5_Ensamblado

2023-02-03

Ensamblado de algoritmos

Nota: El código completo de la practica de ensamblados se encuentra en el siguiente repositorio de GitHub, para no extender innecesariamente el reporte: https://github.com/TiaIvonne/MasterUCM-MachineLearningR/blob/master/5_Ensamblado.R

Preparacion del ensamblado con *caret ensemble*

Antes de comenzar el modelado, se deben tener decididos los parámetros para los algoritmos que serán ensamblados (proceso obtenido anteriormente en el apartado de tunning) para los siguientes algoritmos:

Redes neuronales Gradient boosting machine Random Forest Classifier Support Vector Machines (lineal, polinomial y radial)

Con los resultados obtenidos se construyen los *grids* y se realiza el modelamiento

```
# Semilla
set.seed(12345)
repeticiones <- 10

# Evaluar los modelos
stackControl <- trainControl(
    method = "repeatedcv",
    number = 4,
    repeats = repeticiones,
    savePredictions = TRUE,
    classProbs = TRUE)

# grid gbm
gbmGrid <- expand.grid(
    n.trees = c(2000),
    interaction.depth = c(2),
    shrinkage =c(0.1),
    n.minobsinnode = c(20))

# grid random forest
rfGrid <- expand.grid(mtry=c(3))

# grid svm lineal
svmlinGrid <- expand.grid(C=c(0.08))

# grid svm ply
svmPolyGrid <- expand.grid(C=c(0.03),degree=c(3),scale=c(2))

# grid svm radial
svmRadialGrid <- expand.grid(sigma = c(0.5), C = c(30))

# Modelado
set.seed(12345)
models <- caretList(varObjBin~.,
    data = dengue,
    trControl = stackControl,
    tuneList = list(
        parrf = caretModelSpec(method = "rf", maxnodes = 30, n.trees = 200, nodesize = 10, sampsize = 150, tuneGrid = rfGrid),
        glm = caretModelSpec(method = "glm"),
        gbm = caretModelSpec(method = "gbm", tuneGrid = gbmGrid),
        svmlinear = caretModelSpec(method = "svmLinear", tuneGrid = svmlinGrid),
        svmPoly = caretModelSpec(method = "svmPoly", tuneGrid = svmPolyGrid),
        svmradial = caretModelSpec(method = "svmRadial", tuneGrid = svmRadialGrid)))
```

Se despliegan los resultados obtenidos para el ensamblado

```
results <- resamples(models)
summary(results)
dotplot(results)

modelCor(results)
splom(results)
results[[2]]

ensemble <- caretEnsemble(models)
# Aquí se recomiendan los pesos para el ensamblado # de todos los modelos y se
# ve la tasa de aciertos de cada modelo y ensamblado

summary(ensemble)
```

```r
# 2. Validacion cruzada y kit ensamblado ----
# 2.1 Leer funciones ------
# 2.2 Prepapar archivo, variables , semilla y repeticiones ----
dput(names(dengue))

set.seed(12345)

archivo <- dengue

vardep <- "varObjBin"
listconti <- c("h10pix", "temp90", "trees", "trees90", "Ymin", "Ymax",
    "temp", "humid", "humid90")

listclass <- c("")
grupos <- 4
sinicio <- 1234
repe <- 15


# 2.3 Obtener datos de cv repetida para cada algoritmo y
# procesar resultado ----
medias_1<-cruzadalogistica(data = archivo,
    vardep = vardep,
    listconti=listconti,
    listclass = listclass, grupos = grupos,
    sinicio = sinicio, repe = repe)

medias1bis <- as.data.frame(medias_1[1])
medias1bis$modelo<-"logistica"
predi1<-as.data.frame(medias_1[2])
predi1$logi<-predi1$Yes

medias_2<-cruzadaavnnetbin(data=archivo,
    vardep=vardep,listconti=listconti,
    listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
    size=c(10),decay=c(0.01),repeticiones=5,itera=200)

medias2bis<-as.data.frame(medias_2[1])
medias2bis$modelo<-"avnnet"
predi2<-as.data.frame(medias_2[2])
predi2$avnnet<-predi2$Yes

medias_3<-cruzadarfbin(data=archivo,
    vardep=vardep,listconti=listconti,
    listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
    mtry=3,ntree=500,nodesize=10,replace=TRUE)

medias3bis<-as.data.frame(medias_3[1])
medias3bis$modelo<-"randomforest"
predi3<-as.data.frame(medias_3[2])
predi3$rf<-predi3$Yes

medias_4 <-cruzadagbmbin(data=archivo,
    vardep=vardep,listconti=listconti,
    listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
    n.minobsinnode=5,shrinkage=0.1,n.trees=3000,interaction.depth=2)

medias4bis<-as.data.frame(medias_4[1])
medias4bis$modelo<-"gbm"
predi4<-as.data.frame(medias_4[2])
predi4$gbm<-predi4$Yes

medias_5 <-cruzadaxgbmbin(data = archivo,
    vardep = vardep,listconti = listconti,
    listclass = listclass, grupos = grupos, sinicio = sinicio, repe = repe,
    min_child_weight = 5, eta = 0.10, nrounds = 250, max_depth = 6,
    gamma = 0, colsample_bytree = 1, subsample = 1,
    alpha = 0, lambda = 0, lambda_bias = 0)

medias5bis <-as.data.frame(medias_5[1])
medias5bis$modelo <-"xgbm"
predi5 <-as.data.frame(medias_5[2])
predi5$xgbm<-predi5$Yes

medias_6<-cruzadaSVMbin(data=archivo,
    vardep=vardep,listconti=listconti,
    listclass=listclass,grupos=grupos,
    sinicio=sinicio,repe=repe,C=0.08)

medias6bis<-as.data.frame(medias_6[1])
medias6bis$modelo<-"svmLinear"
predi6<-as.data.frame(medias_6[2])
predi6$svmLinear<-predi6$Yes

medias_7 <- cruzadaSVMbinPoly(data = archivo,
     vardep = vardep, listconti = listconti,
     listclass = listclass, grupos = grupos, sinicio = sinicio, repe = repe,
     C = 0.03, degree = 3, scale = 2)

medias7bis <- as.data.frame(medias_7[1])
medias7bis$modelo <- "svmPoly"
predi7 <- as.data.frame(medias_7[2])
predi7$svmPoly <- predi7$Yes

medias_8 <- cruzadaSVMbinRBF(data = archivo,
    vardep = vardep, listconti = listconti,
    listclass = listclass, grupos = grupos,
    sinicio = sinicio, repe = repe,
    C = 30, sigma = 0.5)

medias8bis<-as.data.frame(medias_8[1])
medias8bis$modelo<-"svmRadial"
predi8<-as.data.frame(medias_8[2])
predi8$svmRadial<-predi8$Yes
```

```
summary(ensemble)
```

```
## The following models were ensembled: parrf, glm, gbm, svmlinear, svmPoly, svmradial
## They were weighted:
## 4.1057 -0.3249 0.4592 -2.7462 -1.4311 -1.5155 -3.104
## The resulting Accuracy is: 0.9468
## The fit for each individual model on the Accuracy is:
##     method  Accuracy  AccuracySD
##      parrf 0.8973276 0.013143839
##        glm 0.8899252 0.012870074
##        gbm 0.9378647 0.009223015
##  svmlinear 0.8885664 0.011650506
##    svmPoly 0.9256787 0.011685866
##  svmradial 0.9385687 0.009063826
```

Observaciones

Validacion cruzada repetida y boxplot

Siguiendo la guia de ensamblados entregada en el material, se realizan los pasos requeridos para realizar pruebas de ensamblado y validacion cruzada repetida

Paso 1, carga del archivo con la funcion "cruzadas ensamblado binaria fuente.R"

Paso 2, Preparacion del archivo

Se define semilla, variables y repeticiones

Paso 3, aplicacion de la funcion cruzadas ensamblado

Con los datos obtenidos del proceso de tunning se construyen las nuevas medias a evaluar bajo la funcion "cruzadas ensamblado binaria fuente.R".

Solo se adjunta una muestra del código generado

```
medias_1<-cruzadalogistica(data = archivo,
    vardep = vardep,
    listconti=listconti,
    listclass = listclass, grupos = grupos,
    sinicio = sinicio, repe = repe)

medias1bis <- as.data.frame(medias_1[1])
medias1bis$modelo<-"logistica"
predi1<-as.data.frame(medias_1[2])
predi1$logi<-predi1$Yes

medias_2<-cruzadaavnnetbin(data=archivo,
    vardep=vardep,listconti=listconti,
    listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
    size=c(10),decay=c(0.01),repeticiones=5,itera=200)

medias2bis<-as.data.frame(medias_2[1])
medias2bis$modelo<-"avnnet"
predi2<-as.data.frame(medias_2[2])
predi2$avnnet<-predi2$Yes

medias_3<-cruzadarfbin(data=archivo,
    vardep=vardep,listconti=listconti,
    listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
    mtry=3,ntree=500,nodesize=10,replace=TRUE)

medias3bis<-as.data.frame(medias_3[1])
medias3bis$modelo<-"randomforest"
predi3<-as.data.frame(medias_3[2])
predi3$rf<-predi3$Yes

# Se ha omitido el resto del codigo
```

Con la función genera graficos y una vez calculadas las medias en el proceso anterior se obtiene el grafico de cajas respectivo:

```
# Genera graficos
genera_graficos(medias1bis, medias2bis, medias3bis, medias4bis, medias5bis,
                medias6bis, medias7bis, medias8bis)
```

Paso 4, construccion del ensamblado

Con las predicciones obtenidas en las respectivas variables predi1, predi2, predi3 etc se crean los ensamblados. Solo se adjunta una muestra del codigo

```
unipredi<-cbind(predi1,predi2,predi3,predi4,predi5,predi6,predi7,predi8)
ncol(unipredi)

unipredi<- unipredi[, !duplicated(colnames(unipredi))]
ncol(unipredi)
```

```
unipredi$predi9<-(unipredi$logi+unipredi$avnnet)/2
unipredi$predi10<-(unipredi$logi+unipredi$rf)/2
unipredi$predi11<-(unipredi$logi+unipredi$gbm)/2
unipredi$predi12<-(unipredi$logi+unipredi$xgbm)/2
unipredi$predi13<-(unipredi$logi+unipredi$svmLinear)/2
unipredi$predi14<-(unipredi$logi+unipredi$svmPoly)/2
```

Paso 5, Procesado de los ensamblados

Solo se adjunta una parte del codigo, se construyen los promedios de tasa de fallos y AUC.

```
dput(names(unipredi))
listado<-c("logi", "avnnet",
           "rf","gbm",  "xgbm", "svmLinear",  "svmPoly",
           "svmRadial","predi9", "predi10", "predi11", "predi12",
           "predi13", "predi14", "predi15", "predi16", "predi17", "predi18",
           "predi19", "predi20", "predi21", "predi22", "predi23", "predi24",
           "predi25", "predi26", "predi27", "predi28", "predi29", "predi30",
           "predi31", "predi32", "predi33", "predi34", "predi35", "predi36",
           "predi37", "predi38", "predi39", "predi40", "predi41", "predi42",
           "predi43", "predi44", "predi45", "predi46", "predi47", "predi48",
           "predi49", "predi50", "predi51", "predi52", "predi53", "predi54",
           "predi55", "predi56", "predi57", "predi58", "predi59", "predi60",
           "predi61", "predi62", "predi63", "predi64", "predi65", "predi66",
           "predi67", "predi68", "predi69")

# Cambio a Yes, No, todas las predicciones
# Defino funcion tasafallos

tasafallos<-function(x,y) {
    confu<-confusionMatrix(x,y)
    tasa<-confu[[3]][1]
    return(tasa)
}

auc<-function(x,y) {
    curvaroc<-roc(response=x,predictor=y)
    auc<-curvaroc$auc
    return(auc)
}

# Se obtiene el numero de repeticiones CV y se calculan las medias por repe en
# el data frame medias0

repeticiones<-nlevels(factor(unipredi$Rep))
unipredi$Rep<-as.factor(unipredi$Rep)
unipredi$Rep<-as.numeric(unipredi$Rep)

medias0<-data.frame(c())
for (prediccion in listado)
{
    unipredi$proba<-unipredi[,prediccion]
    unipredi[,prediccion]<-ifelse(unipredi[,prediccion]>0.5,"Yes","No")
    for (repe in 1:repeticiones)
    {
        paso <- unipredi[(unipredi$Rep==repe),]
        pre<-factor(paso[,prediccion])
        archi<-paso[,c("proba","obs")]
        archi<-archi[order(archi$proba),]
        obs<-paso[,c("obs")]
        tasa=1-tasafallos(pre,obs)
        t<-as.data.frame(tasa)
        t$modelo<-prediccion
        auc<-suppressMessages(auc(archi$obs,archi$proba))
        t$auc<-auc
        medias0<-rbind(medias0,t)
    }
}
```

Paso 6, boxplot inicial

Boxplot con tasa de fallos para todos los ensamblados, en el punto 8 se muestran ordenados

Paso 7, tabla con resultados

Se genera tabla con resultados para la tasa de fallos y el area under curve. **Nota:** Solo se despliegan las primeras salidas para no extender el documento.

```
tablamedias<-medias0 %>%
    group_by(modelo) %>%
    summarise(tasa=mean(tasa))

tablamedias<-as.data.frame(tablamedias[order(tablamedias$tasa),])
knitr::kable(head(tablamedias, n = 10), "pipe")
```

modelo	tasa
predi55	0.0641826
predi16	0.0650554
rf	0.0652904
predi57	0.0656260
predi26	0.0663310
predi60	0.0663646
predi63	0.0663981
predi69	0.0667338
predi52	0.0670695

modelo	tasa
predi56	0.0678751

```
# Para AUC
tablamedias2<-medias0 %>%
    group_by(modelo) %>%
    summarise(auc=mean(auc))

tablamedias2<-tablamedias2[order(-tablamedias2$auc),]
knitr::kable(head(tablamedias2, n=10), "pipe")
```

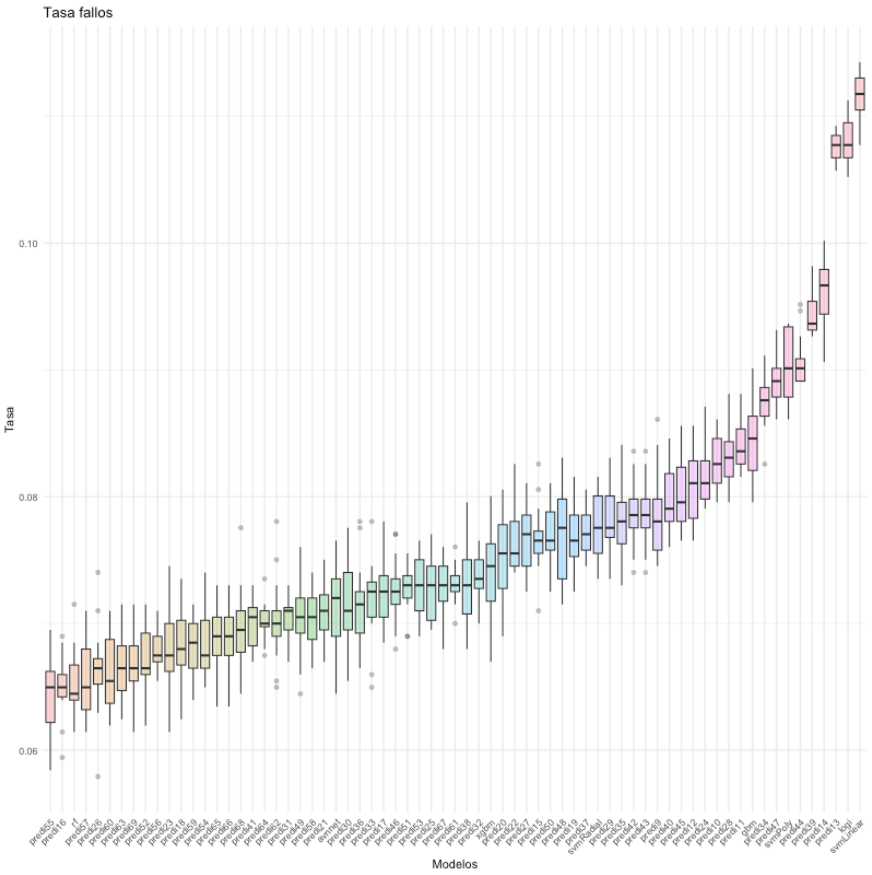
modelo	auc
predi55	0.9789378
predi57	0.9786039
predi52	0.9782842
predi60	0.9782593
predi23	0.9780586
predi69	0.9779774
predi56	0.9778908
predi26	0.9778800
predi16	0.9777369
predi54	0.9775005

En el punto siguiente se grafica para una mejor visualizacion y toma de decision

Paso 8, Boxplot ordenados para comparacion de medias

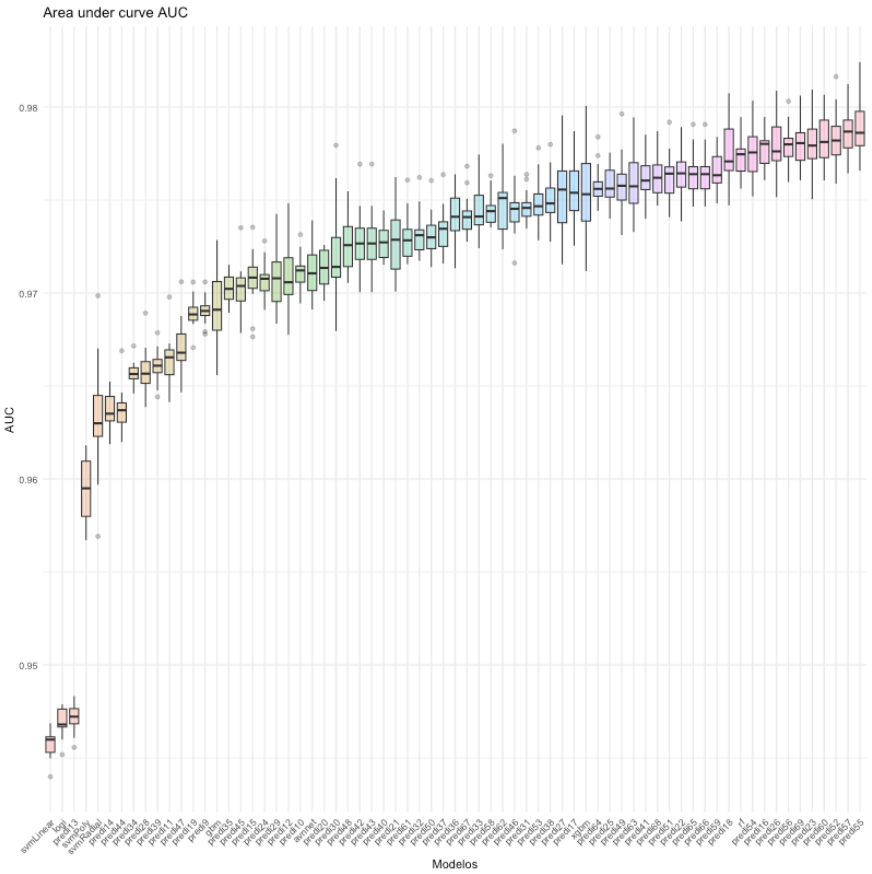
Con los resultados obtenidos de los ensamblados se generan los boxplot ordenados por tasa de fallos y auc respectivamente:

```
knitr::include_graphics("./figure/ensamblado_medias0_tasa.png")
```



En tasa de fallos los modelos con menor tasa son predi55, predi16, randomforest

```
knitr::include_graphics("./figure/ensamblado_medias0_auc.png")
```

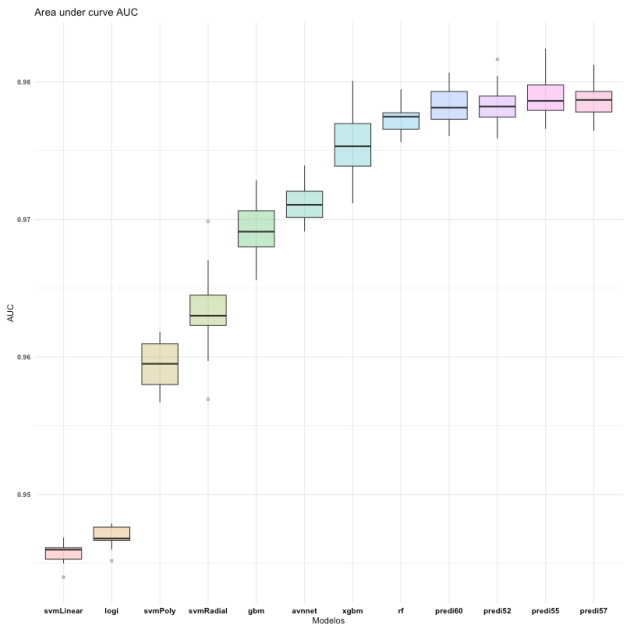


En AUC respectivamente los modelos con mayor *accuracy* son predi55, predi57 y predi52.

Como se observa en el grafico son demasiados modelos, en el apartado siguiente solo se grafican los mejores modelos de ensamblado y se comparan con los alfgoritmos sin ensamblar

Paso 9, comparacion de mejores modelos

Los mejores ensamblados son los modelos predi57, predi55, predi52 y predi 60, estos se muestran a continuacion comparandolos con los modelos originales sin emsamblar



Paso 10, revision a los mejores ensamblados

```
unipredi$predi55<-(unipredi$rf+unipredi$xgbm+unipredi$svmRadial)/3
unipredi$predi57<-(unipredi$rf+unipredi$avnnet+unipredi$xgbm)/3
unipredi$predi60<-(unipredi$rf+unipredi$avnnet+unipredi$svmRadial)/3
unipredi$predi52<-(unipredi$rf+unipredi$gbm+unipredi$svmRadial)/3
```

En los ensamblados el algoritmo común en los cuatro mejores ensamblados es random forest, el mejor ensamblado es el numero 55 que mezcla random forest, xgbm y svm radial.

Observaciones finales

- De los modelos originales los mejores resultados observando el grafico los obtiene random forest, xgbm y avnnet, xgbm presenta mas varianza que los dos anteriores.
- El *accuracy* alcanzado por los ensamblados es mas alto si se compara con los modelos originales sin ensamblar. Si solo fuese tomando como criterio el accuracy, predi57 que es un ensamblado de randomforest con xgbm y svmlineal seria el modelo *ganador*. Sin embargo no hay diferencias dramáticas desde randomforest hacia adelante y considerando sesgo-varianza, randomforest estaria mostrando mejores resultados que los ensamblados.
- La tasa de fallos vs el auc muestra algunas incoherencias, en tasa de fallos el numero menor lo obtienen predi55 y predi16 pero en auc los ensamblados predi55 y predi57 respectivamente estan a la cabecera de los resultados.