ALY 6040 Module 4 Technique Practice

Student's name: Mohammad Hossein Movahedi

Assignment title: Module 4 Technique Practice

Course number and title: ALY6040 71368 Data Mining Applications SEC 09 Fall 2022 CPS [TOR-A-HY]

Term: 202315_A Fall 2022 CPS Quarter First Half

Instructor's name: Hootan Kamran, Ph.D.

Oct 21, 2022,

Introduction

In this assignment, I will use the Support Vector Machines technique to predict whether or not a blood donor would donate in march of 2007 or not. The dataset I used for this assignment can be found in Kaggle.
There are five main steps in this project.

1. Loading the data
2. Cleaning dataset
3. Deleting Heavy influential data
4. Sampling
5. Running the SVM
6. Tunning the SVM
7. Validating the final model

Code walkthrough

Loading the data

The first part of the Code is Just Installing and loading the Libraries.

```
#loading libraries
install.packages('tidyverse')
install.packages('MASS')
install.packages('car')
install.packages('e1071')
install.packages('caret')
install.packages('carTools')
install.packages('cowplot')
install.packages('pROC')
install.packages('ggcorrplot')
install.packages('corrplot')
install.packages('dplyr')
install.packages('janitor')
install.packages("mlbench")
install.packages('repr')
install.packages("caTools")
install.packages("randomForest")
install.packages('effects')
# loading libraries
library(repr)
library(tidyverse)
library(MASS)
library(car)
library(e1071)
library(caret)
library(caTools)
library(randomForest)
library(cowplot)
library(pROC)
library(ggcorrplot)
library(dplyr)
library(janitor)
library(mlbench)
library(data.table)
library(matrixStats)
```

In the next part, I load the dataset from Kaggle the dataset I choose to work on is taken from the Blood Transfusion Service Center in Hsin-Chu City in Taiwan (Chauhan, 2022)

```
#loading the dataset
dt <- read.csv("transfusion.csv")
#looking at the dataset
dim(dt)
summary(dt)
str(dt)
```

The Dataset has 748 rows and 5 variables. The summary of data is shown below

```
> summary(dt)
 Recency..months. Frequency..times. Monetary..c.c..blood. Time..months.
whether.he.she.donated.blood.in.March.2007
 Min.   : 0.000   Min.   : 1.000   Min.   :  250         Min.   : 2.00
Min.   :0.000
 1st Qu.: 2.750   1st Qu.: 2.000   1st Qu.:  500         1st Qu.:16.00
1st Qu.:0.000
 Median : 7.000   Median : 4.000   Median : 1000         Median :28.00
Median :0.000
 Mean   : 9.507   Mean   : 5.515   Mean   : 1379         Mean   :34.28
Mean   :0.238
 3rd Qu.:14.000   3rd Qu.: 7.000   3rd Qu.: 1750         3rd Qu.:50.00
3rd Qu.:0.000
```

```
 Max.   :74.000   Max.   :50.000   Max.   :12500         Max.   :98.00  
Max.   :1.000
```

Cleaning dataset

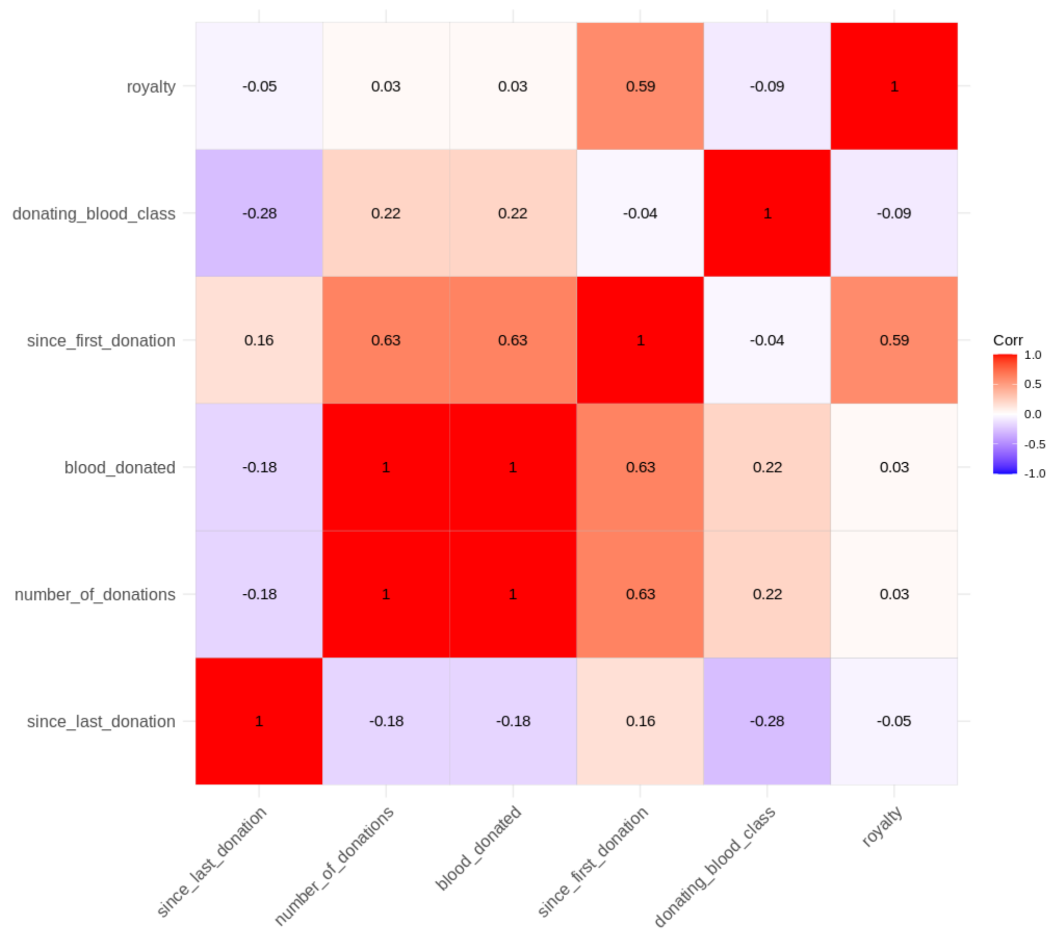
In this part I first corrected the variables name and also calculated a loyalty variable based on the frequency of visits and duration of visits

```
#cleaning data set names
dt<-dt %>% clean_names()
names(dt)
dt<-dt %>%
  rename(
    since_last_donation = recency_months,
    number_of_donations = frequency_times,
    blood_donated = monetary_c_c_blood,
    since_first_donation = time_months,
    donating_blood_class = whether_he_she_donated_blood_in_march_2007

  )

# calculating royalty index ( total visit divided by duration of donation)
dt$royalty = (dt$since_first_donation - dt$since_last_donation)/
dt$number_of_donations
```

Then I checked the dataset for singularity and deleted one of the heavy correlated variables and finally corrected the data structure and reordered dataset

```
#finding singularities
corr<-cor(dt)
ggcorrplot(corr,lab = TRUE)

# number of donation and blood donated are heavily correlated so I drop
number of donation
dt = subset(dt, select = -c(number_of_donations) )
dt$donating_blood_class <- as.factor(dt$donating_blood_class)

#reordering data
col_order <-
c("since_last_donation","blood_donated","since_first_donation",
"royalty","donating_blood_class" )
dt <- dt[, col_order]
```

The Corrplot of variables is shown below :



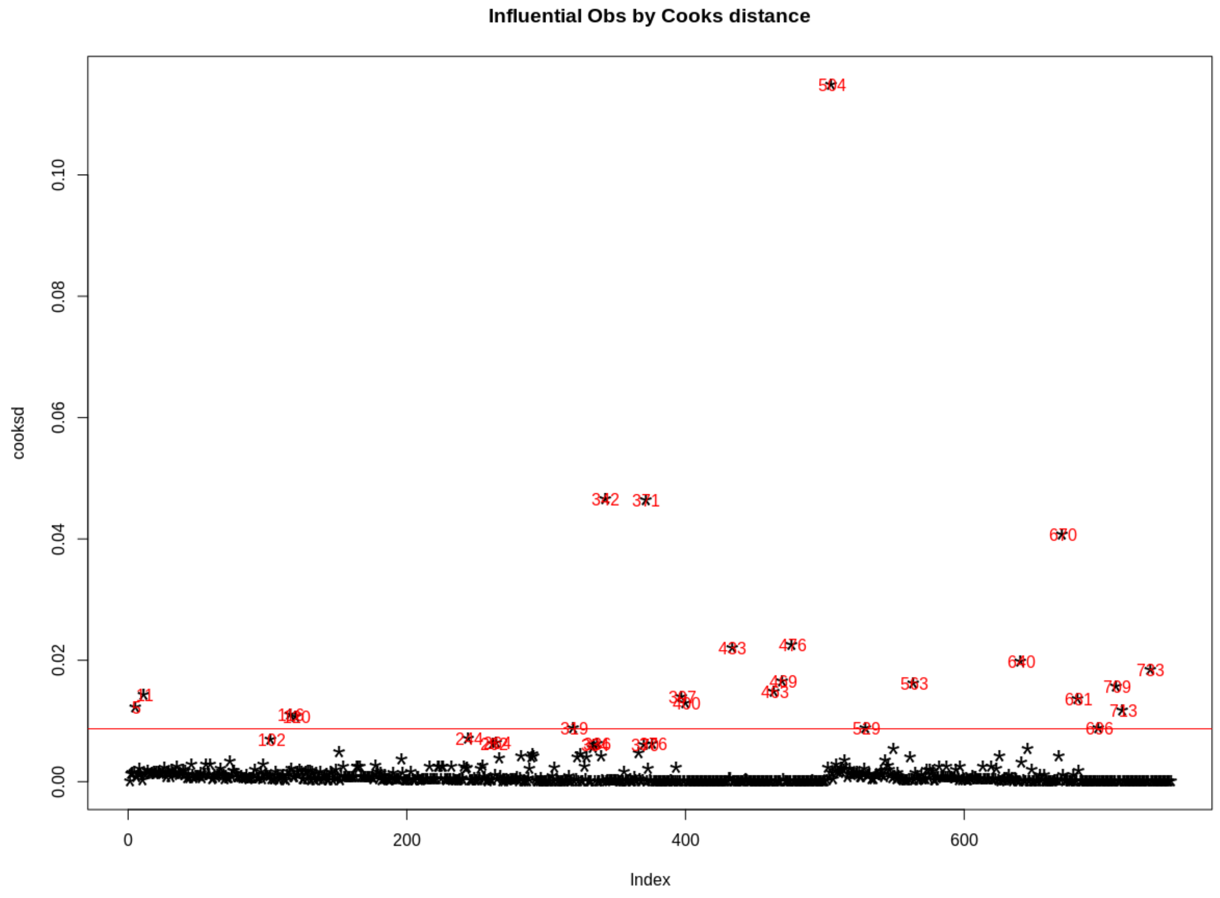
Deleting Heavy influential data

In this part, using the cook's method, I deleted heavy influential data in the prediction this thing improved the specification of the final model dramatically.

```
#finding outliers for glm prediction for donation blood class
## Cook's Distance Method
mod <- glm(donating_blood_class ~., data = dt,family=binomial)
par(mfrow = c(2, 2))
library(effects)
plot(allEffects(mod))
summary(mod)
dev.off()
cooksd <- stats:::cooks.distance.glm(mod)
plot(cooksd, pch="*", cex=2, main="Influential Obs by Cooks distance")  #
plot cook's distance
abline(h = 6*mean(cooksd, na.rm=T), col="red")  # add cutoff line
text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>4*mean(cooksd,
na.rm=T),names(cooksd),""), col="red")  # add labels
```

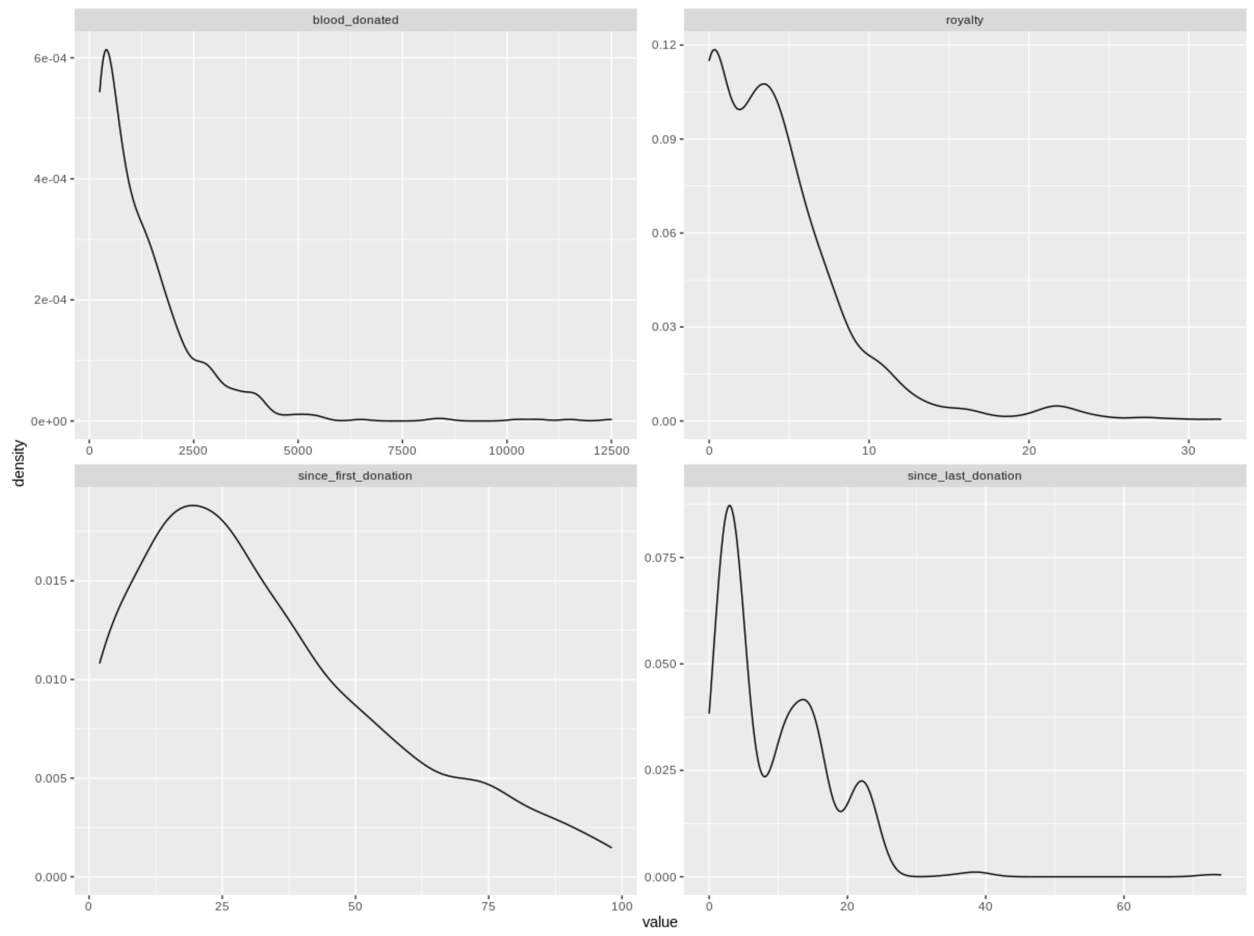
```
influential <- cooksd[(cooksd > (4 * mean(cooksd, na.rm = TRUE)))]
influential
ci <- ifelse(cooksd>4*mean(cooksd, na.rm=T),1,0)  # influential row numbers
table(ci)

names_of_influential <- names(influential)
outliers <- dt[names_of_influential,]
dt_without_outliers <- dt %>% anti_join(outliers)
modn <- glm(donating_blood_class ~., data =
dt_without_outliers,family=binomial)
dev.off()
par(mfrow = c(2, 2))
plot(allEffects(mod))
dev.off()
```

The picture below shows the influential data above 4 cooks' distance



Finally I looked at data before moving to machine learning part

```
#looking at dataset one last time
dt_without_outliers %>%
  keep(is.numeric) %>%                        # Keep only numeric columns
  gather() %>%                                # Convert to key-value pairs
  ggplot(aes(value)) +                        # Plot the values
  facet_wrap(~ key, scales = "free") +    # In separate panels
  geom_density()
```



Sampling Dataset

This is usually an easy part but since the dataset is small and the positive values aren't that common I did a biased sampling to have more than normal proportion positive values in training dataset this also helped the SVM model a lot.

```
# sampling data set for testing with bias due to small dataset
set.seed(123)
prb <- ifelse(dt_without_outliers$donating_blood_class == 1 ,0.7,0.3)
sample <- sample(nrow(dt_without_outliers),replace = FALSE, 300, prob =
prb)
train <- dt_without_outliers[sample,]
test <- dt_without_outliers %>% anti_join(train)
```

```
count(train$donating_blood_class==1)/count(train$donating_blood_class==0)
summary(dt_without_outliers)
```

Running SVM

I used the Carret package SVM method since it’s faster and generated better results with same optimal cost

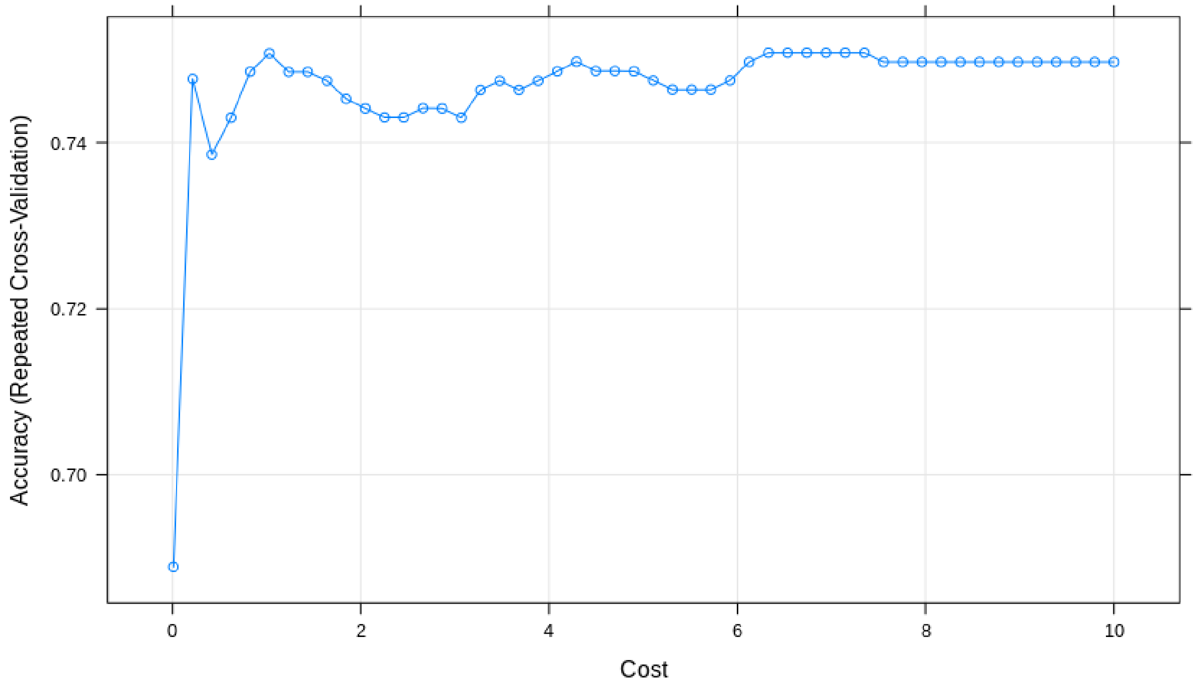
```
# the actual SVM
set.seed(123)
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)


svm_Linear_Grid <- train(donating_blood_class ~., data = train, method =
"svmLinear",
                         trControl=trctrl,
                         preProcess = c("center", "scale"),
                         tuneGrid = expand.grid(C = seq(0.01, 10, length =
50)),
                         tuneLength = 10)
svm_Linear_Grid$finalModel
summary(svm_Linear_Grid)
plot(svm_Linear_Grid)
```

The summary of the model and also the plot is shown below:

```
> svm_Linear_Grid
Support Vector Machines with Linear Kernel

300 samples
  4 predictor
  2 classes: '0', '1'

Pre-processing: centered (4), scaled (4)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 269, 270, 270, 270, 270, 270, ...
Resampling results across tuning parameters:

  C          Accuracy   Kappa
   0.0100000  0.6888988  0.07329322
   0.2138776  0.7476851  0.40757464
```

```
0.4177551  0.7385737  0.40571705
0.6216327  0.7430182  0.42197142
0.8255102  0.7485737  0.43767025
1.0293878  0.7507601  0.44506317
1.2332653  0.7485379  0.43700873
1.4371429  0.7485379  0.43503867
1.6410204  0.7474626  0.43243311
1.8448980  0.7452762  0.42836474
2.0487755  0.7441268  0.42547745
2.2526531  0.7430515  0.42454598
2.4565306  0.7430515  0.42294153
2.6604082  0.7441626  0.42292798
2.8642857  0.7441268  0.42478043
3.0681633  0.7430157  0.42150874
3.2720408  0.7463490  0.42933261
3.4759184  0.7474601  0.43282467
3.6797959  0.7463490  0.42929526
3.8836735  0.7474601  0.43115357
4.0875510  0.7486096  0.43168496
4.2914286  0.7497590  0.43276577
4.4953061  0.7486479  0.42939195
4.6991837  0.7486479  0.42939195
4.9030612  0.7486120  0.42869669
5.1069388  0.7475009  0.42520462
5.3108163  0.7463898  0.42264946
5.5146939  0.7463898  0.42150715
5.7185714  0.7463898  0.42150715
5.9224490  0.7475009  0.42413341
6.1263265  0.7497231  0.42773999
6.3302041  0.7508343  0.42984408
6.5340816  0.7508343  0.42984408
6.7379592  0.7508343  0.42984408
6.9418367  0.7508343  0.42984408
7.1457143  0.7508343  0.42984408
7.3495918  0.7508343  0.42984408
7.5534694  0.7497231  0.42752679
7.7573469  0.7497231  0.42752679
7.9612245  0.7497231  0.42752679
8.1651020  0.7497231  0.42752679
8.3689796  0.7497231  0.42752679
8.5728571  0.7497231  0.42752679
8.7767347  0.7497231  0.42752679
8.9806122  0.7497231  0.42752679
```

```
   9.1844898  0.7497231  0.42752679
   9.3883673  0.7497231  0.42752679
   9.5922449  0.7497231  0.42752679
   9.7961224  0.7497231  0.42752679
  10.0000000  0.7497231  0.42752679

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 6.330204.
```



Validation

Finally using the reset of the data that SVM hasn't been trained on before I tested the accuracy and other factors of the model

```
test_pred_grid <- predict(svm_Linear_Grid, newdata =test)
confusionMatrix(table(test_pred_grid, test$donating_blood_class))
```

```
> confusionMatrix(table(test_pred_grid, test$donating_blood_class))
Confusion Matrix and Statistics


test_pred_grid   0   1
             0 208  22
             1  33  19

               Accuracy : 0.805
                 95% CI : (0.7538, 0.8496)
    No Information Rate : 0.8546
    P-Value [Acc > NIR] : 0.9910

                  Kappa : 0.2938

 Mcnemar's Test P-Value : 0.1775

            Sensitivity : 0.8631
            Specificity : 0.4634
         Pos Pred Value : 0.9043
         Neg Pred Value : 0.3654
             Prevalence : 0.8546
         Detection Rate : 0.7376
   Detection Prevalence : 0.8156
      Balanced Accuracy : 0.6632

       'Positive' Class : 0
```

As can be seen, the model is performing 30 % better than random prediction . the sensitivity of the model is high and specificity is fairly good considering the small sample size.

Recommendation

Based on the Model we can see that the defined variables were able to detect a pattern in the dataset however collecting more data and stacking a model on this model is highly recommended for future research.

References

Chauhan, A. (2022). *Blood Transfusion Dataset*. [online] Kaggle.com. Available at: https://www.kaggle.com/datasets/whenamancodes/blood-transfusion-dataset [Accessed 23 Oct. 2022].

Appendix

```r
print('Mohammad Hossein Movahedi')

#loading libraries
install.packages('tidyverse')
install.packages('MASS')
install.packages('car')
install.packages('e1071')
install.packages('caret')
install.packages('carTools')
install.packages('cowplot')
install.packages('pROC')
install.packages('ggcorrplot')
install.packages('corrplot')
install.packages('dplyr')
install.packages('janitor')
install.packages("mlbench")
install.packages('repr')
install.packages("caTools")
install.packages("randomForest")
install.packages('effects')
# loading libraries
library(repr)
library(tidyverse)
library(MASS)
library(car)
library(e1071)
library(caret)
library(caTools)
library(randomForest)
library(cowplot)
library(pROC)
library(ggcorrplot)
library(dplyr)
library(janitor)
library(mlbench)
library(data.table)
library(matrixStats)

#loading the dataset
dt <- read.csv("transfusion.csv")
```

```r
#looking at the dataset
dim(dt)
summary(dt)
str(dt)

#cleaning data set names
dt<-dt %>% clean_names()
names(dt)
dt<-dt %>%
  rename(
    since_last_donation = recency_months,
    number_of_donations = frequency_times,
    blood_donated = monetary_c_c_blood,
    since_first_donation = time_months,
    donating_blood_class = whether_he_she_donated_blood_in_march_2007

  )

# calculating royalty index ( total visit divided by duration of donation)
dt$royalty = (dt$since_first_donation - dt$since_last_donation)/
dt$number_of_donations

#correcting data structure
dt$donating_blood_class <- as.numeric(dt$donating_blood_class)
dt$blood_donated <- as.numeric(dt$blood_donated)
dt$number_of_donations <-as.numeric(dt$number_of_donations)

#finding singularities
corr<-cor(dt)
ggcorrplot(corr,lab = TRUE)

# number of donation and blood donated are heavily correlated so I drop
number of donation
dt = subset(dt, select = -c(number_of_donations) )
dt$donating_blood_class <- as.factor(dt$donating_blood_class)

#reordering data
col_order <-
c("since_last_donation","blood_donated","since_first_donation",
"royalty","donating_blood_class" )
dt <- dt[, col_order]
#finding outliers for glm prediction for donation blood class
## Cook's Distance Method
```

```
mod <- glm(donating_blood_class ~., data = dt,family=binomial)
par(mfrow = c(2, 2))
library(effects)
plot(allEffects(mod))
summary(mod)
dev.off()
cooksd <- stats:::cooks.distance.glm(mod)
plot(cooksd, pch="*", cex=2, main="Influential Obs by Cooks distance")  #
plot cook's distance
abline(h = 6*mean(cooksd, na.rm=T), col="red")  # add cutoff line
text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>4*mean(cooksd,
na.rm=T),names(cooksd),""), col="red")  # add labels

influential <- cooksd[(cooksd > (4 * mean(cooksd, na.rm = TRUE)))]
influential
ci <- ifelse(cooksd>4*mean(cooksd, na.rm=T),1,0)  # influential row numbers
table(ci)

names_of_influential <- names(influential)
outliers <- dt[names_of_influential,]
dt_without_outliers <- dt %>% anti_join(outliers)
modn <- glm(donating_blood_class ~., data =
dt_without_outliers,family=binomial)
dev.off()
par(mfrow = c(2, 2))
plot(allEffects(mod))
dev.off()
#looking at dataset one last time
dt_without_outliers %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
  facet_wrap(~ key, scales = "free") +   # In separate panels
  geom_density()
# sampling data set for testing with bias due to small dataset
set.seed(123)
prb <- ifelse(dt_without_outliers$donating_blood_class == 1 ,0.7,0.3)
sample <- sample(nrow(dt_without_outliers),replace = FALSE, 300, prob =
prb)
train <- dt_without_outliers[sample,]
test <- dt_without_outliers %>% anti_join(train)
count(train$donating_blood_class==1)/count(train$donating_blood_class==0)
summary(dt_without_outliers)
```

```
# the actual SVM
set.seed(123)
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)


svm_Linear_Grid <- train(donating_blood_class ~., data = train, method =
"svmLinear",
                         trControl=trctrl,
                         preProcess = c("center", "scale"),
                         tuneGrid = expand.grid(C = seq(0.01, 10, length =
50)),
                         tuneLength = 10)
svm_Linear_Grid$finalModel
svm_Linear_Grid
plot(svm_Linear_Grid)
test_pred_grid <- predict(svm_Linear_Grid, newdata =test)
confusionMatrix(table(test_pred_grid, test$donating_blood_class)
```