Universitat Oberta de Catalunya

Patricia Lázaro Tello
plazarotello@uoc.edu

PRA2: Proyecto de minería de datos

Patricia Lázaro Tello

Índice general

Patricia Lázaro Tello
plazarotello@uoc.edu

1 Introducción

Kickstarter es una plataforma de financiación colectiva para proyectos creativos — música, arte, tecnología, videojuegos… — fundada en 2009. A finales de 2012 el modelo de financiación alternativo que proponía la plataforma ganó mucha popularidad y tracción, y aunque a día de hoy ese fervor ha disminuido, la financiación colectiva y este tipo de plataformas se han normalizado y permanecen relevantes.

Kickstarter es la plataforma de financiación colectiva más popular actualmente, pero compite en su nicho de mercado con otras como Indiegogo o GoFundMe. Esta plataforma se utiliza para financiar proyectos (ideas con un objetivo claro y que eventualmente son completadas).

En este trabajo se propone un proyecto de minería de datos sobre los proyectos que han intentado financiarse a través de **Kickstarter** entre el 12 de agosto y el 16 de septiembre de 2021.

---

2 Objetivos del proyecto

El proyecto objeto de estudio de este trabajo consiste en el **estudio de las características de los proyectos financiados con éxito en Kickstarter**.

Su objetivo es averiguar si un proyecto va a ser financiado con éxito a partir las características del mismo; por lo tanto, se trata de un **problema de clasificación**. Se estudiará cuál de los siguientes modelos es más adecuado para el caso de estudio:

- **Algoritmo k-means**
- **Algoritmo DBSCAN + OPTICS**
- **Árboles de decisión**
- **Algoritmo k-NN**
- **Regresión logística**

A este respecto, el problema de clasificación es **binario** (el proyecto es o no financiado con éxito) y **equilibrado** (existe un número similar de observaciones de éxito que de fracaso en la financiación).

Debido a que se asume que las clases están equilibradas, definir una medida de cumplimiento del objetivo resulta más fácil que en casos de clasificación binaria no equilibrada.

Patricia Lázaro Tello
plazarotello@uoc.edu

Se procede a definir las medidas que se utilizarán:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

La **precisión** corresponde con la calidad del modelo a la hora de predecir e informa del porcentaje de observaciones realmente positivas de entre las que el modelo ha identificado como positivas.

El **recall** (o sensibilidad) hace referencia al porcentaje de observaciones realmente positivas que el modelo ha identificado.

El ***F_1 score*** proporciona un balance entre precisión y *recall* mediante la combinación de ambas métricas.

El objetivo del proyecto de minería de datos se alcanzará cuando el modelo de clasificación elegido obtenga una medida de $F_1 > 0.75$, $precision > 0.75$ y $recall > 0.75$.

El trabajo se va a realizar utilizando solo los datos correspondiente al 2021 (hasta el 16 de septiembre), pero se podría aplicar el proyecto de minería de datos sobre el conjunto de datos global, que se remonta hasta 2014.

Patricia Lázaro Tello
plazarotello@uoc.edu

2.1 Limitaciones del *dataset*

El conjunto de datos posee varias limitaciones que impiden una correcta clasificación de los casos de éxito y fracaso, así como el descubrimiento de patrones. Estas limitaciones son:

- Los datos originales poseían varios campos de texto e imágenes; estos atributos no se han tenido en cuenta o no se han explotado más que superficialmente. Un análisis de textos e imágenes obtendría mucha más información, y en ella se conseguirían más atributos discriminantes para la clasificación.
- El conjunto de datos del que se parte en este trabajo posee variables tanto categóricas como numéricas. Algunos de estos algoritmos no aceptan variables categóricas *out of the box*, por lo que en algunos casos es preferible eliminarlas del *dataset* y por tanto, se pierde información relevante.
- El conjunto de datos tiene 9 atributos; es decir, se trata de un *dataset* con muchas dimensiones. No se va a poder representar gráficamente en el espacio original, sino que se llevará a cabo una reducción de dimensionalidad mediante t-SNE para poder representarlo.
- El conjunto de datos tiene 203.094 muestras, un tamaño muy grande para la mayoría de algoritmos aquí presentados. Se muestreará el *dataset* para obtener un volumen de datos aceptable, pero supondrá la pérdida de información relevante.
- El conjunto de datos es una fotografía en el tiempo: las conclusiones que se puedan extraer del análisis pueden no resultar válidas para realizar predicciones futuras.

Patricia Lázaro Tello
plazarotello@uoc.edu

3 Carga de datos

Se procede a cargar en memoria el conjunto de datos de Kickstarter que se filtró y limpió en el trabajo anterior. Se cargará también el conjunto de datos con las variables cuantitativas normalizadas por la diferencia:

```
kickstarter <- read.csv('kickstarter-clean.csv', stringsAsFactors=TRUE)
kickstarter.norm <- read.csv('kickstarter-clean-normalized.csv',
                             stringsAsFactors=TRUE)
kickstarter.dim <- dim(kickstarter)
```

El conjunto de datos tiene 203.094 observaciones con 10 atributos cada una de ellas. Se observan las primeras filas del *dataset*:

```
head(kickstarter, n=4)
```

```
##   words.name words.blurb photo.exists created.projects   category
## 1          8          19            1                1 Technology
## 2          6          16            1                2 Technology
## 3          4          12            1                1 Technology
## 4          4          13            1                1 Technology
##   collection.period launch.month  goal country success
## 1                30    noviembre  5788      NZ       1
## 2                30    noviembre 16388      GB       1
## 3                16       agosto 18000      US       1
## 4                35        julio 46608      GB       1
```

Los atributos se definen tal que:

- **words.name**: número de palabras en el nombre del proyecto.
- **words.blurb**: número de palabras en la descripción del proyecto.
- **photo.exists**: si el proyecto tiene fotografía (si es 1, tiene foto; en caso contrario el valor será 0).
- **created.projects**: la cantidad de proyectos que ha creado el usuario creador del proyecto. Se puede ver como una medida de la experiencia que tiene el usuario en la plataforma.
- **category**: categoría a la que pertenece el proyecto.
- **collection.period**: días que estuvo abierta la campaña de financiación del proyecto.
- **launch.month**: mes en que se lanzó la campaña de financiación del proyecto.
- **goal**: objetivo de dinero en dólares que se quería alcanzar con la campaña de financiación.
- **country**: país de origen del proyecto.

- **success**: si el proyecto se financió con éxito o no (si es 1, fue exitoso; en caso contrario el valor será 0).

Para poder trabajar con el *dataset* sin que los algoritmos tarden demasiado tiempo o no se puedan ejecutar por un *hardware* subyacente insuficiente se va a reducir el volumen de datos, eligiendo una muestra de tamaño aceptable:

```
maxrowsperclass <- 1000
classes <- 2
p <- round((maxrowsperclass*classes)/kickstarter.dim[1], 2)
kickstarter.sample.idx <- caret::createDataPartition(kickstarter$success,
                                                     p=p, list=FALSE)
kickstarter.sample <- kickstarter[kickstarter.sample.idx,]
kickstarter.norm.sample <- kickstarter.norm[kickstarter.sample.idx,]
```

4 Algoritmo k-means

El algoritmo k-means es un algoritmo de clasificación no supervisada. Su objetivo es encontrar patrones en los datos que permitan categorizar las observaciones en grupos o *clusters*.

Se trata de un algoritmo de **segmentación particional** o no-jerárquico, es decir, que los grupos que genera no responden a ningún tipo de organización jerárquica. Se basa en agrupar los datos en k clusters (dada una k); cada observación pertenece al grupo cuya distancia es menor.

Algunas de sus características principales son:

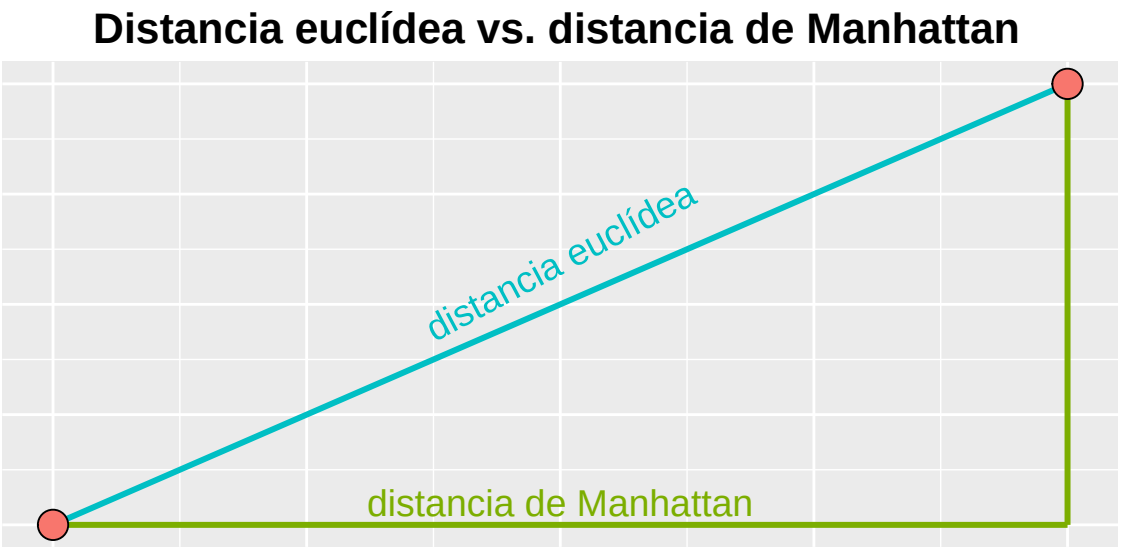
- Parámetros preconfigurados: número de clusters (k).
- Genera *clusters* más o menos esféricos.
- Es sensible al ruido.
- Tiene un buen rendimiento en *datasets* grandes.
- Su rendimiento empeora al aumentar el número de dimensiones.
- Necesita que las variables se muevan en rangos similares.
- Necesita que las variables sean continuas.
- Cada cluster ha de tener más o menos el mismo número de observaciones.

En el *dataset* elegido existen atributos categóricos (`category`, `country` y `launch.month`). Para tratarlos existen 3 opciones: se puede no utilizar en el modelo, se puede utilizar la distancia de Gower para tratarlas o se puede realizar *one-hot encoding* y tratarlos como variables continuas, aunque con eficacia reducida.

En este caso, se decide utilizar la distancia euclídea y la distancia de Manhattan sin incluir las variables categóricas en el modelo. Se procede, entonces, a eliminar al variables categóricas:

```
norm.sample.labels <- kickstarter.norm.sample %>% dplyr::select(success)
norm.sample.nofactors <- kickstarter.norm.sample %>% dplyr::select(
  -category, -launch.month, -country, -success)
```

La distancia euclídea busca la distancia mínima en línea recta entre 2 puntos, mientras que la distancia de Manhattan, también llamada distancia *taxicab*, es la suma de la distancia en horizontal y en vertical.



4.1 Algoritmo k-means con distancia euclídea

En primer lugar se ha de elegir un valor de k adecuado. Como se trata de un problema supervisado de clasificación, se conoce de antemano los *clusters* en que se divide el conjunto de datos.

De todas formas, se calculará también un k adecuado mediante la regla del codo, minimizando la suma de distancias intragrupo (***Sum of Squared Within*** o SSW):

Se crean y evalúan modelos para $k \in \{1..10\}$, utilizando la distancia euclídea como métrica de distancia:

```
choose.k <- function(df, metric){
  candidates <- seq(from=1, to=20, by=1)
  results <- rep(NA, length(candidates))

  for (k in candidates[-1]) {
    set.seed(seed)
    model <- amap::Kmeans(df, k, method=metric, iter.max=50)
    results[k] <- mean(model$withins)
  }

  results.k <- data.frame(k=candidates, ssw=results) %>% drop_na()

  k.plot <- ggplot(data=results.k, mapping=aes(x=k, y=ssw)) +
    scale_x_continuous(breaks=candidates, minor_breaks=NULL) +
```

```
    scale_y_continuous(labels=scales::label_scientific(digits=2)) +
    geom_line(color=default.color.main, size=1.05) +
    geom_point(color=default.color.secondary, size=2.5) +
    labs(title='Sum of Squared Within', x='número de clusters', y='SSW')
  return(list('df'=results.k, 'plot'=k.plot))
}
```

```
euclidean.k <- choose.k(norm.sample.nofactors, 'euclidean')
euclidean.k$plot + title.centered +
  geom_vline(aes(xintercept=11), color=default.color.terciary,
             linetype='dashed', size=1, alpha=0.75) +
  annotate('text', x=11, y=euclidean.k$df$ssw[euclidean.k$df$k==11],
           fontface='bold', label='   k\n', color=default.color.terciary,
           size=4, hjust=0.0)
```

La **regla del codo** aplicada sobre *SSW* encuentra el valor óptimo en $k = 11$, mientras que el conocimiento de expertos indica que debería haber únicamente 2 *clusters*. Se procede a crear ambos modelos y visualizarlos reduciendo su dimensionalidad mediante t-SNE.

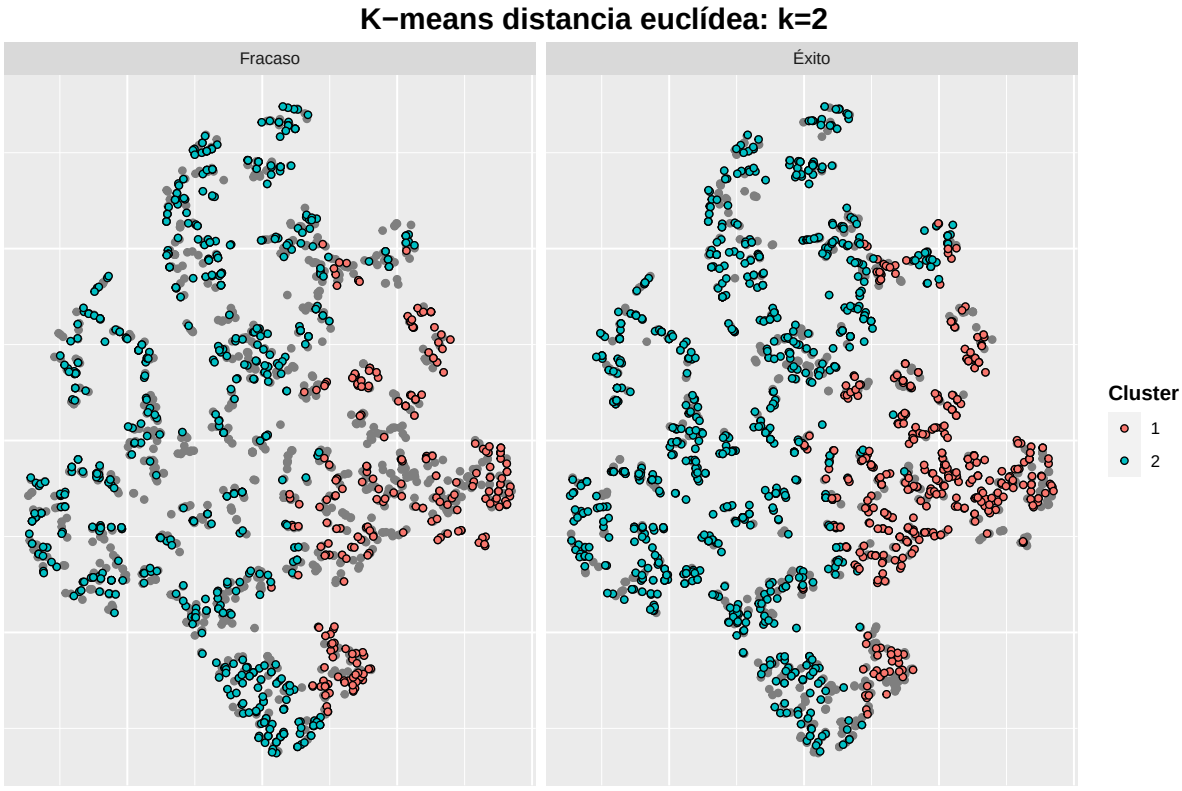
Como el conjunto de datos es muy grande, se cogerá solo una muestra representativa de cada cluster para su visualización.

```
set.seed(seed)
keuc.k2 <- amap::Kmeans(norm.sample.nofactors, centers=2, method='euclidean')
set.seed(seed)
keuc.k11 <- amap::Kmeans(norm.sample.nofactors, centers=11, method='euclidean')
```

```
transform.tsne <- function(kmeans.result, df, labels){
  tsne.sample <- Rtsne::Rtsne(df, dims=2, check_duplicates=FALSE,
                              normalize=FALSE)
  tsne.df <- data.frame(x=tsne.sample$Y[,1], y=tsne.sample$Y[,2],
                        cluster=kmeans.result$cluster, label=labels)
  return(tsne.df)
}
```

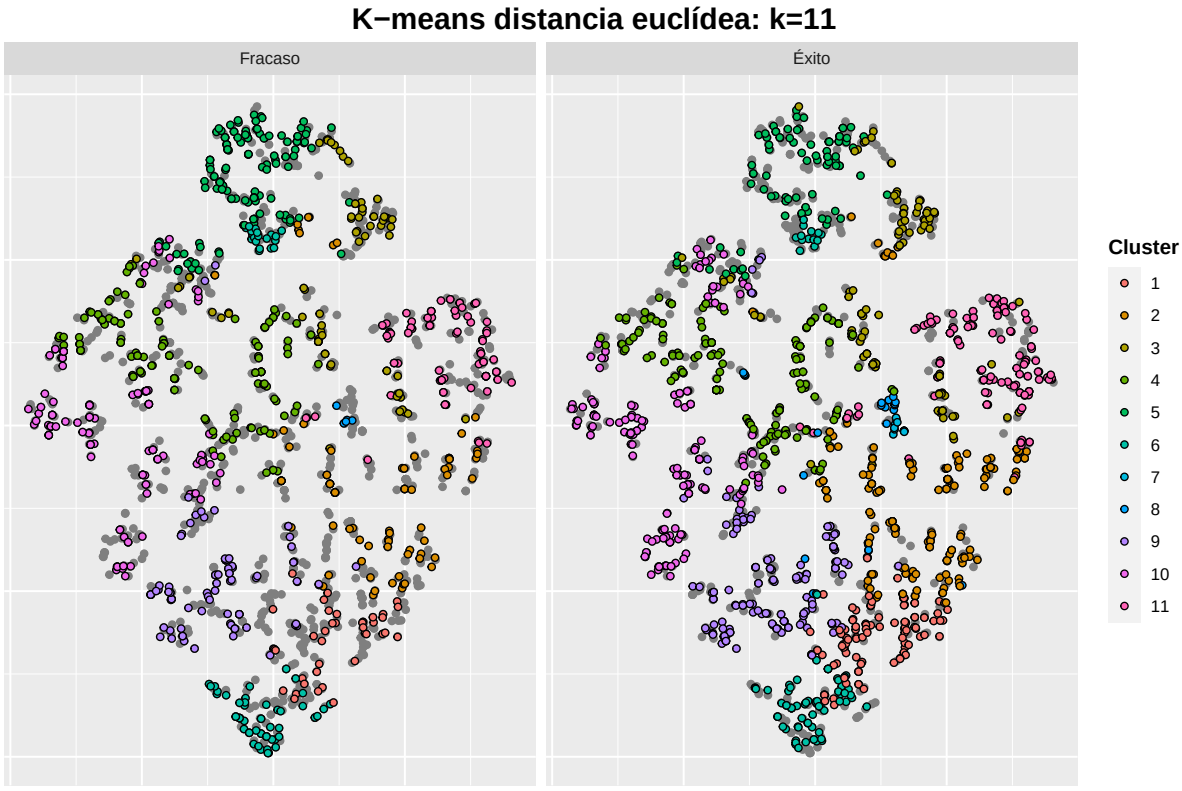
```
tsne.k2 <- transform.tsne(keuc.k2, norm.sample.nofactors,
                          norm.sample.labels$success)
tsne.k11 <- transform.tsne(keuc.k11, norm.sample.nofactors,
                           norm.sample.labels$success)
```

```
ggplot(data=tsne.k2, mapping=aes(x=x, y=y)) +
  geom_point(color='grey50', data=transform(tsne.k2, label=NULL)) +
  geom_point(aes(fill=factor(cluster)), shape=21) +
  scale_fill_manual(name='Cluster', values=palette) +
  theme(legend.position='right', legend.title=element_text(face='bold')) +
  title.centered + no.axis.x + no.axis.y + labs(x=NULL, y=NULL) +
  facet_wrap(vars(label), labeller=success.labeller.2) +
  ggtitle('K-means distancia euclídea: k=2')
```



A primera vista no se observan grupos diferenciados para $k = 2$, aunque se podría apreciar que existen más punto rojos (del *cluster* 1) en los casos de éxito que en los de fracaso.

```
ggplot(data=tsne.k11, mapping=aes(x=x, y=y)) +
  geom_point(color='grey50', data=transform(tsne.k11, label=NULL)) +
  geom_point(aes(fill=factor(cluster)), shape=21) + labs(fill='Cluster') +
  theme(legend.position='right', legend.title=element_text(face='bold')) +
  title.centered + no.axis.x + no.axis.y + labs(x=NULL, y=NULL) +
  facet_wrap(vars(label), labeller=success.labeller.2) +
  ggtitle('K-means distancia euclídea: k=11')
```



En el caso de $k = 11$, no se observan tampoco grupos diferenciados. Se procede a observar el porcentaje de observaciones que encuentra cada *cluster* para poder etiquetarlos con mayor precisión para $k = 2$ y especialmente $k = 11$:

```
round(prop.table(100.0*table(as.factor(tsne.k2$cluster),
                             as.factor(tsne.k2$label),
                             dnn=c('clusters', 'etiquetas')), margin=2), 2)
```

```
##          etiquetas
## clusters    0    1
##        1 0.29 0.33
##        2 0.71 0.67
```

```
round(prop.table(100.0*table(as.factor(tsne.k11$cluster),
                             as.factor(tsne.k11$label),
                             dnn=c('clusters', 'etiquetas')), margin=2), 2)
```

```
##          etiquetas
## clusters    0    1
##        1  0.04 0.08
##        2  0.10 0.14
##        3  0.09 0.09
```

```
##        4  0.16 0.14
##        5  0.15 0.07
##        6  0.09 0.07
##        7  0.00 0.01
##        8  0.01 0.02
##        9  0.10 0.12
##        10 0.15 0.15
##        11 0.10 0.10
```

Se procede a obtener medidas de calidad por **técnicas de validación externa** asignando a cada *cluster* de ambos modelos de agrupamiento la etiqueta que mejor coincide con los grupos:

k	*cluster*	etiqueta
2	1	Éxito
2	2	Fracaso
11	1	Éxito
11	2	Éxito
11	3	Fracaso
11	4	Fracaso
11	5	Fracaso
11	6	Fracaso
11	7	Éxito
11	8	Éxito
11	9	Éxito
11	10	Éxito
11	11	Éxito

```
k2.metrics.df <- data.frame(cluster=keuc.k2$cluster,
                            label=norm.sample.labels$success) %>%
  dplyr::mutate(cluster=ifelse(cluster==1, 1, 0))

k2.confusion <- caret::confusionMatrix(data=as.factor(k2.metrics.df$cluster),
                                       reference=as.factor(k2.metrics.df$label),
                                       positive='1')
k2.confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
```

```
##          0 601 786
##          1 251 393
##
##                Accuracy : 0.489
##                  95% CI : (0.467, 0.511)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.036
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.333
##             Specificity : 0.705
##          Pos Pred Value : 0.610
##          Neg Pred Value : 0.433
##              Prevalence : 0.581
##          Detection Rate : 0.194
##    Detection Prevalence : 0.317
##       Balanced Accuracy : 0.519
##
##        'Positive' Class : 1
##
```

```
k11.metrics.df <- data.frame(cluster=keuc.k11$cluster,
                             label=norm.sample.labels$success) %>%
  dplyr::mutate(cluster=ifelse(cluster %in% c(3,4,5,6), 0, 1))

k11.confusion <- caret::confusionMatrix(data=as.factor(k11.metrics.df$cluster),
                                        reference=as.factor(k11.metrics.df$label),
                                        positive='1')
k11.confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 424 442
##          1 428 737
##
##                Accuracy : 0.572
##                  95% CI : (0.55, 0.593)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 0.797
##
##                   Kappa : 0.122
```

```
##
##  Mcnemar's Test P-Value : 0.659
##
##             Sensitivity : 0.625
##             Specificity : 0.498
##          Pos Pred Value : 0.633
##          Neg Pred Value : 0.490
##              Prevalence : 0.581
##          Detection Rate : 0.363
##    Detection Prevalence : 0.574
##       Balanced Accuracy : 0.561
##
##        'Positive' Class : 1
##
```

	$k = 2$	$k = 11$
Precisión	0,61	0,63
Recall	0,33	0,63
F_1 *Score*	0,43	0,63

Los modelos k-means con distancia euclídea no tienen la calidad esperada. Los valores de precisión no alcanzan valores aceptables (mayores a 70%) y los de *recall*, aunque aceptable en el caso de $k = 11$, siguen siendo muy bajos.

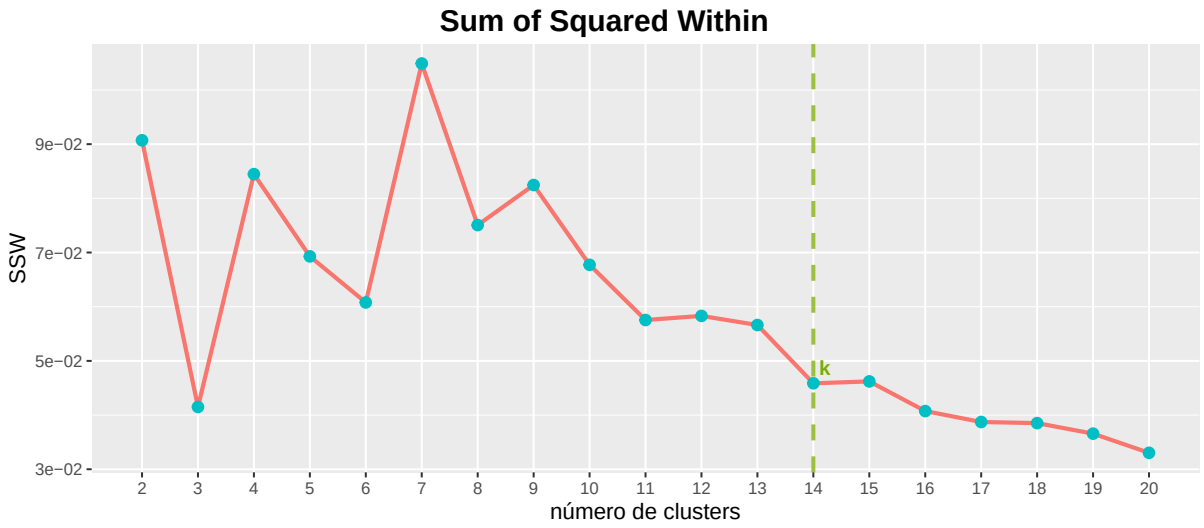
Esto indica que se trata de modelos que no encuentran patrones en los datos para predecir los casos de éxito. En este caso, se reservaría el modelo $k = 11$.

4.2 Algoritmo k-means con distancia de Manhattan

Se procede a realizar los pasos seguidos anteriormente para un algoritmo k-means que utiliza la distancia de Manhattan. Se escogerá un valor de k adecuado mediante la regla del codo, se crearán los modelos y se visualizarán con reducción de dimensionalidad. Después, se asociarán *clusters* con etiquetas y se calcularán las métricas de calidad:

```
manhattan.k <- choose.k(norm.sample.nofactors, 'manhattan')
manhattan.k$plot + title.centered +
  geom_vline(aes(xintercept=14), color=default.color.terciary,
             linetype='dashed', size=1, alpha=0.75) +
  annotate('text', x=14, y=manhattan.k$df$ssw[manhattan.k$df$k==14],
```

```
            fontface='bold', label=' k\n', color=default.color.terciary,
            size=4, hjust=0.0)
```

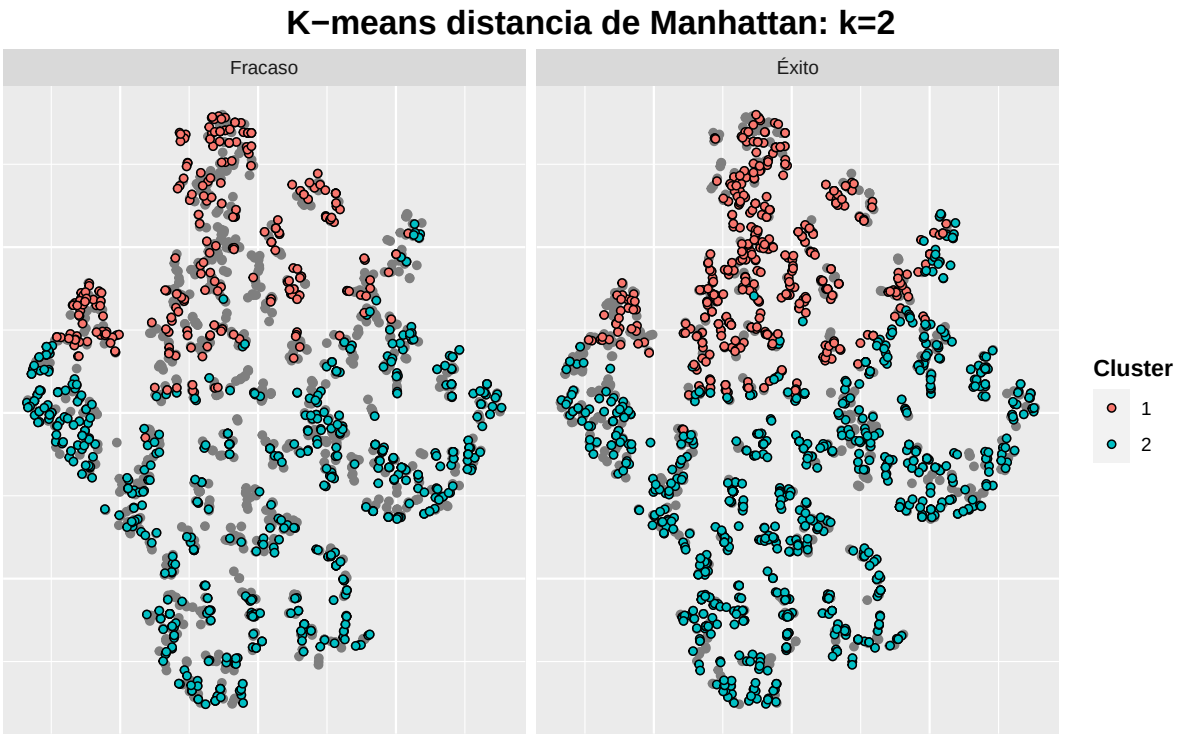


Se elige $k = 14$, aunque también se podría haber elegido $k = 16$. La gráfica no presenta una forma de codo muy clara, por lo que la elección de k queda abierta a interpretación.

```
set.seed(seed)
kman.k2 <- amap::Kmeans(norm.sample.nofactors, centers=2, method='manhattan')
set.seed(seed)
kman.k14 <- amap::Kmeans(norm.sample.nofactors, centers=14, method='manhattan')
```

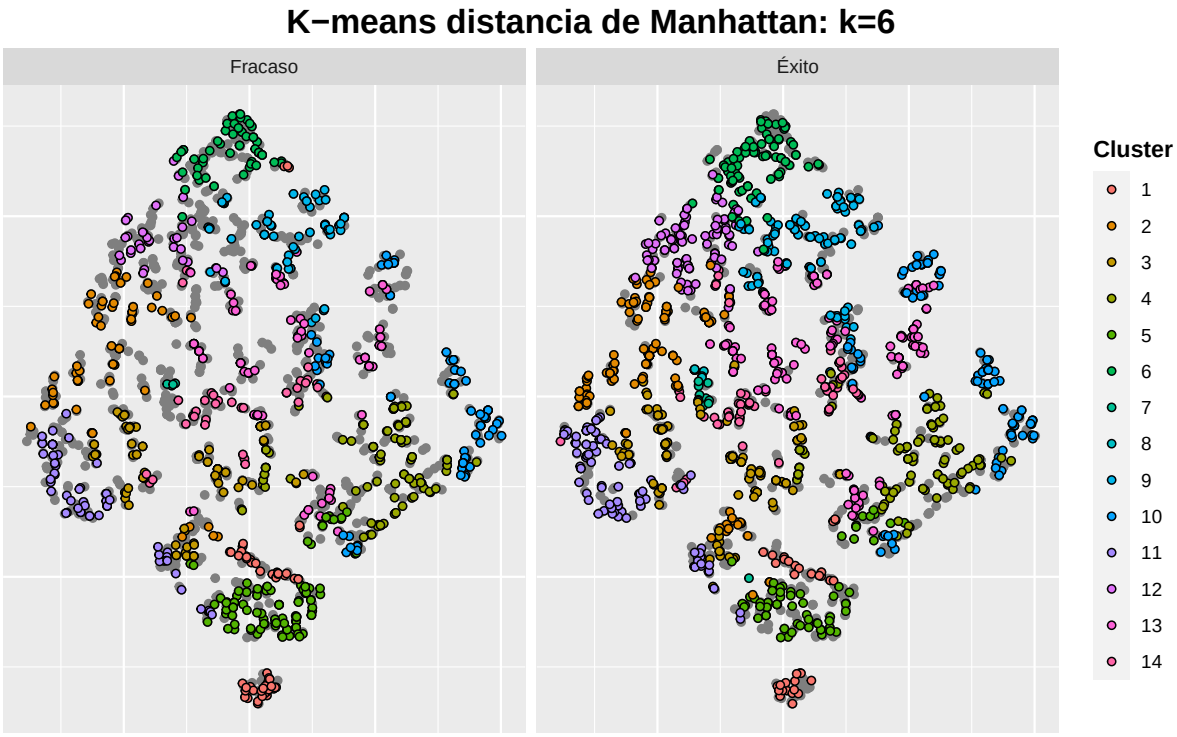
```
tsne.km2 <- transform.tsne(kman.k2, norm.sample.nofactors,
                           norm.sample.labels$success)
tsne.km14 <- transform.tsne(kman.k14, norm.sample.nofactors,
                            norm.sample.labels$success)
```

```
ggplot(data=tsne.km2, mapping=aes(x=x, y=y)) +
  geom_point(color='grey50', data=transform(tsne.km2, label=NULL)) +
  geom_point(aes(fill=factor(cluster)), shape=21) +
  scale_fill_manual(name='Cluster', values=palette) +
  theme(legend.position='right', legend.title=element_text(face='bold')) +
  title.centered + no.axis.x + no.axis.y + labs(x=NULL, y=NULL) +
  facet_wrap(vars(label), labeller=success.labeller.2) +
  ggtitle('K-means distancia de Manhattan: k=2')
```



A primera vista no se observan grupos diferenciados para $k = 2$, aunque se podría apreciar que existen más punto rojos (del *cluster* 1) en los casos de fracaso que en los de éxito.

```
ggplot(data=tsne.km14, mapping=aes(x=x, y=y)) +
  geom_point(color='grey50', data=transform(tsne.km14, label=NULL)) +
  geom_point(aes(fill=factor(cluster)), shape=21) + labs(fill='Cluster') +
  theme(legend.position='right', legend.title=element_text(face='bold')) +
  title.centered + no.axis.x + no.axis.y + labs(x=NULL, y=NULL) +
  facet_wrap(vars(label), labeller=success.labeller.2) +
  ggtitle('K-means distancia de Manhattan: k=6')
```



En $k = 14$ no se observan tampoco grupos claramente diferenciados. Se procede a observar el porcentaje de observaciones que encuentra cada *cluster* para poder etiquetarlos con mayor precisión para $k = 2$ y especialmente $k = 14$:

```
round(prop.table(100.0*table(as.factor(tsne.km2$cluster),
                             as.factor(tsne.km2$label),
                             dnn=c('clusters', 'etiquetas')), margin=2), 2)
```

```
##          etiquetas
## clusters    0    1
##        1 0.30 0.35
##        2 0.70 0.65
```

```
round(prop.table(100.0*table(as.factor(tsne.km14$cluster),
                             as.factor(tsne.km14$label),
                             dnn=c('clusters', 'etiquetas')), margin=2), 2)
```

```
##          etiquetas
## clusters    0    1
##        1  0.07 0.03
##        2  0.07 0.11
##        3  0.10 0.11
##        4  0.13 0.09
##        5  0.12 0.06
```

```
##        6  0.07 0.07
##        7  0.00 0.01
##        8  0.00 0.01
##        9  0.08 0.09
##        10 0.10 0.09
##        11 0.08 0.08
##        12 0.04 0.08
##        13 0.11 0.13
##        14 0.04 0.05
```

Se procede a obtener medidas de calidad por **técnicas de validación externa** asignando a cada *cluster* de ambos modelos de agrupamiento la etiqueta que mejor coincide con los grupos:

k	*cluster*	etiqueta
2	1	Éxito
2	2	Fracaso
14	1	Fracaso
14	2	Éxito
14	3	Éxito
14	4	Fracaso
14	5	Fracaso
14	6	Fracaso
14	7	Éxito
14	8	Éxito
14	9	Éxito
14	10	Fracaso
14	11	Éxito
14	12	Éxito
14	13	Éxito
14	14	Éxito

```
km2.metrics.df <- data.frame(cluster=kman.k2$cluster,
                             label=norm.sample.labels$success) %>%
  dplyr::mutate(cluster=ifelse(cluster==1, 1, 0))

km2.confusion <- caret::confusionMatrix(data=as.factor(
  km2.metrics.df$cluster),reference=as.factor(k2.metrics.df$label),
  positive='1')
km2.confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 594 768
##          1 258 411
##
##                Accuracy : 0.495
##                  95% CI : (0.473, 0.517)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.042
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.349
##             Specificity : 0.697
##          Pos Pred Value : 0.614
##          Neg Pred Value : 0.436
##              Prevalence : 0.581
##          Detection Rate : 0.202
##    Detection Prevalence : 0.329
##       Balanced Accuracy : 0.523
##
##        'Positive' Class : 1
##
```

```
km14.metrics.df <- data.frame(cluster=kman.k14$cluster,
                              label=norm.sample.labels$success) %>%
  dplyr::mutate(cluster=ifelse(cluster %in% c(1,4,5,6,10), 0, 1))

km14.confusion <- caret::confusionMatrix(data=as.factor(
  km14.metrics.df$cluster),reference=as.factor(km14.metrics.df$label),
  positive='1')
km14.confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 411 395
##          1 441 784
##
##                Accuracy : 0.588
##                  95% CI : (0.567, 0.61)
```

```
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 0.243
##
##                   Kappa : 0.148
##
##  Mcnemar's Test P-Value : 0.120
##
##             Sensitivity : 0.665
##             Specificity : 0.482
##          Pos Pred Value : 0.640
##          Neg Pred Value : 0.510
##              Prevalence : 0.581
##          Detection Rate : 0.386
##    Detection Prevalence : 0.603
##       Balanced Accuracy : 0.574
##
##        'Positive' Class : 1
##
```

	$k = 2$	$k = 14$
Precisión	0,61	0,64
Recall	0,35	0,66
F_1 *Score*	0,44	0,65

Los modelos k-means con distancia de Manhattan tampoco tienen la calidad esperada, aunque son ligeramente superiores a los obtenidos con distancia euclídea. Los valores de precisión y *recall* no alcanzan valores aceptables (mayores a 70%), siendo $k = 14$ el que ofrece mejor *recall*. Esto indica que se trata de modelos que no encuentran patrones en los datos para predecir los casos de éxito. Se reserva el modelo $k = 14$.

5 DBSCAN + OPTICS

El algoritmo DBSCAN, como el algoritmo k-means, es un **algoritmo de *clustering* basado en densidad**. Se basa en identificar zonas de alta concentración de observaciones separadas entre sí por zonas de baja concentración de observaciones.

Un concepto que introduce DBSCAN es el de *core samples* o muestras centrales, que son agregaciones de un mínimo de $min\_samples$ observaciones que se encuentran a una distancia máxima de ϵ.

El algoritmo OPTICS, cuyas siglas significan *Ordering Points to Identify Cluster Structure*, supone una generalización del algoritmo DBSCAN. Al contrario que este último, OPTICS no genera *clusters* propiamente, si no que ordena los puntos del *dataset* de entrada en función de su alcanzabilidad (o *reachability*).

Sus parámetros de entrada son los mismos que los de DBSCAN ($min\_samples$ y ϵ), pero ϵ tiene un significado diferente: si en DBSCAN ϵ era la distancia mínima entre observaciones para ser consideradas vecinas, en OPTICS es la distancia máxima a la ***core sample*** en la que se buscarán vecinos.

La combinación de DBSCAN y OPTICS produce los mejores resultados al utilizar OPTICS para encontrar un valor apropiado para los parámetros de DBSCAN. En el trabajo se utilizará esta técnica combinada.

Algunas de sus características principales son:

- Parámetros preconfigurados: distancia mínima entre muestras para ser considerada parte de la muestra central (ϵ) y número mínimo de muestras para ser considerada muestra central ($min\_samples$). Se obtendrán a partir de OPTICS.
- Los *clusters* que genera pueden tener forma no convexa o irregular.
- Puede identificar *outliers*.
- Es robusto; no es sensible al ruido.
- Funciona bien con *datasets* dispersos o *datasets* con densidad variable.

5.1 Elección de parámetros (ϵ y $min\_samples$)

El algoritmo DBSCAN requiere de 2 parámetros: $epsilon$, la distancia mínima entre 2 observaciones para ser consideradas vecinas; y $min\_samples$, el número de observaciones vecinas necesario para considerar la zona que representan como de alta densidad, y por tanto, considerarla *core sample*.

La elección de estos 2 parámetros no es trivial y depende en gran medida de la distribución y densidad de los datos. Si los datos están muy esparcidos, se requerirá aumentar la ϵ o disminuir $min\_samples$; si están muy concentrados, habrá que llevar a cabo las acciones contrarias.

De cara a la elección de estos parámetros, cabe destacar que que un valor de $min\_samples$ alto o un valor de ϵ bajo indican que se requiere una mayor densidad para formar cluster.

Dado que para definir el grado de densidad necesario para formar un cluster solo se necesita variar ϵ o $min\_samples$, y no ambas, se puede simplificar la elección de parámetros fijando uno a un valor arbitrario que tenga sentido.

En este caso, $min\_samples$ es el parámetro más sencillo de entender y escoger inicialmente, dado que solo depende del número de observaciones en el *dataset*. Para elegir $min\_samples$ se utilizará el **conocimiento previo del dominio**, concretamente el número de muestras que tiene la clase con menos observaciones.

Como regla de oro, el parámetro $min\_samples$ ha de ser al menos igual, si no mayor, al número de dimensiones del *dataset*. Algunas heurísticas que suelen funcionar son $min\_samples = features \times 2$ y $min\_samples = features + 1$.

Por último, se utilizará únicamente el conjunto de datos normalizado y sin atributos categóricos, dado que estos algoritmos se basan en la distancia euclídea.

Se procede a comprobar el número de muestras en cada clase y elegir un número mínimo de muestras por *cluster* que tenga sentido al respecto, relajando el número de $min\_samples$. A partir del número de muestras, se obtendrá el valor de ϵ, sabiendo que se buscan 2 *clusters*.

```
summary(as.factor(norm.sample.labels$success))
```

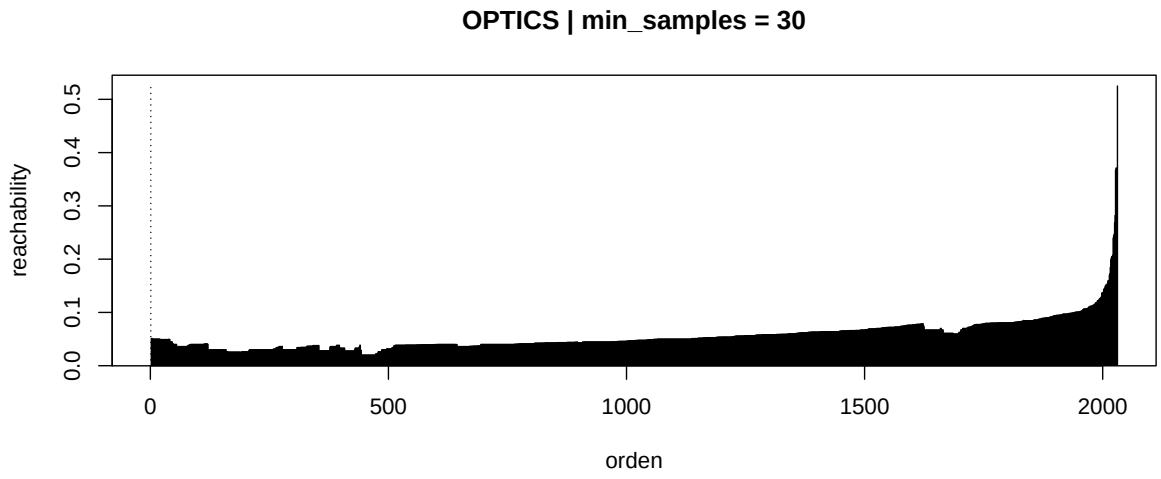
```
##    0    1
##  852 1179
```

```
min_samples <- 30
set.seed(seed)
optics.model <- dbscan::optics(norm.sample.nofactors, minPts=min_samples)
optics.model
```

```
## OPTICS ordering/clustering for 2031 objects.
## Parameters: minPts = 30, eps = 0.837792337207287, eps_cl = NA, xi = NA
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
```

```
##                   eps_cl, xi
```

```
plot(optics.model, main=paste('OPTICS | min_samples = ', min_samples, sep=''),
     xlab='orden', y='reachability')
```



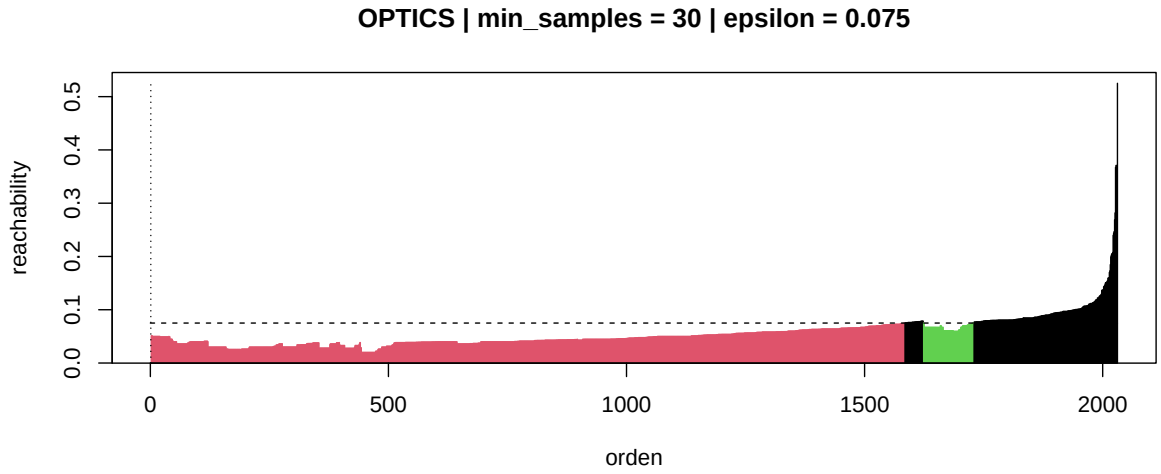
Un valor de ϵ adecuado para 500 muestras como mínimo para formar *cluster* sería de $\epsilon = 0.075$. Se obtendrán algunos puntos marcados como *outliers*, pero la mayoría caerán dentro de uno de los 2 *clusters*. Se probará también con $\epsilon = 0.05$.

Por otro lado, se intuye que no se va a realizar una buena segmentación: el primer pico incluye muy pocas observaciones, mientras que el segundo pico se lleva la mayoría de las observaciones del *dataset*.

```
eps <- 0.075
eps2 <- 0.05
set.seed(seed)
optics.eps <- dbscan::extractDBSCAN(optics.model, eps_cl=eps)
optics.eps
```

```
## OPTICS ordering/clustering for 2031 objects.
## Parameters: minPts = 30, eps = 0.837792337207287, eps_cl = 0.075, xi = NA
## The clustering contains 2 cluster(s) and 341 noise points.
##
##    0    1    2
##  341 1584  106
##
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi, cluster
```

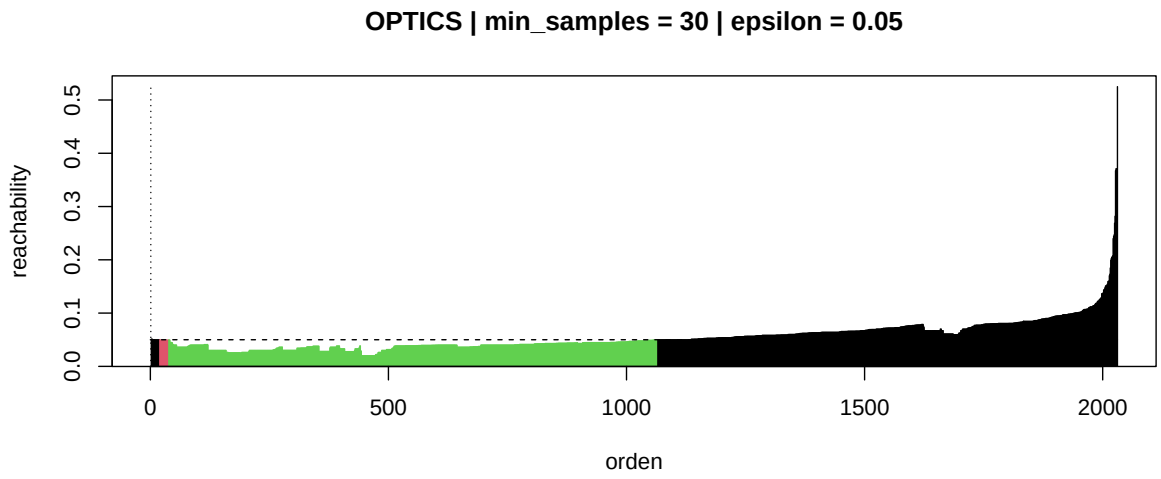
```
plot(optics.eps, main=paste('OPTICS | min_samples = ', min_samples,
                            ' | epsilon = ', eps, sep=''),
     xlab='orden', y='reachability', cl=palette)
```




```
set.seed(seed)
optics.eps2 <- dbscan::extractDBSCAN(optics.model, eps_cl=eps2)
optics.eps2
```

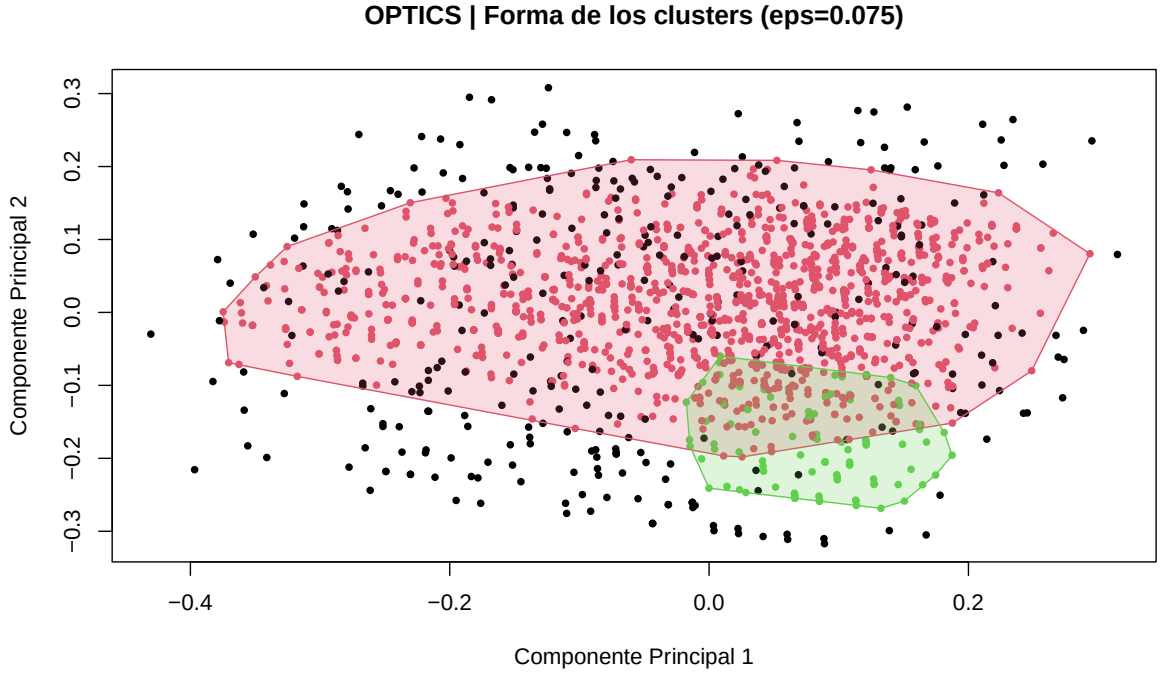
```
## OPTICS ordering/clustering for 2031 objects.
## Parameters: minPts = 30, eps = 0.837792337207287, eps_cl = 0.05, xi = NA
## The clustering contains 2 cluster(s) and 985 noise points.
##
##    0    1    2
##  985   19 1027
##
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi, cluster
```

```
plot(optics.eps2, main=paste('OPTICS | min_samples = ', min_samples,
                            ' | epsilon = ', eps2, sep=''),
     xlab='orden', y='reachability', cl=palette)
```



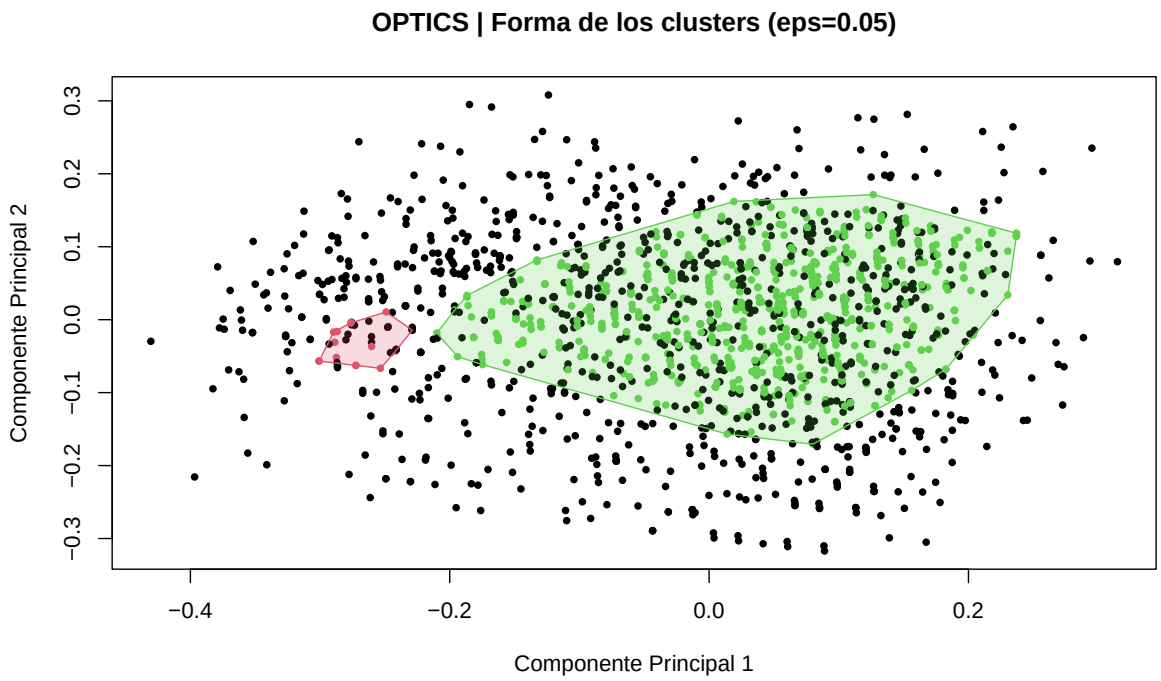
Se han obtenido 2 *clusters*, reservando el $cluster = 0$ para los *outliers*, representados de color negro. Se procede a representar gráficamente este agrupamiento:

```
dbscan::hullplot(norm.sample.nofactors, optics.eps, fill=c('black', palette),
                 pch=16, cex=0.75, main='OPTICS | Forma de los clusters (eps=0.075)',
                 xlab='Componente Principal 1', ylab='Componente Principal 2')
```



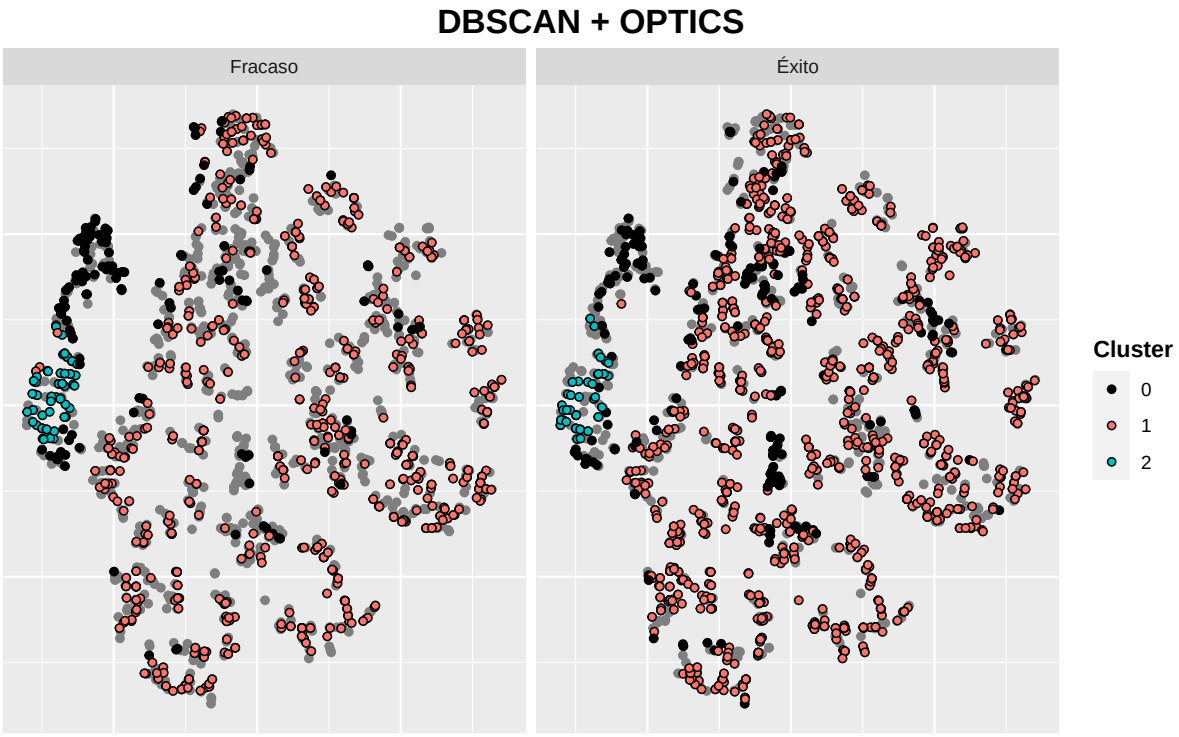
Patricia Lázaro Tello
plazarotello@uoc.edu

```
dbscan::hullplot(norm.sample.nofactors, optics.eps2, fill=c('black', palette),
                 pch=16, cex=0.75, main='OPTICS | Forma de los clusters (eps=0.05)',
                 xlab='Componente Principal 1', ylab='Componente Principal 2')
```

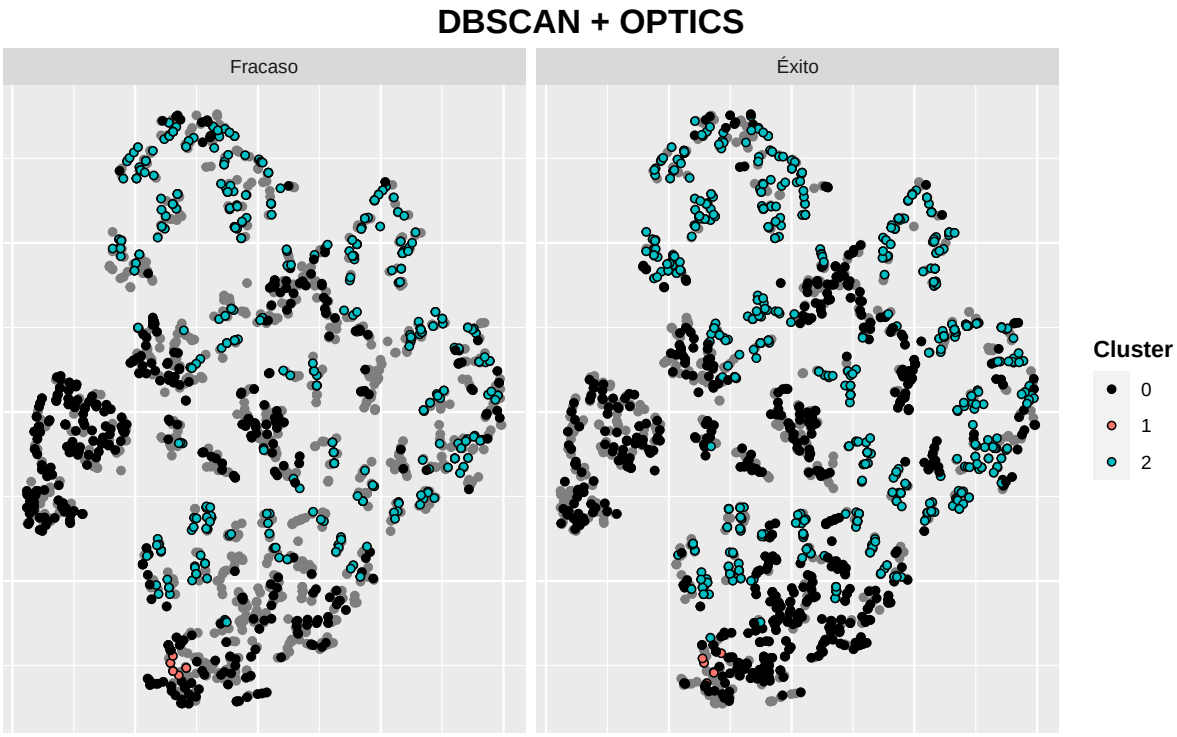



```
tsne.optics <- transform.tsne(optics.eps, norm.sample.nofactors,
                              norm.sample.labels$success)
tsne.optics2 <- transform.tsne(optics.eps2, norm.sample.nofactors,
                              norm.sample.labels$success)

ggplot(data=tsne.optics, mapping=aes(x=x, y=y)) +
  geom_point(color='grey50', data=transform(tsne.optics, label=NULL)) +
  geom_point(aes(fill=factor(cluster)), shape=21) +
  scale_fill_manual(name='Cluster', values=c('black', palette)) +
  theme(legend.position='right', legend.title=element_text(face='bold')) +
  title.centered + no.axis.x + no.axis.y + labs(x=NULL, y=NULL) +
  facet_wrap(vars(label), labeller=success.labeller.2) +
  ggtitle('DBSCAN + OPTICS')
```




```
ggplot(data=tsne.optics2, mapping=aes(x=x, y=y)) +
  geom_point(color='grey50', data=transform(tsne.optics2, label=NULL)) +
  geom_point(aes(fill=factor(cluster)), shape=21) +
  scale_fill_manual(name='Cluster', values=c('black', palette)) +
  theme(legend.position='right', legend.title=element_text(face='bold')) +
  title.centered + no.axis.x + no.axis.y + labs(x=NULL, y=NULL) +
  facet_wrap(vars(label), labeller=success.labeller.2) +
  ggtitle('DBSCAN + OPTICS')
```



Se observa que los datos mostrados con reducción de dimensionalidad (PCA) no muestran grupos fácilmente separables para ninguna ϵ probada. Se procede a calcular y mostrar las métricas de evaluación, y para ello se elegirá primero qué *cluster* corresponde a qué clase:

```
round(prop.table(100.0*table(as.factor(tsne.optics$cluster),
                             as.factor(tsne.optics$label),
                             dnn=c('clusters', 'etiquetas')), margin=2), 2)
```

```
##          etiquetas
## clusters    0    1
##        0 0.18 0.16
##        1 0.75 0.80
##        2 0.08 0.04
```

```
round(prop.table(100.0*table(as.factor(tsne.optics2$cluster),
                             as.factor(tsne.optics2$label),
                             dnn=c('clusters', 'etiquetas')), margin=2), 2)
```

```
##          etiquetas
## clusters    0    1
##        0 0.49 0.48
##        1 0.01 0.01
```

```
##         2 0.50 0.51
```

ϵ	*cluster*	etiqueta
0.075	0	Fracaso
0.075	1	Éxito
0.075	2	Fracaso
0.05	0	Fracaso
0.05	1	Éxito
0.05	2	Éxito

```
optics.df <- data.frame(cluster=optics.eps$cluster,
                            label=norm.sample.labels$success) %>%
  dplyr::mutate(cluster=ifelse(cluster==1, 1, 0))

optics.confusion <- caret::confusionMatrix(data=as.factor(
  optics.df$cluster),reference=as.factor(optics.df$label),
  positive='1')
optics.confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 215 232
##          1 637 947
##
##                Accuracy : 0.572
##                  95% CI : (0.55, 0.594)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 0.784
##
##                   Kappa : 0.059
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.803
##             Specificity : 0.252
##          Pos Pred Value : 0.598
##          Neg Pred Value : 0.481
##              Prevalence : 0.581
##          Detection Rate : 0.466
##    Detection Prevalence : 0.780
```

```
##       Balanced Accuracy : 0.528
##
##        'Positive' Class : 1
##
```

```
optics.df2 <- data.frame(cluster=optics.eps2$cluster,
                         label=norm.sample.labels$success) %>%
  dplyr::mutate(cluster=ifelse(cluster==0, 0, 1))

optics.confusion2 <- caret::confusionMatrix(data=as.factor(
  optics.df2$cluster),reference=as.factor(optics.df2$label),
  positive='1')
optics.confusion2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 421 564
##          1 431 615
##
##                Accuracy : 0.51
##                  95% CI : (0.488, 0.532)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.015
##
##  Mcnemar's Test P-Value : 0.0000286
##
##             Sensitivity : 0.522
##             Specificity : 0.494
##          Pos Pred Value : 0.588
##          Neg Pred Value : 0.427
##              Prevalence : 0.581
##          Detection Rate : 0.303
##    Detection Prevalence : 0.515
##       Balanced Accuracy : 0.508
##
##        'Positive' Class : 1
##
```

	OPTICS ($\epsilon = 0.075$)	OPTICS ($\epsilon = 0.05$)
Precisión	0,6	0,59
Recall	0,8	0,52
F_1 *Score*	0,69	0,55

Los modelos DBSCAN+OPTICS no tienen la calidad deseada. Los valores de precisión son bajos (menores de 70%), aunque el *recall* con $\epsilon = 0.075$ es suficientemente alto. Parece que el modelo necesita de más atributos discriminantes para poder distinguir mejor las 2 clases. Se reserva el modelo $\epsilon = 0.075$.

Patricia Lázaro Tello
plazarotello@uoc.edu

6 Árboles de decisión

El árbol de decisión es uno de los modelos supervisados de clasificación con mayor capacidad explicativa y de más fácil interpretación; por ello es uno de los más populares para problemas de clasificación.

Acepta tanto variables continuas como categóricas, porque a la hora de crear las reglas corta las variables continuas para transformarlas en categóricas.

Además, no es necesario normalizar los datos de entrada, por lo que se utilizará el *dataset* muestreado sin normalizar, que aporta más información.

Por último, será necesario reservar una parte de los datos para realizar la validación:

```
kickstarter.sample <- kickstarter.sample %>% dplyr::mutate(
  success=as.factor(success>0))
kickstarter.norm.sample <- kickstarter.norm.sample %>% dplyr::mutate(
  success=as.factor(success>0))
tmp.success <- kickstarter.sample[kickstarter.sample$success=='TRUE',]
tmp.fail <- kickstarter.sample[kickstarter.sample$success=='FALSE',]

tmp.success.norm <- kickstarter.norm.sample[
  kickstarter.norm.sample$success=='TRUE',]
tmp.fail.norm <- kickstarter.norm.sample[
  kickstarter.norm.sample$success=='FALSE',]

train.pctg <- 0.8  # 80%

tmp.success.split <- round(nrow(tmp.success) * train.pctg)
tmp.fail.split <- round(nrow(tmp.fail) * train.pctg)

train <- bind_rows(tmp.success[1:tmp.success.split,],
                   tmp.fail[1:tmp.fail.split,])
test <- bind_rows(tmp.success[(tmp.success.split+1):nrow(tmp.success),],
                   tmp.fail[(tmp.fail.split+1):nrow(tmp.fail),])

train.norm <- bind_rows(tmp.success.norm[1:tmp.success.split,],
                   tmp.fail[1:tmp.fail.split,])
test.norm <- bind_rows(tmp.success.norm[(tmp.success.split+1):
                                          nrow(tmp.success.norm),],
                   tmp.fail[(tmp.fail.split+1):nrow(tmp.fail),])
```

Patricia Lázaro Tello
plazarotello@uoc.edu

A continuación se muestra la distribución de las clases en los conjuntos de datos de entrenamiento y test. Se comprueba que la proporción de clases en ambos conjuntos sea correcta.

	Dataset muestreado	Entrenamiento	Test
Éxito	1.179	943	236
Fracaso	852	682	170

6.1 Árbol de decisión básico

Se procede a crear un árbol de decisión básico (sin podas ni *adaptative boosting*) y entrenarlo con el conjunto de datos de entrenamiento. Se establecen también ciertos parámetros de control (las hojas han de tener al menos 10 observaciones) para evitar el sobreentrenamiento y obtener un modelo más legible.

```
simple.params <- C50::C5.0Control(minCases=10)

set.seed(seed)
simple.model <- C50::C5.0(dplyr::select(train, -success), train$success,
                          control=simple.params)
```

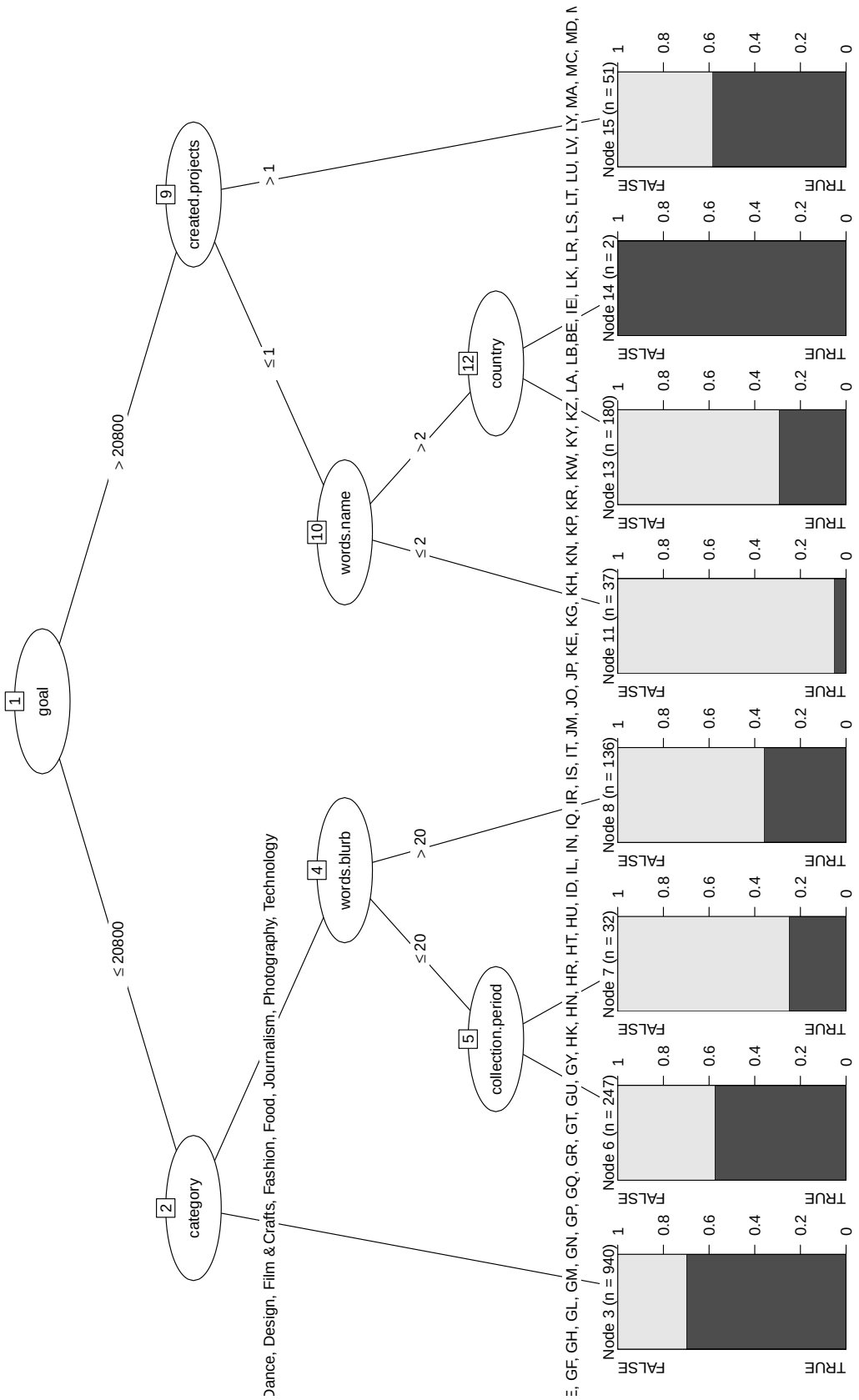


Figura 1: Árbol de clasificación

Patricia Lázaro Tello
plazarotello@uoc.edu

Las reglas que el árbol ha generado son las siguientes:

```
summary(simple.model)
```

```
##
## Call:
## C5.0.default(x = dplyr::select(train, -success), y = train$success, control
##  = simple.params)
##
##
## C5.0 [Release 2.07 GPL Edition]      Fri Jan 07 12:03:14 2022
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 1625 cases (10 attributes) from undefined.data
##
## Decision tree:
##
## goal <= 20800:
## :...category in {Art,Comics,Dance,Design,Film & Video,Games,Music,Publishing,
## :   :            Theater}: TRUE (940/283)
## :   category in {Crafts,Fashion,Food,Journalism,Photography,Technology}:
## :   :...words.blurb > 20: FALSE (136/49)
## :       words.blurb <= 20:
## :       :...collection.period <= 50: TRUE (247/105)
## :           collection.period > 50: FALSE (32/8)
## goal > 20800:
## :...created.projects > 1: TRUE (51/21)
##     created.projects <= 1:
##     :...words.name <= 2: FALSE (37/2)
##         words.name > 2:
##         :...country in {AE,AF,AG,AI,AL,AM,AO,AR,AS,AT,AU,AX,AZ,BA,BB,BD,BF,BG,
##             :           BH,BJ,BM,BO,BR,BS,BT,BW,BY,BZ,CA,CG,CH,CI,CK,CL,CM,CN,
##             :           CO,CR,CU,CV,CY,CZ,DE,DJ,DK,DM,DO,DZ,EC,EE,EG,ES,ET,FI,
##             :           FJ,FM,FO,FR,GA,GB,GE,GF,GH,GL,GM,GN,GP,GQ,GR,GT,GU,GY,
##             :           HK,HN,HR,HT,HU,ID,IL,IN,IQ,IR,IS,IT,JM,JO,JP,KE,KG,KH,
##             :           KN,KP,KR,KW,KY,KZ,LA,LB,LC,LI,LK,LR,LS,LT,LU,LV,LY,MA,
##             :           MC,MD,MG,MK,ML,MM,MN,MQ,MR,MT,MU,MV,MW,MX,MY,MZ,NC,NE,
##             :           NG,NI,NL,NO,NP,NZ,OM,PA,PE,PF,PG,PH,PK,PL,PR,PS,PT,PY,
##             :           QA,RE,RO,RS,RU,RW,SA,SC,SD,SE,SG,SI,SJ,SK,SL,SN,SO,SR,
##             :           SS,SV,SY,TC,TD,TH,TJ,TL,TN,TO,TR,TT,TW,TZ,UA,UG,US,UY,
##             :           UZ,VC,VE,VN,VU,WS,YE,ZA,ZM,ZW}: FALSE (180/53)
##             country in {BE,IE}: TRUE (2)
##
##
```

```
## Evaluation on training data (1625 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       8  521(32.1%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     273   409    (a): class FALSE
##     112   831    (b): class TRUE
##
##
##  Attribute usage:
##
##  100.00% goal
##   83.38% category
##   25.54% words.blurb
##   17.17% collection.period
##   16.62% created.projects
##   13.48% words.name
##   11.20% country
##
##
## Time: 0.0 secs
```

No se han utilizado todos los atributos en la construcción del modelo dado que no todos ellos discriminaban entre las clases suficientemente bien. Además, se ha producido una tasa de error del 32.1% en el conjunto de datos de entrenamiento.

El objetivo de financiación y la categoría del proyecto son los factores más influyentes para considerar el éxito o fracaso de la campaña. Otros atirbutos discriminantes son el número de palabras en la descripción, el periodo de recaudación o cuántos proyectos ha creado el usuario.

Respecto a las reglas que el modelo ha obtenido, se describen como:

- Si el objetivo de financiación es igual o menor a 20800 dólares y se encuentra en una de las siguientes categorías (Arte, Cómics, Danza, Diseño, Películas y vídeos, Juegos, Música, Publicación o Teatro), entonces se financiará con éxito.
- Si el objetivo de financiación es igual o menor a 20800 dólares, se encuentra en una de las categorías de Manualidades, Moda, Comida, Periodismo, Fotografía

o Tecnología, y tiene más de 20 palabras en la descripción, no se financiará con éxito.

- Si el objetivo de financiación es igual o menor a 20800 dólares, se encuentra en una de las categorías de Manualidades, Moda, Comida, Periodismo, Fotografía o Tecnología, y tiene 20 palabras o menos en la descripción, se financiará con éxito.
- Si el objetivo de financiación es mayor a 20800 dólares y el usuario ha creado más de 1 proyecto, entonces se financiará con éxito.
- Si el objetivo de financiación es mayor a 20800 dólares, el usuario ha creado 1 proyecto o menos, y el nombre tiene 2 o menos palabras, entonces no se financiará con éxito.
- Si el objetivo de financiación es mayor a 20800 dólares, el usuario ha creado 1 proyecto o menos, el nombre tiene más de 2 palabras, y el país de origen del proyecto es Bélgica o Irlanda, entonces se financiará con éxito.
- Si el objetivo de financiación es mayor a 20800 dólares, el usuario ha creado 1 proyecto o menos, el nombre tiene más de 2 palabras, y el país de origen del proyecto no es Bélgica o Irlanda, entonces no se financiará con éxito.

Se procede a analizar la calidad del modelo con los datos de entrenamiento y los datos de test:

```
preds.train <- predict(simple.model, dplyr::select(train, -success),
                       type='class')
cm.train <- caret::confusionMatrix(data=as.factor(preds.train),
                                   reference=train$success,
                                   positive='TRUE')
cm.train
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE   273  112
##      TRUE    409  831
##
##                Accuracy : 0.679
##                  95% CI : (0.656, 0.702)
##     No Information Rate : 0.58
##     P-Value [Acc > NIR] : <0.0000000000000002
```

```
##
##                  Kappa : 0.3
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.881
##             Specificity : 0.400
##          Pos Pred Value : 0.670
##          Neg Pred Value : 0.709
##              Prevalence : 0.580
##          Detection Rate : 0.511
##    Detection Prevalence : 0.763
##       Balanced Accuracy : 0.641
##
##        'Positive' Class : TRUE
##
```

```
train.accuracy <- cm.train$byClass[['Balanced Accuracy']][1]
train.precision <- cm.train$byClass[['Precision']][1]
train.recall <- cm.train$byClass[['Recall']][1]
train.f1score <- cm.train$byClass[['F1']][1]

preds.test <- predict(simple.model, dplyr::select(test, -success),
                      type='class')
cm.test <- caret::confusionMatrix(data=as.factor(preds.test),
                                  reference=test$success,
                                  positive='TRUE')
cm.test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    68   28
##      TRUE    102  208
##
##                Accuracy : 0.68
##                  95% CI : (0.632, 0.725)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 0.000028103722
##
##                   Kappa : 0.3
##
##  Mcnemar's Test P-Value : 0.000000000153
##
```

```
##             Sensitivity : 0.881
##             Specificity : 0.400
##          Pos Pred Value : 0.671
##          Neg Pred Value : 0.708
##              Prevalence : 0.581
##          Detection Rate : 0.512
##    Detection Prevalence : 0.764
##       Balanced Accuracy : 0.641
##
##        'Positive' Class : TRUE
##
```

```
test.accuracy <- cm.test$byClass[['Balanced Accuracy']][1]
test.precision <- cm.test$byClass[['Precision']][1]
test.recall <- cm.test$byClass[['Recall']][1]
test.f1score <- cm.test$byClass[['F1']][1]
```

Dataset	*Accuracy*	*Precision*	*Recall*	F_1 *score*
Entrenamiento	64%	67%	88%	0,76
Test	64%	67%	88%	0,76

El árbol de decisión básico no tiene la calidad deseada. Los valores de precisión son correctos (sobre el 70%) y el *recall* es muy alto, provocando que el F_1 *score* sea más alto. Sin embargo, no se obtiene un buen balance entre precisión y *recall*, ya que la primera debería ser algo más alta.

6.2 Árbol de decisión con *adaptative boosting* y matriz de costes

El árbol de decisión básico ha obtenido varias mejoras desde su inicio. Una de ellas es el llamado «*adaptative boosting*», una técnica que genera múltiples árboles de decisión y los utiliza para votar democráticamente qué clase es la predicha cuando se encuentra un caso nuevo.

Otra mejora del árbol de decisión básico consiste en asignar costes a cada tipo de error, de forma que se penaliza más un tipo de error que otro. En este caso, se penalizará más clasificar un proyecto como de éxito cuando era fracaso.

```
costs <- matrix(c(NA, 1,
                  1.5, NA), 2, 2, byrow=TRUE,
```

```
                dimnames=list(c('FALSE', 'TRUE'), c('FALSE', 'TRUE')))

complex.params <- C50::C5.0Control(minCases=5, subset=TRUE)

set.seed(seed)
complex.model <- C50::C5.0(dplyr::select(train, -success), train$success,
                           control=complex.params, costs=costs, trials=50)
```

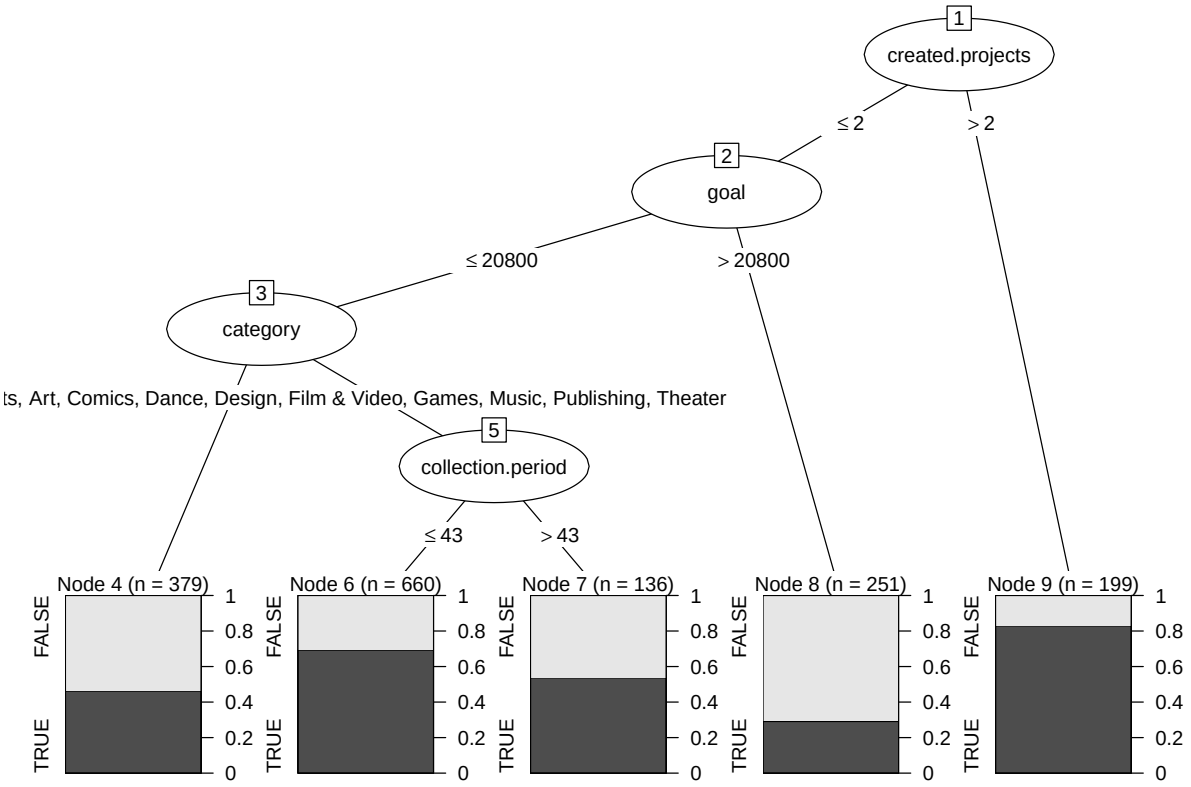


Figura 2: Árbol de clasificación complejo

Las reglas que el árbol ha generado son las siguientes:

```
summary(complex.model)
```

```
##
## Call:
## C5.0.default(x = dplyr::select(train, -success), y = train$success, trials
##  = 50, control = complex.params, costs = costs)
##
##
## C5.0 [Release 2.07 GPL Edition]      Fri Jan 07 12:03:15 2022
```

```
## ------------------------------
##
## Class specified by attribute `outcome'
##
## Read 1625 cases (10 attributes) from undefined.data
## Read misclassification costs from undefined.costs
##
## -----  Trial 0:  -----
##
## Decision tree:
##
## created.projects > 2: TRUE (199/34)
## created.projects <= 2:
## :...goal > 20800: FALSE (251/73)
##     goal <= 20800:
##     :...category in {Crafts,Fashion,Food,Journalism,Photography,
##         :            Technology}: FALSE (379/175)
##         category in {Art,Comics,Dance,Design,Film & Video,Games,Music,
##         :            Publishing,Theater}:
##         :...collection.period <= 43: TRUE (660/203)
##             collection.period > 43: FALSE (136/73)
##
## -----  Trial 1:  -----
##
## Decision tree:
##
## goal > 99999: FALSE (62.5/7.3)
## goal <= 99999:
## :...category in {Comics,Dance}: TRUE (85.6/9.4)
##     category in {Art,Crafts,Design,Fashion,Film & Video,Food,Games,Journalism,
##     :            Music,Photography,Publishing,Technology,Theater}:
##     :...created.projects > 1: TRUE (343.2/100.9)
##         created.projects <= 1:
##         :...words.name <= 2: FALSE (153/60.8)
##             words.name > 2:
##             :...words.blurb > 20: FALSE (373.1/177.1)
##                 words.blurb <= 20:
##                 :...collection.period <= 48: TRUE (538.2/179.1)
##                     collection.period > 48: FALSE (74.1/34.4)
##
## -----  Trial 2:  -----
##
## Decision tree:
##
## goal > 37516: FALSE (153.2/47.3)
## goal <= 37516:
```

```
## :...category in {Art,Comics,Dance,Design,Fashion,Film & Video,Games,Journalism,
##     :            Music,Photography,Publishing,Theater}: TRUE (1177/413.5)
##     category in {Crafts,Food,Technology}: FALSE (297/138.7)
##
## -----  Trial 3:  -----
##
## Decision tree:
##
## goal > 99999: FALSE (55.2/9.7)
## goal <= 99999:
## :...created.projects > 2: TRUE (167.2/44.4)
##     created.projects <= 2:
##     :...words.name <= 6: FALSE (840.4/446.1)
##         words.name > 6:
##         :...created.projects > 1: TRUE (88.9/30.6)
##             created.projects <= 1:
##             :...words.name <= 8: TRUE (259.3/94)
##                 words.name > 8: FALSE (205.3/114.7)
##
## -----  Trial 4:  -----
##
## Decision tree:
##
## goal > 99999: FALSE (52.7/10.4)
## goal <= 99999:
## :...created.projects > 5: TRUE (51.9/10.3)
##     created.projects <= 5:
##     :...created.projects > 1: TRUE (317.8/114.4)
##         created.projects <= 1:
##         :...goal <= 7500: TRUE (763.8/288.8)
##             goal > 7500: FALSE (442.9/224.8)
##
## -----  Trial 5:  -----
##
## Decision tree:
##
## created.projects <= 5: FALSE (1572.6/896.6)
## created.projects > 5: TRUE (50/10.8)
##
## -----  Trial 6:  -----
##
## Decision tree:
##
## created.projects <= 5: FALSE (1584.9/931.4)
## created.projects > 5: TRUE (48.9/11)
##
```

```
## -----  Trial 7:  -----
##
## Decision tree:
##
## goal > 99999: FALSE (49.3/11.4)
## goal <= 99999:
## :...collection.period <= 10: FALSE (31.8/12.2)
##     collection.period > 10: TRUE (1558.2/595.7)
##
## -----  Trial 8:  -----
##
## Decision tree:
##
## category in {Comics,Dance}: TRUE (74/12.3)
## category in {Art,Crafts,Design,Fashion,Film & Video,Food,Games,Journalism,
## :            Music,Photography,Publishing,Technology,Theater}:
## :...created.projects <= 5: FALSE (1517.8/867.1)
##     created.projects > 5: TRUE (37.9/9.6)
##
## -----  Trial 9:  -----
##
## Decision tree:
##
## goal > 99999: FALSE (47.4/12)
## goal <= 99999:
## :...words.name <= 2: FALSE (192.7/102.6)
##     words.name > 2: TRUE (1402/522.4)
##
## -----  Trial 10:  -----
##
## Decision tree:
##
## goal > 48500: FALSE (113.7/44)
## goal <= 48500:
## :...words.blurb <= 18: TRUE (701.6/249.5)
##     words.blurb > 18:
##     :...words.name <= 10: FALSE (769.9/434.8)
##         words.name > 10: TRUE (47.4/12.6)
##
## -----  Trial 11:  -----
##
## Decision tree:
##
## goal > 111503.9: FALSE (31.2/6.6)
## goal <= 111503.9:
## :...created.projects > 2: TRUE (158.4/48)
```

```
##     created.projects <= 2:
##     :...created.projects <= 1: FALSE (1234.9/727.6)
##         created.projects > 1: TRUE (211.7/79.7)
##
## -----  Trial 12:  -----
##
## Decision tree:
##
## goal <= 111503.9: TRUE (1611.7/623.6)
## goal > 111503.9: FALSE (30.6/6.8)
##
## -----  Trial 13:  -----
##
## Decision tree:
##
## goal <= 111503.9: TRUE (1605.5/637.1)
## goal > 111503.9: FALSE (30.3/6.9)
##
## -----  Trial 14:  -----
##
## Decision tree:
##
## goal <= 20800: TRUE (1385.2/536.7)
## goal > 20800: FALSE (247.3/117)
##
## -----  Trial 15:  -----
##
## Decision tree:
##
## collection.period > 48: FALSE (203.4/96.4)
## collection.period <= 48:
## :...created.projects > 2: TRUE (135.9/41.9)
##     created.projects <= 2:
##     :...collection.period <= 10: FALSE (26.3/8.1)
##         collection.period > 10:
##         :...collection.period <= 28: TRUE (246.4/82.1)
##             collection.period > 28: FALSE (1013.8/582.5)
##
## -----  Trial 16:  -----
##
## Decision tree:
##
## goal <= 111503.9: TRUE (1608.4/636.2)
## goal > 111503.9: FALSE (28.7/7.1)
##
## -----  Trial 17:  -----
```

```
##
## Decision tree:
##
## collection.period <= 48: TRUE (1429.6/561.9)
## collection.period > 48: FALSE (204/101.2)
##
## -----  Trial 18:  -----
##
## Decision tree:
##  FALSE (1628.9/954.8)
##
## *** boosting reduced to 18 trials since last classifier is very inaccurate
##
##
## Evaluation on training data (1625 cases):
##
## Trial           Decision Tree
## -----      ----------------------
##    Size      Errors   Cost
##
##    0      5  558(34.3%)   0.42
##    1      7  550(33.8%)   0.43
##    2      3  543(33.4%)   0.45
##    3      6  707(43.5%)   0.48
##    4      5  584(35.9%)   0.48
##    5      2  890(54.8%)   0.55
##    6      2  890(54.8%)   0.55
##    7      3  626(38.5%)   0.57
##    8      3  831(51.1%)   0.52
##    9      3  607(37.4%)   0.53
##   10      4  685(42.2%)   0.50
##   11      4  738(45.4%)   0.48
##   12      2  638(39.3%)   0.59
##   13      2  638(39.3%)   0.59
##   14      2  586(36.1%)   0.51
##   15      5  732(45.0%)   0.48
##   16      2  638(39.3%)   0.59
##   17      2  632(38.9%)   0.56
## boost            496(30.5%)   0.40   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     381   301    (a): class FALSE
##     195   748    (b): class TRUE
##
```

```
##
##  Attribute usage:
##
##  100.00% created.projects
##  100.00% category
##  100.00% collection.period
##  100.00% goal
##   95.82% words.name
##   93.85% words.blurb
##
##
## Time: 0.0 secs
```

Se observa que en este caso se han utilizado los proyectos creados por el usuario, la categoría a la que pertenece el proyecto, el período de recaudación, el objetivo de recaudación y el número de palabras en el nombre y en la descripción como principales atributos del árbol de clasificación.

Se procede a analizar la calidad del modelo con los datos de entrenamiento y los datos de test:

```
ada.preds.train <- predict(complex.model, dplyr::select(train, -success),
                           type='class')
ada.cm.train <- caret::confusionMatrix(data=as.factor(ada.preds.train),
                                       reference=train$success,
                                       positive='TRUE')
ada.cm.train
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE   192   49
##      TRUE    490  894
##
##                Accuracy : 0.668
##                  95% CI : (0.645, 0.691)
##     No Information Rate : 0.58
##     P-Value [Acc > NIR] : 0.000000000000205
##
##                   Kappa : 0.252
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.948
```

```
##             Specificity : 0.282
##          Pos Pred Value : 0.646
##          Neg Pred Value : 0.797
##              Prevalence : 0.580
##          Detection Rate : 0.550
##    Detection Prevalence : 0.852
##       Balanced Accuracy : 0.615
##
##        'Positive' Class : TRUE
##
```

```
ada.train.accuracy <- ada.cm.train$byClass[['Balanced Accuracy']][1]
ada.train.precision <- ada.cm.train$byClass[['Precision']][1]
ada.train.recall <- ada.cm.train$byClass[['Recall']][1]
ada.train.f1score <- ada.cm.train$byClass[['F1']][1]

ada.preds.test <- predict(complex.model, dplyr::select(test, -success),
                          type='class')
ada.cm.test <- caret::confusionMatrix(data=as.factor(ada.preds.test),
                                      reference=test$success,
                                      positive='TRUE')
ada.cm.test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    49   11
##      TRUE    121  225
##
##                Accuracy : 0.675
##                  95% CI : (0.627, 0.72)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 0.0000666
##
##                   Kappa : 0.266
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.953
##             Specificity : 0.288
##          Pos Pred Value : 0.650
##          Neg Pred Value : 0.817
##              Prevalence : 0.581
##          Detection Rate : 0.554
```

```
##    Detection Prevalence : 0.852
##       Balanced Accuracy : 0.621
##
##        'Positive' Class : TRUE
##
```

```
ada.test.accuracy <- ada.cm.test$byClass[['Balanced Accuracy']][1]
ada.test.precision <- ada.cm.test$byClass[['Precision']][1]
ada.test.recall <- ada.cm.test$byClass[['Recall']][1]
ada.test.f1score <- ada.cm.test$byClass[['F1']][1]
```

Dataset	*Accuracy*	*Precision*	*Recall*	F_1 *score*
Entrenamiento	61%	65%	95%	0,77
Test	62%	65%	95%	0,77

El árbol de decisión con *adaptative boosting* tampoco tiene la calidad deseada. Los valores de precisión son algo más bajos que los del árbol de decisión básico, pero el *recall* es muy alto, por lo que no hay un balance entre precisión y *recall*.

6.3 Árbol de decisión RPART

Los árboles de decisión anteriores estaban basados en la metodología C5.0, desarrollada por Ross Quinlan en 1993. En esta sección se utilizará otra metodología para la creación de árboles de decisión llamada RPART, acrónimo de *Recursive PARTitioning*.

```
rpart.model <- rpart::rpart(success~., data=train, minsplit=5, maxdepth=5)
```

Se observa que se utilizan únicamente la categoría, el objetivo de recaudación y el país de origen del proyecto. Se procede a observar la calidad del modelo en entrenamiento y test:

```
rpart.preds.train <- predict(rpart.model, dplyr::select(train, -success),
                             type='class')
rpart.cm.train <- caret::confusionMatrix(data=as.factor(rpart.preds.train),
                                         reference=train$success,
                                         positive='TRUE')
rpart.cm.train
```

```
## Confusion Matrix and Statistics
```

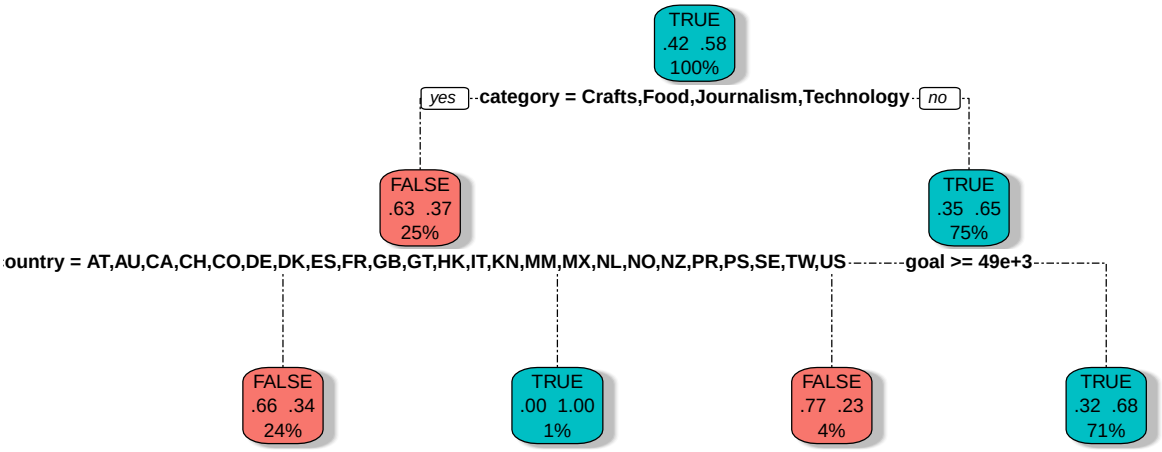


Figura 3: Árbol de clasificación RPART

```
##
##           Reference
## Prediction FALSE TRUE
##      FALSE   306  149
##      TRUE    376  794
##
##                Accuracy : 0.677
##                  95% CI : (0.654, 0.7)
##     No Information Rate : 0.58
##     P-Value [Acc > NIR] : 0.000000000000000746
##
##                   Kappa : 0.305
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.842
##             Specificity : 0.449
##          Pos Pred Value : 0.679
##          Neg Pred Value : 0.673
##              Prevalence : 0.580
##          Detection Rate : 0.489
##    Detection Prevalence : 0.720
##       Balanced Accuracy : 0.645
```

```
##
##        'Positive' Class : TRUE
##
```

```
rpart.train.accuracy <- rpart.cm.train$byClass[['Balanced Accuracy']][1]
rpart.train.precision <- rpart.cm.train$byClass[['Precision']][1]
rpart.train.recall <- rpart.cm.train$byClass[['Recall']][1]
rpart.train.f1score <- rpart.cm.train$byClass[['F1']][1]

rpart.preds.test <- predict(rpart.model, dplyr::select(test, -success),
                            type='class')
rpart.cm.test <- caret::confusionMatrix(data=as.factor(rpart.preds.test),
                                        reference=test$success,
                                        positive='TRUE')
rpart.cm.test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    80   22
##      TRUE     90  214
##
##                Accuracy : 0.724
##                  95% CI : (0.678, 0.767)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 0.000000001603
##
##                   Kappa : 0.4
##
##  Mcnemar's Test P-Value : 0.000000000244
##
##             Sensitivity : 0.907
##             Specificity : 0.471
##          Pos Pred Value : 0.704
##          Neg Pred Value : 0.784
##              Prevalence : 0.581
##          Detection Rate : 0.527
##    Detection Prevalence : 0.749
##       Balanced Accuracy : 0.689
##
##        'Positive' Class : TRUE
##
```

```
rpart.test.accuracy <- rpart.cm.test$byClass[['Balanced Accuracy']][1]
rpart.test.precision <- rpart.cm.test$byClass[['Precision']][1]
rpart.test.recall <- rpart.cm.test$byClass[['Recall']][1]
rpart.test.f1score <- rpart.cm.test$byClass[['F1']][1]
```

Dataset	*Accuracy*	*Precision*	*Recall*	F_1 *score*
Entrenamiento	65%	68%	84%	0,75
Test	69%	70%	91%	0,79

El árbol de decisión RPART tiene una calidad más parecida a la deseada: la precisión y *recall* aumentan en el conjunto de test y se trata de valores adecuados, aunque la precisión es algo baja.

7 Algoritmo k-NN

El algoritmo k-NN o k-vecinos más cercanos es un algoritmo de clasificación supervisada de aprendizaje vago (o *lazy learning method*). Dado un conjunto de datos de entrenamiento, para clasificar una nueva muestra se utilizan los k vecinos más cercanos a esa muestra para decidir su clase.

El algoritmo k-NN utiliza la distancia euclídea; por tanto, será necesario utilizar solo las variables numéricas del conjunto de datos:

```
train.num <- train %>% dplyr::select(-category, -launch.month, -country)
test.num <- test %>% dplyr::select(-category, -launch.month, -country)
```

Se procede a crear el modelo y evaluar su calidad en el conjunto de datos de test, utilizando $k \in \{10, 20\}$:

```
knn.k10 <- class::knn(train=dplyr::select(train.num, -success),
                      test=dplyr::select(test.num, -success),
                      cl=train.num$success, k=10)
knn.k20 <- class::knn(train=dplyr::select(train.num, -success),
                      test=dplyr::select(test.num, -success),
                      cl=train.num$success, k=20)

knn.k10.cm.test <- caret::confusionMatrix(data=as.factor(knn.k10),
                                          reference=test$success,
                                          positive='TRUE')
knn.k10.cm.test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    73   59
##      TRUE     97  177
##
##                Accuracy : 0.616
##                  95% CI : (0.567, 0.663)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 0.08682
##
##                   Kappa : 0.185
##
##  Mcnemar's Test P-Value : 0.00305
##
```

```
##             Sensitivity : 0.750
##             Specificity : 0.429
##          Pos Pred Value : 0.646
##          Neg Pred Value : 0.553
##              Prevalence : 0.581
##          Detection Rate : 0.436
##    Detection Prevalence : 0.675
##       Balanced Accuracy : 0.590
##
##        'Positive' Class : TRUE
##
```

```
knn.k10.test.accuracy <- knn.k10.cm.test$byClass[['Balanced Accuracy']][1]
knn.k10.test.precision <- knn.k10.cm.test$byClass[['Precision']][1]
knn.k10.test.recall <- knn.k10.cm.test$byClass[['Recall']][1]
knn.k10.test.f1score <- knn.k10.cm.test$byClass[['F1']][1]

knn.k20.cm.test <- caret::confusionMatrix(data=as.factor(knn.k20),
                                          reference=test$success,
                                          positive='TRUE')
knn.k20.cm.test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    54   36
##      TRUE    116  200
##
##                Accuracy : 0.626
##                  95% CI : (0.577, 0.673)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 0.0386
##
##                   Kappa : 0.177
##
##  Mcnemar's Test P-Value : 0.000000000148
##
##             Sensitivity : 0.847
##             Specificity : 0.318
##          Pos Pred Value : 0.633
##          Neg Pred Value : 0.600
##              Prevalence : 0.581
##          Detection Rate : 0.493
##    Detection Prevalence : 0.778
```

```
##       Balanced Accuracy : 0.583
##
##        'Positive' Class : TRUE
##
```

```
knn.k20.test.accuracy <- knn.k20.cm.test$byClass[['Balanced Accuracy']][1]
knn.k20.test.precision <- knn.k20.cm.test$byClass[['Precision']][1]
knn.k20.test.recall <- knn.k20.cm.test$byClass[['Recall']][1]
knn.k20.test.f1score <- knn.k20.cm.test$byClass[['F1']][1]
```

k	*Accuracy*	*Precision*	*Recall*	F_1 *score*
$k = 10$	59%	65%	75%	0,69
$k = 20$	58%	63%	85%	0,72

El algoritmo k-NN no produce modelos con la calidad deseada. Los valores de precisión son ligeramente bajos y el *recall* es correcto, provocando que el F_1 *score* sufra por la baja precisión. No se obtiene un buen balance entre precisión y *recall*.

8 Regresión logística

La regresión logística es un modelo supervisado de clasificación cuyos fundamentos se encuentran en los modelos de regresión, que son modelos de predicción de variables continuas.

En el caso de la clasificación, se predice la probabilidad de que una variable categórica tome un valor o no; se utiliza en problemas de clasificación binaria, como el planteado en este trabajo.

Para problemas de clasificación multiclase se podría utilizar este método para crear tantos modelos como clases existan, y calcular la probabilidad de que la observación sea de cada clase.

Como el resultado de este modelo es la probabilidad de que la variable tome un valor, es necesario definir un umbral o *threshold* a partir del cual se considere que la variable toma el valor positivo.

En primer lugar, es necesario transformar la variable dependiente a [0,1], donde 0 es fracaso y 1 es éxito:

```
train$success <- as.numeric(train$success) - 1
test$success <- as.numeric(test$success) - 1
summary(as.factor(train$success))
```

```
##   0   1
## 682 943
```

```
summary(as.factor(test$success))
```

```
##   0   1
## 170 236
```

Se procede a crear un primer modelo con una única variable, al que se le irán añadiendo variables gradualmente hasta llegar al modelo completo, observando en cada paso la variación del criterio de información de Akaike (AIC) — cuanto menor sea el AIC, mejor ajustará el modelo a los datos — y lo significativos que sean los coeficientes de los atributos.

```
lr.model.null <- glm(success~1, data=train, family='binomial')
lr.model.full <- glm(success~., data=train, family='binomial')
lr.model <- step(lr.model.null, direction='forward', trace=0,
                 scope=list(lower=lr.model.null, upper=lr.model.full))
```

```
summary(lr.model)
```

```
##
## Call:
## glm(formula = success ~ category + goal + created.projects +
##     collection.period + words.blurb + words.name, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.872  -1.072   0.617   0.951   2.350
##
## Coefficients:
##                          Estimate  Std. Error z value    Pr(>|z|)
## (Intercept)           1.18757826  0.31059841    3.82     0.00013 ***
## categoryComics        1.72377309  0.47129000    3.66     0.00025 ***
## categoryCrafts       -0.83513524  0.34899244   -2.39     0.01671 *
## categoryDance         1.62072886  0.64972735    2.49     0.01261 *
## categoryDesign        0.80743138  0.33081235    2.44     0.01466 *
## categoryFashion      -0.28218082  0.27278597   -1.03     0.30093
## categoryFilm & Video  0.37080147  0.22947931    1.62     0.10613
## categoryFood         -0.91393866  0.26077655   -3.50     0.00046 ***
## categoryGames         0.33519270  0.29098973    1.15     0.24936
## categoryJournalism   -0.34453667  0.39242644   -0.88     0.37996
## categoryMusic         0.44548623  0.23516870    1.89     0.05818 .
## categoryPhotography  -0.15879709  0.33122758   -0.48     0.63164
## categoryPublishing    0.29825976  0.24925591    1.20     0.23146
## categoryTechnology   -0.57814000  0.24709485   -2.34     0.01930 *
## categoryTheater       0.24875638  0.35880255    0.69     0.48812
## goal                 -0.00001383  0.00000244   -5.67 0.000000015 ***
## created.projects      0.13913444  0.04274740    3.25     0.00113 **
## collection.period    -0.02042599  0.00475547   -4.30 0.000017449 ***
## words.blurb          -0.04084607  0.01018648   -4.01 0.000060762 ***
## words.name            0.08133420  0.02090547    3.89     0.00010 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2210.6  on 1624  degrees of freedom
## Residual deviance: 1910.8  on 1605  degrees of freedom
## AIC: 1951
##
## Number of Fisher Scoring iterations: 7
```

El modelo de regresión logística ha descartado los atributos correspondientes a la exis-

tencia de una imagen en el proyecto, el mes de lanzamiento de la campaña de recaudación y el país de origen del proyecto por no aportar información relevante al modelo.

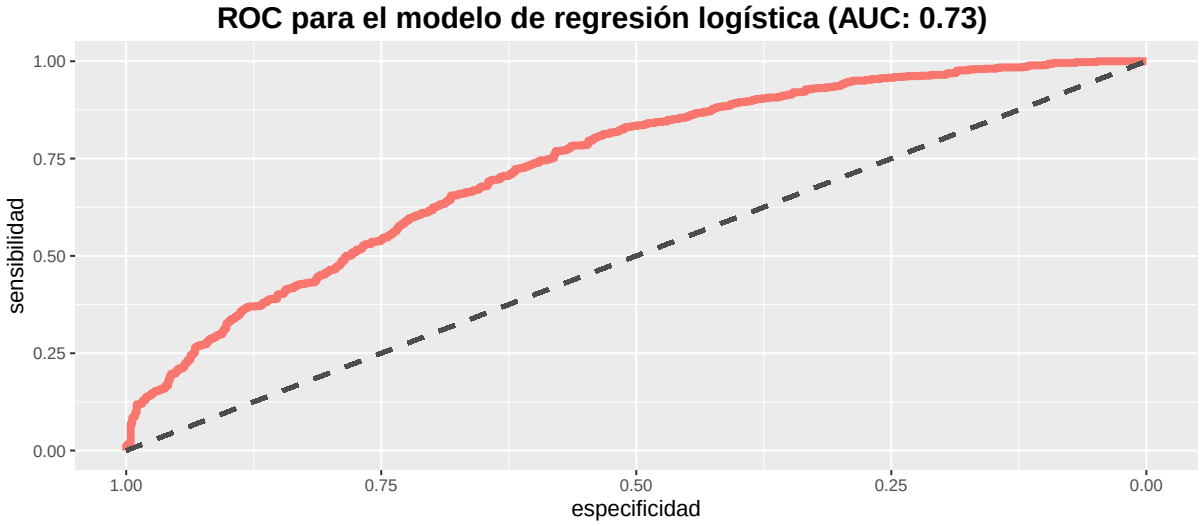
Se procede a comparar el modelo elegido contra el modelo nulo y el modelo completo:

	Modelo nulo	Modelo elegido	Modelo completo
AIC	2.213	1.951	2.007

El modelo elegido tiene el criterio de información de Akaike (AIC) más bajo de los 3; es decir, es el que mejor se ajusta a los datos. Se procede a realizar un análisis ROC para el conjunto de datos de entrenamiento:

```
prob <- predict(lr.model, type='response')
r <- roc(train$success, prob, data=train)

ggroc(r, color=default.color.main, size=2) +
  geom_segment(aes(x=1, xend=0, y=0, yend=1), color='grey30',
               linetype='dashed', size=1.05) +
  title.centered + ggtitle(paste0('ROC para el modelo de regresión logística ',
                                  '(AUC: ', round(auc(r),2), ')')) +
  xlab('especificidad') + ylab('sensibilidad')
```



Se obtiene un área bajo la curva (*area under the ROC*, *area under the curve*, AUROC o AUC) de 0,73, que indica que el modelo discrimina de forma adecuada Si el AUC hubiera sido igual o superior a 0.8, el modelo discriminaría de forma excelente.

Se procede a observar la calidad del modelo en entrenamiento y test:

```
logi.preds.train <- ifelse(predict(lr.model, dplyr::select(train, -success),
                                   type='response') >= 0.5, 1, 0)
logi.cm.train <- caret::confusionMatrix(data=as.factor(logi.preds.train),
                                        reference=as.factor(train$success),
                                        positive='1')
logi.cm.train
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 349 163
##          1 333 780
##
##                Accuracy : 0.695
##                  95% CI : (0.672, 0.717)
##     No Information Rate : 0.58
##     P-Value [Acc > NIR] : < 0.0000000000000002
##
##                   Kappa : 0.351
##
##  Mcnemar's Test P-Value : 0.0000000000000324
##
##             Sensitivity : 0.827
##             Specificity : 0.512
##          Pos Pred Value : 0.701
##          Neg Pred Value : 0.682
##              Prevalence : 0.580
##          Detection Rate : 0.480
##    Detection Prevalence : 0.685
##       Balanced Accuracy : 0.669
##
##        'Positive' Class : 1
##
```

```
logi.train.accuracy <- logi.cm.train$byClass[['Balanced Accuracy']][1]
logi.train.precision <- logi.cm.train$byClass[['Precision']][1]
logi.train.recall <- logi.cm.train$byClass[['Recall']][1]
logi.train.f1score <- logi.cm.train$byClass[['F1']][1]

logi.preds.test <- ifelse(predict(lr.model, dplyr::select(test, -success),
                                  type='response') >= 0.5, 1, 0)
logi.cm.test <- caret::confusionMatrix(data=as.factor(logi.preds.test),
```

```
                                          reference=as.factor(test$success),
                                          positive='1')
logi.cm.test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  88  29
##          1  82 207
##
##                Accuracy : 0.727
##                  95% CI : (0.68, 0.769)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 0.000000000834
##
##                   Kappa : 0.413
##
##  Mcnemar's Test P-Value : 0.000000798959
##
##             Sensitivity : 0.877
##             Specificity : 0.518
##          Pos Pred Value : 0.716
##          Neg Pred Value : 0.752
##              Prevalence : 0.581
##          Detection Rate : 0.510
##    Detection Prevalence : 0.712
##       Balanced Accuracy : 0.697
##
##        'Positive' Class : 1
##
```

```
logi.test.accuracy <- logi.cm.test$byClass[['Balanced Accuracy']][1]
logi.test.precision <- logi.cm.test$byClass[['Precision']][1]
logi.test.recall <- logi.cm.test$byClass[['Recall']][1]
logi.test.f1score <- logi.cm.test$byClass[['F1']][1]
```

Patricia Lázaro Tello
plazarotello@uoc.edu

Dataset	*Accuracy*	*Precision*	*Recall*	F_1 *score*
Entrenamiento	67%	70%	83%	0,76
Test	70%	72%	88%	0,79

El modelo de regresión logística tiene la calidad deseada. Los valores de precisión son adecuados y el *recall* es correcto, incluso alto, provocando que el F_1 *score* sea también bastante alto. Se obtiene un buen balance entre precisión y *recall*.

9 Conclusiones

A continuación se muestran para los modelos seleccionados una comparativa de los valores de las métricas de calidad sobre el conjunto de datos de test:

Modelo	*Precision*	*Recall*	F_1 *score*
k-means (dist. euclídea) $k = 11$	63%	63%	0,63
k-means (dist. Manhattan) $k = 14$	64%	66%	0,65
DBSCAN+OPTICS $\epsilon = 0.075$	60%	80%	0,69
Árbol de decisión básico	67%	88%	0,76
Árbol de decisión con *AdaBoost*	65%	**95%**	0,77
Árbol de decisión RPART	70%	91%	**0,79**
k-NN $k = 10$	65%	75%	0,69
k-NN $k = 20$	63%	85%	0,72
Regresión logística	**72%**	88%	**0,79**

Se observa que los modelos de *clustering* o agrupamiento (k-means y DBSCAN+OPTICS) son los que producen peor precisión y *recall*, y por tanto, peor F_1 *score*. Dado que los valores se encuentran cercanos al 50%, se podría incluso hablar que un modelo aleatorio, donde se lanza una moneda al aire para decidir la clase de la muestra; o de modelos que siempre devuelven el mismo valor, sin importar las entradas.

Por otro lado, los modelos de *clustering* implican que una persona ha de validar los resultados y asignar etiquetas a los grupos formados. Aparte del esfuerzo humano que los modelos de clasificación no añaden dada su naturaleza, en problemas con datos de alta dimensionalidad como el tratado en este trabajo, la representación de los datos contribuye a hacer todo el proceso más confuso.

Respecto a los modelos supervisados de clasificación, solo el árbol de decisión RPART y la regresión logística consiguen resultados adecuados. En general los valores de *recall* son correctos e incluso altos, pero los modelos sufren en comparación de una baja precisión.

Si se considera una precisión adecuada aquella igual o mayor al 70% (una precisión del 50% o algo superior se obtendría con un modelo que devuelve siempre la misma salida), solo RPART y la regresión logística serían modelos adecuados.

En este caso, se elegiría la regresión logística sobre el árbol de decisión RPART, ya

que consigue algo más de precisión a cambio de sacrificar unos puntos de *recall*, pero como el valor de *recall* ya es suficientemente alto, se otorga mayor importancia a la precisión.

El algoritmo k-NN produce los peores resultados, con precisiones parecidas a las obtenidas mediante *clustering*, aunque sus valores de *recall* son superiores. El árbol de decisión básico y con *AdaBoost* tampoco obtienen precisiones del 70% o más.

9.1 Riesgos de modelo

Como se comentaba en la sección de limitaciones del *dataset*, los datos originales incluían más información que los utilizados para crear los modelos; dicha información se ha descartado por cuestiones de tiempo y por no estar dentro del alcance de la asignatura, pero podría haber otorgado atributos que permitieran discriminar mejor las 2 clases objetivo: éxito o fracaso.

Predecir si un proyecto en Kickstarter tendrá éxito o no es un problema muy complejo que depende de muchas variables, unas que se pueden obtener de los datos originales pero otras que tienen un origen sociológico, como la popularidad de la plataforma o la popularidad y *marketing* del proyecto en redes sociales, que están fuera del alcance de la asignatura.

Por último, cabe destacar que el volumen de datos es tal que no se pueden utilizar todos los datos para entrenar el modelo, y eso también hace sufrir a los modelos, dado que cuantas más observaciones se tienen, más podrá aprender el modelo y menos probabilidades de *overfitting* habrá.

Respecto al modelo elegido, la **regresión logística**, cabe destacar que asume ciertos supuestos como la linearidad entre la variable dependiente (variable respuesta) y las variables independientes (los descriptores). Además, requiere preparar concienzudamente los datos antes de poder utilizarlos en el modelo, y se han de procesar los valores nulos o perdidos porque el algoritmo no los admite.

La **regresión logística** utiliza un modelo linear y, por tanto, crea fronteras de decisión lineales, que pueden no ser la mejor opción dependiendo del problema. En este caso, es posible que esto sea problemático, dado que los problemas del mundo real no suelen ser lineales.

Bibliografía

Baruah, Indraneel Dutta. 2020. «K-means, DBSCAN, GMM, Agglomerative clustering - Mastering the popular models in a segmentation problem». *Towards Data Science*, noviembre. https://towardsdatascience.com/k-means-dbscan-gmm-agglomerative-clustering-mastering-the-popular-models-in-a-segmentation-c891a3818e29.

Berhane, Fisseha. 2021. «Data distributions where Kmeans clustering fails». *Data Science Enthusiast*. https://datascience-enthusiast.com/Python/DBSCAN_Kmeans.html.

Fernandez Casal, Ruben, Julian Costa Bouzas, y Manuel Oviedo de la Fuente. 2021. «Arboles de decision». En *Aprendizaje estadistico*. unknown. https://rubenfcasal.github.io/aprendizaje_estadistico/trees.html.

Girones Roig, Jordi. 2021. *Modelos no supervisados*. Universitat Oberta de Catalunya. http://cvapp.uoc.edu/autors/MostraPDFMaterialAction.do?id=284572.

Guillen Estany, Montserrat, y Maria Teresa Alonso Alonso. 2021. *Modelos de regresion logistica*. Universitat Oberta de Catalunya. https://materials.campus.uoc.edu/daisy/Materials/PID_00276229/pdf/PID_00276229.pdf.

Hashemifar, Soroush. 2018. «K-KMeans vs. DBScan». *Medium*, abril. https://soroushhashemifar.medium.com/kmeans-vs-dbscan-d9d5f9dbee8b.

Montoliu Colas, Raul. 2021a. *Evaluacion de modelos*. Universitat Oberta de Catalunya. http://cvapp.uoc.edu/autors/MostraPDFMaterialAction.do?id=284573.

———. 2021b. *Modelos supervisados*. Universitat Oberta de Catalunya. http://cvapp.uoc.edu/autors/MostraPDFMaterialAction.do?id=284578.

Mysiak, Kamil. 2020. «Explaining DBSCAN Clustering». *Towards Data Science*, julio. https://towardsdatascience.com/explaining-dbscan-clustering-18eaf5c83b31.

Navlani, Avinash. 2019. «Neural Network Models in R». *Datacamp Tutorials*, diciembre. https://www.datacamp.com/community/tutorials/neural-network-models-r.

Sharma, Ekta. 2020. «K-Means vs. DBSCAN Clustering - For Beginners». *Towards Data Science*, mayo. https://towardsdatascience.com/k-means-vs-dbscan-clustering-49f8e627de27.

Villalba, Fernando. 2018a. «Arboles de decision». En *Aprendizaje supervisado en R*. unknown. https://fervilber.github.io/Aprendizaje-supervisado-en-R/arboles.html.

———. 2018b. «Regresion logistica binaria». En *Aprendizaje supervisado en R*. unknown. https://fervilber.github.io/Aprendizaje-supervisado-en-R/glm.html.