int 294 232 149 600 567 571 512 347 212 102 ...  
## $ x3: num 8.2 6.9 3 12 10.6 ...  
## $ x4: num 8.2 4.1 4.3 16.1 14.1 ...  
## $ x5: int 11 12 15 1 5 4 10 12 15 8 ...

# Fit the MLR model  
lm1 <- lm(Y ~ x1 + x2 + x3 + x4 + x5, data = D)  
summary(lm1)

##   
## Call:  
## lm(formula = Y ~ x1 + x2 + x3 + x4 + x5, data = D)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -26.338 -9.699 -4.496 4.040 41.139   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -18.85941 30.15023 -0.626 0.538372   
## x1 16.20157 3.54444 4.571 0.000166 \*\*\*  
## x2 0.17464 0.05761 3.032 0.006347 \*\*   
## x3 11.52627 2.53210 4.552 0.000174 \*\*\*  
## x4 13.58031 1.77046 7.671 1.61e-07 \*\*\*  
## x5 -5.31097 1.70543 -3.114 0.005249 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 17.65 on 21 degrees of freedom  
## Multiple R-squared: 0.9932, Adjusted R-squared: 0.9916   
## F-statistic: 611.6 on 5 and 21 DF, p-value: < 2.2e-16

library(car)

## Warning: package 'car' was built under R version 4.2.3

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.2.3

vif(lm1)

## x1 x2 x3 x4 x5   
## 4.240914 10.122480 7.624391 6.912318 5.818768

Here all VIF’s are not < 5,

We’ll drop x2 here

# Fit the MLR model  
lm2 <- lm(Y ~ x1 + x3 + x4 + x5, data = D)  
summary(lm2)

##   
## Call:  
## lm(formula = Y ~ x1 + x3 + x4 + x5, data = D)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -30.422 -12.858 -6.477 16.160 45.255   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -39.460 34.411 -1.147 0.2638   
## x1 20.444 3.815 5.359 2.22e-05 \*\*\*  
## x3 16.966 2.093 8.107 4.73e-08 \*\*\*  
## x4 15.673 1.910 8.206 3.86e-08 \*\*\*  
## x5 -4.043 1.937 -2.088 0.0486 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 20.68 on 22 degrees of freedom  
## Multiple R-squared: 0.9902, Adjusted R-squared: 0.9884   
## F-statistic: 555.4 on 4 and 22 DF, p-value: < 2.2e-16

library(car)  
vif(lm2)

## x1 x3 x4 x5   
## 3.579850 3.795323 5.861520 5.468943

Now, we’ll drop x4 because the VIF’s is >5 for x4

# Fit the MLR model  
lm3 <- lm(Y ~ x1 + x3 + x5, data = D)  
summary(lm3)

##   
## Call:  
## lm(formula = Y ~ x1 + x3 + x5, data = D)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -59.816 -17.514 -5.536 20.708 79.746   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 110.141 57.521 1.915 0.06804 .   
## x1 36.338 6.478 5.610 1.04e-05 \*\*\*  
## x3 18.605 4.106 4.531 0.00015 \*\*\*  
## x5 -12.290 3.263 -3.766 0.00100 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 40.75 on 23 degrees of freedom  
## Multiple R-squared: 0.9602, Adjusted R-squared: 0.955   
## F-statistic: 184.9 on 3 and 23 DF, p-value: 3.088e-16

library(car)  
vif(lm3)

## x1 x3 x5   
## 2.657032 3.760743 3.996868

Now all the VIF’s are < 5 and all the P-values are <0.05.

We can test normality of residuals:

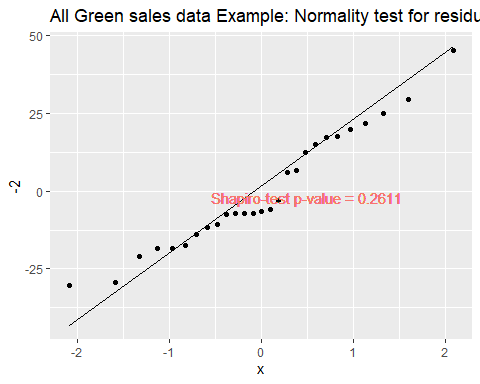
#test Normality of residuals  
  
shapiro.test(lm3$residuals)

##   
## Shapiro-Wilk normality test  
##   
## data: lm3$residuals  
## W = 0.95355, p-value = 0.2611

# type p-value below  
  
  
# create data frame of residuals from the SLR model  
df.resid <- as.data.frame(lm2$residuals)  
colnames(df.resid) <- "Residuals"  
  
#qq plot with normal line (normality test for residuals from lm2)  
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.2.3

ggplot(df.resid)+stat\_qq(aes(sample=Residuals)) +   
 geom\_qq\_line(aes(sample=Residuals))+  
 geom\_text(aes(x=0.5, y=-2, color="red", label="Shapiro-test p-value = 0.2611"))+  
 theme(legend.position="none")+ggtitle("All Green sales data Example: Normality test for residuals")



**++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++(b) Fit an SVM model to Y as a function of the potential predictors in the datafile. ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

# Load required libraries  
library(e1071)

# Fit the SVM model  
svm1 <- svm(Y ~ x1 + x2 + x3 + x4 + x5, kernel = "radial", type = "eps-regression", probability = TRUE, data = D, cost = 10)  
svm1

##   
## Call:  
## svm(formula = Y ~ x1 + x2 + x3 + x4 + x5, data = D, kernel = "radial",   
## type = "eps-regression", probability = TRUE, cost = 10)  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: radial   
## cost: 10   
## gamma: 0.2   
## epsilon: 0.1   
##   
## Sigma: 0.1439572   
##   
##   
## Number of Support Vectors: 12

names(svm1)

## [1] "call" "type" "kernel" "cost"   
## [5] "degree" "gamma" "coef0" "nu"   
## [9] "epsilon" "sparse" "scaled" "x.scale"   
## [13] "y.scale" "nclasses" "levels" "tot.nSV"   
## [17] "nSV" "labels" "SV" "index"   
## [21] "rho" "compprob" "probA" "probB"   
## [25] "sigma" "coefs" "na.action" "fitted"   
## [29] "decision.values" "residuals" "terms"

#SVR performance measures are: (1) RMSE = Root Mean Squarred Error, (2) The R-square value (not my preferred choice)  
#RMSE = average of squarred svm2residuals = sqrt(mean(svm2residuals\*\*2))  
  
  
RMSE <- function(obs,pred)  
{  
temp <-sqrt(sum(obs-pred)\*\*2)  
temp  
}  
obs <- c(1,1)  
pred <- c(1,1)  
RMSE(obs,pred)

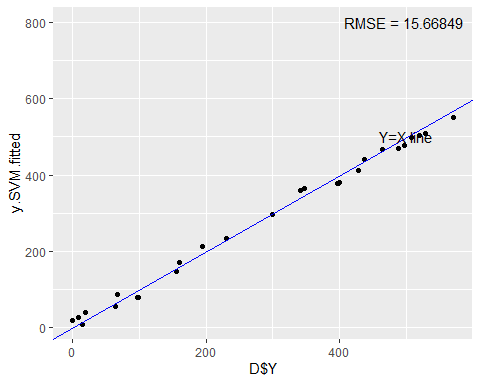
## [1] 0

y.SVM.fitted <- predict(svm1, D)

svm\_predictions <- predict(svm1, D)  
svm\_mse <- mean((D$Y - svm\_predictions)\*\*2)  
svm\_rmse <- sqrt(svm\_mse)  
cat("SVM Model RMSE:", svm\_rmse, "\n")

## SVM Model RMSE: 15.66849

#install.packages("ggplot2")  
library(ggplot2)  
df1 <- cbind.data.frame(D$Y,y.SVM.fitted)  
  
P1 <- ggplot(df1, aes(x=D$Y,y=y.SVM.fitted)) + geom\_point()+  
geom\_abline(intercept = 0, slope = 1,color="blue")+  
annotate("text", x = 500, y = 500, label = "Y=X line")+  
annotate("text", x = 500, y = 800, label = "RMSE = 15.66849 ")  
P1



**+++++++++++++++++++++**

**(c) Compare the two models.**

**+++++++++++++++++++++**

# Step 3: Calculate RMSE for MLR and SVM models  
  
mlr\_predictions <- predict(lm3, D)  
mlr\_mse <- mean((D$Y - mlr\_predictions)\*\*2)  
mlr\_rmse <- sqrt(mlr\_mse)  
  
svm\_predictions <- predict(svm1, D)  
svm\_mse <- mean((D$Y - svm\_predictions)\*\*2)  
svm\_rmse <- sqrt(svm\_mse)  
  
# Step 4: Compare RMSE between MLR and SVM models  
cat("MLR Model RMSE:", mlr\_rmse, "\n")

## MLR Model RMSE: 37.60968

cat("SVM Model RMSE:", svm\_rmse, "\n")

## SVM Model RMSE: 15.66849

**Observation:**

**Here we have, MLR Model RMSE: 37.60968 & SVM Model RMSE: 15.66849.**

**The SVM model has a significantly lower RMSE (15.66849) compared to the MLR model (37.60968). A lower RMSE indicates that the SVM model’s predictions are closer to the actual values compared to the MLR model. Therefore, the SVM model is performing better in terms of minimizing prediction errors.**

**In general, when comparing models for regression tasks, a model with a lower RMSE is preferred as it demonstrates better predictive performance. The SVM model’s lower RMSE suggests that it is a better fit for the data and has a higher accuracy in predicting the target variable compared to the MLR model.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Problem : 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++(a) Fit a logistic regression model for Florence as a function of the potential predictors in the datafile CBC.csv ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

C <- read.csv("CBC.csv")  
dim(C)

## [1] 2000 18

names(C)

## [1] "Seq." "ID." "Gender" "M" "R"   
## [6] "F" "FirstPurch" "ChildBks" "YouthBks" "CookBks"   
## [11] "DoltYBks" "RefBks" "ArtBks" "GeogBks" "ItalCook"   
## [16] "ItalHAtlas" "ItalArt" "Florence"

head(C)

## Seq. ID. Gender M R F FirstPurch ChildBks YouthBks CookBks DoltYBks RefBks  
## 1 1 2 0 138 28 3 40 0 1 0 1 0  
## 2 2 30 1 240 14 1 14 1 0 0 0 0  
## 3 3 59 1 97 6 2 10 0 0 0 0 0  
## 4 4 89 1 348 2 7 38 1 1 1 0 1  
## 5 5 96 0 239 20 2 28 0 0 1 0 0  
## 6 6 120 1 253 10 4 20 1 0 0 0 0  
## ArtBks GeogBks ItalCook ItalHAtlas ItalArt Florence  
## 1 0 1 0 0 0 0  
## 2 0 0 0 0 0 0  
## 3 0 0 0 0 0 0  
## 4 0 1 0 0 0 0  
## 5 0 1 0 0 0 0  
## 6 1 0 0 0 0 1

tail(C)

## Seq. ID. Gender M R F FirstPurch ChildBks YouthBks CookBks DoltYBks  
## 1995 1995 49781 1 192 8 1 8 0 0 0 0  
## 1996 1996 49801 1 164 12 5 32 0 0 1 0  
## 1997 1997 49866 0 294 10 1 10 0 0 0 0  
## 1998 1998 49872 0 261 4 2 10 0 0 0 0  
## 1999 1999 49914 1 41 32 1 32 0 0 1 0  
## 2000 2000 49962 1 308 12 1 12 0 0 0 0  
## RefBks ArtBks GeogBks ItalCook ItalHAtlas ItalArt Florence  
## 1995 0 0 0 0 0 0 0  
## 1996 0 1 2 1 0 1 1  
## 1997 0 0 0 0 0 0 0  
## 1998 0 0 0 0 0 0 0  
## 1999 0 0 0 0 0 0 0  
## 2000 0 0 0 0 0 0 0

summary(C)

## Seq. ID. Gender M   
## Min. : 1.0 Min. : 2 Min. :0.0000 Min. : 15.0   
## 1st Qu.: 500.8 1st Qu.:12699 1st Qu.:0.0000 1st Qu.:126.8   
## Median :1000.5 Median :24201 Median :1.0000 Median :207.0   
## Mean :1000.5 Mean :24753 Mean :0.7085 Mean :206.8   
## 3rd Qu.:1500.2 3rd Qu.:37300 3rd Qu.:1.0000 3rd Qu.:281.2   
## Max. :2000.0 Max. :49962 Max. :1.0000 Max. :477.0   
## R F FirstPurch ChildBks   
## Min. : 2.00 Min. : 1.000 Min. : 2.00 Min. :0.000   
## 1st Qu.: 8.00 1st Qu.: 1.000 1st Qu.:14.00 1st Qu.:0.000   
## Median :12.00 Median : 2.000 Median :22.00 Median :0.000   
## Mean :13.52 Mean : 4.005 Mean :27.42 Mean :0.711   
## 3rd Qu.:16.00 3rd Qu.: 6.000 3rd Qu.:38.00 3rd Qu.:1.000   
## Max. :36.00 Max. :12.000 Max. :99.00 Max. :6.000   
## YouthBks CookBks DoltYBks RefBks   
## Min. :0.000 Min. :0.0000 Min. :0.000 Min. :0.0000   
## 1st Qu.:0.000 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:0.0000   
## Median :0.000 Median :0.0000 Median :0.000 Median :0.0000   
## Mean :0.314 Mean :0.7385 Mean :0.391 Mean :0.2705   
## 3rd Qu.:0.000 3rd Qu.:1.0000 3rd Qu.:1.000 3rd Qu.:0.0000   
## Max. :5.000 Max. :8.0000 Max. :5.000 Max. :4.0000   
## ArtBks GeogBks ItalCook ItalHAtlas   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :0.0000 Median :0.0000 Median :0.0000 Median :0.0000   
## Mean :0.3145 Mean :0.4115 Mean :0.1285 Mean :0.0395   
## 3rd Qu.:0.0000 3rd Qu.:1.0000 3rd Qu.:0.0000 3rd Qu.:0.0000   
## Max. :5.0000 Max. :5.0000 Max. :2.0000 Max. :2.0000   
## ItalArt Florence   
## Min. :0.000 Min. :0.0000   
## 1st Qu.:0.000 1st Qu.:0.0000   
## Median :0.000 Median :0.0000   
## Mean :0.052 Mean :0.1085   
## 3rd Qu.:0.000 3rd Qu.:0.0000   
## Max. :2.000 Max. :1.0000

str(C)

## 'data.frame': 2000 obs. of 18 variables:  
## $ Seq. : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ ID. : int 2 30 59 89 96 120 128 131 139 177 ...  
## $ Gender : int 0 1 1 1 0 1 1 1 1 1 ...  
## $ M : int 138 240 97 348 239 253 118 326 249 294 ...  
## $ R : int 28 14 6 2 20 10 16 14 12 10 ...  
## $ F : int 3 1 2 7 2 4 1 2 3 12 ...  
## $ FirstPurch: int 40 14 10 38 28 20 16 22 20 58 ...  
## $ ChildBks : int 0 1 0 1 0 1 0 0 1 4 ...  
## $ YouthBks : int 1 0 0 1 0 0 1 1 0 1 ...  
## $ CookBks : int 0 0 0 1 1 0 0 0 0 4 ...  
## $ DoltYBks : int 1 0 0 0 0 0 0 0 1 0 ...  
## $ RefBks : int 0 0 0 1 0 0 0 1 0 0 ...  
## $ ArtBks : int 0 0 0 0 0 1 0 0 0 0 ...  
## $ GeogBks : int 1 0 0 1 1 0 0 0 0 2 ...  
## $ ItalCook : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ItalHAtlas: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ItalArt : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Florence : int 0 0 0 0 0 1 0 0 0 0 ...

# Data Overview

# counts of missing values  
n.NA <- colSums(is.na(C))  
n.NA

## Seq. ID. Gender M R F FirstPurch   
## 0 0 0 0 0 0 0   
## ChildBks YouthBks CookBks DoltYBks RefBks ArtBks GeogBks   
## 0 0 0 0 0 0 0   
## ItalCook ItalHAtlas ItalArt Florence   
## 0 0 0 0

library(Amelia)

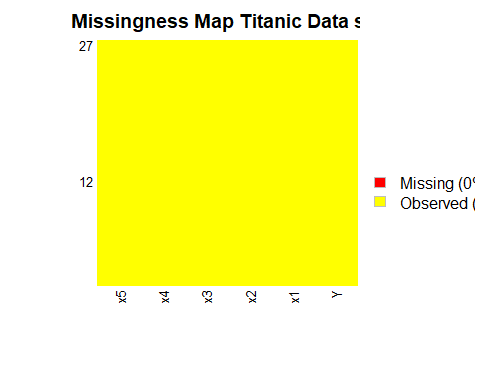
## Warning: package 'Amelia' was built under R version 4.2.3

## Loading required package: Rcpp

## Warning: package 'Rcpp' was built under R version 4.2.3

## ##   
## ## Amelia II: Multiple Imputation  
## ## (Version 1.8.1, built: 2022-11-18)  
## ## Copyright (C) 2005-2023 James Honaker, Gary King and Matthew Blackwell  
## ## Refer to http://gking.harvard.edu/amelia/ for more information  
## ##

missmap(D, col=c("red","yellow"),main = "Missingness Map Titanic Data set")



**We don’t have any missing values in this data set, so we can start our analysis**,

# Split Dataset:.

In predictive analytics, it is always a good idea (mandatory in many situations) to randomly split the data set into a 75% training set and a 25% test set.

We build the model on the training set set, and evaluate its performance on both the training and the test sets.

M <- 0.25\*nrow(C) # # of rows in the test set  
M

## [1] 500

#to be able to replicate the results, set initial seed for random   
#number generator  
set.seed(11731)  
holdout <- sample(1:nrow(C), M, replace=F)  
holdout

## [1] 629 1339 297 1027 1124 405 865 85 504 1352 503 819 1102 1231 1817  
## [16] 441 1495 111 1648 1149 894 605 18 1139 1145 1850 1086 1048 1397 188  
## [31] 1121 1900 719 889 1672 1936 458 1701 223 1903 783 1107 722 1478 181  
## [46] 879 928 1699 546 1004 248 1951 1506 96 1919 1767 1216 1508 33 279  
## [61] 292 220 810 1637 1230 1744 58 1616 3 1320 1999 28 131 333 528  
## [76] 1130 1074 764 1026 1569 1494 983 1888 561 1458 1644 1290 1451 1089 17  
## [91] 930 311 954 1753 1237 1695 1485 1400 284 902 808 1703 1024 335 598  
## [106] 262 161 1714 589 678 1902 520 1462 721 1155 645 1411 368 1855 1656  
## [121] 1244 1076 1239 2 912 1548 692 1555 958 1651 625 1473 1073 761 1985  
## [136] 74 52 1698 581 1365 1916 463 949 177 1815 1465 955 674 1249 371  
## [151] 941 583 86 218 1986 1208 875 1514 1863 1521 899 334 344 1732 1263  
## [166] 984 1898 652 153 1306 1178 975 138 627 1302 1584 654 1329 1094 1918  
## [181] 1949 1546 596 1391 522 1665 1709 974 483 1935 16 1066 1747 1746 555  
## [196] 1588 1971 1912 133 559 1556 1786 286 822 1112 659 1565 94 487 803  
## [211] 1245 1436 462 1844 1893 1878 12 465 687 107 356 379 88 343 575  
## [226] 185 1115 735 1222 579 1250 777 1235 1214 409 973 1298 1813 1552 109  
## [241] 577 758 1280 1452 298 982 1613 491 519 738 440 1954 1923 1624 1399  
## [256] 346 1177 1721 1881 435 1720 112 826 750 1980 1929 1189 788 859 167  
## [271] 372 244 1866 1319 502 1953 1831 341 1840 347 1006 1580 1108 1680 1218  
## [286] 1688 1474 1258 1146 1283 1106 200 414 241 1536 1105 381 468 1615 1489  
## [301] 582 1666 1491 1388 774 923 1078 179 962 323 273 221 1917 640 896  
## [316] 796 751 772 1674 1000 345 1774 454 1447 1663 472 525 1225 848 1323  
## [331] 1592 1300 1649 1252 703 664 1240 1143 996 236 1533 534 1975 1116 232  
## [346] 183 42 753 646 1640 1123 1883 95 1997 683 909 37 747 1602 481  
## [361] 795 841 1196 648 1272 390 1842 1799 1693 83 401 222 215 437 68  
## [376] 811 209 1422 1945 477 1253 618 1928 1463 395 613 621 1274 1229 1328  
## [391] 1209 1848 277 1705 1070 480 1273 863 120 1896 1526 1570 726 1523 1653  
## [406] 115 199 1974 1369 1453 1350 1547 1809 1142 1412 1419 1818 1743 1349 786  
## [421] 1635 265 936 197 1016 518 653 1162 51 1180 1694 1172 1978 331 661  
## [436] 408 1658 98 342 1759 727 90 1924 1035 1021 1894 116 1593 1395 34  
## [451] 498 1822 1496 876 1031 981 564 1557 1641 1668 375 1036 513 995 1282  
## [466] 801 915 1993 608 1816 1629 1617 1927 662 1224 1783 1370 1879 358 688  
## [481] 1944 673 1206 1731 760 1808 403 66 299 815 1779 631 201 746 1294  
## [496] 985 1728 398 1600 1061

C.train <- C[-holdout, ]   
C.test <- C[holdout, ]

dim(C.train) #

## [1] 1500 18

dim(C.test) #

## [1] 500 18

# Fit the logistic regression model  
lr1 <- glm(Florence ~ Gender + M + R + F + FirstPurch + ChildBks + YouthBks + CookBks + DoltYBks + RefBks + ArtBks + GeogBks + ItalCook + ItalHAtlas + ItalArt,  
 data = C.train, family = binomial )  
summary(lr1)

##   
## Call:  
## glm(formula = Florence ~ Gender + M + R + F + FirstPurch + ChildBks +   
## YouthBks + CookBks + DoltYBks + RefBks + ArtBks + GeogBks +   
## ItalCook + ItalHAtlas + ItalArt, family = binomial, data = C.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.8425 -0.4718 -0.3220 -0.1951 2.9850   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.8159972 0.3683265 -4.930 8.21e-07 \*\*\*  
## Gender -0.8269788 0.1879613 -4.400 1.08e-05 \*\*\*  
## M 0.0008611 0.0010732 0.802 0.422339   
## R -0.0619704 0.0216769 -2.859 0.004252 \*\*   
## F 0.4413971 0.1087820 4.058 4.96e-05 \*\*\*  
## FirstPurch 0.0032053 0.0133932 0.239 0.810857   
## ChildBks -0.6710131 0.1618330 -4.146 3.38e-05 \*\*\*  
## YouthBks -0.8238403 0.2170283 -3.796 0.000147 \*\*\*  
## CookBks -0.8114902 0.1740752 -4.662 3.14e-06 \*\*\*  
## DoltYBks -0.9289765 0.2085671 -4.454 8.43e-06 \*\*\*  
## RefBks -0.2799765 0.2153622 -1.300 0.193592   
## ArtBks 0.5959189 0.1783640 3.341 0.000835 \*\*\*  
## GeogBks 0.0608700 0.1647373 0.369 0.711757   
## ItalCook 0.5607102 0.2927117 1.916 0.055420 .   
## ItalHAtlas 0.3487661 0.4532468 0.769 0.441606   
## ItalArt 0.2688061 0.3928893 0.684 0.493863   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1035.36 on 1499 degrees of freedom  
## Residual deviance: 832.21 on 1484 degrees of freedom  
## AIC: 864.21  
##   
## Number of Fisher Scoring iterations: 6

library(car)  
vif(lr1)

## Gender M R F FirstPurch ChildBks YouthBks   
## 1.021949 1.436746 2.389057 19.783430 7.925053 3.250331 1.788888   
## CookBks DoltYBks RefBks ArtBks GeogBks ItalCook ItalHAtlas   
## 3.356715 1.982037 2.266648 2.562628 2.729896 1.873899 1.781217   
## ItalArt   
## 1.750172

Here we’ll drop VIf’s >5 one by one first, then we’ll look p-values for our model.

Let’s drop “F” first, which has highest VIF.

# Fit the logistic regression model  
lr2 <- glm(Florence ~ Gender + M + R + FirstPurch + ChildBks + YouthBks + CookBks + DoltYBks + RefBks + ArtBks + GeogBks + ItalCook + ItalHAtlas + ItalArt,  
 data = C.train, family = binomial )  
summary(lr2)

##   
## Call:  
## glm(formula = Florence ~ Gender + M + R + FirstPurch + ChildBks +   
## YouthBks + CookBks + DoltYBks + RefBks + ArtBks + GeogBks +   
## ItalCook + ItalHAtlas + ItalArt, family = binomial, data = C.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0915 -0.4885 -0.3344 -0.1989 2.9817   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.901687 0.287981 -3.131 0.00174 \*\*   
## Gender -0.833252 0.185596 -4.490 7.14e-06 \*\*\*  
## M 0.001101 0.001060 1.039 0.29902   
## R -0.120946 0.016857 -7.175 7.25e-13 \*\*\*  
## FirstPurch 0.016936 0.012963 1.307 0.19136   
## ChildBks -0.266344 0.125008 -2.131 0.03312 \*   
## YouthBks -0.435560 0.189865 -2.294 0.02179 \*   
## CookBks -0.421318 0.141715 -2.973 0.00295 \*\*   
## DoltYBks -0.530662 0.179235 -2.961 0.00307 \*\*   
## RefBks 0.100943 0.193137 0.523 0.60122   
## ArtBks 0.939130 0.159240 5.898 3.69e-09 \*\*\*  
## GeogBks 0.458288 0.135054 3.393 0.00069 \*\*\*  
## ItalCook 0.588315 0.289117 2.035 0.04186 \*   
## ItalHAtlas 0.334859 0.450352 0.744 0.45715   
## ItalArt 0.292514 0.390114 0.750 0.45337   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1035.4 on 1499 degrees of freedom  
## Residual deviance: 848.2 on 1485 degrees of freedom  
## AIC: 878.2  
##   
## Number of Fisher Scoring iterations: 6

library(car)  
vif(lr2)

## Gender M R FirstPurch ChildBks YouthBks CookBks   
## 1.018138 1.444634 1.294913 7.960965 2.042601 1.417164 2.310325   
## DoltYBks RefBks ArtBks GeogBks ItalCook ItalHAtlas ItalArt   
## 1.537505 1.874539 2.044659 1.865681 1.890261 1.772833 1.734066

Now, We’ll drop “first Purchase”

# Fit the logistic regression model  
lr3 <- glm(Florence ~ Gender + M + R + ChildBks + YouthBks + CookBks + DoltYBks + RefBks + ArtBks + GeogBks + ItalCook + ItalHAtlas + ItalArt,  
 data = C.train, family = binomial )  
summary(lr3)

##   
## Call:  
## glm(formula = Florence ~ Gender + M + R + ChildBks + YouthBks +   
## CookBks + DoltYBks + RefBks + ArtBks + GeogBks + ItalCook +   
## ItalHAtlas + ItalArt, family = binomial, data = C.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.9656 -0.4884 -0.3358 -0.1973 3.0243   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.820663 0.281046 -2.920 0.00350 \*\*   
## Gender -0.826161 0.185262 -4.459 8.22e-06 \*\*\*  
## M 0.001213 0.001054 1.150 0.25000   
## R -0.113206 0.015918 -7.112 1.14e-12 \*\*\*  
## ChildBks -0.191622 0.110290 -1.737 0.08231 .   
## YouthBks -0.349405 0.176407 -1.981 0.04763 \*   
## CookBks -0.344296 0.128206 -2.685 0.00724 \*\*   
## DoltYBks -0.445936 0.164967 -2.703 0.00687 \*\*   
## RefBks 0.195610 0.178865 1.094 0.27412   
## ArtBks 1.013946 0.148820 6.813 9.54e-12 \*\*\*  
## GeogBks 0.540893 0.118953 4.547 5.44e-06 \*\*\*  
## ItalCook 0.572268 0.287195 1.993 0.04630 \*   
## ItalHAtlas 0.267704 0.448582 0.597 0.55066   
## ItalArt 0.260741 0.387192 0.673 0.50068   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1035.36 on 1499 degrees of freedom  
## Residual deviance: 849.91 on 1486 degrees of freedom  
## AIC: 877.91  
##   
## Number of Fisher Scoring iterations: 6

library(car)  
vif(lr3)

## Gender M R ChildBks YouthBks CookBks DoltYBks   
## 1.016453 1.438128 1.127899 1.593129 1.238018 1.894463 1.317875   
## RefBks ArtBks GeogBks ItalCook ItalHAtlas ItalArt   
## 1.607937 1.807176 1.462383 1.878508 1.744037 1.711365

In above Model all VIF’s are <5

Now we’ll drop all the P- Values, which are > 0.05

# Fit the logistic regression model  
lr4 <- glm(Florence ~ Gender + R + ChildBks + YouthBks + CookBks + DoltYBks + ArtBks + GeogBks + ItalCook, data = C, family = binomial )  
summary(lr4)

##   
## Call:  
## glm(formula = Florence ~ Gender + R + ChildBks + YouthBks + CookBks +   
## DoltYBks + ArtBks + GeogBks + ItalCook, family = binomial,   
## data = C)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0486 -0.4939 -0.3421 -0.1964 3.0238   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.56907 0.18119 -3.141 0.00169 \*\*   
## Gender -0.86644 0.15914 -5.444 5.20e-08 \*\*\*  
## R -0.11448 0.01357 -8.435 < 2e-16 \*\*\*  
## ChildBks -0.09970 0.08882 -1.122 0.26166   
## YouthBks -0.21748 0.15026 -1.447 0.14779   
## CookBks -0.29524 0.10564 -2.795 0.00519 \*\*   
## DoltYBks -0.41125 0.13921 -2.954 0.00313 \*\*   
## ArtBks 1.03818 0.12058 8.610 < 2e-16 \*\*\*  
## GeogBks 0.56548 0.10319 5.480 4.25e-08 \*\*\*  
## ItalCook 0.49927 0.22249 2.244 0.02483 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1373.5 on 1999 degrees of freedom  
## Residual deviance: 1142.2 on 1990 degrees of freedom  
## AIC: 1162.2  
##   
## Number of Fisher Scoring iterations: 6

library(car)  
vif(lr4)

## Gender R ChildBks YouthBks CookBks DoltYBks ArtBks GeogBks   
## 1.015605 1.101905 1.438549 1.169037 1.791731 1.316252 1.502148 1.426216   
## ItalCook   
## 1.375667

**Now all the VIF’s are <5, and P-value’s are also < 0.05**

#Performance Measures of a Binary Classifier #Precision and Recall for both categories, training set #CM is confusion matrix #——————————————- # Observed #———0——–1—- #Pred #0 CM[1,1] CM[1,2] = TN FN #1 CM[2,1] CM[2 ,2] FP TP

#Precision and Recall formulas: #Category 1 # Precision = TP/(TP+FP), diag/row sum # Recall = TP/(TP+FN) diag/column sum

#Precision1 <- CM.train[2,2]/(CM.train[2,1]+CM.train[2,2]) # diag/row sum #Recall1 <- CM.train[2,2]/(CM.train[1,2]+CM.train[2,2]) # diag/column sum #—————————————————————— #Category 0 #Precision0 <- CM.train[1,1]/(CM.train[1,1]+CM.train[1,2]) # diag/row sum #Recall0 <- CM.train[1,1]/(CM.train[1,1]+CM.train[2,1]) # diag/column sum #——————————————————————

#Predict training data using the model lr4  
observed.train <- C.train$Florence  
predicted.train <-predict(lr4, C.train, type = 'response')  
predicted.train <- round(predicted.train)  
  
#Predict testing data using the model lr4  
observed.test <- C.test$Florence  
predicted.test <-predict(lr4, C.test, type = 'response')  
predicted.test <- round(predicted.test)

CM <- function(x,y)  
{  
# x = predicted, y = observed  
table(x,y)  
}  
  
CM.train <- CM(C.train$Florence, predicted.train)  
CM.test <- CM(C.test$Florence, predicted.test)

PRF(CM.train)

## [1] 0.1219512 0.6451613 0.2051282 0.9917665 0.9019741 0.9447415

PRF(CM.test)

## [1] 0.05660377 0.60000000 0.10344828 0.99552573 0.89898990 0.94479830

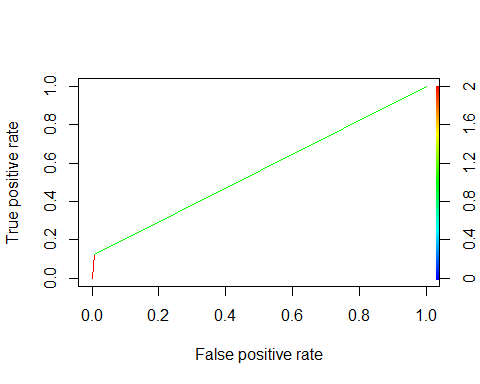
# ---------------------------------------------------------------------  
#install.packages("ROCR",dependencies=TRUE)  
library(ROCR)

## Warning: package 'ROCR' was built under R version 4.2.3

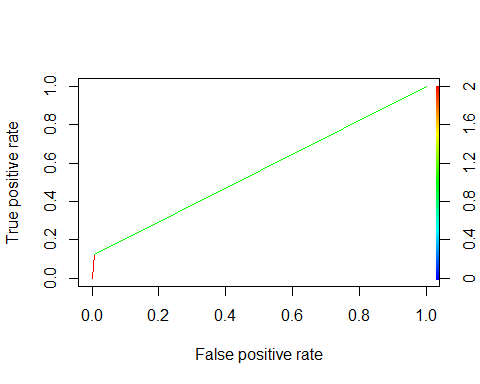
#Predict training data using the model LR1  
observed.train <- C.train$Florence  
predicted.train <-predict(lr4, C.train, type = 'response')  
predicted.train <- round(predicted.train)  
  
#Predict testing data using the model LR1  
observed.test <- C.test$Florence  
predicted.test <-predict(lr4, C.test, type = 'response')  
predicted.test <- round(predicted.test)  
  
Y <- observed.train  
str(Y) # int [1:784] 1 0 0 1 0 1 0 1 1 1 ...

## int [1:1500] 0 0 0 1 0 0 0 0 0 1 ...

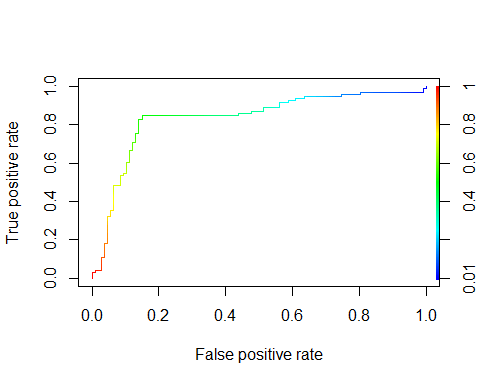
pred <- prediction(predicted.train, Y)  
perf <- performance(pred,"tpr","fpr")  
plot(perf,colorize=TRUE)



# -----------------------------------------------------------  
library(ROCR)  
# plot a ROC curve for a single prediction run  
# and color the curve according to cutoff.  
  
pred <- prediction(predicted.train, Y)  
perf <- performance(pred,"tpr","fpr")  
plot(perf,colorize=TRUE)



data(ROCR.simple)  
df <- data.frame(ROCR.simple)  
pred <- prediction(df$predictions, df$labels)  
perf <- performance(pred,"tpr","fpr")  
plot(perf,colorize=TRUE)



# ------------------------------------------------------  
  
library(ROCR)  
p <- predict(lr4, newdata=subset(C.train), type="response")  
pr <- prediction(p, C.train$Florence)  
prf <- performance(pr, measure = "tpr", x.measure = "fpr")  
#plot(prf)  
  
DF.PR <- cbind.data.frame(prf@x.values[[1]], prf@y.values[[1]], prf@alpha.values[[1]])  
colnames(DF.PR) <- c("FPR","TPR","cutoff")  
head(DF.PR)

## FPR TPR cutoff  
## 1 0.000000000 0.000000000 Inf  
## 2 0.000000000 0.006097561 0.8749787  
## 3 0.000000000 0.012195122 0.8506791  
## 4 0.000000000 0.018292683 0.8450449  
## 5 0.000000000 0.024390244 0.8288504  
## 6 0.000748503 0.024390244 0.8241600

library(ggplot2)  
proc.train <- ggplot(DF.PR, aes(x=FPR, y=TPR)) + geom\_line()  
proc.train <- proc.train + geom\_segment(aes(x = 0, y = 0, xend = 1, yend = 1), data = DF.PR)+  
 geom\_segment(aes(x = 0, y = 0, xend = 1, yend = 0), data = DF.PR)+  
 geom\_segment(aes(x = 1, y = 0, xend = 1, yend = 1), data = DF.PR) +  
 ggtitle("ROC Curve from Logistic Regression for Titanic Data - Training Set")  
  
  
  
auc <- performance(pr, measure = "auc")  
auc <- auc@y.values[[1]]  
auc # 0.78493

## [1] 0.7849332

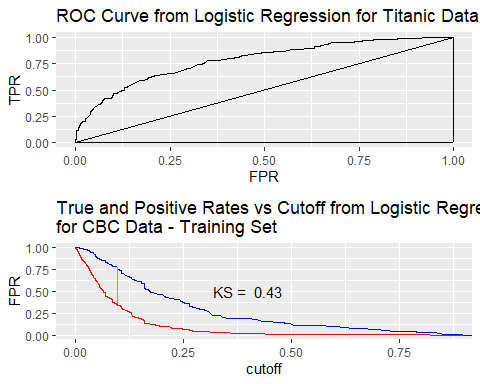
#———————# KS (Kolomogorov-Smirnov) Statistic #———————#

# KS = maximum(TPR-FPR)

pK1 <- ggplot()+geom\_line(data=DF.PR,aes(x=cutoff,y=FPR), color='red') +   
 geom\_line(data=DF.PR,aes(x=cutoff,y=TPR), color='blue')  
  
DF.PR$diff <- DF.PR$TPR - DF.PR$FPR  
KS.train <- max(DF.PR$diff) # 0.38  
print(KS.train)

## [1] 0.4324339

i.m <- which.max(DF.PR$diff) # 2  
xM <- DF.PR$cutoff[i.m]  
yML <- DF.PR$FPR[i.m]  
yMU <- DF.PR$TPR[i.m]  
  
pKS.train <- pK1 + geom\_segment(aes(x = xM, y = yML,  
 xend = xM, yend = yMU, colour="black"))+  
 annotate("text", x=0.4, y=0.5, label= "KS = 0.43")+   
 theme(legend.position = "none")+  
 ggtitle("True and Positive Rates vs Cutoff from Logistic Regression \nfor CBC Data - Training Set")  
  
library(gridExtra)  
grid.arrange(proc.train,pKS.train,nrow=2)



**++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++(b) Fit an SVM model for Florence as a function of the potential predictors in the datafile CBC.csv data. ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

library(e1071)  
svm1\_CBC <- svm(Florence ~ Gender + M + R + F + FirstPurch + ChildBks + YouthBks + CookBks + DoltYBks + RefBks + ArtBks + GeogBks + ItalCook + ItalHAtlas + ItalArt,  
 family = binomial, kernel ="radial", type="C-classification", probability = TRUE, data = C.train)  
summary(svm1\_CBC)

##   
## Call:  
## svm(formula = Florence ~ Gender + M + R + F + FirstPurch + ChildBks +   
## YouthBks + CookBks + DoltYBks + RefBks + ArtBks + GeogBks + ItalCook +   
## ItalHAtlas + ItalArt, data = C.train, family = binomial, kernel = "radial",   
## type = "C-classification", probability = TRUE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
##   
## Number of Support Vectors: 505  
##   
## ( 341 164 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## 0 1

y.obs\_train <- C.train$Florence  
y.SVM.fitted\_train <- svm1\_CBC$fitted  
CM.SVM\_train <- table(y.obs\_train,y.SVM.fitted\_train)  
CM.SVM\_train

## y.SVM.fitted\_train  
## y.obs\_train 0 1  
## 0 1336 0  
## 1 149 15

**The confusion matrix shows the number of true negatives (1336), false negatives (149), true positives (15), and false positives (0) for the training data.**

OA.SVM\_train <- sum(diag(CM.SVM\_train))/sum(CM.SVM\_train)  
PRF1(CM.SVM\_train)

## Prec.1 Rec.1 F1.1 Prec.0 Rec.0 F1.0   
## 0.09146341 1.00000000 0.16759777 1.00000000 0.89966330 0.94718185

OA.SVM\_train

## [1] 0.9006667

y.obs\_test <- C.test$Florence  
y.SVM.fitted\_test <- predict(svm1\_CBC,C.test)  
CM.SVM\_test <- table(y.obs\_test,y.SVM.fitted\_test)  
CM.SVM\_test

## y.SVM.fitted\_test  
## y.obs\_test 0 1  
## 0 446 1  
## 1 53 0

OA.SVM\_test <- sum(diag(CM.SVM\_test))/sum(CM.SVM\_test)  
PRF1(CM.SVM\_test)

## Prec.1 Rec.1 F1.1 Prec.0 Rec.0 F1.0   
## 0.0000000 0.0000000 0.0000000 0.9977629 0.8937876 0.9429175

OA.SVM\_test

## [1] 0.892

**++++++++++++++++++++++++++++++++++++++++++++++++**

**(c) Compare the two models.**

**+++++++++++++++++++++++++++++++++++++++++++++**

# Logistic Regression Metrics  
CM.lr\_train <- CM(C.train$Florence, predicted.train)  
CM.lr\_test <- CM(C.test$Florence, predicted.test)  
PRF.lr\_train <- PRF1(CM.lr\_train)  
PRF.lr\_test <- PRF1(CM.lr\_test)  
OA.lr\_train <- sum(diag(CM.lr\_train)) / sum(CM.lr\_train)  
OA.lr\_test <- sum(diag(CM.lr\_test)) / sum(CM.lr\_test)  
  
# SVM Metrics  
CM.svm\_train <- table(y.obs\_train, y.SVM.fitted\_train)  
CM.svm\_test <- table(y.obs\_test, y.SVM.fitted\_test)  
PRF.svm\_train <- PRF1(CM.svm\_train)  
PRF.svm\_test <- PRF1(CM.svm\_test)  
OA.svm\_train <- sum(diag(CM.svm\_train)) / sum(CM.svm\_train)  
OA.svm\_test <- sum(diag(CM.svm\_test)) / sum(CM.svm\_test)

# Print the metrics for both models  
  
  
  
print("Logistic Regression Metrics:")

## [1] "Logistic Regression Metrics:"

print(paste("Train Accuracy:", OA.lr\_train))

## [1] "Train Accuracy: 0.896666666666667"

print(paste("Test Accuracy:", OA.lr\_test))

## [1] "Test Accuracy: 0.896"

print(paste("Train Precision:", PRF.lr\_train["Prec.1"]))

## [1] "Train Precision: 0.121951219512195"

print(paste("Test Precision:", PRF.lr\_test["Prec.1"]))

## [1] "Test Precision: 0.0566037735849057"

print(paste("Train Recall:", PRF.lr\_train["Rec.1"]))

## [1] "Train Recall: 0.645161290322581"

print(paste("Test Recall:", PRF.lr\_test["Rec.1"]))

## [1] "Test Recall: 0.6"

print(paste("Train F1-score:", PRF.lr\_train["F1.1"]))

## [1] "Train F1-score: 0.205128205128205"

print(paste("Test F1-score:", PRF.lr\_test["F1.1"]))

## [1] "Test F1-score: 0.103448275862069"

print("\nSVM Metrics:")

## [1] "\nSVM Metrics:"

print(paste("Train Accuracy:", OA.svm\_train))

## [1] "Train Accuracy: 0.900666666666667"

print(paste("Test Accuracy:", OA.svm\_test))

## [1] "Test Accuracy: 0.892"

print(paste("Train Precision:", PRF.svm\_train["Prec.1"]))

## [1] "Train Precision: 0.0914634146341463"

print(paste("Test Precision:", PRF.svm\_test["Prec.1"]))

## [1] "Test Precision: 0"

print(paste("Train Recall:", PRF.svm\_train["Rec.1"]))

## [1] "Train Recall: 1"

print(paste("Test Recall:", PRF.svm\_test["Rec.1"]))

## [1] "Test Recall: 0"

print(paste("Train F1-score:", PRF.svm\_train["F1.1"]))

## [1] "Train F1-score: 0.167597765363128"

print(paste("Test F1-score:", PRF.svm\_test["F1.1"]))

## [1] "Test F1-score: 0"

**Observation:**

**From, observation**

**Both models have similar accuracy on the test set, with the logistic regression model having a slightly higher accuracy (0.896 vs. 0.892).**

**The logistic regression model has a higher precision (0.0566) compared to SVM (0.0000), indicating that it is better at correctly identifying positive cases (Florence = 1) in the test set.**

**Similarly, the logistic regression model has a higher recall (0.600) compared to SVM (0.000), meaning that it can capture a greater proportion of actual positive cases in the test set.**

**The F1-score, which considers both precision and recall, is also higher for the logistic regression model (0.103 vs. 0.000) on the test set.**

**Overall, the logistic regression model shows better performance on the test set compared to the SVM model. However, it is worth noting that both models have relatively low precision, indicating that they struggle in correctly predicting positive cases.**