# KDD_dataset

## Data set introduction

This is the data set used for The Second International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-98 The Fourth International Conference on Knowledge Discovery and Data Mining. The competition task is a regression problem where the goal is to estimate the return from a direct mailing in order to maximize donation profits. The details about the dataset could be found here.

The details about the columns included in the data set could be found here

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.2

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
pf <- read.csv('cup98LRN.txt', header = TRUE, stringsAsFactors = FALSE)
```

```
## Warning in scan(file = file, what = what, sep = sep, quote = quote, dec =
## dec, : embedded nul(s) found in input
```

## Data analysis

Each record show one of the doner. The unique identity key is the column of **CONTROLN**. Each doner is contacted though mail. There are events organized by the organization. On those events the state of doner is recorded. In total there are 23 events for each doner. The dates for those events is recorded in columnts ADATE_2, … ADATE_24, in format of YYMM. The state of doner is recorded in columns RFA_2, . . . , RFA_24.

Based on the donations from the doner the organization have divided the state into 7 states. The description about the states is described below

- **FIRST TIME DONOR**: Anyone who has made their first donation in the last 6 months and has made just one donation.
- **NEW DONOR**: Anyone who has made their first donation in the last 12 months and is not a First time donor. This is everyone who made their first donation 7-12 months ago, or people who made their first donation between 0-6 months ago and have made 2 or more donations.
- **ACTIVE DONOR**: Anyone who made their first donation more than 12 months ago and has made a donation in the last 12 months.
- **LAPSING DONOR**: A previous donor who made their last donation between 13-24 months ago.
- **INACTIVE DONOR**: A previous donor who has not made a donation in the last 24 months. It is people who made a donation 25+ months ago.

- **STAR DONOR STAR**: Donors are individuals who have given to 3 consecutive card mailings.
- **null**: Means the donor has not responded or has not yet joined.

The data set is pruned to contain only the features of interest

```
prunedPf <- pf[c( 'ADATE_2', 'ADATE_3', 'ADATE_4', 'ADATE_5', 'ADATE_6', 'ADATE_7', 'ADATE_8', 'ADATE_9
                  'ADATE_11', 'ADATE_12', 'ADATE_13', 'ADATE_14', 'ADATE_15', 'ADATE_16', 'ADATE_17', '
                  'ADATE_20', 'ADATE_21', 'ADATE_23', 'ADATE_24',
                  'RFA_2', 'RFA_3', 'RFA_4', 'RFA_5', 'RFA_6', 'RFA_7', 'RFA_8', 'RFA_9', 'RFA_10', 'RF
                  'RFA_13', 'RFA_14', 'RFA_15', 'RFA_16', 'RFA_17', 'RFA_18', 'RFA_19', 'RFA_20', 'RFA_
                  'RFA_24' , 'CONTROLN', 'TARGET_B', 'TARGET_D', 'HPHONE_D')]
```

## State Transformation

The above defined transformations could be converted into algorithm friendly states. The following mapping is used to transform the states

- Inactive doner || ” (Dont Care) 0
- LAPSING DONOR || FIRST TIME DONOR || NEW DONOR || ACTIVE USER (Interested) 1
- STAR DONOR STAR (Very Interested) 2

```
convertRFAToState <- function (rfa) {
  if(is.na(rfa) || is.null(rfa)) {
    return(0)
  } else {
    letter = substr(rfa, 1, 1)
    if(letter == 'I' || letter == 'L') {
      return(0)
    }
    if(letter == 'F' || letter == 'N' || letter == 'A') {
      return(1)
    }
    if(letter == 'S') {
      return(2)
    }
  }
  return(0)
}

getColName <- function(ind) {
  return(paste('RFA_', as.character(ind) , sep=''))
}
```

All the dataset should be traversed throught to convert the states. The following code does that

```
convertRecords <- function(df) {
  for(i in 1:nrow(df)) {
    if(i%%5000 == 0) {
      print(paste('percentage completed',as.character(i/nrow(df))));
    }
    for(j in 2:24) {
      df[i,getColName(j)] = convertRFAToState(df[i,getColName(j)])
    }
  }
  return(df)
}
```

The dataframe is created which contains the transformed states

```
prunedCopy <- data.frame(prunedPf, stringsAsFactors=FALSE)
transforedPrune <- readRDS('transforedPrune.rds')
#transforedPrune <- convertRecords(prunedCopy)
#saveRDS(transforedPrune, "transforedPrune.rds")
```

## State Transition calculation

After the states are defined the transition between the states should be calculated. The following snippet define the transition and traverse the data to calculate transitions for each user. The code snippet below is used to calculate the transition probabilities for each user

```
getTransitionType <- function (past, current) {
    if (past == 0 && current == 0) {
      return('dctodc')
    }
    if(past == 0 && current == 1) {
      return('dctoi')
    }
    if(past == 0 && current == 2) {
    return('dctovi')
    }

    if (past == 1 && current == 0) {
      return('itodc')
    }
    if(past == 1 && current == 1) {
      return('itoi')
    }
    if(past == 1 && current == 2) {
      return('itovi')
    }

    if (past == 2 && current == 0) {
      return('vitodc')
    }
    if(past == 2 && current == 1) {
      return('vitoi')
    }
    if(past == 2 && current == 2) {
      return('vitovi')
    }
}

# create a dataframe with transition probabilities
transitionDataFrame <- function(df) {
  transitionDF <- data.frame('id'= 0, 'TARGET_B' = 0, 'TARGET_D' = 0,'dctodc' = 0, 'dctoi' = 0, 'dctovi
  for(i in 1:nrow(df)) {
 # for(i in 1:10) {
    if(i%%5000 == 0) {
      print(paste('percentage completed',as.character(i/nrow(df))));
    }
    transitionDF<- rbind(transitionDF, data.frame('id'= df[i,'CONTROLN'], 'TARGET_B' = df[i,'TARGET_B']
    currentState = df[i,'RFA_24']
    for(j in 23:2) {
      transitionDF[i +1, getTransitionType(currentState, df[i,getColName(j)])] = transitionDF[i +1, get
      currentState = df[i,getColName(j)]
    }
    transitionDF[i+1, c('dctodc', 'dctoi', 'dctovi', 'itodc', 'itoi', 'itovi', 'vitodc', 'vitoi', 'vito
  }
  return(transitionDF)
}
```

New data frame is created which contains the transition probabilities for each user

```
transitionProb <- readRDS('transitionProb.rds')
#transitionProb <- transitionDataFrame(transforedPrune)
#transitionProb <- transitionCounts[-1,]
#saveRDS(transitionProb, "transitionProb.rds")
positiveRes <- subset(transitionProb, TARGET_B == 1)
negRes <- subset(transitionProb, TARGET_B != 1)

colMeans(positiveRes[, c('dctodc', 'dctoi', 'dctovi', 'itodc', 'itoi', 'itovi', 'vitodc', 'vitoi', 'vit
```

```
##       dctodc        dctoi       dctovi        itodc         itoi
## 0.6168790945 0.0260357029 0.0299401198 0.0299682766 0.0766523380
##        itovi       vitodc        vitoi       vitovi
## 0.0007696206 0.0386969009 0.0003284966 0.1807294502
```

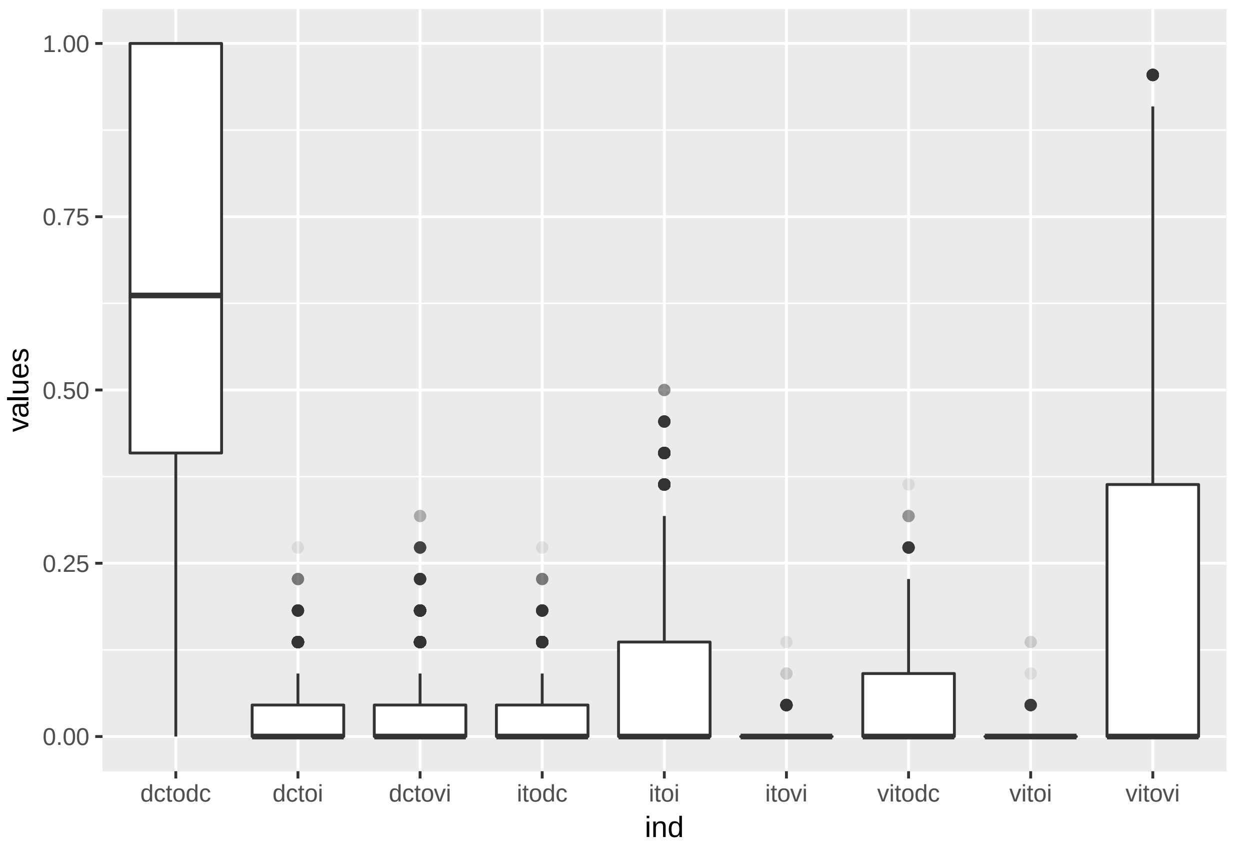
```
colMeans(negRes[, c('dctodc', 'dctoi', 'dctovi', 'itodc', 'itoi', 'itovi', 'vitodc', 'vitoi', 'vitovi')]
```

```
##       dctodc        dctoi       dctovi        itodc         itoi
## 0.6607942626 0.0299411806 0.0203704819 0.0342622984 0.0865041605
##        itovi       vitodc        vitoi       vitovi
## 0.0005520592 0.0274679555 0.0001726440 0.1399239162
```

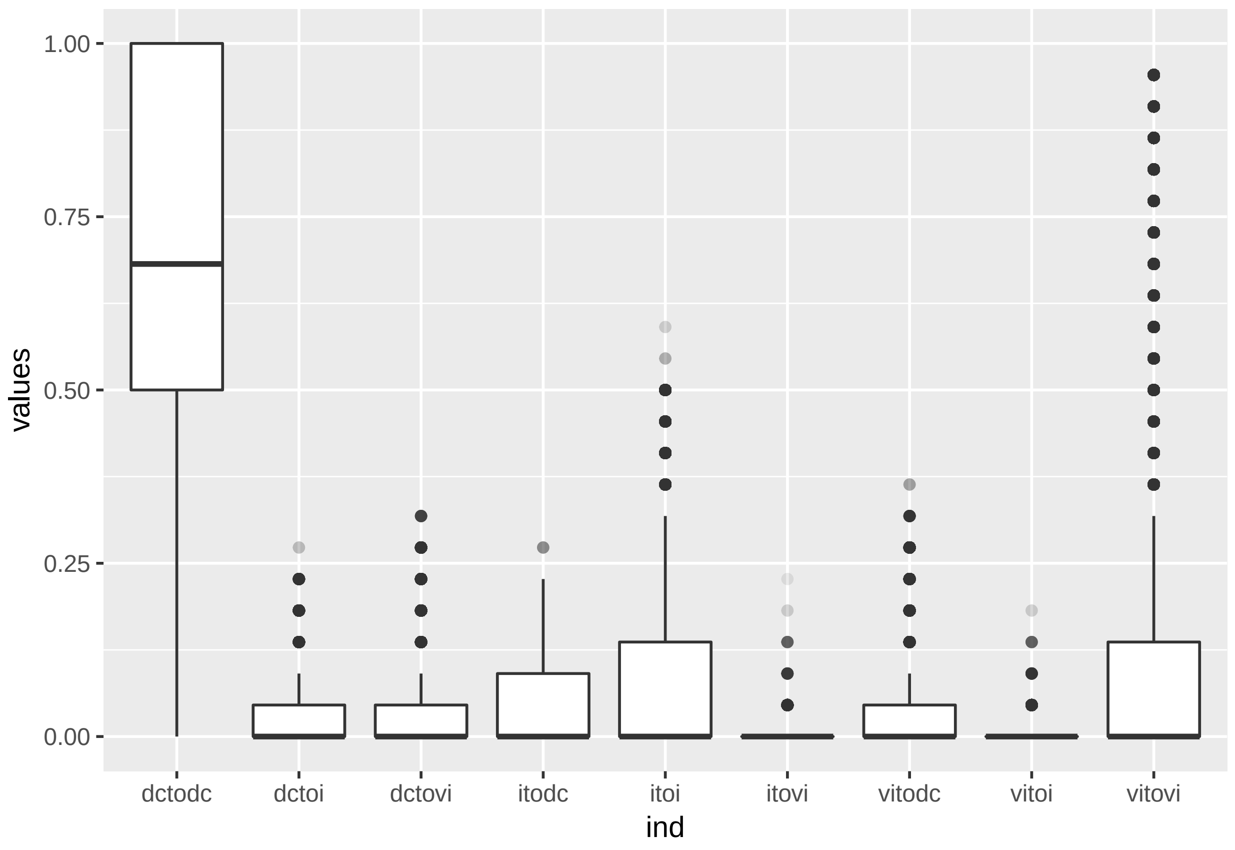
Plots and transition analysis

```
library(ggplot2)

ggplot(stack(positiveRes[-(1:3)]), aes(x = ind, y = values)) +
  geom_boxplot(aes(group = cut_width(ind, 0.25)), outlier.alpha = 0.1)
```

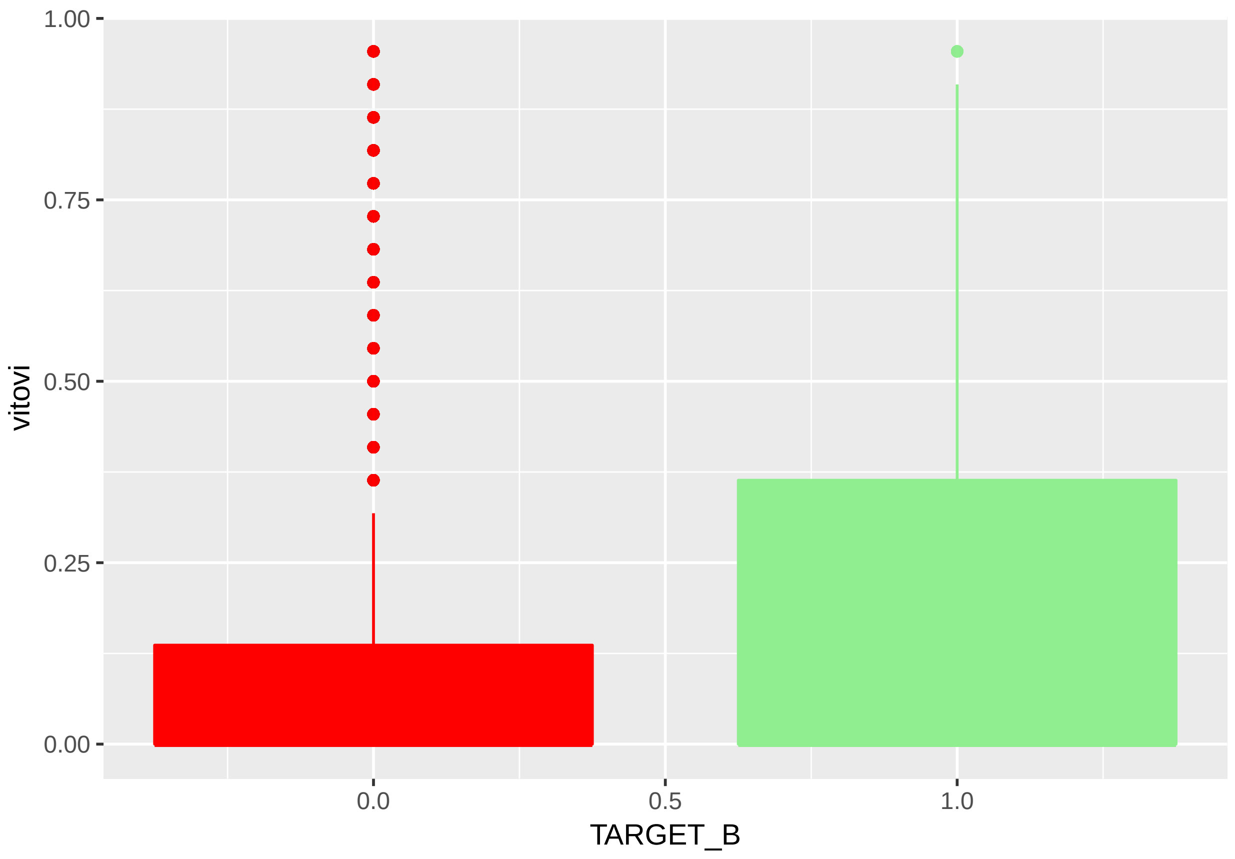


```r
ggplot(stack(negRes[-(1:3)]), aes(x = ind, y = values)) +
  geom_boxplot(aes(group = cut_width(ind, 0.25)), outlier.alpha = 0.1)
```



There is clear distinction between in the distribution of **vi-to-vi** for positive and negative response. This supports our state model. Furhter data analysis is performed on these two data sets

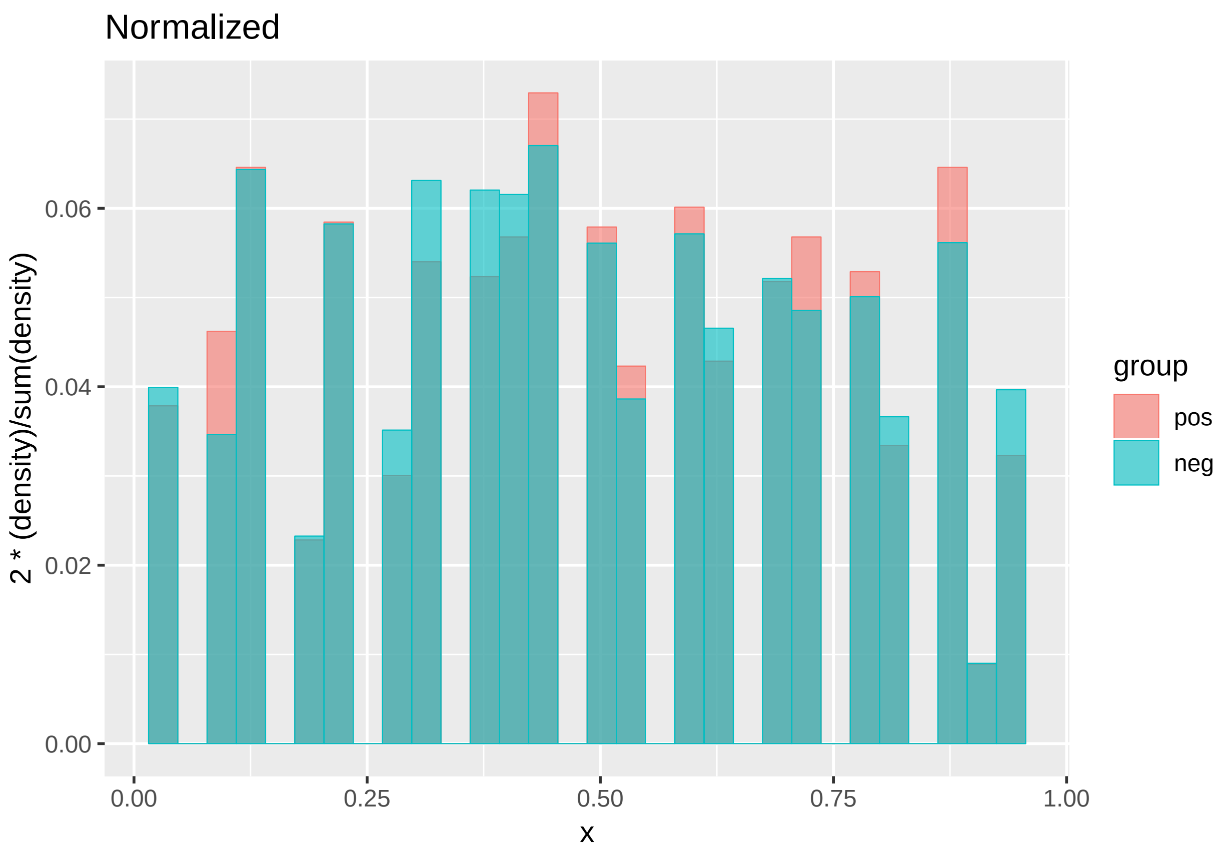
```
ggplot(transitionProb, aes(x = TARGET_B, y = vitovi)) +
  geom_boxplot(aes(group = cut_width(TARGET_B, 0.25)), col=(c("red","lightgreen")), fill=(c("red","ligh
```



```r
dat1 = data.frame(x=subset(positiveRes, vitovi != 0)$vitovi, group="pos")
dat2 = data.frame(x=subset(negRes, vitovi != 0)$vitovi, group="neg")
dat = rbind(dat1, dat2)

ggplot(dat, aes(x, fill=group, colour=group)) +
  geom_histogram(aes(y=2*(..density..)/sum(..density..)), alpha=0.6,
                 position="identity", lwd=0.2) +
  ggtitle("Normalized")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


Normalized
2 * (density)/sum(density)
0.00
0.02
0.04
0.06
0.00
0.25
0.50
0.75
1.00
x
group
pos
neg