ANEXO 7 – SCRIPTS Y CÓDIGO

En este anexo se incluyen algunos de los programas y scripts en R utilizados para realizar las diferentes fases del trabajo.

Contenido

A7-1. PRIMERA SELECCIÓN DE CAMPOS

A partir de a BBDD intermedia MIMICSEL se analizan los campos y se realizan las transformaciones y el procesado de los datos para utilizarlos en los diferentes modelos.

load_format_dataset_1.Rmd

````
---
title: "Load-Format dataset 1"
output: html_notebook
---

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.
#libraries
```{r}
require("RPostgreSQL")
require("caret")
require("dplyr")
require("pROC")
```

#connect database
```{r}
drv <- dbDriver("PostgreSQL")
# creates a connection to the postgres database
# note that "con" will be used later in each connection to the database
con <- dbConnect(drv, dbname = "mimicsel",
                 host = "192.168.1.106", port = 5432,
                 user = "mimicsel", password = "mimicsel")


# check for the cartable
dbExistsTable(con, "ds_icu_stays_1")
# TRUE
```

#datatable from table


```{r}
ds_icu_2m<-dbGetQuery(con,"select
PAT_GENDER,PAT_EXPIRE_HOSP,ADT_HOS_AGE,ADT_HSTAYS,ADT_READMIT,ICU_DBSOURCE,IC
U_LOS,ICU_TYPE_FSERVICE,ICU_DEAD_BEFORE_28,ICU_HAS_PRESC_INSULIN,
ICU_HAS_ADMIV_INSULIN,ICU_HAS_ADMIT_INSULIN,ICU_STAY_MORE72,ICD9P_M,ICD9D_M,I
CD9D_M_CHAPTER,DIAG_M_HAS_DIABET,DIAG_S_HAS_DIABET,COM_CONGEST_HEA_FAILURE
,COM_CARDIAC_ARRHYTHMIAS ,
COM_VALVULA_DISEASE   ,COM_PULMONA_CIR_DISORDERS   ,COM_PERIPHE_VAS_DISORDERS
,COM_HYPERTENSION          ,COM_PARALYSIS          ,COM_OTHER_NEUR_DISORDERS
,COM_CHRONIC_PUL_DISEASE ,
COM_DIABETES_UNCOMPLICATED    ,COM_DIABETES_COMPLICATED    ,COM_HYPOTHYROIDISM
,COM_RENAL_FAILURE ,COM_LIVER_DISEASE ,COM_PEPTIC_ULC_DISEASE ,COM_AIDS ,
````
````

```
COM_LYMPHOMA        ,COM_METASTATIC_CANCER        ,COM_SOLID_TUM_WO_METASTASIS
,COM_RHEUMATOID_ART_DISEASES  ,COM_COAGULOPATHY  ,COM_OBESITY  ,COM_WEIGHT_LOSS
,COM_FLUID_ELEC_DISORDERS ,
COM_BLOOD_LOS_ANEMIA        ,COM_DEFICIENCY_ANEMIAS        ,COM_ALCOHOL_ABUSE
,COM_DRUG_ABUSE ,COM_PSYCHOSES ,COM_DEPRESSION ,SCORE_APSIII,SCORE_LODS,
SCORE_MLODS,SCORE_OASIS,SCORE_QSOFA,SCORE_SAPS,SCORE_SAPSII,SCORE_SOFA,SCORE_
SIRS,MED_HAS_ALLERGIE_INSULIN,MED_HAS_INFECTION,MED_HAS_EXPLICIT_SEPSIS,
MED_HAS_ORGAN_DYSFUNCTION,MED_HAS_SEPSIS,MED_HAS_VASO,MED_HAS_24H_VASO,MED_HA
S_DIALYSIS,MED_HAS_24H_DIALYSIS,MED_HAS_MECHVENT,MED_HAS_24H_MECHVENT,
MED_HAS_ESTEROID_PRESC,PH_AVG_24H,PH_STD_24H,PO2_AVG_24H,PO2_STD_24H,WHITEBLO
OD_AVG_24H,WHITEBLOOD_STD_24H,LACTATE_AVG_24H,LACTATE_STD_24H,
HEMOGLOBIN_AVG_24H,HEMOGLOBIN_STD_24H,PCO2_AVG_24H,PCO2_STD_24H,BICARBONATE_A
VG_24H,BICARBONATE_STD_24H,CREATININE_AVG_24H,CREATININE_STD_24H,PH_AVG_48H,
PH_STD_48H,PO2_AVG_48H,PO2_STD_48H,WHITEBLOOD_AVG_48H,WHITEBLOOD_STD_48H,LACT
ATE_AVG_48H,LACTATE_STD_48H,HEMOGLOBIN_AVG_48H,HEMOGLOBIN_STD_48H,PCO2_AVG_48
H,
PCO2_STD_48H,BICARBONATE_AVG_48H,BICARBONATE_STD_48H,CREATININE_AVG_48H,CREAT
ININE_STD_48H,PH_AVG_72H,PH_STD_72H,PO2_AVG_72H,PO2_STD_72H,WHITEBLOOD_AVG_72
H,
WHITEBLOOD_STD_72H,LACTATE_AVG_72H,LACTATE_STD_72H,HEMOGLOBIN_AVG_72H,HEMOGLO
BIN_STD_72H,PCO2_AVG_72H,PCO2_STD_72H,BICARBONATE_AVG_72H,BICARBONATE_STD_72H
,
CREATININE_STD_72H,CREATININE_AVG_72H,GLUCOSE_VARIABILITY_ICU,TIME_HIPO_CRITI
CAL_SUM_24H,TIME_HIPO_MODERATE_SUM_24H,TIME_HYPER_GLUCE_SUM_24H,TIME_RANGE_PR
OTOCOL_SUM_24H,
TIME_RANGE_CLINIC_SUM_24H,GLUCOSE_VARIABILITY_24H,TIME_HIPO_CRITICAL_SUM_48H,
TIME_HIPO_MODERATE_SUM_48H,TIME_HYPER_GLUCE_SUM_48H,TIME_RANGE_PROTOCOL_SUM_4
8H,
TIME_RANGE_CLINIC_SUM_48H,GLUCOSE_VARIABILITY_48H,TIME_HIPO_CRITICAL_SUM_72H,
TIME_HIPO_MODERATE_SUM_72H,TIME_HYPER_GLUCE_SUM_72H,TIME_RANGE_PROTOCOL_SUM_7
2H,
TIME_RANGE_CLINIC_SUM_72H,
HIPO_CRITICAL_PERCENT_ICU,HIPO_MODERATE_PERCENT_ICU,HYPER_GLUCE_PERCENT_ICU,I
N_RANGE_CLINIC_PERCENT_ICU,IN_RANGE_PROTOCOL_PERCENT_ICU,
HIPO_CRITICAL_PERCENT_24H,HIPO_MODERATE_PERCENT_24H,HYPER_GLUCE_PERCENT_24H,I
N_RANGE_CLINIC_PERCENT_24H,IN_RANGE_PROTOCOL_PERCENT_24H,
HIPO_CRITICAL_PERCENT_48H,HIPO_MODERATE_PERCENT_48H,HYPER_GLUCE_PERCENT_48H,I
N_RANGE_CLINIC_PERCENT_48H,IN_RANGE_PROTOCOL_PERCENT_48H,
HIPO_CRITICAL_PERCENT_72H,HIPO_MODERATE_PERCENT_72H,HYPER_GLUCE_PERCENT_72H,I
N_RANGE_CLINIC_PERCENT_72H,IN_RANGE_PROTOCOL_PERCENT_72H,
GLUCOSE_VARIABILITY_72H,NUT_TPN_IN_24H,NUT_NPO_IN_24H,NUT_PPN_IN_24H,
NUT_TFE_IN_24H,NUT_TPN_IN_48H,NUT_NPO_IN_48H,NUT_PPN_IN_48H,NUT_TFE_IN_48H,NU
T_TPN_IN_72H,NUT_NPO_IN_72H,NUT_PPN_IN_72H,NUT_TFE_IN_72H
from
ds_icu_stays_1")
```


20445 of 168 variables

#first we converted the logical values to 1=TRUE / 0=FALSE
```{r}
ds_icu_2m[,sapply(ds_icu_2m,is.logical)]<-
ds_icu_2m[,sapply(ds_icu_2m,is.logical)]+0

```


#PAT_GENDER to 0,1 FEMALE->1, MALE->0

```{r}
ds_icu_2m$pat_gender=ds_icu_2m$pat_gender=="F"
ds_icu_2m$pat_gender=ds_icu_2m$pat_gender+0

tbl1<-table(ds_icu_2m$pat_gender)
prop.table(tbl1)
addmargins(tbl1)
```

#PAT_EXPIRE_HOSP

```{r}
tbl1<-table(ds_icu_2m$pat_expire_hosp)
prop.table(tbl1)
addmargins(tbl1)
```

```{r}
#delete column
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("pat_expire_hosp"))]
```

#ADT_HOS_AGE

```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$adt_hos_age))+geom_histogram(alpha=0.4,
binwidth = 10,fill="blue")+
  ggtitle("Age distribution")+
  xlab("Age")+
  theme(plot.title = element_text(hjust = 0.5))


#nulos
sum(is.na(ds_icu_2m$pat_gender))
#summary
summary(ds_icu_2m$adt_hos_age)
```

```{r}
# asign to decades

ds_icu_2m$adt_hos_age<-
cut(ds_icu_2m$adt_hos_age,breaks=seq(0,100,10),labels=seq(0,9))

```

#ADT_HSTAYS
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$adt_hstays))+geom_histogram(alpha=0.4,b
inwidth = 1,fill="blue")

#nulos
sum(is.na(ds_icu_2m$adt_hstays))
#summary
summary(ds_icu_2m$adt_hstays)
```

#ADT_READMIT

```{r}
#nulos
sum(is.na(ds_icu_2m$adt_readmit))
#summary
summary(ds_icu_2m$adt_readmit)

tbl1<-table(ds_icu_2m$adt_readmit)
prop.table(tbl1)
addmargins(tbl1)
```
```{r}
#delete column
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("adt_readmit"))]
```

#ICU_DBSOURCE
```{r}
ds_icu_2m$icu_dbsource<-as.factor(ds_icu_2m$icu_dbsource)

summary(ds_icu_2m$icu_dbsource)
```

#ICU_LOS
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$icu_los))+geom_histogram(alpha=0.4,binw
idth = 1,fill="blue")+
  ggtitle("ICU LOS")+
  xlab("days")+
  theme(plot.title = element_text(hjust = 0.5))


#nulos
sum(is.na(ds_icu_2m$icu_los))
#summary
summary(ds_icu_2m$icu_los)
```

#ICU_TYPE_FSERVICE
```{r}
ds_icu_2m$icu_type_fservice<-as.factor(ds_icu_2m$icu_type_fservice)

summary(ds_icu_2m$icu_type_fservice)


```

```{r}
#IMPUTATION + REFACTOR
ds_icu_2m$icu_type_fservice[which(ds_icu_2m$icu_type_fservice=="-")]<-
"MEDICAL"
ds_icu_2m$icu_type_fservice<-droplevels(ds_icu_2m$icu_type_fservice)
summary(ds_icu_2m$icu_type_fservice)
```

#ICU_DEAD_BEFORE_28
```{r}
#nulos
sum(is.na(ds_icu_2m$icu_dead_before_28))
```

```
#summary

tbl1<-table(ds_icu_2m$icu_dead_before_28)
prop.table(tbl1)
addmargins(tbl1)
```

#ICU_HAS_PRESC_INSULIN
```{r}
#nulos
sum(is.na(ds_icu_2m$icu_has_presc_insulin))
#summary

tbl1<-table(ds_icu_2m$icu_has_presc_insulin)
prop.table(tbl1)
addmargins(tbl1)
```

#ICU_HAS_ADMIV_INSULIN
```{r}
#nulos
sum(is.na(ds_icu_2m$icu_has_admiv_insulin))
#summary

tbl1<-table(ds_icu_2m$icu_has_admiv_insulin)
prop.table(tbl1)
addmargins(tbl1)
```

#ICU_HAS_ADMIT_INSULIN
```{r}
#nulos
sum(is.na(ds_icu_2m$icu_has_admit_insulin))
#summary

tbl1<-table(ds_icu_2m$icu_has_admit_insulin)
prop.table(tbl1)
addmargins(tbl1)
```
#MED_HAS_ALLERGIE_INSULIN

```{r}
nearZeroVar(ds_icu_2m$med_has_allergie_insulin)
```
```{r}
#delete column
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in%
c("med_has_allergie_insulin"))]
```

#ICD9P_M
```{r}

#FACTOR
ds_icu_2m$icd9p_m<-as.factor(as.character(ds_icu_2m$icd9p_m))
#summary


```

```
```{r}
#delete column
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("icd9p_m"))]
```

#ICD9D_M
```{r}

#FACTOR
ds_icu_2m$icd9d_m<-as.factor(as.character(ds_icu_2m$icd9d_m))
#summary
```


#DIAG_M_HAS_DIABET / DIAG_S_HAS_DIABET
```{r}

#combine columns OR

ds_icu_2m$diag_has_diabet<-(ds_icu_2m$diag_m_has_diabet                    |
ds_icu_2m$diag_s_has_diabet)
ds_icu_2m$diag_has_diabet<-ds_icu_2m$diag_has_diabet+0
```

```{r}
#delete column
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m)                              %in%
c("diag_s_has_diabet","diag_m_has_diabet"))]
```

```{r}
tbl1<-table(ds_icu_2m$diag_has_diabet)
prop.table(tbl1)
addmargins(tbl1)
```
#COM_
```{r}

#nulos

df_com<-ds_icu_2m %>% dplyr::select(starts_with("COM_"))

sum(is.na(df_com))

```


#near zero var
```{r}
nzv<-nearZeroVar(df_com)

(czv<-names(df_com[nzv]))

ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% czv)]
```

```{r}
rm(df_com)
```
```

```

#SCORES

#SCORE_APSIII
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$score_apsiii))+geom_density(alpha=0.4,f
ill="blue")

#nulos
sum(is.na(ds_icu_2m$score_apsiii))
#summary
summary(ds_icu_2m$score_apsiii)
```

```{r}
g1<-glm(ds_icu_2m$icu_dead_before_28~ds_icu_2m$score_apsiii,family          =
binomial)
(r1<-roc(ds_icu_2m$icu_dead_before_28,predict(g1,type = "response")))
plot(r1)
```

```{r}
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("score_apsiii"))]
```


#SCORE_LODS
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$score_lods))+geom_histogram(alpha=0.4,b
inwidth = 1,fill="blue")

#nulos
sum(is.na(ds_icu_2m$score_lods))
#summary
summary(ds_icu_2m$score_lods)
```

```{r}
g1<-glm(ds_icu_2m$icu_dead_before_28~ds_icu_2m$score_lods,family = binomial)
(r1<-roc(ds_icu_2m$icu_dead_before_28,predict(g1,type = "response")))
plot(r1)
```


```{r}
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("score_lods"))]
```


#SCORE_MLODS
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$score_mlods))+geom_histogram(alpha=0.4,
binwidth = 1,fill="blue")

#nulos
sum(is.na(ds_icu_2m$score_mlods))
#summary
summary(ds_icu_2m$score_mlods)
```
```

```

```{r}
g1<-glm(ds_icu_2m$icu_dead_before_28~ds_icu_2m$score_mlods,family = binomial)
(r1<-roc(ds_icu_2m$icu_dead_before_28,predict(g1,type = "response")))
plot(r1)
```

```{r}
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("score_mlods"))]
```

#SCORE_OASIS
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$score_oasis))+geom_density(alpha=0.4,fi
ll="blue")

#nulos
sum(is.na(ds_icu_2m$score_oasis))
#summary
summary(ds_icu_2m$score_oasis)
```

```{r}
g1<-glm(ds_icu_2m$icu_dead_before_28~ds_icu_2m$score_oasis,family = binomial)
(r1<-roc(ds_icu_2m$icu_dead_before_28,predict(g1,type = "response")))
plot(r1)
```

```{r}
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("score_oasis"))]
```

#SCORE_QSOFA
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$score_qsofa))+geom_histogram(binwidth =
1,alpha=0.4,fill="blue")

#nulos
sum(is.na(ds_icu_2m$score_qsofa))
#summary
summary(ds_icu_2m$score_qsofa)
```

```{r}
g1<-glm(ds_icu_2m$icu_dead_before_28~ds_icu_2m$score_qsofa,family = binomial)
(r1<-roc(ds_icu_2m$icu_dead_before_28,predict(g1,type = "response")))
plot(r1)
```

```{r}
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("score_qsofa"))]
```

#SCORE_SAPS
```

```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$score_saps))+geom_histogram(binwidth  =
1,alpha=0.4,fill="blue")

#nulos
sum(is.na(ds_icu_2m$score_saps))
#summary
summary(ds_icu_2m$score_saps)
```
```{r}
g1<-glm(ds_icu_2m$icu_dead_before_28~ds_icu_2m$score_saps,family = binomial)
(r1<-roc(ds_icu_2m$icu_dead_before_28,predict(g1,type = "response")))
plot(r1)
```

```{r}
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("score_saps"))]
```


#SCORE_SAPSII
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$score_sapsii))+geom_density(alpha=0.4,f
ill="blue")+
  ggtitle("SAPSII distribution")+
  xlab("Value")+
  theme(plot.title = element_text(hjust = 0.5))

#nulos
sum(is.na(ds_icu_2m$score_sapsii))
#summary
summary(ds_icu_2m$score_sapsii)
```
```{r}
g1<-glm(ds_icu_2m$icu_dead_before_28~ds_icu_2m$score_sapsii,family          =
binomial)
(r1<-roc(ds_icu_2m$icu_dead_before_28,predict(g1,type = "response")))
plot(r1)
```


#SCORE_SOFA
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$score_sofa))+geom_histogram(binwidth=1,
alpha=0.4,fill="blue")+
  ggtitle("SOFA distribution")+
  xlab("Value")+
  theme(plot.title = element_text(hjust = 0.5))



#nulos
sum(is.na(ds_icu_2m$score_sofa))
#summary
summary(ds_icu_2m$score_sofa)
```

```{r}
g1<-glm(ds_icu_2m$icu_dead_before_28~ds_icu_2m$score_sofa,family = binomial)
(r1<-roc(ds_icu_2m$icu_dead_before_28,predict(g1,type = "response")))
plot(r1)
```

#SCORE_SIRS
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$score_sirs))+geom_histogram(binwidth=1,
alpha=0.4,fill="blue")

#nulos
sum(is.na(ds_icu_2m$score_sirs))
#summary
summary(ds_icu_2m$score_sirs)
```

```{r}
g1<-glm(ds_icu_2m$icu_dead_before_28~ds_icu_2m$score_sirs,family = binomial)
(r1<-roc(ds_icu_2m$icu_dead_before_28,predict(g1,type = "response")))
plot(r1)
```

```{r}
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("score_sirs"))]
```

#MED_
```{r}

#nulos

df_com<-ds_icu_2m %>% dplyr::select(starts_with("MED_"))

sum(is.na(df_com))

#near zero var
nzv<-nearZeroVar(df_com)
```

```{r}
colSums(df_com[nzv])
```

#AVG

```{r}
avgl<-names(ds_icu_2m[grep("_avg_24",names(ds_icu_2m))])
colSums(is.na(ds_icu_2m[avgl]))/204.45
```

```{r}
avgl<-names(ds_icu_2m[grep("_avg_48",names(ds_icu_2m))])
```

```
colSums(is.na(ds_icu_2m[avgl]))/204.45
```

```{r}
avgl<-names(ds_icu_2m[grep("_avg_72",names(ds_icu_2m))])
colSums(is.na(ds_icu_2m[avgl]))/204.45
```

```{r}
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in%
c("ph_avg_48h","po2_avg_48h","lactate_avg_48h","pco2_avg_48h","ph_avg_72h",
"po2_avg_72h","lactate_avg_72h","pco2_avg_72h","ph_std_48h","po2_std_48h","la
ctate_std_48h","pco2_std_48h","ph_std_72h",
"po2_std_72h","lactate_std_72h","pco2_std_72h"))]
```

```{r}
avgl<-names(ds_icu_2m[grep("_avg_",names(ds_icu_2m))])
par(mar=c(10,1,1,1))
boxplot(ds_icu_2m[avgl],las=2,cex.axis=1,cex=.4,cex.main=1,main="Clinic
Variables (AVG)", col="grey", outcol="red",pch=20)
```

```{r}
#OUTLIERS WHITEBLOOD
ds_icu_2m$whiteblood_avg_24h[ds_icu_2m$whiteblood_avg_24h>200]<-NA
ds_icu_2m$whiteblood_avg_48h[ds_icu_2m$whiteblood_avg_48h>200]<-NA
ds_icu_2m$whiteblood_avg_48h[ds_icu_2m$whiteblood_avg_48h>200]<-NA
```

#STD

```{r}
stdl<-names(ds_icu_2m[grep("_std_",names(ds_icu_2m))])
boxplot(ds_icu_2m[stdl])
```

#DELETE STD
```{r}
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% stdl)]
```

#GLUCOSE_VARIABILITY_ICU
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$glucose_variability_icu))+geom_density(
alpha=0.4,fill="blue")
```
```{r}
summary(ds_icu_2m$glucose_variability_icu)
sum(is.na(ds_icu_2m$glucose_variability_icu))
ds_icu_2m$glucose_variability_icu[ds_icu_2m$glucose_variability_icu>500]
```

#GLUCOSE_VARIABILITY_24H

```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$glucose_variability_24h))+geom_density(
alpha=0.4,fill="blue")+
  ggtitle("Glucose Variability 24H distribution")+
  xlab("Value")+
  theme(plot.title = element_text(hjust = 0.5))
```

```{r}
summary(ds_icu_2m$glucose_variability_24h)
sum(is.na(ds_icu_2m$glucose_variability_24h))

ds_icu_2m$glucose_variability_icu[ds_icu_2m$glucose_variability_24h>300]
```

#GLUCOSE_VARIABILITY_48H
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$glucose_variability_48h))+geom_density(
alpha=0.4,fill="blue")+
  ggtitle("Glucose Variability 48H distribution")+
  xlab("Value")+
  theme(plot.title = element_text(hjust = 0.5))
```

```{r}
summary(ds_icu_2m$glucose_variability_48h)
sum(is.na(ds_icu_2m$glucose_variability_48h))

ds_icu_2m$glucose_variability_icu[ds_icu_2m$glucose_variability_48h>300]
```

#GLUCOSE_VARIABILITY_72H
```{r}
ggplot(data=ds_icu_2m,aes(x=ds_icu_2m$glucose_variability_72h))+geom_density(
alpha=0.4,fill="blue")
```

```{r}
summary(ds_icu_2m$glucose_variability_72h)
sum(is.na(ds_icu_2m$glucose_variability_72h))

ds_icu_2m$glucose_variability_icu[ds_icu_2m$glucose_variability_72hh>300]
```

#TIME_HIPOCRITICAL_SUM

```{r}

timel<-names(ds_icu_2m[grep("time_hipo",names(ds_icu_2m))])

boxplot(ds_icu_2m[timel])

summary(ds_icu_2m[timel])


```

#TIME_HYPER

```{r}

timel<-names(ds_icu_2m[grep("time_hyper",names(ds_icu_2m))])

boxplot(ds_icu_2m[timel])

summary(ds_icu_2m[timel])


```


#TIME_RANGE

```{r}

timel<-names(ds_icu_2m[grep("time_range",names(ds_icu_2m))])

boxplot(ds_icu_2m[timel])

summary(ds_icu_2m[timel])


```

#DELETE TIME_

```{r}
#finally didnt include any TIME_

timel<-names(ds_icu_2m[grep("time_",names(ds_icu_2m))])

ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% timel)]
```


#NUT_NPO
#DELETE

```{r}
npol<-names(ds_icu_2m[grep("nut_npo",names(ds_icu_2m))])

ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% npol)]
```


#NUT_TPN

```{r}

tpnl<-names(ds_icu_2m[grep("nut_tpn",names(ds_icu_2m))])

summary(ds_icu_2m[tpnl])

```
```{r}
ds_icu_2m[tpnl][is.na(ds_icu_2m[tpnl])]<-0
summary(ds_icu_2m[tpnl])
```

```
nearZeroVar(ds_icu_2m[tpnl])

colSums(ds_icu_2m[tpnl])

```
#DELETE TPN
```{r}
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% tpnl)]
```

#NUT_PPN

```{r}

ppnl<-names(ds_icu_2m[grep("nut_ppn",names(ds_icu_2m))])

summary(ds_icu_2m[ppnl])

nearZeroVar(ds_icu_2m[ppnl])

colSums(ds_icu_2m[ppnl],na.rm = TRUE)
```

#DELETE PPN
```{r}
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% ppnl)]
```

#NUT_TFE


```{r}

tfel<-names(ds_icu_2m[grep("nut_tfe",names(ds_icu_2m))])

summary(ds_icu_2m[tfel])

nearZeroVar(ds_icu_2m[tfel])

colSums(ds_icu_2m[tfel],na.rm = TRUE)
```


```{r}
#IMPUTATION

ds_icu_2m[tfel][is.na(ds_icu_2m[tfel])]<-0
summary(ds_icu_2m[tfel])

```


#HIPO_CRITICAL

```{r}
hipol<-names(ds_icu_2m[grep("hipo_",names(ds_icu_2m))])

boxplot(ds_icu_2m[hipol])
```

```
summary(ds_icu_2m[hipol])

nearZeroVar(ds_icu_2m[hipol])

```

#DELETE HIPO
```{r}
#ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% hipol)]
```


```{r}
par(mar=c(14,2.3,2.3,2))
boxplot(ds_icu_2m[hipol],las=2,cex.axis=1,cex=.4,cex.main=1,main="Hipoglucemi
a - %", col="grey", outcol="red",pch=20)

```


#HYPER

```{r}
hypel<-names(ds_icu_2m[grep("hyper_",names(ds_icu_2m))])

boxplot(ds_icu_2m[hypel])

summary(ds_icu_2m[hypel])

nearZeroVar(ds_icu_2m[hypel])

```

#remove outliers
```{r}
ds_icu_2m$hyper_gluce_percent_72h[ds_icu_2m$hyper_gluce_percent_72h<0]<-NA
ds_icu_2m$hyper_gluce_percent_72h[ds_icu_2m$hyper_gluce_percent_72h>100]<-NA
```

```{r}
par(mar=c(14,2.3,2.3,2))
boxplot(ds_icu_2m[hypel],las=2,cex.axis=1,cex=.4,cex.main=1,main="Hyperglucem
ia - %", col="grey", outcol="red",pch=20)
```


#IN_RANGE

```{r}
inrl<-names(ds_icu_2m[grep("in_range_",names(ds_icu_2m))])

boxplot(ds_icu_2m[inrl])

summary(ds_icu_2m[inrl])

nearZeroVar(ds_icu_2m[inrl])

```
#remove outliers

```{r}
ds_icu_2m$in_range_clinic_percent_72h[ds_icu_2m$in_range_clinic_percent_72h<0
]<-NA
ds_icu_2m$in_range_clinic_percent_72h[ds_icu_2m$in_range_clinic_percent_72h>1
00]<-NA
ds_icu_2m$in_range_protocol_percent_72h[ds_icu_2m$in_range_protocol_percent_7
2h<0]<-NA
ds_icu_2m$in_range_protocol_percent_72h[ds_icu_2m$in_range_protocol_percent_7
2h>100]<-NA
```


```{r}
par(mar=c(14,2.3,2.3,2))
boxplot(ds_icu_2m[inrl],las=2,cex.axis=1,cex=.4,cex.main=1,main="Glucose - In
range %", col="grey", outcol="red",pch=20)
```


#FINAL FORMAT DATASET
```{r}
#only variables obtained for <72H
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in%
names(ds_icu_2m[grep("_icu",names(ds_icu_2m))]))]

#delete icu_los and adt_hstays
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in%
c("icu_los","adt_hstays"))]

#db_sources
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("icu_dbsource"))]

#problems with factor
ds_icu_2m$adt_hos_age<-as.numeric(ds_icu_2m$adt_hos_age)

```

```{r}
#delete only score_sapii data
ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("score_sapsii"))]
```

```{r}
#auxiliar variable
ds_icu_2m$score_sofa_more4<-as.numeric(ds_icu_2m$score_sofa>=4)
```


```{r}
#group icd9_m_chapter
dicd9<-(ds_icu_2m %>% group_by(icd9d_m_chapter) %>% summarise(n=n()) %>%
arrange(n))

#refactoring as others
dicd_others<-filter(dicd9,n<1000)
ds_icu_2m$icd9d_m_chapter<-as.character(ds_icu_2m$icd9d_m_chapter)
```

```
ds_icu_2m<-ds_icu_2m  %>%  mutate(icd9d_m_chapter=ifelse(icd9d_m_chapter  %in%
dicd_others$icd9d_m_chapter,"00-Others",icd9d_m_chapter))
ds_icu_2m$icd9d_m_chapter<-as.factor(ds_icu_2m$icd9d_m_chapter)

ds_icu_2m<-ds_icu_2m[,-which(names(ds_icu_2m) %in% c("icd9d_m"))]
```

```{r}
#imputation and normalize with caret
preproc_ds_range<-preProcess(ds_icu_2m,method = c("medianImpute","range"))
ds_icu_2m<-predict(preproc_ds_range,ds_icu_2m)
```

```{r}
#factorize outcome
ds_icu_2m$icu_dead_before_28<-as.factor(ds_icu_2m$icu_dead_before_28)
levels(ds_icu_2m$icu_dead_before_28)<-
make.names(levels(ds_icu_2m$icu_dead_before_28))
```

```{r}
str(ds_icu_2m)
names(ds_icu_2m)

```


```{r}
save(file="~/ds_icu_2m",ds_icu_2m)
```


```

## A7-2.  SELECCIÓN DE MODELOS

Selección de modelos realizada en Caret

### Caret_model_tests_selection.Rmd

```
---
title: "Caret Model Selection"
output:
  html_document: default
  html_notebook: default

```{r}
require(caret)
require(doMC)
require(RANN)
require(pROC)
```


#Parallel
```{r}
```

```
registerDoMC(cores=3) # best compromise
```

#random seed
```{r}
set.seed(1000)
```


LOAD DATASET
```{r}
load(file = "~/ds_icu_2m")
```


#small dataset to begin tests
```{r}
small_sample<-createDataPartition(y=ds_icu_2m$icu_dead_before_28,p=.1,list  =
FALSE)
ds_small_sample<-ds_icu_2m[small_sample,]
```


#training and test datasets
```{r}
small_train<-
createDataPartition(y=ds_small_sample$icu_dead_before_28,p=.60,list = FALSE)
ds_small_train<-ds_small_sample[small_train,]
ds_small_test<-ds_small_sample[-small_train,]
```

#predictors and decision
```{r}
ds_predictors<-ds_small_train[!names(ds_small_train)                    %in%
c("icu_dead_before_28") ]
ds_decision<-ds_small_train$icu_dead_before_28
```

#MODELS TWO CLASSES
FAST CV
```{r}
ctrl <- trainControl(method = "cv",
  number=3,
  classProbs = TRUE,
  savePredictions = "final",
  sampling = "down",  #subsampling inside resampling,
  summaryFunction = twoClassSummary)
```


#glm
```{r}

set.seed(333)

time_train<-system.time(glmFit <- train(  x=ds_predictors, y=ds_decision,
    method = "glm",
    trControl = ctrl,
    metric = "ROC"
```

```
    ))

summary(glmFit)
glmFit

#plot(glmFit)

predClasses<-predict(glmFit,newdata=ds_small_test,type="prob")
(mr<-roc(response = ds_small_test$icu_dead_before_28,
         predictor = predClasses$X1,
         levels=levels(ds_small_test$icu_dead_before_28)))
plot(mr)


time_train
```


#ranger random forest
```{r}
set.seed(333)

time_train<-system.time(rangerFit <-   train( x=ds_predictors, y=ds_decision,
    method = "ranger",
    trControl = ctrl,
    metric = "ROC",
    tuneGrid=expand.grid(mtry=c(8,9))
    ))

summary(rangerFit)
rangerFit

plot(rangerFit)

predClasses<-predict(rangerFit,newdata=ds_small_test,type="prob")
(mr<-roc(response = ds_small_test$icu_dead_before_28,
         predictor = predClasses$X1,
         levels=levels(ds_small_test$icu_dead_before_28)))
plot(mr)

time_train
```

#regularized random forest
```{r}
set.seed(333)

time_train<-system.time(rrfFit <-   train( x=ds_predictors, y=ds_decision,
    method = "RRF",
    trControl = ctrl,
    metric = "ROC",

tuneGrid=expand.grid(mtry=c(12,13),coefReg=c(0.96,0.98,0.97,0.99),coefImp=c(0
.004,0.005))
    ))

summary(rrfFit)
rrfFit
```

```
plot(rrfFit)

predClasses<-predict(rrfFit,newdata=ds_small_test,type="prob")
(mr<-roc(response = ds_small_test$icu_dead_before_28,
         predictor = predClasses$X1,
         levels=levels(ds_small_test$icu_dead_before_28)))
plot(mr)

time_train
```

#stochastic gradient boosting
```{r}
set.seed(333)

time_train<-system.time(gbmFit <-  train( x=ds_predictors, y=ds_decision,
    method = "gbm",
    trControl = ctrl,
    metric = "ROC",
    tuneGrid=expand.grid(   interaction.depth=c(4),   #   Depth   of   variable
interactions
                    n.trees=c(300),                   # Num trees to fit
                    shrinkage=c(0.07,0.08,0.09),           # Try 2 values for
learning rate
                    n.minobsinnode = c(10,20))
    ))

summary(gbmFit)
gbmFit

plot(gbmFit)

predClasses<-predict(gbmFit,newdata=ds_small_test,type="prob")
(mr<-roc(response = ds_small_test$icu_dead_before_28,
         predictor = predClasses$X1,
         levels=levels(ds_small_test$icu_dead_before_28)))
plot(mr)

time_train
```


#Neural network
```{r}
set.seed(333)

time_train<-system.time(nnetFit <-  train( x=ds_predictors, y=ds_decision,

    method = "nnet",
    trControl = ctrl,
    metric = "ROC",
    tuneGrid=expand.grid(decay = c(1e-3,3e-3), size = c(2))
    ))

summary(nnetFit)
nnetFit

plot(nnetFit)
```

```
predClasses<-predict(nnetFit,newdata=ds_small_test,type="prob")
(mr<-roc(response = ds_small_test$icu_dead_before_28,
         predictor = predClasses$X1,
         levels=(levels(ds_small_test$icu_dead_before_28))))
plot(mr)

time_train
```

#Neural network PCA
```{r}
set.seed(333)

time_train<-system.time(nnetPCAFit <-  train( x=ds_predictors, y=ds_decision,
    method = "pcaNNet",
    trControl = ctrl,
    metric = "ROC",
    tuneGrid=expand.grid(.decay = c(7e-3,9e-3), .size = c(2,4))
    ))

summary(nnetPCAFit)
nnetPCAFit

plot(nnetPCAFit)

predClasses<-predict(nnetPCAFit,newdata=ds_small_test,type="prob")
(mr<-roc(response = ds_small_test$icu_dead_before_28,
         predictor = predClasses$X1,
         levels=levels(ds_small_test$icu_dead_before_28)))
plot(mr)

time_train
```


#Naive Bayes
```{r}
set.seed(333)

time_train<-system.time(nbFit <-  train(x=ds_predictors, y=ds_decision,
    method = "nb",
    trControl = ctrl,
    tuneGrid = data.frame(fL=0, usekernel=TRUE, adjust=c(3,4)),
    metric = "ROC"
    ))

summary(nbFit)

nbFit

plot(nbFit)

predClasses<-predict(nbFit,newdata=ds_small_test,type="prob")
(mr<-roc(response = ds_small_test$icu_dead_before_28,
         predictor = predClasses$X1,
         levels=rev(levels(ds_small_test$icu_dead_before_28))))
plot(mr)

time_train
```

```
```

#ADA (SLOW)
```{r}

set.seed(333)

time_train<-system.time(adaFit <- train( x=ds_predictors, y=ds_decision,
    method = "ada",
    trControl = ctrl,
    metric = "ROC",
    tuneGrid=expand.grid(iter=c(2000,3000),maxdepth=c(4,3),nu=c(0.006))
    ))



summary(adaFit)
adaFit

plot(adaFit)

predClasses<-predict(adaFit,newdata=ds_small_test,type="prob")
(mr<-roc(response = ds_small_test$icu_dead_before_28,
         predictor = predClasses$X1,
         levels=levels(ds_small_test$icu_dead_before_28)))


plot(mr)

time_train

```
```

A7-3. EVALUACIÓN MODELOS H2O

Selección de modelos realizada en H2O

H2o_model_tests_selection.Rmd

```
---
title: "H2O_model_tests_selection"
output:output:
  html_document: default
  html_notebook: default
---


```{r}
require(h2o)
require(caret)
require(pROC)
```


#random seed

```{r}
set.seed(1000)
```


#start h2o
```{r}
h2o.init(nthreads=-1, max_mem_size = "6G")
#h2o.removeAll()
#h2o.shutdown(prompt = FALSE)

```


LOAD DATASET
```{r}
load(file = "~/ds_icu_2m")
```


#small dataset to begin tests
```{r}
small_sample<-createDataPartition(y=ds_icu_2m$icu_dead_before_28,p=1,list   =
FALSE)
ds_small_sample<-ds_icu_2m[small_sample,]
```

#training and test datasets
```{r}
small_train<-
createDataPartition(y=ds_small_sample$icu_dead_before_28,p=.60,list = FALSE)
ds_small_train<-ds_small_sample[small_train,]
ds_small_test<-ds_small_sample[-small_train,]
```

#predictors and decision
```{r}
ds_predictors<-ds_small_train[!names(ds_small_train)                      %in%
c("icu_dead_before_28") ]
ds_decision<-ds_small_train$icu_dead_before_28

ds_test_predictors<-ds_small_test[!names(ds_small_train)                  %in%
c("icu_dead_before_28") ]
ds_test_decision<-ds_small_test$icu_dead_before_28

```


#variables
```{r}
y<-"icu_dead_before_28"
```


#gbm Gradient Boosting Machine
```{r}
time_train<-system.time(gbm_1 <- h2o.gbm(y = y,
                  training_frame = as.h2o(ds_small_train),
                  validation_frame = as.h2o(ds_small_test),
```

```
                  distribution = "bernoulli",
                  balance_classes = TRUE,
                  ntrees = 1000,
                  max_depth = 4,
                  min_rows = 2,
                  learn_rate = 0.001,
                  nfolds = 10,
                  fold_assignment = "Modulo", # for stacked
                  keep_cross_validation_predictions = TRUE,
                  keep_cross_validation_fold_assignment = TRUE,
                  seed = 1))
```

```{r}
gbm_1
plot(gbm_1)

gbm_perf<-h2o.performance(gbm_1,newdata=as.h2o(ds_small_test))
gbm_perf
plot(gbm_perf)

time_train
```
#GLM Generalized Linear Models
```{r}
time_train<-system.time(glm_1 <- h2o.glm(y = y,
                  training_frame = as.h2o(ds_small_train),
                  validation_frame = as.h2o(ds_small_test),
                  family = "binomial",
                  link="logit",
                  lambda_search = TRUE,
                  balance_classes = TRUE,
                  nfolds = 10,
                  fold_assignment = "Modulo",
                  keep_cross_validation_predictions = TRUE,
                  keep_cross_validation_fold_assignment = TRUE,
                  seed = 1))

```



```{r}
glm_1

glm_perf<-h2o.performance(glm_1,newdata=as.h2o(ds_small_test))
glm_perf
plot(glm_perf)

time_train
```

RF random forest
```{r}
time_train<-system.time(rf_1 <- h2o.randomForest(y = y,
                          training_frame = as.h2o(ds_small_train),
                          validation_frame = as.h2o(ds_small_test),
                          ntrees = 1000,
                          mtries = 8,
                          nfolds = 10,
```

```
                          balance_classes = TRUE,
                          fold_assignment = "Modulo",
                          keep_cross_validation_predictions = TRUE,
                          keep_cross_validation_fold_assignment = TRUE,
                          seed = 1))
```

```{r}
rf_1

plot(rf_1)
rf_perf<-h2o.performance(rf_1,newdata=as.h2o(ds_small_test))
rf_perf
plot(rf_perf)

time_train
```

#naive bayes
```{r}
time_train<-system.time(nb_1 <- h2o.naiveBayes(y = y,
                          training_frame = as.h2o(ds_small_train),
                          validation_frame = as.h2o(ds_small_test),
                          nfolds = 10,
                          balance_classes = FALSE, ## ERROR CON TRUE
                          fold_assignment = "Modulo",
                          keep_cross_validation_predictions = TRUE,
                          keep_cross_validation_fold_assignment = TRUE,
                          seed = 1))
```

```{r}
nb_1


nb_perf<-h2o.performance(nb_1,newdata=as.h2o(ds_small_test))
nb_perf
plot(nb_perf)

time_train
```

#deep learning
```{r}
time_train<-system.time(dl_1 <- h2o.deeplearning(y = y,
                          training_frame = as.h2o(ds_small_train),
                          validation_frame = as.h2o(ds_small_test),
                          nfolds = 10,
                          hidden = 10,
                          variable_importances = TRUE,
                          input_dropout_ratio = 0.1,
                          distribution = "bernoulli",
                          balance_classes = TRUE,
                          fold_assignment = "Modulo",
                          keep_cross_validation_predictions = TRUE,
                          keep_cross_validation_fold_assignment = TRUE,
```

```
                          seed = 1))
```


```{r}
dl_1

plot(dl_1)

dl_perf<-h2o.performance(dl_1,newdata=as.h2o(ds_small_test))
dl_perf
plot(dl_perf)

time_train
```



```{r}
time_train<-system.time(ensemble <- h2o.stackedEnsemble(y = y,
                                training_frame = as.h2o(ds_small_train),
                                validation_frame = as.h2o(ds_small_test),
                                selection_strategy = "choose_all",
                                base_models                                    =
list(gbm_1@model_id,rf_1@model_id,
                                                        nb_1@model_id,
dl_1@model_id)))
```



```{r}
ensemble


en_perf<-h2o.performance(ensemble,newdata=as.h2o(ds_small_test))
en_perf
plot(en_perf)

time_train

```

A7-4. EVALUACIÓN MODELOS MLR

Selección de modelos realizada en MLR

mlr_model_tests_selection.Rmd

```
---
title: "mlr_models_tunning"
output:
  html_document: default
  html_notebook: default
---
```

```{r}
require(ggplot2)

require(mlr)
require(doMC)
require(ROCR)

require(irace)
```


#Parallel
```{r}
registerDoMC(cores=1) # best compromise
```

#random seed
```{r}
set.seed(1000)
```


LOAD DATASET
```{r}
load(file = "~/ds_icu_2m")
```


#small stratified dataset CARET
```{r}
small_sample<-
caret::createDataPartition(y=ds_icu_2m$icu_dead_before_28,p=0.1,list = FALSE)
ds_small_sample<-ds_icu_2m[small_sample,]
```

#training and test datasets CARET
```{r}
small_train<-
caret::createDataPartition(y=ds_small_sample$icu_dead_before_28,p=.60,list  =
FALSE)
small_test<-setdiff((1:nrow(ds_small_sample)),small_train)

ds_small_train<-ds_small_sample[small_train,]
ds_small_test<-ds_small_sample[small_test,]
```


#default parameters
```{r}
cltask_1 = makeClassifTask(id = "c1", data = ds_small_sample, target =
"icu_dead_before_28",positive="X1")
```


#tunning
```{r}

lrns<-list(
  makeLearner("classif.ada",predict.type = "prob"),
```

```
  makeLearner("classif.earth",predict.type = "prob"),
  makeLearner("classif.C50",predict.type = "prob"),
  makeLearner("classif.ctree",predict.type = "prob"),
  makeLearner("classif.ranger",predict.type = "prob"),
  makeLearner("classif.gbm",predict.type = "prob"),
  makeLearner("classif.nnet",predict.type = "prob")
  )


lrn <- makeModelMultiplexer(lrns)

ps <- makeModelMultiplexerParamSet(lrn,
 classif.ranger=
   makeParamSet(
     makeIntegerParam("num.trees", lower = 500L, upper = 2000L),
     makeDiscreteParam("mtry", values=c(8L,10L,12L))
    ),
  classif.ada=
    makeParamSet(
    makeIntegerParam("iter", lower = 300L, upper = 2000L),
    makeIntegerParam("maxdepth", lower = 3L, upper = 30L),
    makeNumericParam("nu", lower = 0.001, upper = 0.1)
   ),
 classif.earth=
    makeParamSet(
      makeDiscreteParam("degree", values=c(3L,5L,6L)),
      makeIntegerParam("nprune", lower = 15L, upper = 25L)
  ),
  classif.C50=
    makeParamSet(
      makeIntegerParam("trials", lower = 10L, upper = 100L)
  ),
  classif.ctree=
    makeParamSet(
      makeDiscreteParam("mtry", values=c(8L,10L,12L,14L))
  ),
  classif.nnet=
    makeParamSet(
      makeDiscreteParam("size", values=c(2L,3L,4L)),
      makeNumericParam("decay", lower = 0.0005, upper = 0.002)
    ),
 classif.gbm=
  makeParamSet(
    makeNumericParam("shrinkage", lower = 0.001, upper = 0.003),
    makeIntegerParam("n.trees", lower = 100L, upper = 1000L),
    makeDiscreteParam("interaction.depth", values=c(1L,2L,3L,4L,5L))
  ))



rdesc <- makeResampleDesc("CV", iters = 3L)

#irace
ctrl <- makeTuneControlIrace(maxExperiments = 200L)

time_tune<-system.time(res <- tuneParams(lrn, cltask_1, rdesc, par.set = ps,
control = ctrl, show.info = TRUE, measures = list(mmce,fpr,fnr)))

str(res)
```

```
data_res<-(as.data.frame(trafoOptPath(res$opt.path)))

sum(data_res$exec.time)

data = generateHyperParsEffectData(res)

#ada
(data_learner<-subset(data$data,selected.learner=="classif.ada"))

data_learner_test<-data_learner[,which(names(data_learner) %in%

names(data_learner[grep("classif.ada|test.mean|iteration|selected",names(data
_learner))]))]

data_learner_test[which.min(data_learner_test$mmce.test.mean),]


#earth
(data_learner<-subset(data$data,selected.learner=="classif.earth"))

data_learner_test<-data_learner[,which(names(data_learner) %in%

names(data_learner[grep("classif.earth|test.mean|iteration|selected",names(da
ta_learner))]))]

data_learner_test[which.min(data_learner_test$mmce.test.mean),]

#C50

(data_learner<-subset(data$data,selected.learner=="classif.C50"))

data_learner_test<-data_learner[,which(names(data_learner) %in%

names(data_learner[grep("classif.C50|test.mean|iteration|selected",names(data
_learner))]))]

data_learner_test[which.min(data_learner_test$mmce.test.mean),]


#ctree

(data_learner<-subset(data$data,selected.learner=="classif.ctree"))

data_learner_test<-data_learner[,which(names(data_learner) %in%

names(data_learner[grep("classif.ctree|test.mean|iteration|selected",names(da
ta_learner))]))]

data_learner_test[which.min(data_learner_test$mmce.test.mean),]


#ranger

(data_learner<-subset(data$data,selected.learner=="classif.ranger"))

data_learner_test<-data_learner[,which(names(data_learner) %in%
```

```
names(data_learner[grep("classif.ranger|test.mean|iteration|selected",names(d
ata_learner))]))]

data_learner_test[which.min(data_learner_test$mmce.test.mean),]


#gbm

(data_learner<-subset(data$data,selected.learner=="classif.gbm"))

data_learner_test<-data_learner[,which(names(data_learner) %in%

names(data_learner[grep("classif.gbm|test.mean|iteration|selected",names(data
_learner))]))]

data_learner_test[which.min(data_learner_test$mmce.test.mean),]


#nnet

(data_learner<-subset(data$data,selected.learner=="classif.nnet"))

data_learner_test<-data_learner[,which(names(data_learner) %in%

names(data_learner[grep("classif.nnet|test.mean|iteration|selected",names(dat
a_learner))]))]

data_learner_test[which.min(data_learner_test$mmce.test.mean),]

```

A7-5. TUNNING MODELOS MLR

A7-5.1Tunning de modelos glm/gbm/randomForest en MLR

mlr_tunning_h2o_irace_1.Rmd

```
---
title: "Tunning models mlr 1"
output:
  html_document: default
  html_notebook: default
---


```{r}
require(h2o)
require(caret)
require(pROC)
```


#random seed
```{r}
set.seed(1000)
```
```

#start cluster h2o
```{r}

#local

h2o.init(nthreads=-1, max_mem_size = "7G")
#h2o.removeAll()
#h2o.shutdown(prompt = FALSE)

```


LOAD DATASET
```{r}
load(file = "~/ds_icu_2m")
```

#small dataset to begin tests
```{r}
small_sample<-createDataPartition(y=ds_icu_2m$icu_dead_before_28,p=.1,list  =
FALSE)
ds_small_sample<-ds_icu_2m[small_sample,]
```

#training and test datasets
```{r}
small_train<-
createDataPartition(y=ds_small_sample$icu_dead_before_28,p=.60,list = FALSE)
ds_small_train<-ds_small_sample[small_train,]
ds_small_test<-ds_small_sample[-small_train,]


full_train<-createDataPartition(y= ds_icu_2m $icu_dead_before_28,p=.60,list =
FALSE)
ds_train<- ds_icu_2m [full_train,]
ds_test<- ds_icu_2m [-full_train,]
```


#variables
```{r}
y<-"icu_dead_before_28"
```


#possible parameters
```{r}
getParamSet("classif.h2o.glm")
getParamSet("classif.h2o.gbm")
getParamSet("classif.h2o.randomForest")
```

#tasks
```{r}

#small dataset
```

```
cltask_1 = makeClassifTask(id = "c1", data = ds_small_sample, target =
"icu_dead_before_28",positive="X1")

#full dataset
cltask_2 = makeClassifTask(id = "c2", data = ds_icu_2f_preproc, target =
"icu_dead_before_28",positive="X1")
```

#tunning
```{r}

#default parameters

lrns<-list(
  makeLearner("classif.h2o.gbm",predict.type = "prob",par.vals =
list(balance_classes=TRUE)),
  makeLearner("classif.h2o.randomForest",predict.type = "prob",par.vals =
list(balance_classes=TRUE)),
  makeLearner("classif.h2o.glm",predict.type = "prob")
  )


lrn <- makeModelMultiplexer(lrns)


ps <- makeModelMultiplexerParamSet(lrn,
 classif.h2o.glm=
   makeParamSet(
     makeNumericParam("alpha", lower = 0.7, upper = 0.9)
    ),
   classif.h2o.gbm=
    makeParamSet(
    makeIntegerParam("ntrees", lower = 1200, upper = 1500),
    makeIntegerParam("max_depth", lower = 4, upper = 10),
    makeNumericParam("learn_rate", lower = 0.001, upper = 0.01)
    ),
 classif.h2o.randomForest=
    makeParamSet(
      makeIntegerParam("mtries", lower = 10, upper = 18),
      makeIntegerParam("ntrees", lower = 1500, upper = 2000),
      makeIntegerParam("max_depth", lower = 4, upper = 10)
    )
  )


rdesc <- makeResampleDesc("CV", iters = 3L)


#irace
ctrl <- makeTuneControlIrace(maxExperiments = 500L)

#misclassification rate (mmce) for classification.
# http://mlr-org.github.io/mlr-tutorial/release/html/measures/index.html

res <- tuneParams(lrn, cltask_1, rdesc, par.set = ps, control = ctrl, show.info
= TRUE, measures = list(mmce))

data_res<-(as.data.frame(trafoOptPath(res$opt.path)))
```

```
sum(data_res$exec.time)
```

```{r}
data = generateHyperParsEffectData(res,partial.dep = TRUE)

lparams<-list()

for (learner in c("classif.h2o.glm","classif.h2o.gbm","classif.h2o.randomForest"))
{
  data_learner<-subset(data$data,selected.learner==learner)
  data_learner_test<-data_learner[,which(names(data_learner) %in%
names(data_learner[grep(paste(learner,"test.mean|iteration|selected",sep="|")
,names(data_learner))]))]
  val<-learner
  lparams[[val]]<-
data_learner_test[which.min(data_learner_test$mmce.test.mean),]

}

lparams
```

#set the optimal hyperparameters found
```{r}

#glm
pars_glm<-as.list(lparams$classif.h2o.glm)
pars_glm[which(names(pars_glm) %in%
c("selected.learner","mmce.test.mean","iteration"))] <- NULL
names(pars_glm)<-(sub("classif.h2o.glm.","",names(pars_glm)))


#gbm
pars_gbm<-as.list(lparams$classif.h2o.gbm)
pars_gbm[which(names(pars_gbm) %in%
c("selected.learner","mmce.test.mean","iteration"))] <- NULL
names(pars_gbm)<-(sub("classif.h2o.gbm.","",names(pars_gbm)))

#randomForest
pars_randomForest<-as.list(lparams$classif.h2o.randomForest)
pars_randomForest[which(names(pars_randomForest) %in%
c("selected.learner","mmce.test.mean","iteration"))] <- NULL
names(pars_randomForest)<-
(sub("classif.h2o.randomForest.","",names(pars_randomForest)))

```

#benchmark parameters Irace full dataset
```{r}

  pars_gbm$balance_classes=TRUE
  lrn_gbm<-makeLearner("classif.h2o.gbm",predict.type = "prob",par.vals =
pars_gbm)

  pars_randomForest$balance_classes=TRUE


```

```
  lrn_randomForest<-makeLearner("classif.h2o.randomForest",predict.type     =
"prob",par.vals = pars_randomForest)

  lrn_glm<-makeLearner("classif.h2o.glm",predict.type                       =
"prob",par.vals=pars_glm)

  lrns<-list(lrn_gbm,lrn_glm,lrn_randomForest)

  rdesc<-makeResampleDesc("CV",stratify = TRUE,iter=5)

  bmr<-benchmark(lrns,cltask_2,rdesc)


sum(bmr$results$c2$classif.h2o.gbm$runtime,bmr$results$c2$classif.h2o.glm$run
time ,bmr$results$c2$classif.h2o.randomForest$runtime)


```

#results irace
```{r}
bmr

plt = plotBMRBoxplots(bmr, measure = mmce)
plt

df = generateThreshVsPerfData(bmr, measures = list(fpr, tpr), aggregate=FALSE)
plotROCCurves(df)

```

#ROC PLOTS irace
```{r}

## Extract predictions
preds = getBMRPredictions(bmr, drop = TRUE)


calculateROCMeasures(preds$classif.h2o.glm)
calculateROCMeasures(preds$classif.h2o.gbm)
calculateROCMeasures(preds$classif.h2o.randomForest)

## Convert predictions
ROCRpreds = lapply(preds, asROCRPrediction)

## Calculate true and false positive rate
ROCRperfs = lapply(ROCRpreds, function(x) ROCR::performance(x, "tpr", "fpr"))

plot(ROCRperfs[[1]],col=2,lwd=0)
for (i in 1:4){
  ## lda average ROC curve
  plot(ROCRperfs[[i]], col = i+1, avg = "vertical", lwd = 1, add=TRUE)
}

legend("bottomright",  legend  =  getBMRLearnerIds(bmr),  lty  =  1,  lwd  =
1,col=c(2,3,4,5,6,7,8,9,10,11))


```

mlr_tunning_h2o_irace_2.Rmd

````
---
title: "MLR Tunning deeplearning"
output:
  html_document: default
  html_notebook: default
---


```{r}
require(h2o)
require(caret)
require(pROC)
```


#random seed
```{r}
set.seed(1000)
```


#start cluster h2o
```{r}

#local

h2o.init(nthreads=-1, max_mem_size = "7G")
#h2o.removeAll()
#h2o.shutdown(prompt = FALSE)

#cluster
#localH2O <- h2o.init(nthreads=-1,ip = '192.168.1.126', port =54321)
```


# LOAD DATASET
```{r}
load(file = "~/ds_icu_2m")
```

#small dataset to begin tests
```{r}
small_sample<-createDataPartition(y=ds_icu_2m$icu_dead_before_28,p=.20,list =
FALSE)
ds_small_sample<-ds_icu_2m[small_sample,]
```

#training and test datasets
```{r}
small_train<-
createDataPartition(y=ds_small_sample$icu_dead_before_28,p=.60,list = FALSE)
ds_small_train<-ds_small_sample[small_train,]
ds_small_test<-ds_small_sample[-small_train,]
````
````

```
full_train<-createDataPartition(y=ds_icu_2m$icu_dead_before_28,p=.60,list   =
FALSE)
ds_train<-ds_icu_2m[full_train,]
ds_test<-ds_icu_2m[-full_train,]
```


#variables
```{r}
y<-"icu_dead_before_28"
```


#tasks
```{r}

#small dataset

cltask_1 = makeClassifTask(id = "c1", data = ds_small_sample, target =
"icu_dead_before_28",positive="X1")


#full dataset


cltask_2 = makeClassifTask(id = "c2", data = ds_icu_2m, target =
"icu_dead_before_28",positive="X1")

```

#tunning
```{r}


#deeplearning

lrns<-list(
  makeLearner("classif.h2o.deeplearning",predict.type = "prob", par.vals =
list(balance_classes=TRUE,shuffle_training_data=TRUE,fast_mode=TRUE,l1=1e-6))
  )


lrn <- makeModelMultiplexer(lrns)

hc1=c(50,50,50);hc2=c(600,600);hc3=c(1000,1000);hc4=c(300,300)

ps <- makeModelMultiplexerParamSet(lrn,
  classif.h2o.deeplearning=
    makeParamSet(
      makeDiscreteParam("hidden",values=list(c1=hc1,c2=hc2,c3=hc3,c4=hc4)),
      makeIntegerParam("epochs", lower = 30, upper = 200),
      makeNumericParam("rho", lower = 0.95, upper = 0.999),
      makeNumericParam("epsilon", lower = 1e-8, upper = 1e-6)
    )
  )
```

```
rdesc <- makeResampleDesc("CV", iters = 2L)


#irace
ctrl <- makeTuneControlIrace(maxExperiments = 300L)

#misclassification rate (mmce) for classification.
# http://mlr-org.github.io/mlr-tutorial/release/html/measures/index.html

res <- tuneParams(lrn, cltask_1, rdesc, par.set = ps, control = ctrl, show.info
= TRUE, measures = list(mmce))


data_res<-(as.data.frame(trafoOptPath(res$opt.path)))
sum(data_res$exec.time)
```


```{r}
data = generateHyperParsEffectData(res,partial.dep = TRUE)

lparams<-list()

for (learner in c("classif.h2o.deeplearning"))
{
  data_learner<-subset(data$data,selected.learner==learner)
  data_learner_test<-data_learner[,which(names(data_learner) %in%

names(data_learner[grep(paste(learner,"test.mean|iteration|selected",sep="|")
,names(data_learner))]))]
  val<-learner
  lparams[[val]]<-
data_learner_test[which.min(data_learner_test$mmce.test.mean),]

}

lparams
```

#set the optimal hyperparameters found
```{r}

#deeplearning
pars_deeplearning<-as.list(lparams$classif.h2o.deeplearning)
pars_deeplearning[which(names(pars_deeplearning)                          %in%
c("selected.learner","mmce.test.mean","iteration"))] <- NULL
names(pars_deeplearning)<-
(sub("classif.h2o.deeplearning.","",names(pars_deeplearning)))

```

#benchmark parameters Irace full dataset
```{r}

  pars_deeplearning$balance_classes=TRUE
  pars_deeplearning$shuffle_training_data=TRUE
  pars_deeplearning$fast_mode=TRUE
  pars_deeplearning$l1=1e-6
```

```
pars_deeplearning$hidden=eval(parse(text=paste("h",pars_deeplearning$hidden,s
ep="")))
  pars_deeplearning$l1=as.numeric(as.character(pars_deeplearning$l1))

pars_deeplearning$input_dropout_ratio=as.numeric(as.character(pars_deeplearni
ng$input_dropout_ratio))


  lrn_deeplearning<-makeLearner("classif.h2o.deeplearning",predict.type     =
"prob", par.vals = pars_deeplearning)

  lrn_deeplearning

  lrns<-list(lrn_deeplearning)

  rdesc<-makeResampleDesc("CV",stratify = TRUE,iter=5)

  bmr<-benchmark(lrns,cltask_2,rdesc)

  sum(bmr$results$c2$classif.h2o.deeplearning$runtime)


```

#results irace
```{r}
bmr

plt = plotBMRBoxplots(bmr, measure = mmce)
plt

df = generateThreshVsPerfData(bmr, measures = list(fpr, tpr), aggregate=FALSE)
plotROCCurves(df)

```

#ROC PLOTS irace
```{r}

## Extract predictions
preds = getBMRPredictions(bmr, drop = TRUE)


calculateROCMeasures(preds$classif.h2o.glm)
calculateROCMeasures(preds$classif.h2o.gbm)
calculateROCMeasures(preds$classif.h2o.randomForest)
calculateROCMeasures(preds$classif.h2o.deeplearning)

calc<-calculateROCMeasures(preds)


tt<-asROCRPrediction(preds)

## Convert predictions
ROCRpreds = lapply(preds, asROCRPrediction)
```

```
## Calculate true and false positive rate
ROCRperfs = lapply(ROCRpreds, function(x) ROCR::performance(x, "tpr", "fpr"))

ss<-ROCR::performance(tt, "tpr", "fpr")

plot(ss)

plot(ROCRperfs[[1]],col=2,lwd=0)
for (i in 1:4){
  ## lda average ROC curve
  plot(ROCRperfs[[i]], col = i+1, avg = "vertical", lwd = 1, add=TRUE)
}

legend("bottomright", legend = getBMRLearnerIds(bmr), lty = 1, lwd =
1,col=c(2,3,4,5,6,7,8,9,10,11))


```

A7-6. BENCHMARKING

A7-6.1 Benchmarking con MLR y modelos H2O. Conjunto completo de datos.

mlr_h2o_benchmark.Rmd

```
---
title: "MLR H20 Benchmark"
output:
  html_document: default
  html_notebook: default
---


```{r}
require(mlr)
require(h2o)
require(caret)
require(pROC)
```


#random seed
```{r}
set.seed(1000)
```


#start h2o
```{r}
h2o.init(nthreads=-1, max_mem_size = "7G")
#h2o.removeAll()
#h2o.shutdown(prompt = FALSE)

```
```

LOAD DATASET
```{r}
load(file = "~/ds_icu_2m")
```

#
```{r}
ds_sample<-ds_icu_2m
```

#variables
```{r}
y<-"icu_dead_before_28"
```

#task
```{r}
cltask = makeClassifTask(id = "c1", data = ds_sample, target = "icu_dead_before_28",positive="X1")
```

#parameters
```{r}

getParamSet("classif.h2o.gbm")
getParamSet("classif.h2o.glm")
getParamSet("classif.h2o.randomForest")
getParamSet("classif.h2o.deeplearning")

#gbm
par_gbm<-list()
par_gbm$balance_classes = TRUE
par_gbm$ntrees = 1255
par_gbm$max_depth = 6
par_gbm$min_rows = 10
par_gbm$learn_rate = 0.007709469
par_gbm$balance_classes = TRUE

#glm
par_glm<-list()
par_glm$alpha=0.7473238

#dl
par_dl<-list()
par_dl$hidden = c(1000,1000)
par_dl$epochs =72
par_dl$rho =0.9615025
par_dl$epsilon =1.883177e-07
par_dl$l1=1e-6
par_dl$input_dropout_ratio = 0.1
```

```
par_dl$balance_classes = TRUE
par_dl$variable_importances = TRUE
par_dl$shuffle_training_data=TRUE
par_dl$fast_mode=TRUE

#rf
par_rf<-list()
par_rf$ntrees = 1763
par_rf$mtries = 14
par_rf$max_depth = 5
par_rf$balance_classes = TRUE
```


```{r}
lrns<-list(
  makeLearner("classif.h2o.gbm",predict.type = "prob",par.vals=par_gbm),
  makeLearner("classif.h2o.glm",predict.type = "prob",par.vals=par_glm),
  makeLearner("classif.h2o.randomForest",predict.type                    =
"prob",par.vals=par_rf),
  makeLearner("classif.h2o.deeplearning",predict.type                    =
"prob",par.vals=par_dl)
  )


rdesc <- makeResampleDesc("CV", iters = 10L)
meas = list(auc,mmce,timetrain,f1)

```

```{r}
bmr = benchmark(lrns, cltask, rdesc, meas, show.info = TRUE)
```

#results
```{r}
bmr
```

#boxplot
```{r}
plotBMRBoxplots(bmr,measure = auc)
plotBMRBoxplots(bmr,measure = mmce)
```

#violin plots
```{r}
plotBMRBoxplots(bmr, measure = auc, style = "violin", pretty.names = TRUE) +
  aes(color = learner.id) +
  theme(strip.text.x = element_text(size = 8))


plotBMRBoxplots(bmr, measure = mmce, style = "violin", pretty.names = TRUE) +
  aes(color = learner.id) +
  theme(strip.text.x = element_text(size = 8))
```
```{r}
df = generateThreshVsPerfData(bmr, measures = list(fpr,tpr,mmce))
plotROCCurves(df)
```

```
```{r}
## Extract predictions
preds = getBMRPredictions(bmr, drop = TRUE)

## Convert predictions
ROCRpreds = lapply(preds, asROCRPrediction)

## Calculate true and false positive rate
ROCRperfs = lapply(ROCRpreds, function(x) ROCR::performance(x, "tpr", "fpr"))

## gbm average ROC curve
plot(ROCRperfs[[1]], col = "blue", avg = "vertical", spread.estimate =
"stderror",
  show.spread.at = seq(0.1, 0.8, 0.1), plotCI.col = "blue", plotCI.lwd = 1,
lwd = 1)

## gbm individual ROC curves
plot(ROCRperfs[[1]], col = "blue", lty = 2, lwd = 0.15, add = TRUE)

## glm average ROC curve
plot(ROCRperfs[[2]], col = "red", avg = "vertical", spread.estimate =
"stderror",
  show.spread.at = seq(0.1, 0.8, 0.1), plotCI.col = "red", plotCI.lwd = 1, lwd
= 1,add=TRUE)

## glm individual ROC curves
plot(ROCRperfs[[2]], col = "red", lty = 2, lwd = 0.15, add = TRUE)

## rf average ROC curve
plot(ROCRperfs[[3]], col = "green", avg = "vertical", spread.estimate =
"stderror",
  show.spread.at = seq(0.1, 0.8, 0.1), plotCI.col = "green", plotCI.lwd = 1,
lwd = 1,add=TRUE)

## rf individual ROC curves
plot(ROCRperfs[[3]], col = "green", lty = 2, lwd = 0.15, add = TRUE)

## dl average ROC curve
plot(ROCRperfs[[4]], col = "orange", avg = "vertical", spread.estimate =
"stderror",
  show.spread.at = seq(0.1, 0.8, 0.1), plotCI.col = "orange", plotCI.lwd = 1,
lwd = 1,add=TRUE)

## dl individual ROC curves
plot(ROCRperfs[[4]], col = "orange", lty = 2, lwd = 0.15, add = TRUE)

legend("bottomright", legend = getBMRLearnerIds(bmr), lty = 1, lwd = 2, col =
c("blue", "red","green","orange"))
title("Benchmark H2O classifiers - MLR")

```
```

A7-6.2Benchmarking con MLR y modelos H2O. Grupos.

mlr_h2o_benchmark_groups.Rmd

````
---
title: "MLR H20 Benchmark"
output:
  html_document: default
  html_notebook: default
---


```{r}
require(mlr)
require(h2o)
require(caret)
require(pROC)
```


#random seed
```{r}
set.seed(1000)
```


#start h2o
```{r}
h2o.init(nthreads=-1, max_mem_size = "7G")
#h2o.removeAll()
#h2o.shutdown(prompt = FALSE)

```


# LOAD DATASET
```{r}
load(file = "~/ds_icu_2m")
```



#variables
```{r}
y<-"icu_dead_before_28"
```


#datasets
```{r}

#----------------------------------------------------------------------
#more/less than 3 icu stays
ds_icu_stay_more72<-filter(ds_icu_2m,icu_stay_more72==TRUE)
ds_icu_stay_less72<-filter(ds_icu_2m,icu_stay_more72==FALSE)
````
````

```
#more/less than 4 sofa score
ds_icu_sofa_score_more4<-filter(ds_icu_2m,score_sofa_more4==TRUE)
ds_icu_sofa_score_less4<-filter(ds_icu_2m,score_sofa_more4==FALSE)


#yes/no diabetics
ds_icu_diabetes_yes<-filter(ds_icu_2m,diag_has_diabet==TRUE)
ds_icu_diabetes_no<-filter(ds_icu_2m,diag_has_diabet==FALSE)


#yes/no insuline IV
ds_icu_insulineiv_yes<-filter(ds_icu_2m,icu_has_admiv_insulin==TRUE)
ds_icu_insulineiv_no<-filter(ds_icu_2m,icu_has_admiv_insulin==FALSE)


#yes/no sepsis in
ds_icu_sepsis_yes<-filter(ds_icu_2m,med_has_sepsis==TRUE)
ds_icu_sepsis_no<-filter(ds_icu_2m,med_has_sepsis==FALSE)

```


#tasks
```{r}
cltask1 = makeClassifTask(id = "isu_stay_more72", data = ds_icu_stay_more72,
target = "icu_dead_before_28",positive="X1")
cltask2 = makeClassifTask(id = "isu_stay_less72", data = ds_icu_stay_less72,
target = "icu_dead_before_28",positive="X1")
cltask3 = makeClassifTask(id = "isu_sofa_score_more4", data =
ds_icu_sofa_score_more4, target = "icu_dead_before_28",positive="X1")
cltask4 = makeClassifTask(id = "isu_sofa_score_less4", data =
ds_icu_sofa_score_less4, target = "icu_dead_before_28",positive="X1")
cltask5 = makeClassifTask(id = "isu_diabetes_yes", data = ds_icu_diabetes_yes,
target = "icu_dead_before_28",positive="X1")
cltask6 = makeClassifTask(id = "isu_diabetes_no", data = ds_icu_diabetes_no,
target = "icu_dead_before_28",positive="X1")
cltask7 = makeClassifTask(id = "isu_insulineiv_yes", data =
ds_icu_insulineiv_yes, target = "icu_dead_before_28",positive="X1")
cltask8 = makeClassifTask(id = "isu_insulineiv_no", data =
ds_icu_insulineiv_no, target = "icu_dead_before_28",positive="X1")
cltask9 = makeClassifTask(id = "isu_sepsis_yes", data = ds_icu_sepsis_yes,
target = "icu_dead_before_28",positive="X1")
cltask10 = makeClassifTask(id = "isu_sepsis_no", data = ds_icu_sepsis_no,
target = "icu_dead_before_28",positive="X1")

ltasks=list(cltask1,cltask2,cltask3,cltask4,cltask5,cltask6,cltask7,cltask8,c
ltask9,cltask10)
```


#parameters
```{r}
getParamSet("classif.h2o.gbm")
getParamSet("classif.h2o.glm")
getParamSet("classif.h2o.randomForest")
getParamSet("classif.h2o.deeplearning")
```

```
#gbm
par_gbm<-list()
par_gbm$balance_classes = TRUE
par_gbm$ntrees = 1255
par_gbm$max_depth = 6
par_gbm$min_rows = 10
par_gbm$learn_rate = 0.007709469
par_gbm$balance_classes = TRUE



#glm
par_glm<-list()
par_glm$alpha=0.7473238

#dl
par_dl<-list()
par_dl$hidden = c(1000,1000)
par_dl$epochs =72
par_dl$rho =0.9615025
par_dl$epsilon =1.883177e-07
par_dl$l1=1e-6
par_dl$input_dropout_ratio = 0.1
par_dl$balance_classes = TRUE
par_dl$variable_importances = TRUE
par_dl$shuffle_training_data=TRUE
par_dl$fast_mode=TRUE

#rf
par_rf<-list()
par_rf$ntrees = 1763
par_rf$mtries = 14
par_rf$max_depth = 5
par_rf$balance_classes = TRUE
```



```{r}
lrns<-list(
  makeLearner("classif.h2o.gbm",predict.type = "prob",par.vals=par_gbm),
  makeLearner("classif.h2o.glm",predict.type = "prob",par.vals=par_glm),
  makeLearner("classif.h2o.randomForest",predict.type                    =
"prob",par.vals=par_rf),
  makeLearner("classif.h2o.deeplearning",predict.type                    =
"prob",par.vals=par_dl)
  )


rdesc <- makeResampleDesc("CV", iters = 10L)
meas = list(auc,mmce,timetrain,f1)

```

```{r}
bmr = benchmark(lrns, ltasks , rdesc, meas, show.info = TRUE)
```

```{r}
```

```
save(file="~/Master_ICM/TFM/r_Projects/memoria/mlr_h2o_benchmark_groups/bmr",
bmr)
```

#results
```{r}
bmr
```

#boxplot
```{r}
plotBMRBoxplots(bmr,measure = auc)
plotBMRBoxplots(bmr,measure = mmce)
```

#violin plots
```{r}
plotBMRBoxplots(bmr, measure = auc, style = "violin", pretty.names = TRUE,
facet.wrap.ncol = 2) +
  aes(color = learner.id) +
  theme(strip.text.x = element_text(size = 8))


plotBMRBoxplots(bmr, measure = mmce, style = "violin", pretty.names = TRUE,
facet.wrap.ncol = 2) +
  aes(color = learner.id) +
  theme(strip.text.x = element_text(size = 8))```
#rocs
```{r}
df = generateThreshVsPerfData(bmr, measures = list(fpr,tpr,mmce))
plotROCCurves(df)
```


#summary
```{r}
plt=plotBMRSummary(bmr)
plt+ggtitle("Benchmark aggregate Summary")+
  theme(plot.title = element_text(hjust = 0.5))
```
#ranks
```{r}
m = convertBMRToRankMatrix(bmr, mmce)
m


m = convertBMRToRankMatrix(bmr, auc)
m
```



```{r}
plt=plotBMRRanksAsBarChart(bmr, pos = "tile", measure=mmce)
plt+ggtitle("mmce rank")+
  theme(plot.title = element_text(hjust = 0.5))
```
```{r}

```
plt=plotBMRRanksAsBarChart(bmr, pos = "tile", measure=auc)
plt+ggtitle("auc rank")+
  theme(plot.title = element_text(hjust = 0.5))
```
#Hypothesis test
```{r}

friedmanTestBMR(bmr,auc)
friedmanTestBMR(bmr,mmce)


friedmanPostHocTestBMR(bmr,p.value=0.05,auc)
friedmanPostHocTestBMR(bmr,p.value=0.05,mmce)

g = generateCritDifferencesData(bmr, p.value = 0.05, test = "bd",auc)
plt=plotCritDifferences(g) + coord_cartesian(xlim = c(-1,5), ylim = c(0,2))
plt+ggtitle("critical diferences, test bd, auc")+
  theme(plot.title = element_text(hjust = 0.5))

g = generateCritDifferencesData(bmr, p.value = 0.05, test = "bd",mmce)
plt=plotCritDifferences(g) + coord_cartesian(xlim = c(-1,5), ylim = c(0,2))
plt+ggtitle("critical diferences, test bd, mmce")+
  theme(plot.title = element_text(hjust = 0.5))

```
```

A7-7. APLICACIÓN MODELOS H2O

Aplicación de glm, gbm, rf, dl para dataset completo con H2O agrupando con y sin comorbilidades

h2o_model_groups.R

```
---

require(h2o)
require(caret)
require(ROCR)
require(dplyr)

#random seed

set.seed(1000)

#start h2o

h2o.init(nthreads=-1, max_mem_size = "7G")
#h2o.removeAll()
#h2o.shutdown(prompt = FALSE)

# LOAD DATASET
load(file = "~/ds_icu_2m")


#variables
y<-"icu_dead_before_28"
```
```

```
#-----------------------------------------------------------------
#more/less than 3 icu stays
ds_icu_stay_more72<-filter(ds_icu_2m,icu_stay_more72==TRUE)
ds_icu_stay_less72<-filter(ds_icu_2m,icu_stay_more72==FALSE)



#more/less than 4 sofa score
ds_icu_sofa_score_more4<-filter(ds_icu_2m,score_sofa_more4==TRUE)
ds_icu_sofa_score_less4<-filter(ds_icu_2m,score_sofa_more4==FALSE)


#yes/no diabetics
ds_icu_diabetes_yes<-filter(ds_icu_2m,diag_has_diabet==TRUE)
ds_icu_diabetes_no<-filter(ds_icu_2m,diag_has_diabet==FALSE)


#yes/no insuline IV
ds_icu_insulineiv_yes<-filter(ds_icu_2m,icu_has_admiv_insulin==TRUE)
ds_icu_insulineiv_no<-filter(ds_icu_2m,icu_has_admiv_insulin==FALSE)


#yes/no sepsis in
ds_icu_sepsis_yes<-filter(ds_icu_2m,med_has_sepsis==TRUE)
ds_icu_sepsis_no<-filter(ds_icu_2m,med_has_sepsis==FALSE)

#-----------------------------------------------------
list_groups=c("ds_icu_stay_more72","ds_icu_stay_less72","ds_icu_sofa_score_mo
re4","ds_icu_sofa_score_less4",

"ds_icu_diabetes_yes","ds_icu_diabetes_no","ds_icu_insulineiv_yes","ds_icu_in
sulineiv_no",
              "ds_icu_sepsis_yes","ds_icu_sepsis_no")


#loop

lmdl<-list()

for (d in list_groups){


print("#########################################################################
##################")
  print(d)

print("#########################################################################
##################")

  ds_group<-eval(parse(text=d))

  train<-createDataPartition(y=ds_group$icu_dead_before_28,p=.60,list       =
FALSE)
  d_train<-ds_group[train,]
  d_test<-ds_group[-train,]

  d_train_wocom<-d_train[,!grepl("^com_",names(d_train))]
  d_test_wocom<-d_test[,!grepl("^com_",names(d_train))]
```

```
  #glm
  time_train<-system.time(glm_iter <- h2o.glm(y = y,
                                              training_frame                =
as.h2o(d_train),
                                              validation_frame              =
as.h2o(d_test),
                                              family = "binomial",
                                              link="logit",
                                              lambda_search = FALSE,
                                              balance_classes = TRUE,
                                              alpha=0.7473238,
                                              nfolds = 10,
                                              fold_assignment = "Modulo",

keep_cross_validation_predictions = TRUE,

keep_cross_validation_fold_assignment = TRUE,
                                              seed = 1))


  lmdl[[paste0("glm_",d)]]<-glm_iter
  lmdl[[paste0("glm_",d,"_time")]]<-time_train

  time_train<-system.time(glm_iter_wocom <- h2o.glm(y = y,
                                                    training_frame          =
as.h2o(d_train_wocom),
                                                    validation_frame        =
as.h2o(d_test_wocom),
                                                    family = "binomial",
                                                    link="logit",
                                                    lambda_search = FALSE,
                                                    balance_classes = TRUE,
                                                    alpha=0.7473238,
                                                    nfolds = 10,
                                                    fold_assignment         =
"Modulo",

keep_cross_validation_predictions = TRUE,

keep_cross_validation_fold_assignment = TRUE,
                                                    seed = 1))


  lmdl[[paste0("glm_",d,"_wocom")]]<-glm_iter_wocom
  lmdl[[paste0("glm_",d,"_wocom_time")]]<-time_train

  #gbm Gradient Boosting Machine

  time_train<-system.time(gbm_iter <- h2o.gbm(y = y,
                                              training_frame                =
as.h2o(d_train),
                                              validation_frame              =
as.h2o(d_test),
                                              distribution = "bernoulli",
                                              balance_classes = TRUE,
                                              ntrees = 1255,
                                              max_depth = 6,
                                              min_rows = 10,
```

```
                                            learn_rate = 0.007709469,
                                            nfolds = 10,
                                            fold_assignment  =  "Modulo",  #
for stacked

keep_cross_validation_predictions = TRUE,

keep_cross_validation_fold_assignment = TRUE,
                                            seed = 1))



  lmdl[[paste0("gbm_",d)]]<-gbm_iter
  lmdl[[paste0("gbm_",d,"_time")]]<-time_train

  time_train<-system.time(gbm_iter_wocom <- h2o.gbm(y = y,
                                                    training_frame          =
as.h2o(d_train_wocom),
                                                    validation_frame        =
as.h2o(d_test_wocom),
                                                    distribution            =
"bernoulli",
                                                    balance_classes = TRUE,
                                                    ntrees = 1255,
                                                    max_depth = 6,
                                                    min_rows = 10,
                                                    learn_rate = 0.007709469,
                                                    nfolds = 10,
                                                    fold_assignment         =
"Modulo", # for stacked

keep_cross_validation_predictions = TRUE,

keep_cross_validation_fold_assignment = TRUE,
                                                    seed = 1))


  lmdl[[paste0("gbm_",d,"_wocom")]]<-gbm_iter_wocom
  lmdl[[paste0("gbm_",d,"_wocom_time")]]<-time_train


  # RF random forest

  time_train<-system.time(rf_iter <- h2o.randomForest(y = y,
                                                      training_frame        =
as.h2o(d_train),
                                                      validation_frame      =
as.h2o(d_test),
                                                      ntrees = 1763,
                                                      mtries = 14,
                                                      max_depth = 5,
                                                      nfolds = 10,
                                                      balance_classes = TRUE,
                                                      fold_assignment       =
"Modulo",

keep_cross_validation_predictions = TRUE,

keep_cross_validation_fold_assignment = TRUE,
```

```
                                                    seed = 1))


  lmdl[[paste0("rf_",d)]]<-rf_iter
  lmdl[[paste0("rf_",d,"_time")]]<-time_train


  time_train<-system.time(rf_iter_wocom <- h2o.randomForest(y = y,
                                                            training_frame  =
as.h2o(d_train_wocom),
                                                            validation_frame
= as.h2o(d_test_wocom),
                                                            ntrees = 1763,
                                                            mtries = 14,
                                                            max_depth = 5,
                                                            nfolds = 10,
                                                            balance_classes =
TRUE,
                                                            fold_assignment =
"Modulo",

keep_cross_validation_predictions = TRUE,

keep_cross_validation_fold_assignment = TRUE,
                                                            seed = 1))


  lmdl[[paste0("rf_",d,"_wocom")]]<-rf_iter_wocom
  lmdl[[paste0("rf_",d,"_wocom_time")]]<-time_train

  #dl

  time_train<-system.time(dl_iter <- h2o.deeplearning(y = y,
                                                      training_frame          =
as.h2o(d_train),
                                                      validation_frame        =
as.h2o(d_test),
                                                      nfolds = 10,
                                                      hidden = c(1000,1000),
                                                      epochs =72,
                                                      rho =0.9615025,
                                                      epsilon =1.883177e-07,
                                                      l1=1e-6,
                                                      variable_importances    =
TRUE,
                                                      input_dropout_ratio = 0.1,
                                                      distribution            =
"bernoulli",
                                                      balance_classes = TRUE,

shuffle_training_data=TRUE,
                                                      fast_mode=TRUE,
                                                      fold_assignment         =
"Modulo",

keep_cross_validation_predictions = TRUE,

keep_cross_validation_fold_assignment = TRUE,
                                                      seed = 1))
```

```
  lmdl[[paste0("iter_",d)]]<-dl_iter
  lmdl[[paste0("iter_",d,"_time")]]<-time_train


  time_train<-system.time(dl_iter_wocom <- h2o.deeplearning(y = y,
                                                            training_frame        =
as.h2o(d_train_wocom),
                                                            validation_frame      =
as.h2o(d_test_wocom),
                                                            nfolds = 10,
                                                            hidden = c(1000,1000),
                                                            epochs =72,
                                                            rho =0.9615025,
                                                            epsilon =1.883177e-07,
                                                            l1=1e-6,
                                                            variable_importances  =
TRUE,
                                                            input_dropout_ratio   =
0.1,
                                                            distribution          =
"bernoulli",
                                                            balance_classes = TRUE,

shuffle_training_data=TRUE,
                                                            fast_mode=TRUE,
                                                            fold_assignment       =
"Modulo",

keep_cross_validation_predictions = TRUE,

keep_cross_validation_fold_assignment = TRUE,
                                                            seed = 1))


  lmdl[[paste0("dl_",d,"_wocom")]]<-dl_iter_wocom
  lmdl[[paste0("dl_",d,"_wocom_time")]]<-time_train

  #ensemble

  try(
   {time_train<-system.time(ensemble2_iter <- h2o.stackedEnsemble(y = y,

training_frame = as.h2o(d_train),

validation_frame = as.h2o(d_test),

selection_strategy = "choose_all",
                                                                 base_models =
list(gbm_iter@model_id,rf_iter@model_id,dl_iter@model_id)))



  lmdl[[paste0("ensemble2_",d)]]<-ensemble2_iter
  lmdl[[paste0("ensemble2_",d,"_time")]]<-time_train
   }
  )
```

```
  try({
    time_train<-system.time(ensemble2_iter_wocom <- h2o.stackedEnsemble(y = y,

training_frame = as.h2o(d_train_wocom),

validation_frame = as.h2o(d_test_wocom),

selection_strategy = "choose_all",

base_models = list(gbm_iter_wocom@model_id,rf_iter_wocom@model_id,

dl_iter_wocom@model_id)))


  lmdl[[paste0("ensemble2_",d,"_wocom")]]<-ensemble2_iter_wocom
  lmdl[[paste0("ensemble2_",d,"_wocom_time")]]<-time_train
  })


}
```