# Proyecto final Aprendizaje de Máquina: Modelo predictivo para la base de datos "Speed dating" de Kaggle

Dalia Camacho, Gabriela Vargas, Paulina Gómez Mont

Noviembre 2018

## Descripción de la base de datos

En este proyecto, se realizará un modelo predictivo para la base de datos de Kaggle "Speed Dating", la cual corresponde a un experimento realizado por investigadores de Columbia Business School para el documento titulado "Gender differences in mate selection: Evidence from a speed dating experiment" [1].

La información se recopiló a partir de rondas de citas rápidas llevadas a cabo entre los años 2002 y 2004. Cada participante del experimento se involucraba en una cita de cuatro minutos con otro participante del sexo opuesto y al final del encuentro se les preguntaba si estaban interesados en tener una segunda cita.

La base de datos incluye cuestionarios aplicados a los participantes en diferentes momentos del proceso: antes de la cita, después de la cita y 3-4 semanas después de la cita. Cada cuestionario requería que el participante respondiera preguntas acerca de su percepción respecto a sí mismo y al sexo opuesto en cuanto a seis atributos.

## Limpieza de datos

El tamaño inicial de la base es de $8,378$ observaciones y 195 variables. El primer paso fue eliminar variables que pudieran filtrar información sobre el *match*[1] entre dos personas. Para lograr esto eliminamos todas aquellas variables obtenidas una vez iniciado el experimento. Además eliminamos algunas variables relacionadas con la estructura del experimento de número de *id* en cada ola, el orden de las citas, la ronda, el identificador por género. Sin embargo conservamos el identificador único de cada individuo y la ola a la que pertenece para así poder obtener los conjuntos de entrenamiento, validación y prueba sin filtrar información.

Otras variables que se eliminaron fueron *career*, *career_c* y *field*, ya que están muy relacionadas con la variable *field_cd* que indica el área profesional en la que se desarrolla el individuo de acuerdo a un código predefinido. Eliminamos otras variables como *income*, *undergrad* y *tuition* ya que no se tienen observaciones para todas las encuestas. Así mismo solamente utilizamos variables que hayan sido obtenidas en todas las olas. A partir de estas modificaciones nos queda una base con 52 variables.

Después eliminamos las variables con más de 200 valores faltantes, estas variables fueron *zipcode*, *expnum* y *met* y nos quedan 49 variables. A partir de las variables restantes eliminamos las observaciones con datos faltantes, por lo que nos quedan 8164 observaciones. La descripción de las variables que utilizamos se encuentra en la Tabla 2 en el apéndice.

El siguiente paso fue escalar las variables. Aquellas variables relacionadas con el gusto por ciertas actividades y la satisfacción que uno creía tener durante la cita, así como la importancia que se le da a la raza y a la religión se dividieron entre diez, ya que estaban en una escala de cero a diez. La variable de edad la dividimos entre 60, como el valor máximo para la variable edad era 55 todas las observaciones se mantienen entre cero y uno. A la variable relacionada con la frecuencia con que el individuo acude a citas se le restó 8, se multiplicó

[1] La variable *match* fue nuestra variable de interés

por $-1$ y se dividó entre 7, esto con la finalidad de darle un mayor valor a aquellas personas que acuden más frecuentemente a citas.

A lo largo de las encuestas hubo dos formas distintas de obtener valores para las variables relacionadas con los atributos que uno desea en una pareja, los atributos que uno considera tener y lo que uno cree que el sexo opuesto busca respecto a esos atributos. En las olas de la uno a la cinco y de la 10 a la 21 se pidió a los participantes distribuir 100 puntos a través de los 6 atributos según su preferencia. Mientras que en las olas de la 6 a la 9 se le pidió asignar un puntaje entre uno y diez según qué tanto valoraban los atributos. Para poder utilizar la información de todas las olas dividimos el puntaje para cada atributo entre la suma total de los atributos.

Buscamos si existía una correlación entre los distintos hobbies y entre las distintas expectativas. Si la correlación era mayor a 0.8 consideramos incluir la interacción, únicamente agregamos la interacción entre gusto por el arte y por los museos.

La variable relacionada con el lugar de origen es de tipo categórica con 266 distintos valores. Condensamos esta variable en cinco regiones: *USA/Canada*, *Latin America*, *Europe*, *Asia* y *Other/Unknown*.

Dado que el *match* depende de ambas personas agregamos a cada observación los valores correspondientes a su pareja. Como las parejas de algunos individuos se habían eliminado previamente hay entradas con datos faltantes, por lo que nuevamente seleccionamos únicamente los casos completos. Esto nos da un total de $7,962$ observaciones.

Transformamos a variables tipo *dummy* todas aquellas que eran originalmente categóricas, para así poder hacer uso de modelos basados en elementos de tipo numérico.

Finalmente, agregamos variables que representan la diferencia entre las expectativas de una persona y la apreciación que tiene la otra sobre sí misma; la diferencia entre las expectativas de una persona y lo que la otra cree que el sexo opuesto busca; la diferencia de edades; y la diferencia en valor absoluto del gusto por las distintas actividades. Con todos estos cambios nuestra base final tiene $7,962$ observaciones y 194 variables.

## Partición de datos

Los datos los dividimos en un conjunto de entrenamiento, uno de validación y uno de prueba. Para evitar filtración de datos de los conjuntos de validación y prueba al conjunto de entrenamiento, no dividimos los conjuntos por muestreo de observaciones, si no por muestreo de las olas. De esta forma ningún individuo estará en dos conjuntos y las condiciones no controladas en cada experimento no afectarán a la estimación de error de clasificación en la muestra de prueba.

De forma aleatoria elegimos nueve olas para el conjunto de entrenamiento, seis para el de validación y seis para el de prueba. El número de observaciones en cada conjunto se muestra en la Tabla 1.

| **Conjunto** | **Olas** | **Observaciones** |
|---|---|---|
| Entrenamiento | 1 3 4 7 9 13 14 20 21 | 4,108 |
| Validación | 2 6 8 10 11 19 | 2,274 |
| Prueba | 5 12 15 16 17 18 | 1,580 |

Table 1: Olas elegidas para cada muestra y número de observaciones

## Modelos

El objetivo del proyecto es predecir un *match* entre dos individuos a partir de características tales como edad, género, raza, área de desarrollo profesional. Además de las características se consideran seis atributos: el atractivo, la inteligencia, la ambición, la sinceridad y los intereses en común. Cada individuo en la muestra

señala la importancia que le da a cada atributo en un individuo del sexo opuesto, los atributos que considera que el sexo opuesto busca y la apreciación que tiene de sí mismo respecto a los primeros cinco atributos.

Como *benchmark* utilizaremos regresión logística. Además emplearemos modelos con redes neuronales, bosques aleatorios y *gradient boosting*. Utilizaremos t-SNE para intentar ver si reduciendo el espacio a dimensión dos y considerando la distancia entre las observaciones es posible distinguit un grupo en el que se obtenga un *match* de otro grupo. Debido a que el número de observaciones con *match* positivo es únicamente el 16.43% de la muestra también utilizaremos distintas técnicas de *oversampling* y *down-sampling*.

## Regresión logística

Primero generamos un modelo de regresión logística sin regularización, para observar lo que ocurre con el modelo más simple.

```
modeloLog1 <- glmnet(x=Xentrena, y=YEntrena, family = "binomial",
                     intercept = FALSE, lambda =0 )
```

```
PredEntrena <- predict(modeloLog1, Xentrena, type = "response")
devianza(PredEntrena,YEntrena)
```

La devianza para la muestra de entrenamiento es 0.846.

```
PredMatch   <- as.numeric(predict(modeloLog1, Xentrena, type = "class"))
1-ErrClass(PredMatch,YEntrena)
```

La tasa de clasificación correcta para la muestra de entrenamiento es 0.829.

```
PredPrueba <- predict(modeloLog1, XPrueba, type = "response")
devianza(PredPrueba,YPrueba)
```

La devianza para la muestra de prueba es 0.946.

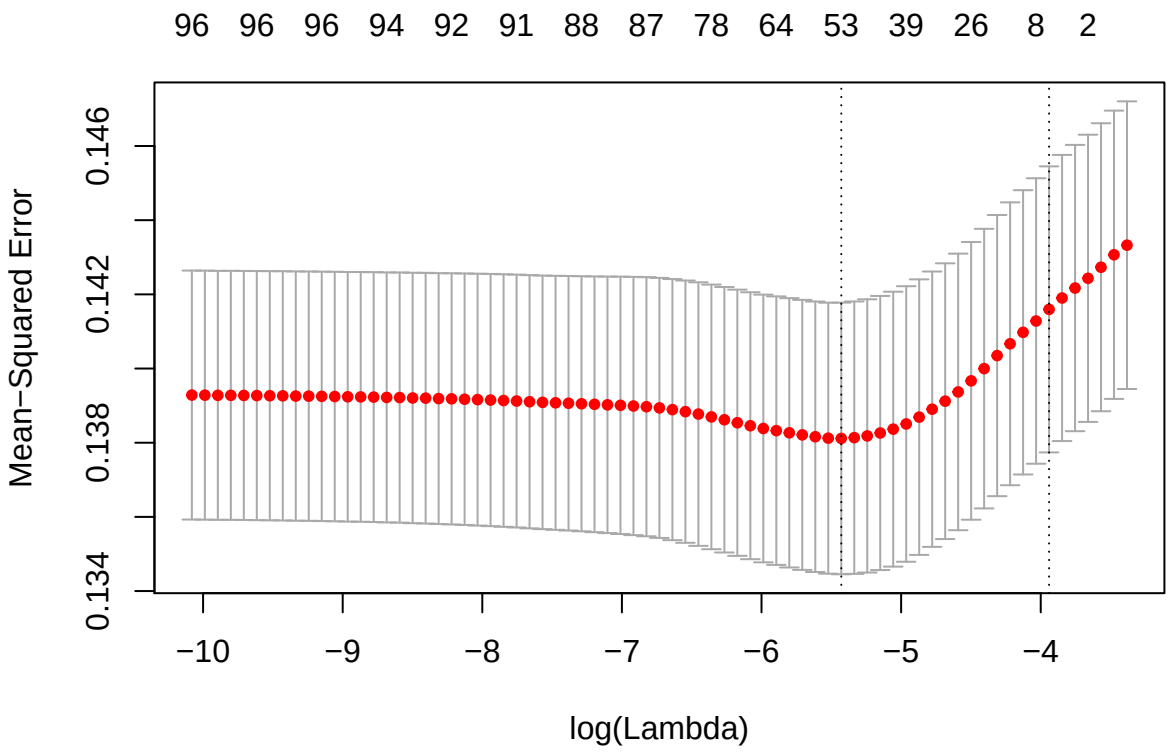
```
PredMatchPrueba   <- as.numeric(predict(modeloLog1, XPrueba, type = "class"))
1-ErrClass(PredMatchPrueba,YPrueba)
```

Para el modelo de regresión logística sin regularización la tasa de clasificación correcta para el conjunto de prueba es 0.810, lo cual resulta en una peor predicción que el modelo en el que se le asigna *match* cero a todas las variables. Ya que para el modelo de puros ceros la tasa de clasificación correcta en el modelo de prueba es de 0.835.

### Agregando regularización

Ahora agregamos regularización Lasso para disminuir el número de variables y mejorar el modelo para el conjunto de prueba.

```
modeloLogLasso <- cv.glmnet(x = Xentrena, y = YEntrena, alpha = 1) #alpha=1 para lasso
plot(modeloLogLasso)
```




```
LambdaMin <- modeloLogLasso$lambda.min
Lambda1se <- modeloLogLasso$lambda.1se
```

### Validación del modelo con dos valores para $\lambda$

A partir de validación cruzada se obtuvieron la $_{min}$ con error mínimo de validación cruzada y otra $\lambda_{1se}$ para el modelo más simple a un error estándar del mejor modelo.

**Lambda Min** Primero probamos con la $\lambda_{min}$

```
modeloLogLassoLambdamin <- glmnet(x=Xentrena, y=YEntrena, family = "binomial",
                     intercept = FALSE, alpha = 1, lambda = LambdaMin )
```

```
PredEntrena <- predict(modeloLogLassoLambdamin, Xentrena, type = "response")
devianza(PredEntrena,YEntrena)
```

La deivanza de entrenamiento es 0.863.

```
PredMatch   <- as.numeric(predict(modeloLogLassoLambdamin, Xentrena, type = "class"))
1-ErrClass(PredMatch,YEntrena)
```

La tasa de clasificación correcta para el conjunto de entrenamiento es 0.828.

```
PredValida <- predict(modeloLogLassoLambdamin, XValida, type = "response")
devianza(PredValida,YValida)
```

La devianza de validación es 0.906.

```
PredMatchValida   <- as.numeric(predict(modeloLogLassoLambdamin, XValida, type = "class"))
1-ErrClass(PredMatchValida,YValida)
```

El error de clasificación con la muestra de validación es 0.8236588.

### Lambda 1se

Ahora probamos con $\lambda_{1se}$.

```
modeloLogLassoLambda1se <- glmnet(x=Xentrena, y=YEntrena, family = "binomial",
                      intercept = FALSE, alpha = 1, lambda = Lambda1se )
```

```
PredEntrena <- predict(modeloLogLassoLambda1se, Xentrena, type = "response")
devianza(PredEntrena,YEntrena)
```

La devianza de entrenamiento es 0.904.

```
PredMatch   <- as.numeric(predict(modeloLogLassoLambda1se, Xentrena, type = "class"))
1-ErrClass(PredMatch,YEntrena)
```

La tasa de clasificación correcta para el conjunto de entrenamiento es 0.827.

```
PredValida <- predict(modeloLogLassoLambda1se, XValida, type = "response")
devianza(PredValida,YValida)
```

La devianza para el conjunto de validación en 0.851.

```
PredMatchValida   <- as.numeric(predict(modeloLogLassoLambda1se, XValida, type = "class"))
1-ErrClass(PredMatchValida,YValida)
```

La tasa de predicción de correctos en el conjunto de validación es 0.851.

Como la tasa de predicción de correctos fue mayor para $\lambda_{1se}$ que para $\lambda_{min}$, nos quedamos con $\lambda_{1se}$ para probar el modelo con la muestra de prueba.

### Muestra de Prueba

Generamos las predicciones con el modelo de regresión logística con regularización Lasso y $\lambda_{1se}$.

```
PredPrueba <- predict(modeloLogLassoLambda1se, XPrueba, type = "response")
devianza(PredPrueba, YPrueba)
```

La devianza en el conjunto de prueba es 0.885,

```
PredMatchPrueba   <- as.numeric(predict(modeloLogLassoLambda1se, XPrueba, type = "class"))
1-ErrClass(PredMatchPrueba,YPrueba)
```

La tasa de clasificación correcta en el conjunto de prueba es 0.838.

```
table(YPrueba, PredMatchPrueba)
```

La matriz de confusión usando el modelo de regresión logística para el conjunto de prueba queda de la siguiente manera:

```
       Predichos
Obs        0    1
      0 1318    2
      1  254    6
```

Podemos ver que el modelo obtenido con regresión logística y regularización Lasso no logra separar el conjunto de observaciones que contienen un *match*. La tasa de falsos positivos es de 0.977 y la tasa de verdaderos positivoes es 0.998. Casi todas las observaciones se clasifican como cero respecto al *match*. El modelo en que se clasifican todas las observaciones como cero tiene una tasa de clasificación correcta igual a 0.835, por lo que regresión logística sólo merjora ligeramente la clasificación.

## Redes neuronales

Intentamos con varios modelos utilizando redes neuronales, empezamos con modelos más pequeños y terminamos con un modelo con 3 capas ocultas, cada una con 100 neuronas y activación sigmoide. En otros modelos probamos con activaciones tipo ReLU, sin embargo el resultado es el mismo que obtenemos en este caso.

```
set.seed(29)
modeloRed1 <- keras_model_sequential()
modeloRed1 %>%

  layer_dense(units = 100,
              activation = 'sigmoid',
              kernel_regularizer = regularizer_l1(l = 1e-3),
              input_shape = 190) %>%

  layer_dropout(0.5) %>%
  layer_dense(units = 100,
              activation = 'sigmoid',
              kernel_regularizer = regularizer_l1(l = 1e-3)) %>%
  layer_dropout(0.5) %>%
  layer_dense(units = 100,
              activation = 'sigmoid',
              kernel_regularizer = regularizer_l1(l = 1e-3)) %>%
  layer_dense(units = 1,
              activation = 'sigmoid',
              kernel_regularizer = regularizer_l1(l = 1e-3))

modeloRed1 %>% compile(loss = 'binary_crossentropy',
                       optimizer = optimizer_sgd(lr = 0.1, momentum = 0.2,
                                                 decay = 0),
                       metrics = c('accuracy')
)
```

Ajustamos el modelo para los datos de entrenamiento y con el conjunto de validación, hicimos 50 épocas y *batches* de tamaño 32 .

```
history <- modeloRed1 %>%
  fit(Xentrena, YEntrena,
      epochs = 50, batch_size = 32,
      validation_data = list(XValida, YValida), verbose=3)
```

En la Figura 1 podemos observar que a través de las épocas la tasa de clasificación de correctos se mantiene para los conjuntos de validación y prueba. Mientras que la devianza a partir de la época 10 no tiene mejoría evidente.

Observamos los resultados del ajuste del modelo a partir de la matriz de confusión para los conjuntos de entrenamieno, validación y prueba.

```
tab_confusion <- table(modeloRed1 %>% predict_classes(Xentrena),YEntrena)
tab_confusion
```

Tabla de Confusión para datos de entrenamiento:

```
       Observados
Pred       0    1
   0    3396  712
```

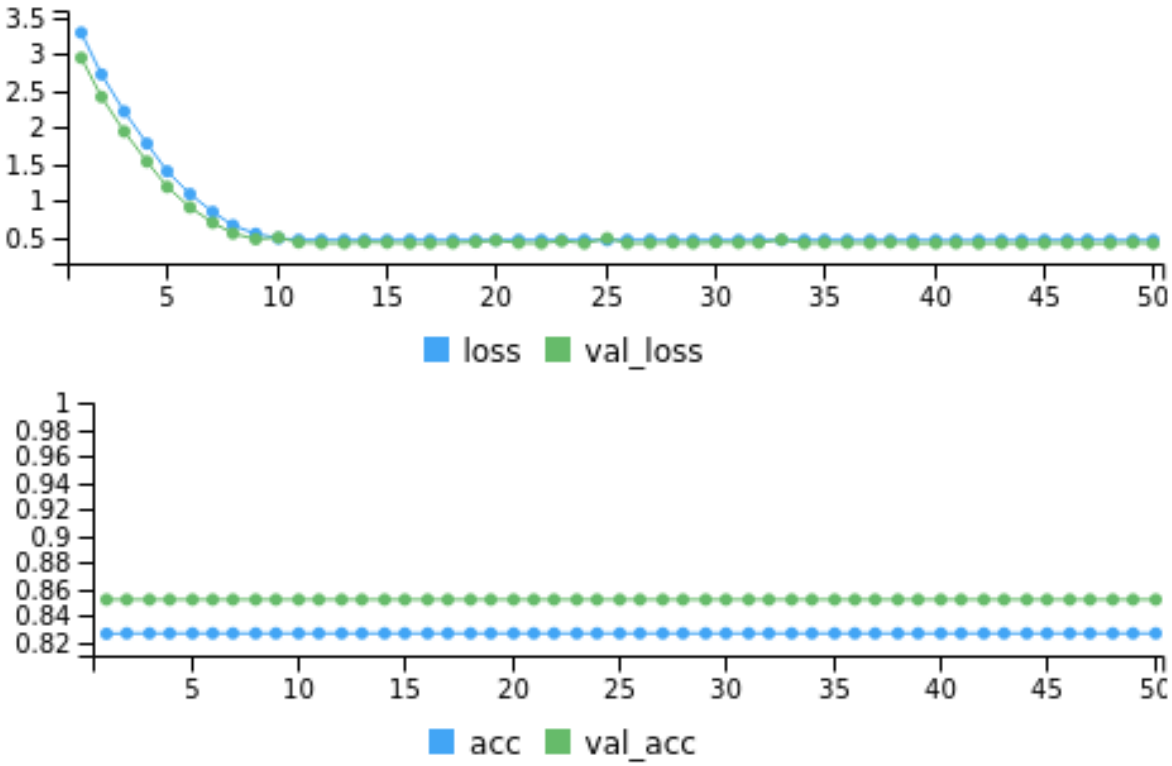



Figure 1: Ajuste del modelo en términos de devianza y precisión.

```
tab_confusion <- table(modeloRed1 %>% predict_classes(XValida),YValida)
tab_confusion
```

Tabla de Confusión para datos de validación:

```
        Observados
Pred      0    1
 0      1938  336
```

```
tab_confusion <- table(modeloRed1 %>% predict_classes(XPrueba),YPrueba)
tab_confusion
```

Tabla de confusión para datos de prueba.

```
         Observados
Pred       0    1
0         1320  260
```

En todos los casos la clasificación es siempre hacia la clase cero, por lo que en cuanto a clasificación. Para este conjunto de datos los modelos con redes neuronales no dan buenos resultados y son equivalentes a clasificar todas las observaciones con *match* igual a cero.

## Modelos basados en árboles

En esta sección utilizamos métodos basados en árboles para intentar resolver el problema de clasificación. Utilizamos bosques aleatorios y *gradient boosting*.

### Árbol

Para darnos una idea de cómo se van generando los árboles mostramos las primeras ramas del árbol que se construye sobre los datos de entrenamiento. Indicamos $\alpha = 0.004$, donde $\alpha$ es el parámetro de penalización sobre la complejidad del árbol. Estas primeras ramas del árbol se pueden observar en la figura 2.

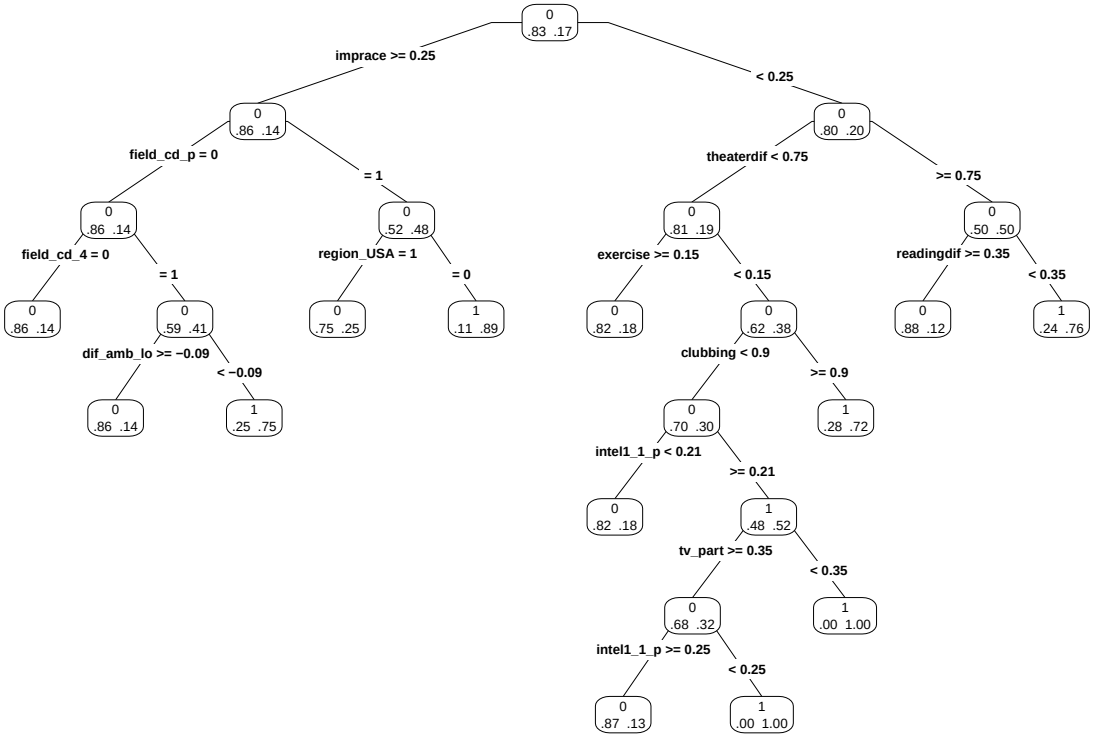



Figure 2: Primeras Ramas del árbol de clasificación binaria para el problema de *speed dating*.

### Bosques aleatorios

Para evitar el sobreajuste en vez de evaluar los modelos sobre un sólo árbol decidimos correr bosques aleatorios para $1,000$ árboles y 10 variables candidatas en cada corte.

```
bosque_SD <-randomForest(factor(match) ~ ., data = m_e,ntree = 1000, mtry = 10,
importance=TRUE)
```

Generamos la matriz de confusión para poder observar qué tan bien se ajusta el modelo a los datos de entrenamiento.

```
bosque_SD$confusion
```

```
##      0  1  class.error
## 0 4451  3 0.0006735519
## 1  881 15 0.9832589286
```

Los modelos con redes neuronales no predecían la clase de *match* igual a uno. Por lo tanto los bosques aleatorios son un mejor modelo al menos para datos de entrenamiento. Ahora probamos el modelo con datos de validación.

```
probas <- predict(bosque_SD, newdata = m_v, type='prob')
head(probas)
```

```
##       0     1
## 1 0.769 0.231
## 2 0.856 0.144
## 3 0.768 0.232
## 4 0.819 0.181
## 5 0.873 0.127
## 6 0.826 0.174
```

Obtenemos la matriz de confusión para un corte en 0.225, es decir aquellas observaciones con probabilidad mayor a 0.225 se les asignó el valor de 1.

```
prop_bosque <- probas[,2]
tabla_conf <- table(prop_bosque> 0.225, m_v$match)
#tabla_conf
prop.table(tabla_conf,2)
```

```
##
##                0        1
##   FALSE 0.655000 0.631068
##   TRUE  0.345000 0.368932
```

Ahora obtenemos un resumen de las probabilidades obtenidas con el modelo.

```
c <- which(m_v$match==1)
summary(probas[c,2])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0850  0.1700  0.2070  0.2072  0.2422  0.3650
```

```
summary(probas[-c(c),2])
```

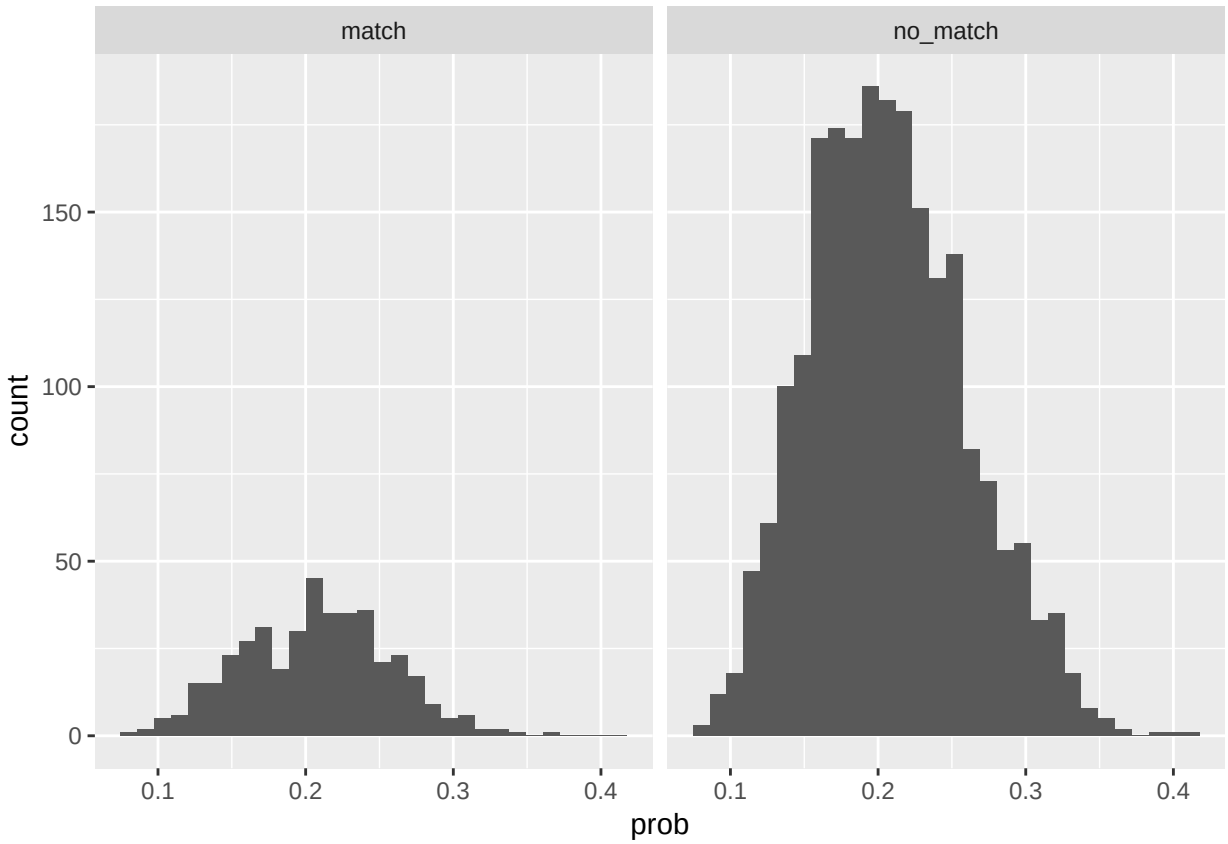
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0750  0.1680  0.2030  0.2072  0.2420  0.4070
```

Las tablas que mostramos anteriormente nos indican que la probabilidad de tener un *match* es similar independientemente de si el valor observado fue o no un *match*. Esto puede reflejar que los conjuntos son indistinguibles. Más aún con corte de clasificación en 0.225 la tasa de falsos positivos es mayor que la de verdaderos positivos.

Ahora hacemos un histograma en el que se muestran las probabilidades de *match* para el conjunto que hizo *match* y el que no.

```
m_v_graf <- m_v
m_v_graf$c <- "no_match"
m_v_graf$c[c] <-"match"
m_v_graf$prob <- probas[,2]

ggplot(m_v_graf)+geom_histogram(aes(prob))+facet_wrap(~c)
```




Nuevamente el histograma de las probabilidades nos muestra que usando bosques aleatorio no podemos separar las observaciones de parejas que hicieron *match* y las que no.

### Bosques aleatorios con ajuste de costos

Intentamos obtener un mejor modelo añadiendo una matriz de costos con la que no se castigan las observaciones clasificadas correctamente, se le da un costo igual a 10 cada falso negativo y con un costo de 1 cada falso positivo. Nuevamente generamos $1,000$ árboles con 10 variables en cada árbol.

```
library(tidyverse)
library(readr)
library(randomForest)

costMatrix <- matrix(c(0,10,1,0), nrow=2)

bosque_SD <-randomForest(factor(match) ~ ., data = m_e,
ntree = 1000, mtry = 10, importance=TRUE, parms = list(loss=costMatrix))



probas <- predict(bosque_SD, newdata = m_v, type='prob')
head(probas)
```

```
##       0     1
## 1 0.787 0.213
## 2 0.836 0.164
## 3 0.758 0.242
## 4 0.832 0.168
```

```
## 5 0.841 0.159
## 6 0.838 0.162
```

Generamos la matriz de confusión de forma análoga al modelo de bosques aleatorios obtenido previamente.

```
prop_bosque <- probas[,2]
tabla_conf <- table(prop_bosque> 0.225, m_v$match)
#tabla_conf
prop.table(tabla_conf,2)
```

```
##
##                 0         1
##   FALSE 0.6736364 0.6286408
##   TRUE  0.3263636 0.3713592
```

Obtenemos los valores resumen de las probabilidades de *match* obtenidas con el bosque aleatorio con costos para ambas categorías.

```
c <- which(m_v$match==1)
summary(probas[c,2])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0780  0.1690  0.2065  0.2064  0.2420  0.3490
```

```
summary(probas[-c(c),2])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0860  0.1680  0.2020  0.2055  0.2390  0.4050
```

Nuevamente mostramos los histogramas de las probabilidades que se obtienen en cada categoría.

```
m_v_graf <- m_v
m_v_graf$c <- "no_match"
m_v_graf$c[c] <-"match"
m_v_graf$prob <- probas[,2]

ggplot(m_v_graf)+geom_histogram(aes(prob))+facet_wrap(~c)
```

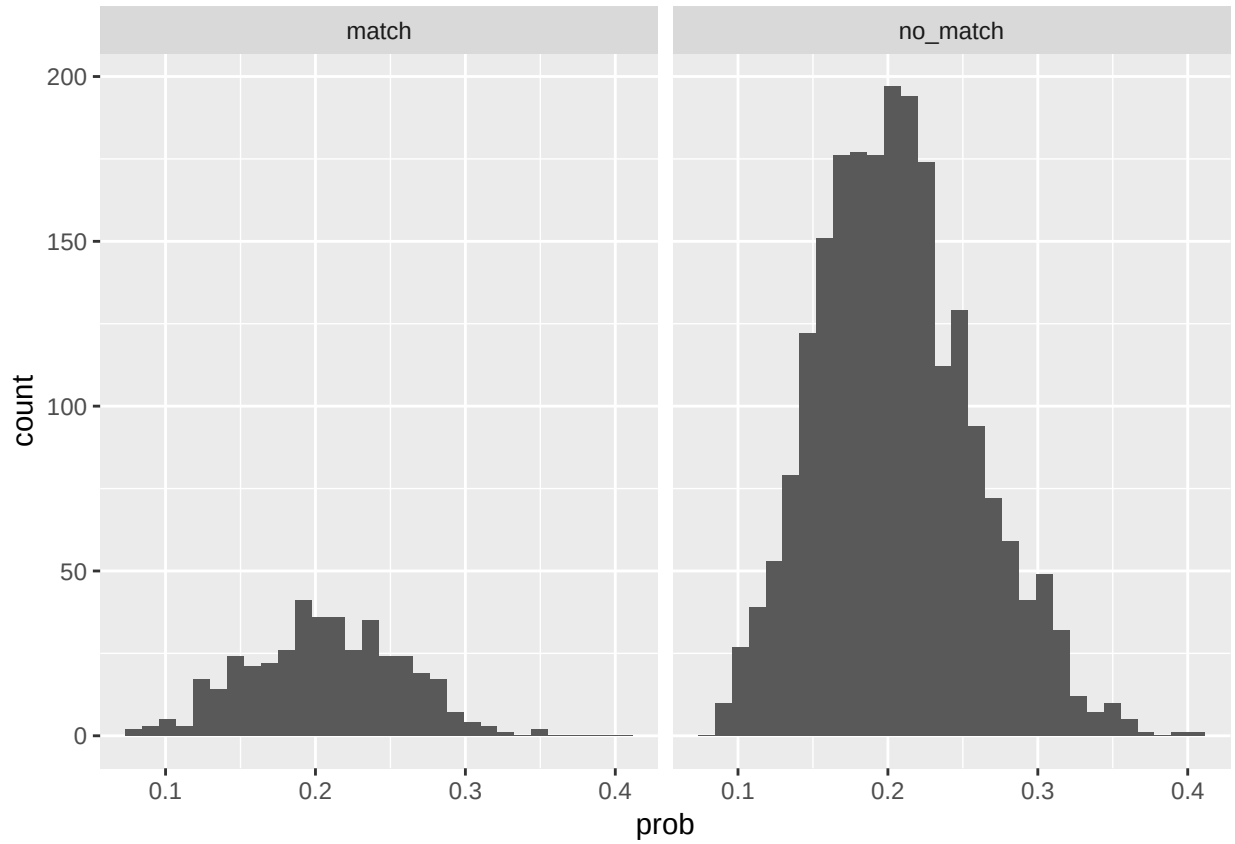
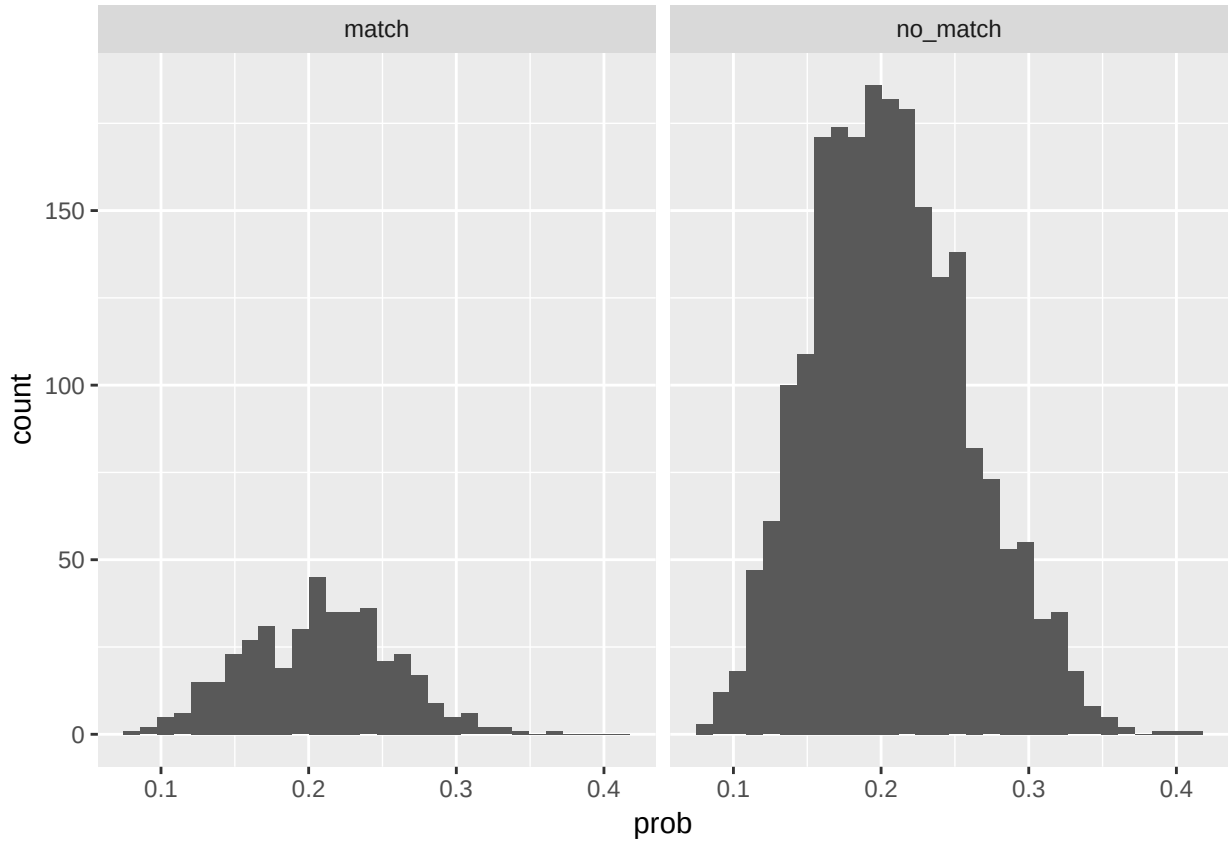



Figure 3: Caption



Con el ajuste de costos vemos una mejora en la tabla de confusión. Sin embargo, como podemos ver en el histograma aún no logramos encontrar una manera de separar claramente a las parejas que hacen match contra las que no.

### Gradient Boosting

Continuando con los métodos basados en árboles probamos el método *gradient boosting* con la librería `xgboost`.

```
library(xgboost)
```

Definimos la profundidad máxima de los árboles y lo corrimos $1,500$ rondas.

```
set.seed(1293)
d_entrena <- xgb.DMatrix(x_ent_s, label = y_ent)
d_valida <- xgb.DMatrix(x_valid_s, label = y_valid)
watchlist <- list(eval = d_valida, train = d_entrena)
params <- list(booster = "gbtree",
               max_depth = 3,
               eta = 0.03,
               nthread = 1,
               subsample = 0.75,
               lambda = 0.001,
               objective = "binary:logistic",
               eval_metric = "mae")
bst <- xgb.train(params, d_entrena, nrounds = 1500, watchlist = watchlist, verbose=0)
probas <- predict(bst, d_valida, type='prob')
prop_bosque <- probas
tabla_conf <- table(prop_bosque> 0.16, m_v$match)
prop.table(tabla_conf,2)
```

Obtuvimos la matriz de confusión y podemos observar que la tasa de verdaderos positivos es 0.405 lo cual mejora los modelos previos.

```
##
##                  0         1
##   FALSE 0.6854545 0.5946602
##   TRUE  0.3145455 0.4053398
```

Podemos observar que la tabla de confusión muestra una mejora en cuanto al recall y el accuracy si los comparamos contra los modelos obtenidos con el método de bosques aleatorios.

Nuevamente obtenemos el histograma de las probabilidades para este modelo. Este histograma está en la Figura 4

```
m_v_graf <- m_v
m_v_graf$c <- "no_match"
m_v_graf$c[c] <-"match"
m_v_graf$prob <- probas

ggplot(m_v_graf)+geom_histogram(aes(prob))+facet_wrap(~c)
```

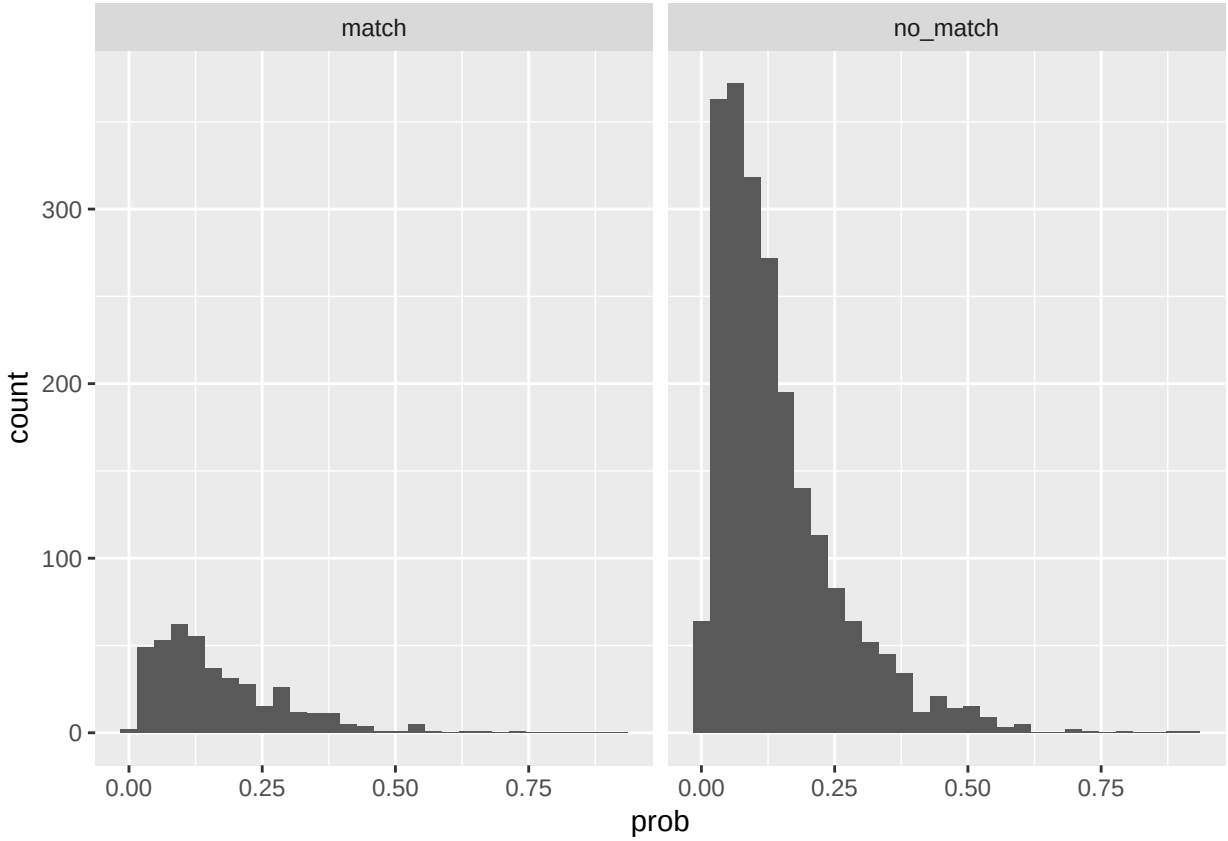



Figure 4:

### Gradient boosting con costos

Siguiendo el ejemplo de bosques aleatorios con costos en la sección  consideramos correr un modelo de *gradient bossting* con los mismos costos.

```
library(xgboost)


set.seed(1293)
costMatrix <- matrix(c(0,10,1,0), nrow=2)
d_entrena <- xgb.DMatrix(x_ent_s, label = y_ent)
d_valida <- xgb.DMatrix(x_valid_s, label = y_valid)
watchlist <- list(eval = d_valida, train = d_entrena)
params <- list(booster = "gbtree",
               max_depth = 3,
               eta = 0.03,
               nthread = 1,
               subsample = 0.75,
               lambda = 0.001,
               objective = "binary:logistic",
               eval_metric = "mae",
               loss=costMatrix)
bst <- xgb.train(params, d_entrena, nrounds = 1500, watchlist = watchlist, verbose=0)
probas <- predict(bst, d_valida, type='prob')
```

```
prop_bosque <- probas
tabla_conf <- table(prop_bosque> 0.16, m_v$match)
prop.table(tabla_conf,2)
```

Ahora observamos los resultados de este modelo, los cuales son muy similares al modelo con *gradient boosting* sin costos.

```
##
##                  0         1
##   FALSE 0.6854545 0.5946602
##   TRUE  0.3145455 0.4053398
```

Nuevamente generamos el histograma con las probabilidades, el cual se encuentra en la Figura 5

```
m_v_graf <- m_v
m_v_graf$c <- "no_match"
m_v_graf$c[c] <-"match"
m_v_graf$prob <- probas

ggplot(m_v_graf)+geom_histogram(aes(prob))+facet_wrap(~c)
```

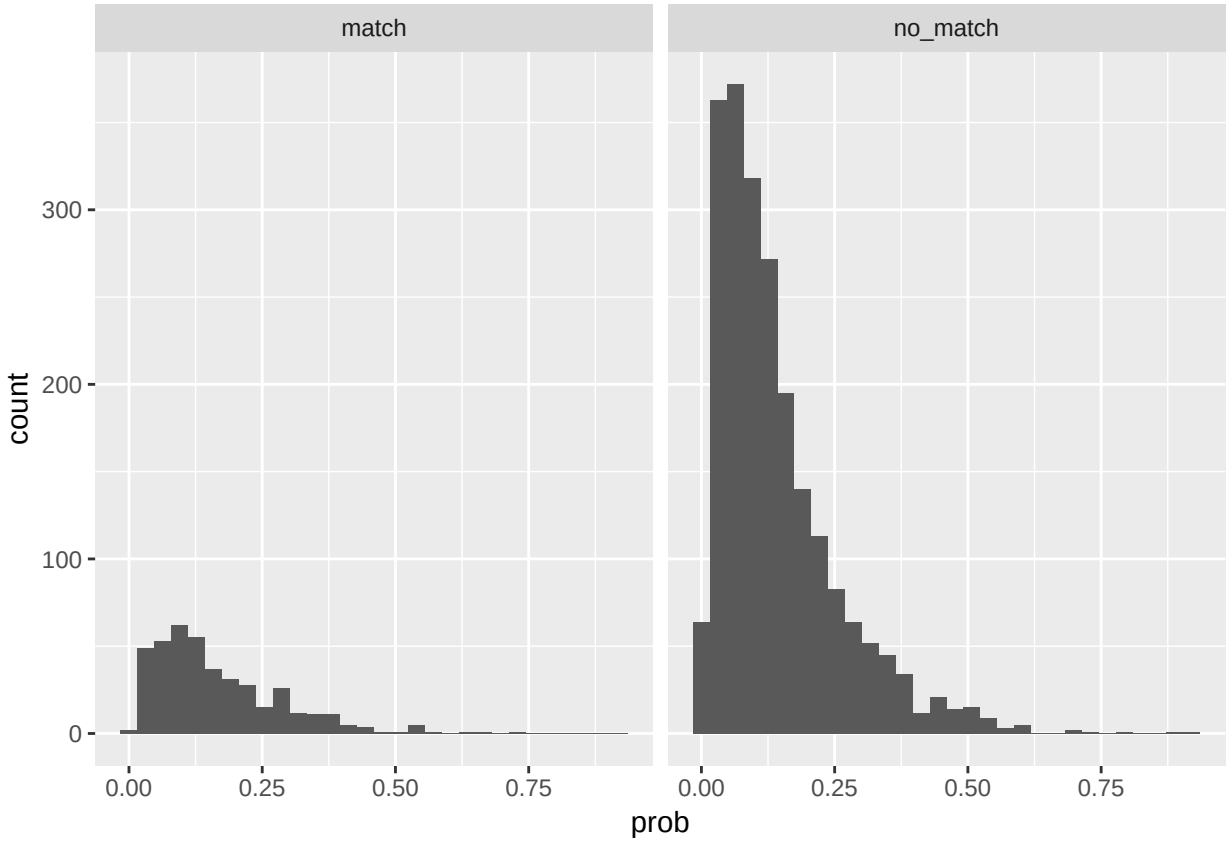



Figure 5:

A partir de los modelos basados en árboles pudimos reconocer de una forma más explícita que los conjuntos de observaciones que tuvieron *match* y los que no no son separables. Mostramos esto más claramente con t-SNE un método de aprendizaje no supervisado. Además como último recurso probamos métodos de *down-sampling* y *over-sampling*.

## Problema observado en los modelos y acciones para corregirlo

Al implementar los modelos mencionados anteriormente y hacer experimentos cambiando el número de variables independientes, añadiendo interacciones y cambiando el valor de los parámetros, observamos que todos los modelos son similares en términos de sensibilidad *vp/pos* y especificidad *vn/neg*. En cada técnica obtuvimos resultados no satisfactorios para nuestro indicador de interés, que es la sensibilidad del modelo, es decir, la proporción de *matches* que detectamos correctamente. El indicador de sensibilidad contrasta con el resultado de especificidad que obtuvimos en los modelos, ya que la tasa de correctos para el caso de la categoría cero era cercana a 1. Esto se debe a que la proporción de observaciones correspondiente a cada clase no es la misma, provocando un problema de sesgo de predictores hacia la clase mayoritaria.

```
library(tsne)
#ojo tarda siglo y medio en correr
tsne_match <- tsne(x_ent_s, perplexity = 80,
                     max_iter=180)
```

```
## sigma summary:
Min. : 0.49613987011077
1st Qu. : 0.650626289276977
Median : 0.690524428760104
Mean : 0.695107773957742
3rd Qu. : 0.735347811771763
Max. : 0.900905952876035
```

```
## Epoch: Iteration #100 error is: 21.5192775827827
```

```
dat_tsne <- data.frame(tsne_match)
dat_tsne$match <- as.character(y_ent)
ggplot(dat_tsne, aes(x=X1, y=X2, colour=match, label=match)) + geom_text()
```

### t-Distributed Stochastic Neighbor Embedding (t-SNE)

Utilizamos el algoritmo de t-SNE buscani que un método que nos ayude a separarar a los datos de tal manera que las parejas que hacen match se queden juntos (en un cluster) y los que hacen match queden en otro cluster, sin embargo, como podemos ver en la imagen 6 el algoritmo no logra separar de manera exitosa a las observaciones por lo que vemos mezcladas a las parejas que hacen match y a las que no.

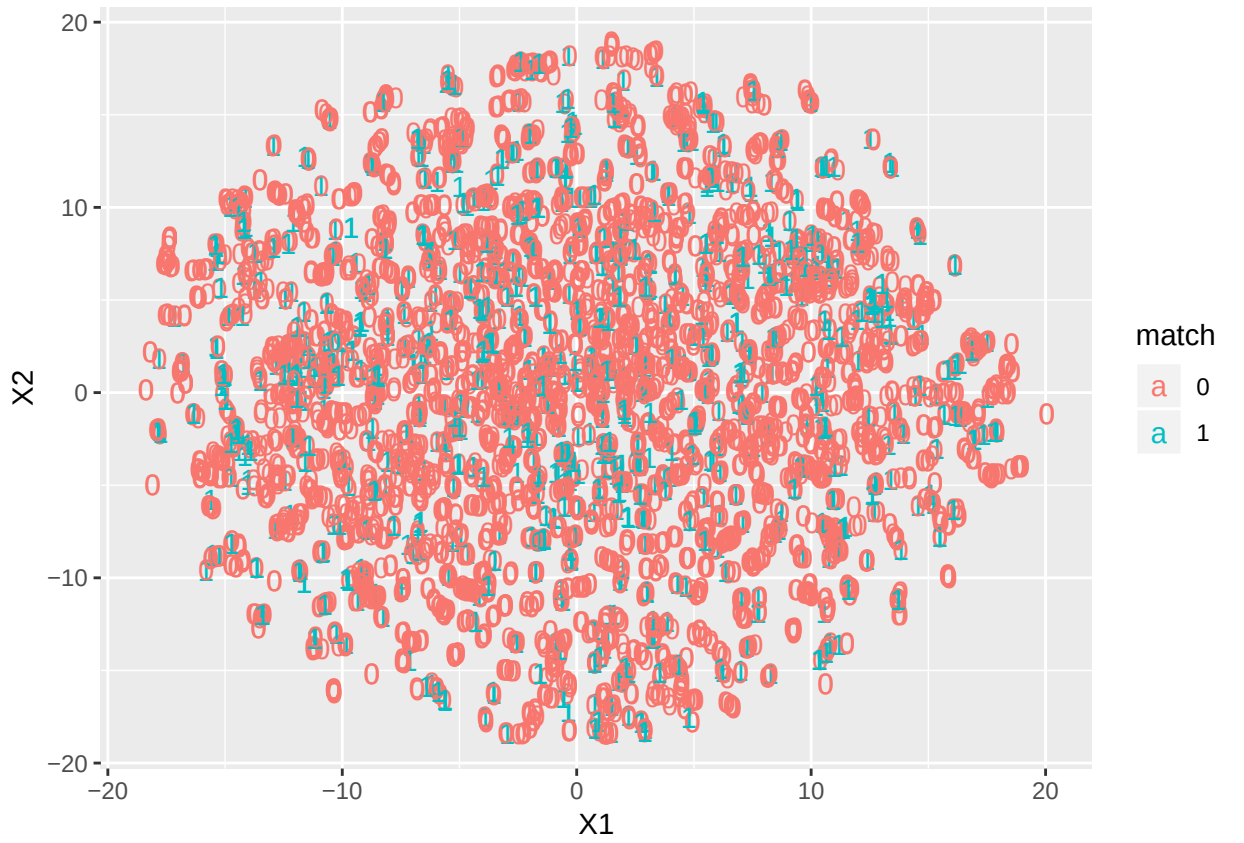



Figure 6: Resultado de t-SNE.

## Técnicas de balanceo de datos

Cuando tratamos con problemas de clasificación en bases de datos con clases no balanceadas, comunmente se presenta el problema de sesgo de predicciones hacia la clase mayoritaria. Es ello que se han desarrollado diversas técnicas para tratar este incoveniente. De acuerdo con un survey sobre tratamiento de bases de datos no balanceadas publicado en 2015 [2], existen al menos 220 artículos académicos que explican técnicas para mitigar este problema. El paquete *unbalanced* de R [3] implementa las técnicas más conocidas para problemas de clasificación con clases no balanceadas. En este paquete encontramos las técnicas basadas en muestreo y las técnicas basadas en distancia. Como parte del primer grupo encontramos la técnica de *Over-Sampling*, la cual consiste en una replicación aleatoria de observaciones de la clase minoritaria con el objetivo de obtener el mismo número de observaciones para cada clase. En el caso contrario, la técnica de *Under-Sampling*, descarta observaciones de la clase minoritaria para llegar a tener un 50/50 en la base de datos. Estas técnicas tienen sus respectivas desventajas, ya que la implementación de Over-Sampling incurre en riesgo de sobreajuste del modelo, mientras en Under-Sampling se enfrenta el problema de pérdida de información. Ahora bien, dentro del grupo de técnicas basadas en distancia encontramos la técnica SMOTE (*Synthetic Minority Oversampling TEchnique*), cuya idea principal es crear nuevas instancias de la clase minoritaria mediante la interpolación de las instancias ya existentes usando k-vecinos más cercanos.

A continuación se presenta una aplicación de las tres técnicas anteriormente mencionadas al problema de predicción de *matches* en citas rápidas.

Cargamos los paquetes utilizados en este análisis.

```r
library(tidyverse)
library(unbalanced)
library(randomForest)
library(readxl)
```

El siguiente código corresponde a la implementación de tres técnicas para tratar datos no balanceados sugeridas por Andrea Dal Pozzolo, desarrolladora del paquete unbalanced de R. Se probaron los métodos de Over-samplig, Down-samplig y SMOTE. Comparamos resultados contra la salida de un bosque aleatorio con datos no balanceados.

Código original disponible en la liga https://github.com/dalpozz/unbalanced.

1. Resultado del modelo de bosques aleatorios con datos no balanceados

```r
#Definimos semilla
set.seed(119465)

#Se define función que calcula los indicadores del modelo
indicadores<-function(table){
  tasa_fp<-table[2,1]/sum(table[,1])
  tasa_fn<-table[1,2]/sum(table[,2])
  sensibilidad<-table[2,2]/sum(table[,2])
  especificidad<-table[1,1]/sum(table[,1])
  # precision<-table[2,2]/sum(table[2,])
  # vpn<-table[1,1]/sum(table[1,])
  indicadores<-data.frame(tasa_fp=tasa_fp,tasa_fn=tasa_fn,
sensibilidad=sensibilidad,   especificidad=especificidad)
  return(indicadores)
}

#Carga de base con datos limpios
SD_limpios <- read_excel("datos_Speed_Dating_limpios1.xlsx")

#Se filtran variables de interés para el modelo
SD_limpios<-SD_limpios %>% select(-c("X__1","iid","wave","pid"))

#Número d eobservaciones por clase
table(SD_limpios$match)
```

```
##
##    0    1
## 6654 1308
```

```r
#Checamos porcentaje de observaciones correspondiente a la base de datos no balanceada
prop.table(table(SD_limpios$match))
```

```
##
##         0         1
## 0.8357197 0.1642803
```

```r
#Definimos variable dependiente
output <- as.factor(as.matrix(SD_limpios[ ,2]))
#Definimos variables independientes
input <- SD_limpios[ ,-2]

#Definimos conjunto de entrenamiento y conjunto de prueba considerando la siguiente partición de datos:
N <- nrow(SD_limpios)
```

```
N.tr <- floor(0.7*N)
id.tr <- sample(1:N, N.tr)
id.ts <- setdiff(1:N, id.tr)
X.tr  <- input[id.tr, ]
Y.tr <- output[id.tr]
X.ts <- input[id.ts, ]
Y.ts <- output[id.ts]

#Definimos base con datos no balanceados
unbalTrain <- data.frame(X.tr, match=Y.tr)

#Creamos bosque aleatorio con datos no balanceados
model1 <- randomForest(factor(match) ~ ., data = unbalTrain, ntree = 500, mtry = 100,
importance=TRUE)
#Guardamos las predicciones del modelo
preds1 <- predict(model1, X.ts, type="class")
#Obtenemos matriz de confusión
confusionMatrix1 <- table(prediction=preds1, actual=Y.ts)
#Mostramos matriz de confusión
print(confusionMatrix1)
```

```
##           actual
## prediction    0    1
##          0 1983  392
##          1    5    9
```

```
t1<-format(indicadores(confusionMatrix1),digits=2,nsmall=2)
```

2. Resultado del modelo de bosques aleatorios con datos balanceados utilizando la técnica de Over-Sampling

```
#Aplicamos técnica de OverSampling a los datos de entrenamiento.
data_ubOver <- ubBalance(X=X.tr, Y=Y.tr, type="ubOver", k=0)
#Creamos base de datos balanceada
overData <- data.frame(data_ubOver$X, match=data_ubOver$Y)
#Comprobamos el número de observaciones correspondiente a cada clase
table(overData$match)
```

```
##
##    0    1
## 4666 4666
```

```
#Configuramos modelo con set de entrenamiento con datos balanceados
model2 <- randomForest(factor(match) ~ ., data = overData, ntree = 500, mtry = 100,
importance=TRUE)
#Guardamos predicciones
preds2 <- predict(model2, X.ts, type="class")
confusionMatrix2 <- table(prediction=preds2, actual=Y.ts)
#Matriz de confusión
print(confusionMatrix2)
```

```
##           actual
## prediction    0    1
##          0 1969  380
##          1   19   21
```

```
#Indicadores del modelo
t2<-format(indicadores(confusionMatrix2),digits=2,nsmall=2)
```

3. Resultado del modelo de bosques aleatorios con datos balanceados utilizando la técnica de Under-Sampling

```
#Aplicamos técnica de UnderSampling a los datos de entrenamiento.
data_ubUnder <- ubBalance(X=X.tr, Y=Y.tr, type="ubUnder", k=0)
#Creamos base de datos balanceada
underData <- data.frame(data_ubUnder$X, match=data_ubUnder$Y)
#Comprobamos el número de observaciones correspondiente a cada clase
table(underData$match)
```

```
##
##   0   1
## 907 907
```

```
#Configuramos modelo con set de entrenamiento con datos balanceados
model3 <- randomForest(factor(match) ~ ., data = underData, ntree = 500, mtry = 100, importance=TRUE)
#Guardamos predicciones
preds3 <- predict(model3, X.ts, type="class")
confusionMatrix3 <- table(prediction=preds3, actual=Y.ts)
#Matriz de confusión
print(confusionMatrix3)
```

```
##           actual
## prediction    0    1
##          0 1264  143
##          1  724  258
```

```
#Indicadores del modelo
t3<-format(indicadores(confusionMatrix3),digits=2,nsmall=2)
```

4. Resultado del modelo de bosques aleatorios con datos balanceados utilizando la técnica SMOTE

```
#Aplicamos técnica de UnderSampling a los datos de entrenamiento.
data_ubSMOTE <- ubBalance(X=X.tr, Y=Y.tr, type="ubSMOTE", percOver = 200,percUnder = 150)
#Creamos base de datos balanceada
smoteData <- data.frame(data_ubSMOTE$X, match=data_ubSMOTE$Y)
#Comprobamos el número de observaciones correspondiente a cada clase
table(smoteData$match)
```

```
##
##    0    1
## 2721 2721
```

```
#Configuramos modelo con set de entrenamiento con datos balanceados
model4 <- randomForest(factor(match) ~ ., data = smoteData, ntree = 500, mtry = 100,
importance=TRUE)
#Guardamos predicciones
preds4 <- predict(model4, X.ts, type="class")
confusionMatrix4 <- table(prediction=preds4, actual=Y.ts)
#Matriz de confusión
print(confusionMatrix4)
```

```
##           actual
## prediction    0    1
##          0 1911  348
##          1   77   53
```

```
#Indicadores del modelo
t4<-format(indicadores(confusionMatrix4),digits=2,nsmall=2)
```

Mostramos los indicadores para cada experimento:

```
Tecnica<-c("No balanceado","Over-Sampling","Under-Sampling","SMOTE")
cbind(Tecnica,rbind(t1,t2,t3,t4))
```

```
##           Tecnica tasa_fp tasa_fn sensibilidad especificidad
## 1   No balanceado  0.0025    0.98        0.022          1.00
## 11  Over-Sampling  0.0096    0.95        0.052          0.99
## 12 Under-Sampling    0.36    0.36         0.64          0.64
## 13          SMOTE   0.039    0.87         0.13          0.96
```

Considerando el indicador de sensibilidad, observamos que el modelo con técnica de Under-Sampling clasifica correctamente el 64% de los casos positivos en el conjunto de prueba, teniendo como contrapeso una pérdida de precisión al generar más falsos positivos que los otros modelos. Otras desventajas de este modelo son: Pérdida de información al ignorar un porcentaje de las observaciones de la clase mayoritaria. Tener que calibrar las probabilidades obtenidas en el modelo.

# References


[1] Raymond Fisman, Sheena S Iyengar, Emir Kamenica, and Itamar Simonson. Gender differences in mate selection: Evidence from a speed dating experiment. *The Quarterly Journal of Economics*, 121(2):673–697, 2006.

[2] Paula Branco, Luis Torgo, and Rita Ribeiro. A survey of predictive modelling under imbalanced distributions. arxiv preprint. *arXiv preprint arXiv:1505.01658*, 2015.

[3] Andrea Dal Pozzolo, Olivier Caelen, Serge Waterschoot, and Gianluca Bontempi. Racing for unbalanced methods selection. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 24–31. Springer, 2013.

# Variables utilizadas de la base de *speed dating*

## Tabla general con las variables de la base

| Variable | Descripción |
|---|---|
| iid | Identificador único del individuo |
| gender | Género Mujer=0 y Hombre=1 |
| wave | Ola de 1-21 |
| pid | Identificador único de la pareja |
| match | 1 si ambas personas coincideiron en verse de nuevo, 0 en otro caso |
| samerace | Si los dos individuos son de la misma raza |
| age | Edad |
| field_cd | Código de área profesional, ver Tabla 3. |
| race | Raza del individuo codificada, ver Tabla 4. |
| imprace | Importancia que se le da a que la pareja sea del mismo grupo racial o étnico |
| imprelig | Importancia que se le da a que la pareja tenga la misma religión |
| from | Lugar de origen |
| goal | Objetivo de la cita codificado, ver Tabla 5 |
| date | Frecuencia con la que se acude a citas |
| sports | Gusto por practicar deporte |
| tvsports | Gusto por ver deporte |
| exercise | Gusto por hacer ejercicio |
| dining | Gusto por comer fuera |
| museums | Gusto por los museos o galerías |
| art | Gusto por el arte |
| hiking | Gusto por excursionismo o campamento |
| gaming | Gusto por jugar videojuegos |
| clubbing | Gusto por salir a clubes nocturnos o bailar |
| reading | Gusto por la lectura |
| tv | Gusto por ver televisión |
| theater | Gusto por el teatro |
| movies | Gusto por las películas |
| concerts | Gusto por ir a conciertos |
| music | Gusto por la música |
| shopping | Gusto por ir de compras |
| yoga | Gusto por el yoga o la meditación |
| exphappy | Expectativa de satisfacción en las citas |
| $attr1_1$ | La importancia del atractivo en alguien del sexo opuesto |
| $sinc1_1$ | La importancia de la sinceridad en alguien del sexo opuesto |
| $intel1_1$ | La importancia de la inteligencia en alguien del sexo opuesto |
| $fun1_1$ | La importancia de lo divertido en alguien del sexo opuesto |
| $amb1_1$ | La importancia de la ambición en alguien del sexo opuesto |
| $shar1_1$ | Importancia que uno le da a los intereses compartidos |
| $sinc2_1$ | La importancia que creo que el sexo opuesto le da a la sinceridad |
| $intel2_1$ | La importancia que creo que el sexo opuesto le da a la inteligencia |
| $fun2_1$ | La importancia que creo que el sexo opuesto le da a lo divertido en uno |
| $amb2_1$ | La importancia que creo que el sexo opuesto le da a la ambición |
| $shar2_1$ | La importancia que creo que el sexo opuesto le da a tener intereses en común. |
| $attr3_1$ | Qué tan atractivo se considera uno mismo |
| $sinc3_1$ | Qué tan sincero se considera uno mismo |
| $fun3_1$ | Qué tan divertido se considera uno mismo |
| $intel3_1$ | Qué tan inteligente se considera uno mismo |
| $amb3_1$ | Qué tan ambicioso se considera uno mismo |

Table 2: Variables y descripción

| Código | Valor |
|---|---|
| 1 | Law |
| 2 | Math |
| 3 | Social Science, Psychologist |
| 4 | Medical Science, Pharmaceuticals, and Bio Tech |
| 5 | Engineering |
| 6 | English/Creative Writing/ Journalism |
| 7 | History/Religion/Philosophy |
| 8 | Business/Econ/Finance |
| 9 | Education, Academia |
| 10 | Biological Sciences/Chemistry/Physics |
| 11 | Social Work |
| 12 | Undergrad/undecided |
| 13 | Political Science/International Affairs |
| 14 | Film |
| 15 | Fine Arts/Arts Administration |
| 16 | Languages |
| 17 | Architecture |
| 18 | Other |

Table 3: Codificación de área profesional

## Tabla de codificación de área profesional

## Tabla de codificación de grupo racial

| Código | Valor |
|---|---|
| 1 | Black/African American |
| 2 | European/Caucasian-American |
| 3 | Latino/Hispanic American |
| 4 | Asian/Pacific Islander/Asian-American |
| 5 | Native American |
| 6 | Other |

Table 4: Codificación del grupo racial

## Tabla de codificación de objetivo de la cita

| Código | Valor |
|---|---|
| 1 | Seemed like a fun night out |
| 2 | To meet new people |
| 3 | To get a date |
| 4 | Looking for a serious relationship |
| 5 | To say I did it |
| 6 | Other |

Table 5: Caption