



UNIVERSIDAD DE GRANADA

PRÁCTICA 3: CLASIFICACIÓN

Autor

Juan Manuel Castillo Nievas



MÁSTER PROFESIONAL EN INGENIERÍA INFORMÁTICA

Granada, 17 de diciembre de 2020

Índice

1. Introducción	2
2. Preprocesamiento de datos	2
2.1. Cambiar todos los precios a la misma moneda	2
2.2. Creación de la variable Ganancia	4
2.3. Variable Category a variable numérica	4
2.4. Variable endDay a variable numérica	4
2.5. Discretización de la variable endDay	5
2.6. Discretización de la variable Duration	6
2.7. Discretización de la variable OpenPriceUS	8
2.8. Selección de características	9
3. Clasificación	11
3.1. Árboles de decisión	11
3.1.1. Rpart partición 1	11
3.1.2. Rpart partición 2	16
3.1.3. Rpart partición 3	19
3.1.4. Conclusiones	23
3.2. Clasificación k-NN	25
3.2.1. kNN con k=3	26
3.2.2. kNN con k=5	28
3.2.3. kNN con k=7	30
3.2.4. Conclusiones	32
3.3. Clasificación Naïve-Bayes	34
3.3.1. Naïve Bayes con la partición 1	35
3.3.2. Naïve Bayes con la partición 2	37
3.3.3. Naïve Bayes con la partición 3	39
3.3.4. Conclusiones	41
4. Conclusiones	43

1. Introducción

En esta práctica se nos ofrece un dataset que contiene datos correspondientes a subastas a través de Ebay.com. Se incluyen variables que describen el objeto (categoría), el vendedor (mediante su valoración o rating) y los términos de la subasta fijados por el vendedor (duración de la subasta, precio de inicio, moneda y día de la semana en el que finaliza la subasta).

Se pretende predecir si la subasta será o no competitiva para de esa forma diseñar diferentes estrategias de negocio para que los vendedores aumenten la tasa de subastas competitivas en sus productos.

2. Preprocesamiento de datos

Antes de la clasificación, es conveniente hacer un preprocesamiento de los datos.

2.1. Cambiar todos los precios a la misma moneda

La variable **Currency** contiene el tipo de moneda que se usa en las variables **ClosePrice** y **OpenPrice**. Hay 3 posibles valores: **US**, **EUR** y **GBP**. Sería conveniente pasar todos los precios a la misma moneda.

En este caso, ya que la moneda mayoritaria es **US**, se van a convertir los precios que actualmente están en EUR y GBP a US. Para ello se ha usado un editor de cálculo y se ha creado una nueva variable (columna) que se llama **ClosePriceUS** y tomará como valor (ver Figura 1):

- Si la moneda es **US**, se deja el mismo precio que **ClosePrice**
- Si la moneda es **EUR**, se multiplica el precio de **ClosePrice** por **1.22** (ver Figura 3)
- Si la moneda es **GBP**, se multiplica el precio de **ClosePrice** por **1.34** (ver Figura 1)

Este mismo proceso se ha hecho para la variable **OpenPrice**, para la cual se ha creado la variable **OpenPriceUS**.

=SI(B2="US";F2;SI(B2="EUR";F2*1,22;F2*1,34))						
A	B	C	D	E	F	G
Category	currency	sellerRating	Duration	endDay	ClosePrice	ClosePriceUS
Music/Movie/Game	US	3249	5	Mon	0,01	0,01
Music/Movie/Game	US	3249	5	Mon	0,01	0,01

Figura 1: Fórmula para pasar todos los precios a la misma moneda



Figura 2: Valor actual de 1EUR en US

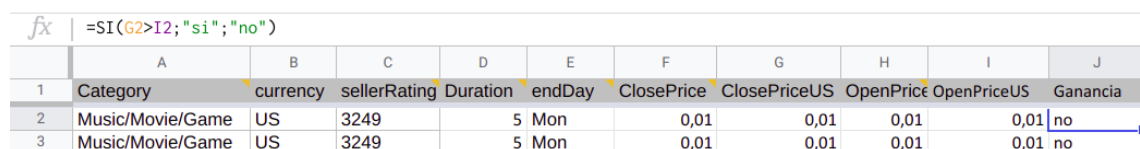


Figura 3: Valor actual de 1GBP en US

2.2. Creación de la variable Ganancia

Todas las subastas tienen un precio de inicio y un precio de cierre y ahora mismo todas tienen el precio en la misma moneda. Aún así, se puede discretizar esta variable creando una nueva variable que se llame **Ganancia** y que indique si la subasta se ha vendido al mismo precio que cuando se abrió o no. De esta forma, esta variable tendrá dos posibles valores (ver Figura 4):

- **Si:** el valor es “si” si el producto se ha vendido a un **precio mayor que su precio inicial**.
- **No:** el valor es “no” si el producto se ha vendido al **mismo precio que su precio inicial**.



	A	B	C	D	E	F	G	H	I	J
1	Category	currency	sellerRating	Duration	endDay	ClosePrice	ClosePriceUS	OpenPrice	OpenPriceUS	Ganancia
2	Music/Movie/Game	US	3249	5	Mon	0,01	0,01	0,01	0,01	no
3	Music/Movie/Game	US	3249	5	Mon	0,01	0,01	0,01	0,01	no

Figura 4: Creación de la variable **Ganancia** con la fórmula usada en la hoja de cálculo

2.3. Variable Category a variable numérica

La variable **Category** contiene la categoría del producto. Hay 14 categorías en total, con lo cual se puede discretizar esta variable y convertirla a numérica (primero debe convertirse a factor). Para ello, se ha usado el código que se muestra en la Listing 1.

```
1 # Convierto la primera columna en numérica
2 ebayActions$Category<-as.factor(ebayActions$Category)
3 ebayActions$Category<-as.numeric(ebayActions$Category)
```

Listing 1: Convertir la variable **Category** en variable numérica

2.4. Variable endDay a variable numérica

La variable **endDay** contiene el día de la semana en el que finaliza una subasta. Contiene las tres iniciales del día de la semana (Mon, Tue, Wed...). He decidido asignar a cada día de la semana un número, asignando el número 1 al lunes (Mon) y el número 7 al domingo (Sun). El código usado

```

1  # Convierto la cuarta columna en numérica
2  # El lunes es 1 (Mon) y el domingo es 7 (Sun)
3  ebayActions$endDay<-sapply(as.character(ebayActions$endDay), switch, "Mon" = 1,
4                             "Tue" = 2, "Wed" = 3, "Thu" = 4, "Fri" = 5, "Sat" = 6,
5                             "Sun" = 7, USE.NAMES = F)

```

Listing 2: Convertir la variable **endDay** en variable numérica

2.5. Discretización de la variable **endDay**

Ya se ha comentado que esta variable contiene el día de la semana en el que finaliza una subasta y se ha convertido en variable numérica. Si observamos la frecuencia de sus datos (ver Figura 5), se pueden agrupar los datos en 3 rangos:

- **Alto:** si la frecuencia es mayor de 500, significa que hay muchas subastas que finalizan ese día (lunes)
- **Normal:** si la frecuencia está entre [200, 500) (jueves, viernes, sábado y domingo).
- **Bajo:** si la frecuencia es menor a 200 (martes y miércoles).

Evidentemente, después de esta discretización se ha convertido el valor en numérico, asignando un **1 a Alto**, **2 a Normal**, y **3 a Bajo**. El código de esta discretización se encuentra en la Listing 3.

```

1  # Discretización de la variable endDay
2  barplot(table(ebayActions$endDay))
3  ebayActions$endDay <- cut(ebayActions$endDay,
4                           breaks = c(-Inf, 2, 4, +Inf),
5                           labels = c("Alto", "Bajo", "Normal"),
6                           right = FALSE)
7  ebayActions$endDay<-sapply(as.character(ebayActions$endDay), switch,
8                             "Alto" = 1, "Normal" = 2, "Bajo" = 3, USE.NAMES = F)

```

Listing 3: Discretización de **endDay**

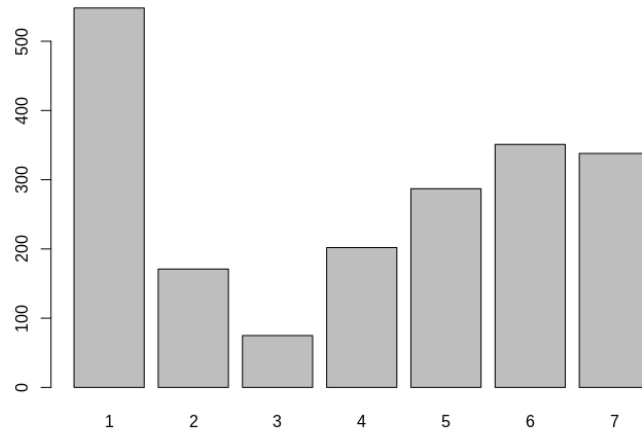


Figura 5: Frecuencias de **endDay**

2.6. Discretización de la variable Duration

Esta variable describe la duración de la subasta. De manera similar a la variable **endDay**, si observamos la frecuencia de sus datos (ver Figura 6) se pueden agrupar sus valores en 3 rangos:

- **Alta:** agrupa al número 7 (7 días), que tiene una frecuencia de más de 900.
- **Normal:** agrupa a los números 3, 5 y 10 días, que tienen una frecuencia similar entre 200 y 500.
- **Baja:** agrupa al número 1 (1 día), que tiene una frecuencia de menos de 50.

Evidentemente, después de esta discretización se ha convertido el valor en numérico, asignando un **1 a Alta**, **2 a Normal**, y **3 a Baja**. El código de esta discretización se encuentra en la Listing 4.

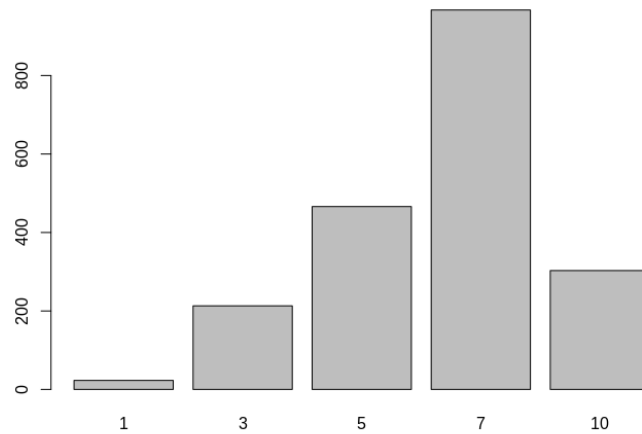


Figura 6: Frecuencias de **Duration**

```

1  # Discretización de la variable Duration
2  barplot(table(ebayActions$Duration))
3  ebayActions$Duration <- cut(ebayActions$Duration,
4                             breaks = c(-Inf, 2, 6, 8, +Inf),
5                             labels = c("Baja", "Normal", "Alta", "Normal"),
6                             right = FALSE)
7  ebayActions$Duration<-sapply(as.character(ebayActions$Duration), switch,
8                             "Alta" = 1, "Normal" = 2, "Baja" = 3, USE.NAMES = F)

```

Listing 4: Discretización de **Duration**

2.7. Discretización de la variable `OpenPriceUS`

Esta variable indica el precio de inicio que el vendedor pone en su subasta. Para discretizar esta variable, se ha optado por dividir el rango usando una igualdad de frecuencia. Para ello se ha usado la librería **arules**. En la Figura 7 se puede ver que se han seleccionado 5 rangos y se muestran los valores de cada uno de ellos.

Una vez se tienen los rangos, se ha procedido a dividir la variable en esos rangos y posteriormente se ha convertido en variable numérica. El código asociado a estas operaciones se puede ver en la Listing 5.

```
> table(discretize(ebayActions$OpenPriceUS, breaks = 5))  
  
[0.01,1.5) [1.5,3.11) [3.11,7) [7,13.5) [13.5,999]  
          395         394         392         378         413
```

Figura 7: Rangos en los que se divide `OpenPriceUS` en igualdad de frecuencia

```
1  # Discretización de la variable OpenPriceUS  
2  library(arules)  
3  table(discretize(ebayActions$OpenPriceUS, breaks = 5))  
4  ebayActions <- discretizeDF(ebayActions, methods = list(  
5    OpenPriceUS = list(method = "frequency", breaks = 5,  
6                      labels = c("0.01-1.5", "1.5-3.11", "3.11-7", "7-13.5", "13.5-999"))  
7  ),  
8  default = list(method = "none")  
9  )  
10 ebayActions$OpenPriceUS<-as.numeric(ebayActions$OpenPriceUS)
```

Listing 5: Discretización de la variable `OpenPriceUS`

2.8. Selección de características

Originalmente hay 7 características en este dataset más 1 característica que es la variable clase que clasifica si la subasta es competitiva o no. A lo largo del preprocesamiento se han añadido 3 características más. El dataset queda con las siguientes 10 características (la variable que clasifica no cuenta):

1. **Category**
2. **Currency**
3. **sellerRating**
4. **Duration**
5. **endDay**
6. **ClosePrice**
7. **ClosePriceUS**
8. **OpenPrice**
9. **OpenPriceUS**
10. **Ganancia**

De estas características se pueden eliminar las siguientes:

- **Currency**: porque todos los precios se han pasado a la misma moneda
- **ClosePrice** y **OpenPrice**: porque se han creado dos nuevas variables (**ClosePriceUS** y **OpenPriceUS**) que contienen los precios en US.
- **ClosePriceUS**: realmente lo que nos interesa es el precio de inicio para intentar predecir un precio de inicio que desemboque en una subasta competitiva. Esta variable se sustituye por la variable **Ganancia** ya que es indiferente el precio con el que se cierra, pero sí es bueno saber si se ha vendido por un precio mayor al de inicio.
- **Ganancia**: la idea es indicar a los vendedores lo que deben hacer para obtener una subasta competitiva, y la ganancia nunca la van a saber de antemano, por lo tanto no es necesaria para saber si una subasta va a ser competitiva o no.

Esto quiere decir que finalmente se se van a considerar solamente las siguientes características:

1. **Category**
2. **sellerRating**

3. **Duration**

4. **endDay**

5. **OpenPriceUS**

3. Clasificación

En esta sección se van a clasificar los datos utilizando diferentes herramientas de clasificación.

3.1. Árboles de decisión

A través de un árbol de decisión se puede tomar una decisión final que viene determinada por las condiciones que se cumplen desde la raíz hasta alguna de sus hojas (decisión final).

3.1.1. Rpart partición 1

Se ha usado la librería **rpart** para crear un árbol de decisión. Se ha usado una partición dividiendo el conjunto en un conjunto de entrenamiento (80 %) y un conjunto test (20 %). El código usado para esto se muestra en la Listing 6.

```

1  # Genero el conjunto de entrenamiento (80%) y test(20%)
2  set.seed(1234)
3  ind=sample(2,nrow(ebay),replace=TRUE,prob=c(0.8,0.2))
4  entrenamiento=ebay[ind==1,]
5  test=ebay[ind==2,]
6  # Utilizo rpart para generar un árbol de decisión
7  # Formula = Competitive ~ . -> indica que se intenta clasificar Competitive
8  # a partir de las otras variables
9  library("rpart")
10 arbol1<-rpart(formula = Competitive ~ ., data = entrenamiento)
11 plot(arbol1) ; text(arbol1)
12 testpred=predict(arbol1,newdata=test,type="class")
13 tablepred=table(testpred,test$Competitive)
14 tablepred
15 # Genero el vector diagonal y lo sumo
16 diag=diag(tablepred)
17 bien=sum(diag)
18 bien_clasificados=(bien/nrow(test))*100
19 # Bien y mal clasificados
20 bien_clasificados
21 100-bien_clasificados
22 m1=predict(arbol1, newdata=test,type = "prob")[,2]
23 pred1=prediction(m1,test$Competitive)
24 perf1=performance(pred1,"tpr","fpr")
25 plot(perf1, main="Árbol de decisión")
26 # AUC
27 as.numeric(performance(pred1, "auc")@y.values)

```

Listing 6: Árbol de decisión con **rpart**

El **árbol de decisión** generado se muestra en la Figura 8. El **porcentaje de aciertos y de error** se muestra en la Figura 9. La **matriz de confusión** se muestra en la Figura 10.

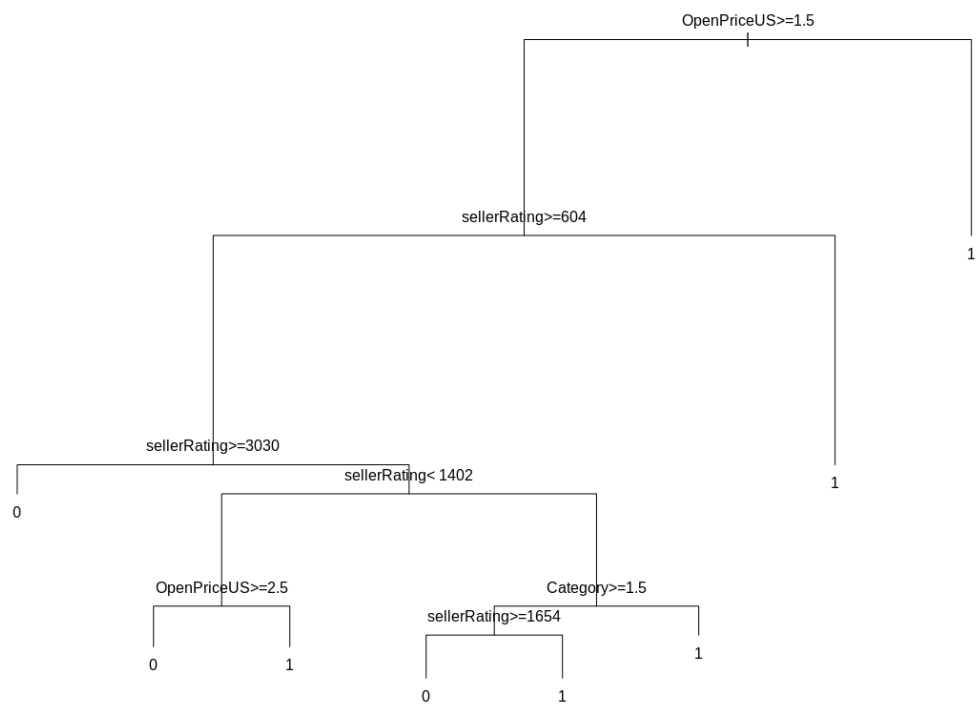


Figura 8: Árbol de decisión creado por **rpart** en la partición 1

```
> bien_clasificados
[1] 73.08707
> 100-bien_clasificados
[1] 26.91293
```

Figura 9: Porcentaje de errores y aciertos con **rpart** en la partición 1

```
> tablepred

testpred   0   1
      0 128  48
      1  54 149
```

Figura 10: Matriz de confusión con **rpart** en la partición 1

En la Figura 11 se muestra la **curva ROC**. Esta curva representa la **proporción de verdaderos positivos frente a falsos positivos** y se utiliza en sistemas de predicción binarios, como es este caso (competitivo o no). Cada resultado de la matriz de confusión se representa con un punto en el espacio ROC. El mejor método de predicción dibujaría una curva que está lo más cercana posible a la esquina superior izquierda (coordenada $[0,1]$). Cuanto mayor sea el área bajo la curva, mejor será la predicción.

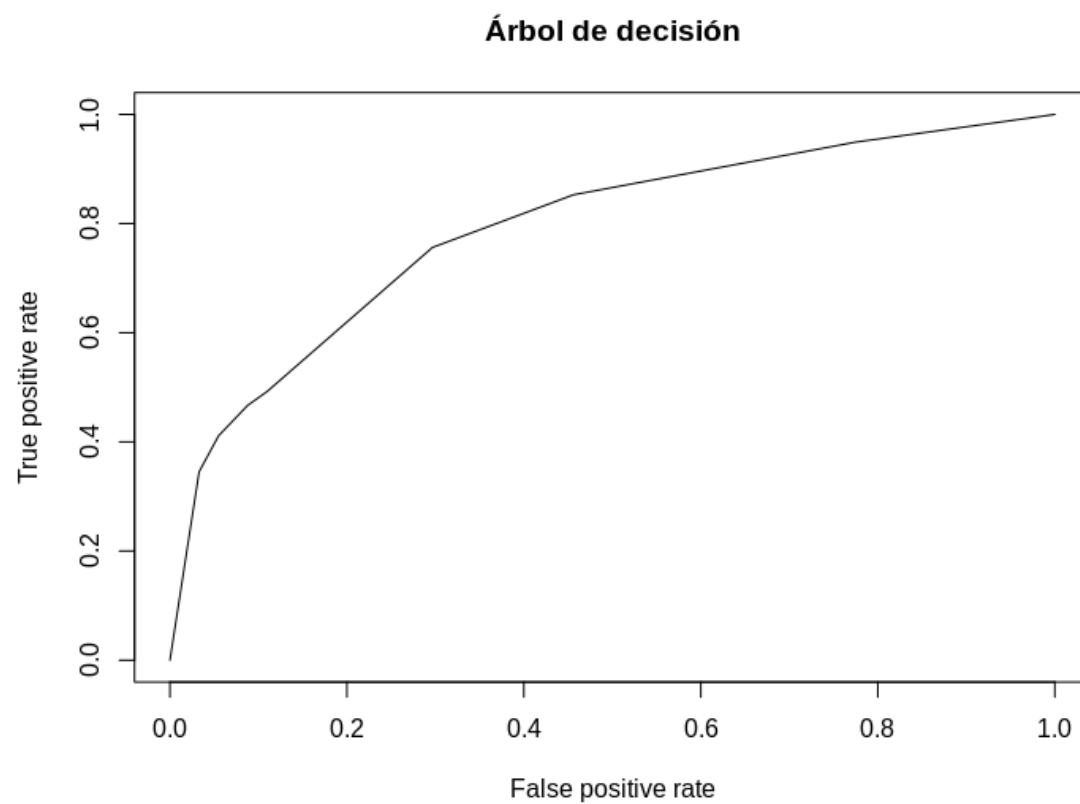


Figura 11: Curva ROC con **rpart** en la partición 1

3.1.2. Rpart partición 2

Se ha usado una partición diferente, tal y como se muestra en la Listing 7.

```
1  # Genero el conjunto de entrenamiento (80%) y test(20%)
2  set.seed(8453)
3  ind=sample(2,nrow(ebay),replace=TRUE,prob=c(0.8,0.2))
4  entrenamiento2=ebay[ind==1,]
5  test2=ebay[ind==2,]
```

Listing 7: Partición 2 con **rpart**

El **árbol de decisión** generado se muestra en la Figura 12. El **porcentaje de aciertos y de error** se muestra en la Figura 13. La **matriz de confusión** se muestra en la Figura 14. En la Figura 15 se muestra la **curva ROC**.

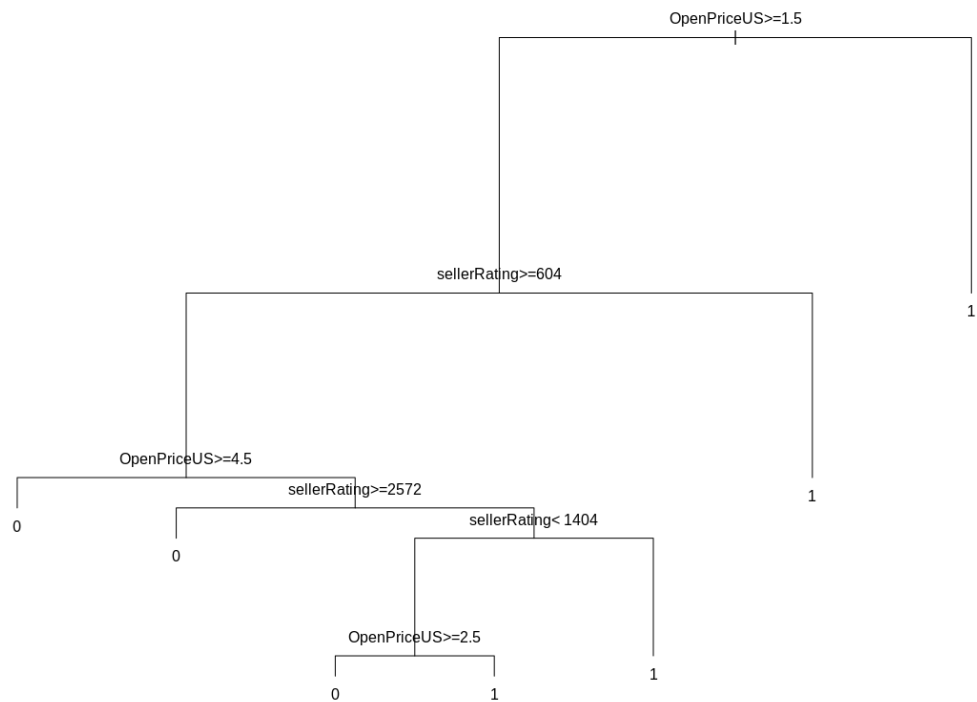


Figura 12: Árbol de decisión creado por **rpart** en la partición 2

```

> bien_clasificados
[1] 71.76781
> 100-bien_clasificados
[1] 28.23219

```

Figura 13: Porcentaje de errores y aciertos con **rpart** en la partición 2

```

> tablepred

testpred   0   1
      0 123  52
      1  49 149

```

Figura 14: Matriz de confusión con **rpart** en la partición 2

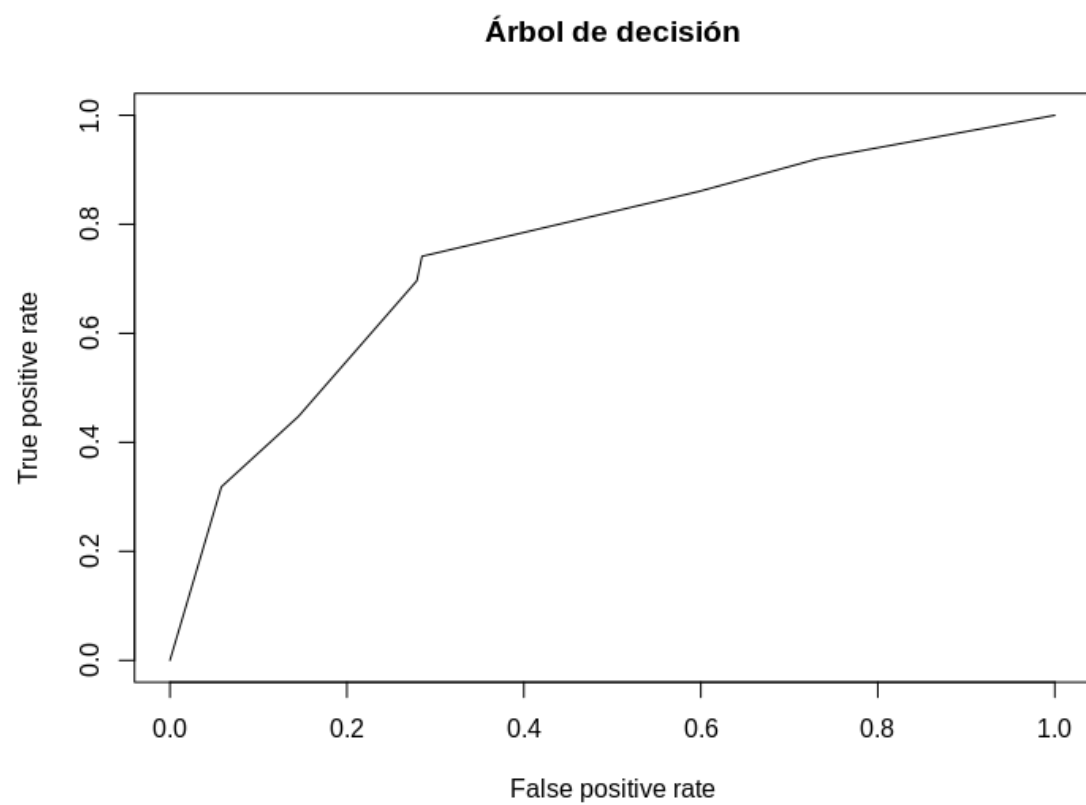


Figura 15: Curva ROC con **rpart** en la partición 2

3.1.3. Rpart partición 3

Se ha usado una partición diferente, tal y como se muestra en la Listing 8. **Esta partición es la que mejor resultado porcentaje de aciertos ha obtenido.**

```
1  # Genero el conjunto de entrenamiento (80%) y test(20%)
2  set.seed(5555)
3  ind=sample(2,nrow(ebay),replace=TRUE,prob=c(0.8,0.2))
4  entrenamiento3=ebay[ind==1,]
5  test3=ebay[ind==2,]
```

Listing 8: Partición 3 con **rpart**

El **árbol de decisión** generado se muestra en la Figura 16. El **porcentaje de aciertos y de error** se muestra en la Figura 17. La **matriz de confusión** se muestra en la Figura 18. En la Figura 19 se muestra la **curva ROC**.

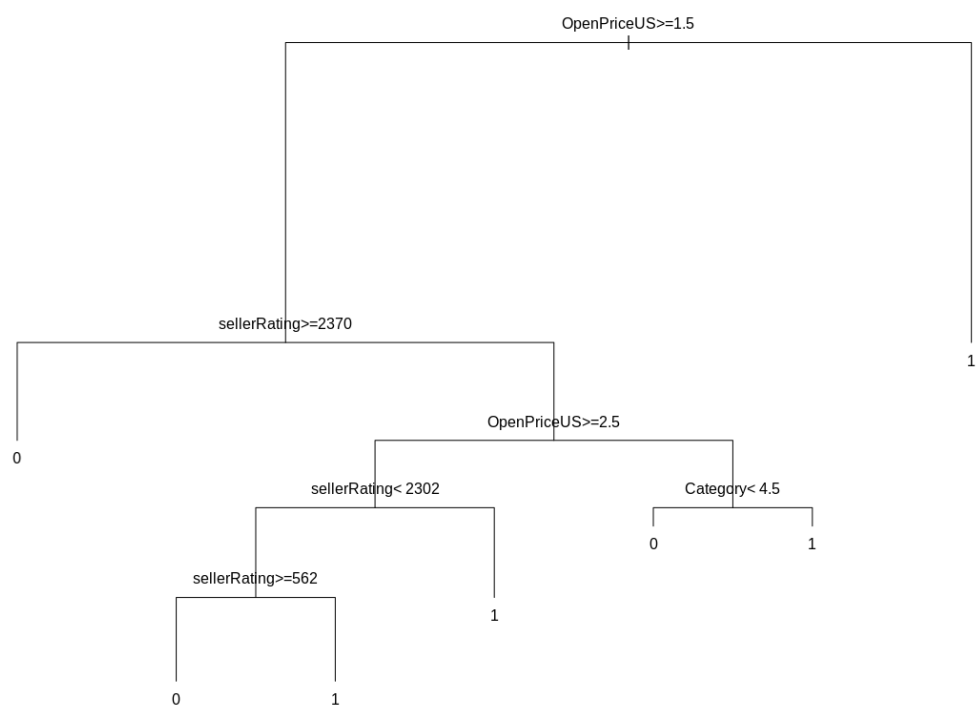


Figura 16: Árbol de decisión creado por **rpart** en la partición 3

```
> bien_clasificados
[1] 76.98413
> 100-bien_clasificados
[1] 23.01587
```

Figura 17: Porcentaje de errores y aciertos con **rpart** en la partición 3

```
> tablepred

testpred   0   1
      0 127  58
      1  29 164
```

Figura 18: Matriz de confusión con **rpart** en la partición 3

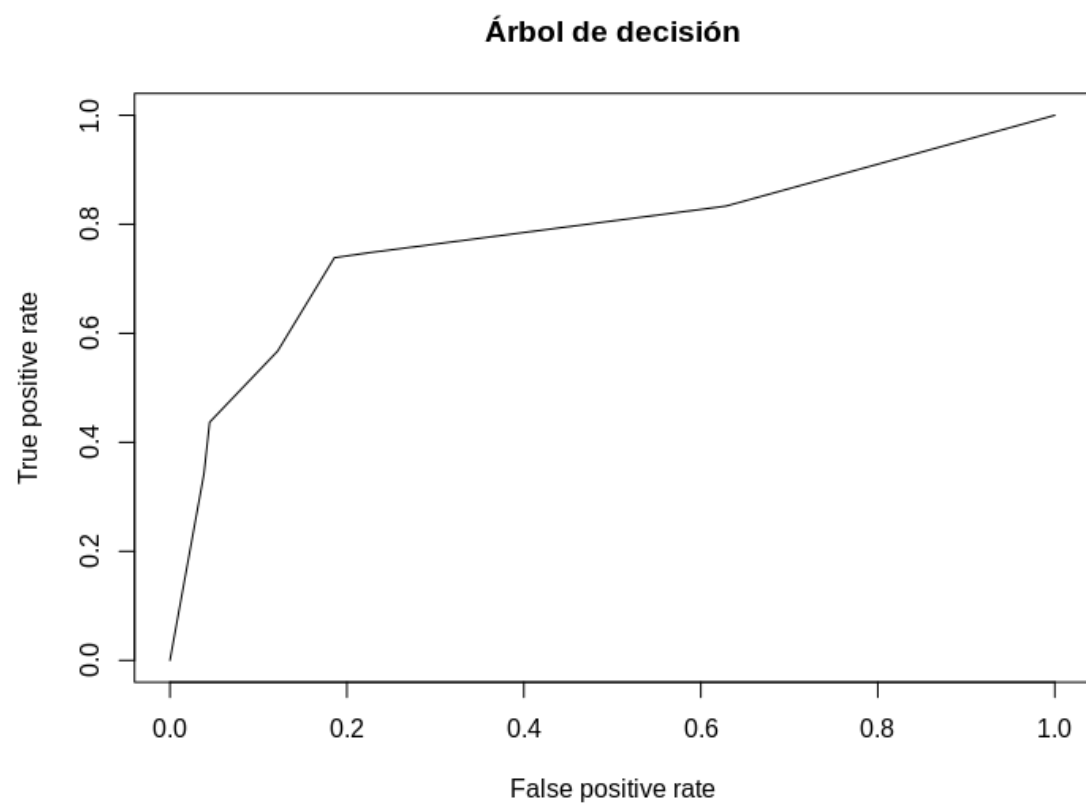


Figura 19: Curva ROC con **rpart** en la partición 1

3.1.4. Conclusiones

La partición 3 ha sido la que mejor resultado ha obtenido en el árbol de decisión con un porcentaje de aciertos del 76.98 %.

En la Figura 20 se muestran las **3 curvas ROC** de cada partición. En **negro** está la curva ROC de la partición 1, en **rojo** la partición 2 y en **azul** la partición 3. En la Figura 21 se muestran las áreas debajo de cada curva. A pesar de que la partición 1 obtiene un poco menos de aciertos, obtiene un área mayor.

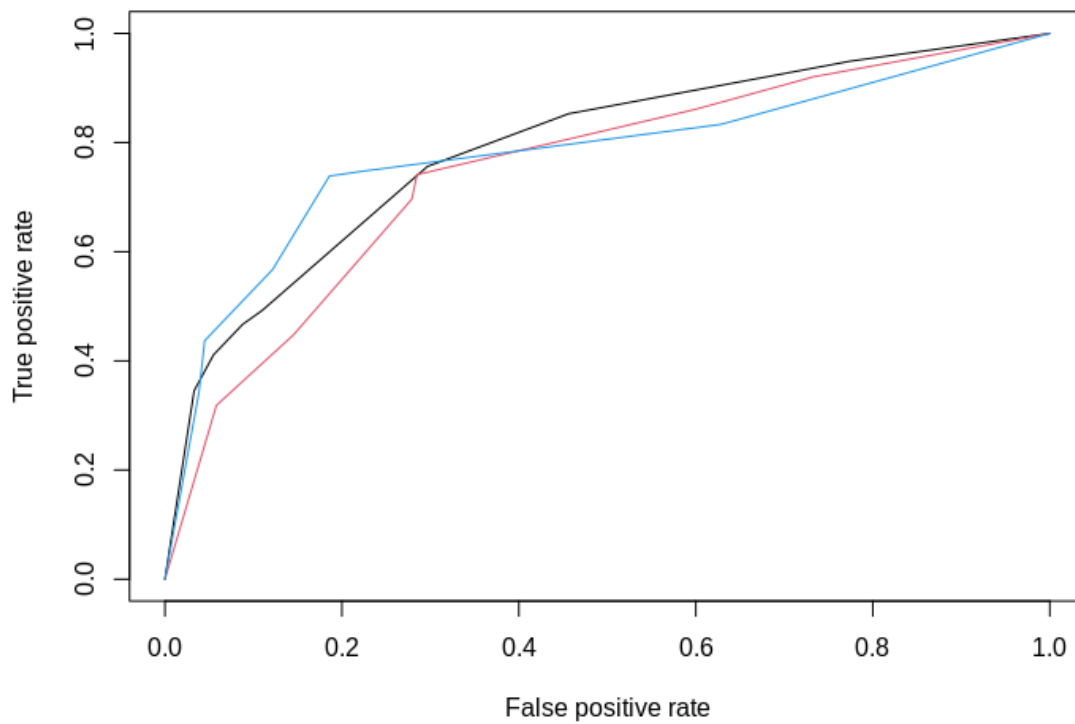


Figura 20: Curvas ROC


```
> # AUC
> as.numeric(performance(pred1, "auc")@y.values)
[1] 0.7905534
> # AUC
> as.numeric(performance(pred2, "auc")@y.values)
[1] 0.7507087
> # AUC
> as.numeric(performance(pred3, "auc")@y.values)
[1] 0.7782398
```

Figura 21: AUC de las curvas ROC

3.2. Clasificación k-NN

En esta clasificación se utiliza una distancia, que normalmente es la euclídea, para buscar a los k vecinos más cercanos y clasifica según la clase más repetida entre los vecinos. Como en la Sección 3.1 se obtuvo un mejor resultado con la **partición 3**, se va a utilizar esta partición para hacer esta clasificación con diferentes k : **3, 5 y 7**. Se hace con números impares ya que se evitan los empates en la clasificación.

El código usado para las 3 clasificaciones se puede ver en la Listing 9. Lo único que cambia es la k en la línea 1. Se hace uso de la librería **kkn**.

```
1 knn1=kkn(formula=Competitive ~ .,entrenamiento3,test3,k=3)
2 table=table(knn1$fit,test3$Competitive)
3 table
4 #Genero el vector diagonal y lo sumo
5 diag=diag(table)
6 bien=sum(diag)
7 bien_clasificados=(bien/nrow(test3))*100
8 #Bien y mal clasificados
9 bien_clasificados
10 100-bien_clasificados
11 #Calculo de la curva ROC
12 library("ROCR")
13 m=knn1$prob[,2]
14 pred=prediction(m,test3$Competitive)
15 perf=performance(pred,"tpr","fpr")
16 plot(perf, main="Knn")
17 # AUC
18 as.numeric(performance(pred, "auc")@y.values)
```

Listing 9: Clasificación con k-NN

3.2.1. kNN con k=3

En la Figura 22 se muestra el **porcentaje de errores y aciertos** obtenido. En la Figura 23 se muestra la **matriz de confusión**. En la Figura 24 se muestra la **curva ROC**.

```
> bien_clasificados  
[1] 67.54617  
> 100-bien_clasificados  
[1] 32.45383
```

Figura 22: Porcentaje de errores y aciertos con **k=3**

```
> table  
  
      0    1  
0 140  94  
1  44 116
```

Figura 23: Matriz de confusión con **k=3**

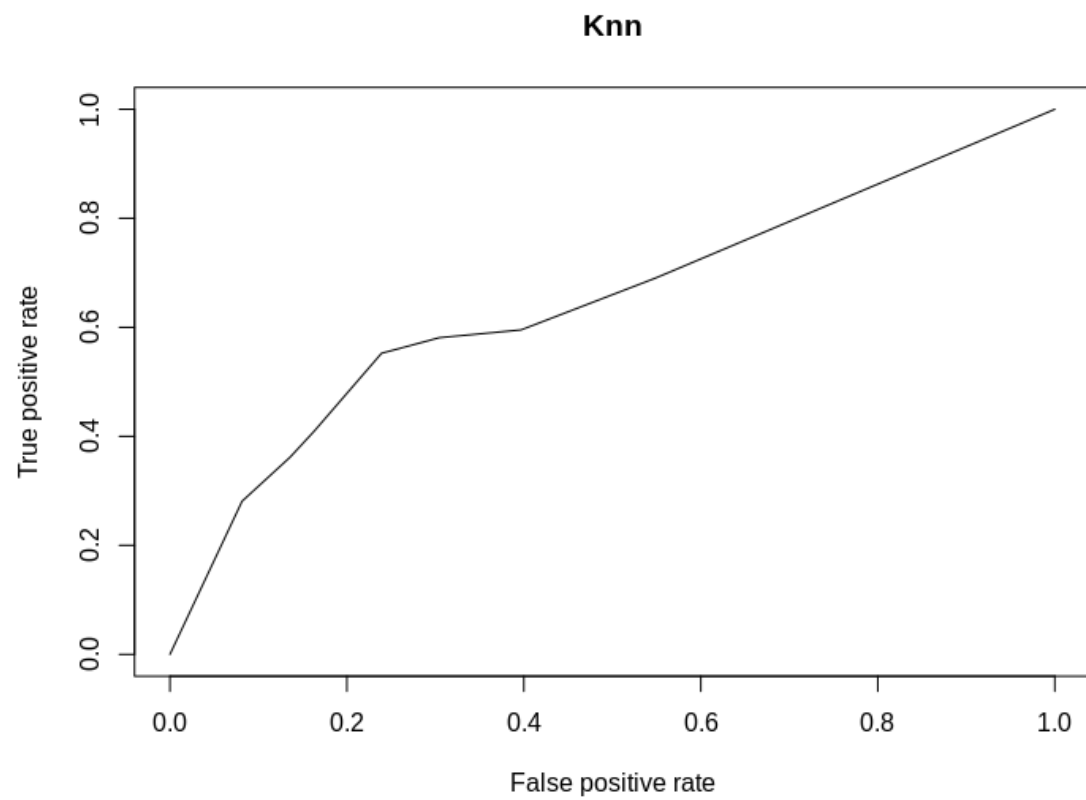


Figura 24: Curva ROC con $k=3$

3.2.2. kNN con k=5

En la Figura 25 se muestra el **porcentaje de errores y aciertos** obtenido. En la Figura 26 se muestra la **matriz de confusión**. En la Figura 28 se muestra la **curva ROC**.

```
> bien_clasificados  
[1] 61.47757  
> 100-bien_clasificados  
[1] 38.52243
```

Figura 25: Porcentaje de errores y aciertos con **k=5**

```
> table  
  
      0    1  
0 136 113  
1  48  97
```

Figura 26: Matriz de confusión con **k=5**

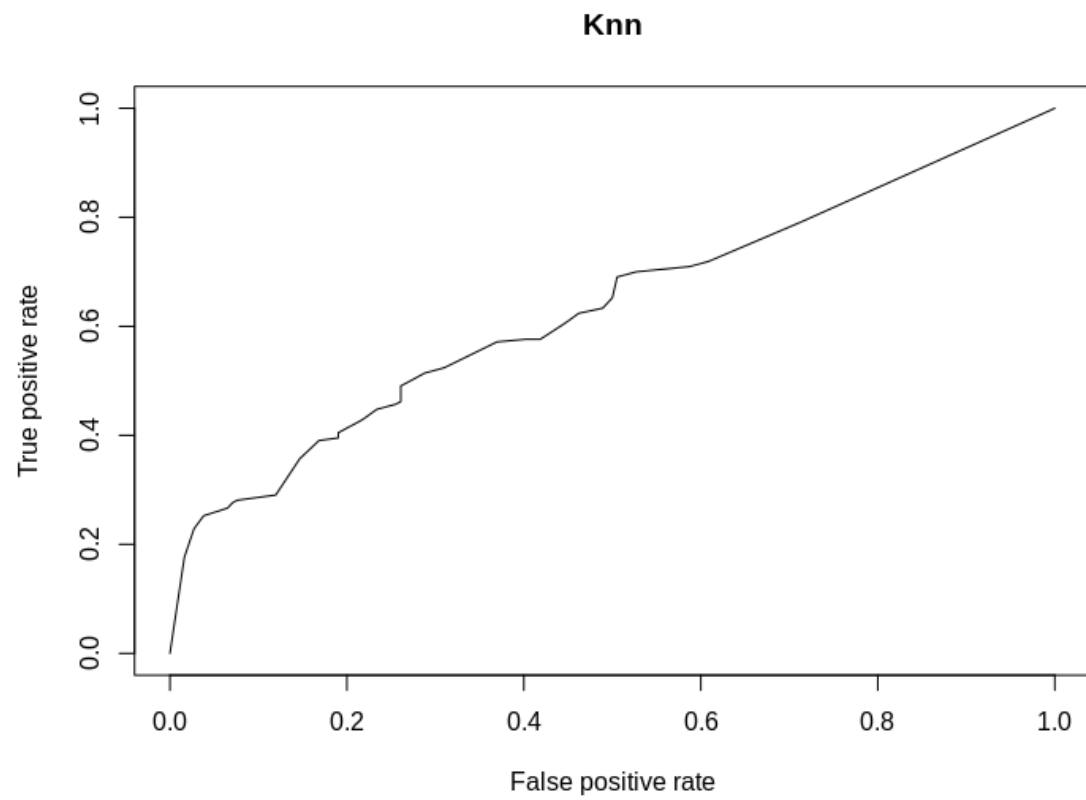


Figura 27: Curva ROC con $k=5$

3.2.3. kNN con k=7

En la Figura 28 se muestra el **porcentaje de errores y aciertos** obtenido. En la Figura 29 se muestra la **matriz de confusión**. En la Figura 30 se muestra la **curva ROC**.

```
> bien_clasificados  
[1] 64.90765  
> 100-bien_clasificados  
[1] 35.09235
```

Figura 28: Porcentaje de errores y aciertos con k=7

```
> table  
  
      0    1  
0 146 110  
1  38 100
```

Figura 29: Matriz de confusión con k=7

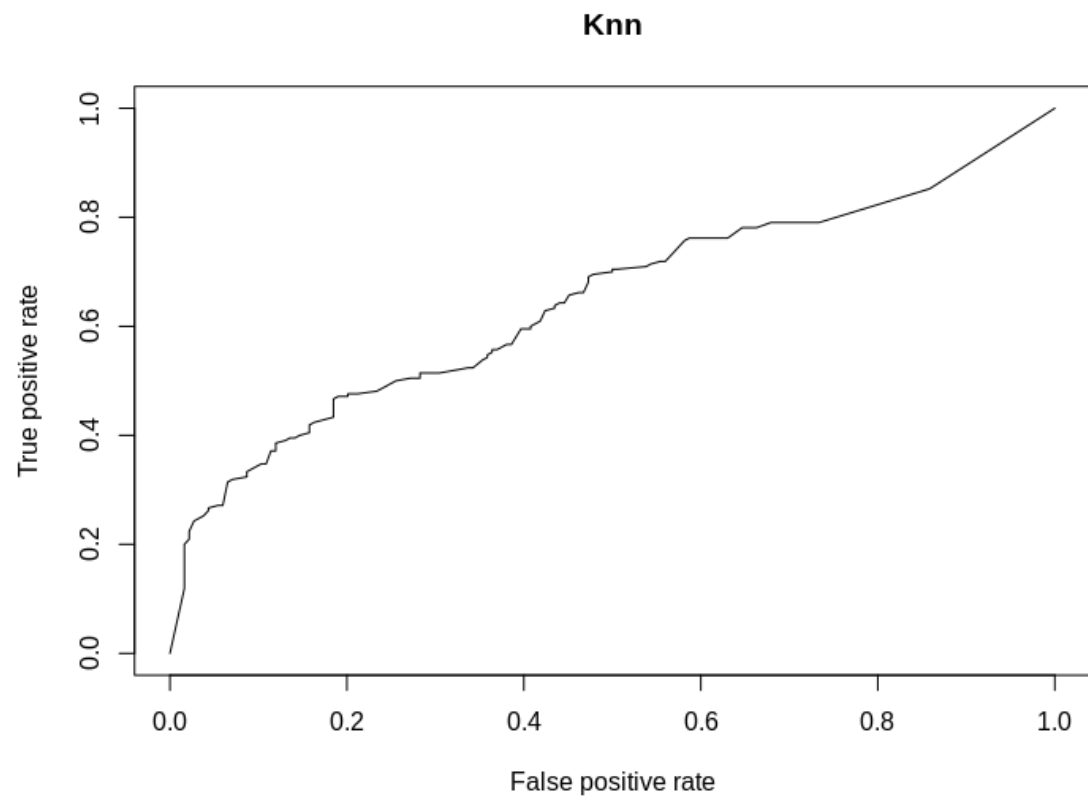


Figura 30: Curva ROC con $k=7$

3.2.4. Conclusiones

La mejor clasificación se ha obtenido con **k=3**, con un 67.54% de aciertos y un 32.45% de fallos. Aún así, este porcentaje es peor que el que se obtuvo usando árboles de decisión en la Sección anterior.

En la Figura 31 se muestra una gráfica con las **3 curvas ROC** generadas en esta Sección. Las curvas son bastante similares entre sí y, honestamente, son curvas que representan una clasificación un poco regular. En color **negro** se muestra la curva ROC en k=3, en **rojo** se muestra la curva ROC en k=5 y en **azul** se muestra la curva ROC en k=7.

En la Figura 32 se muestra el área por debajo de cada curva. Tal y como se ve, la predicción 1 (k=3) es la curva que tiene mayor área por debajo, tan solo 2 milésimas por encima de la predicción 3 (k=7), y es que realmente sus porcentajes de aciertos y fallos son parecidos.

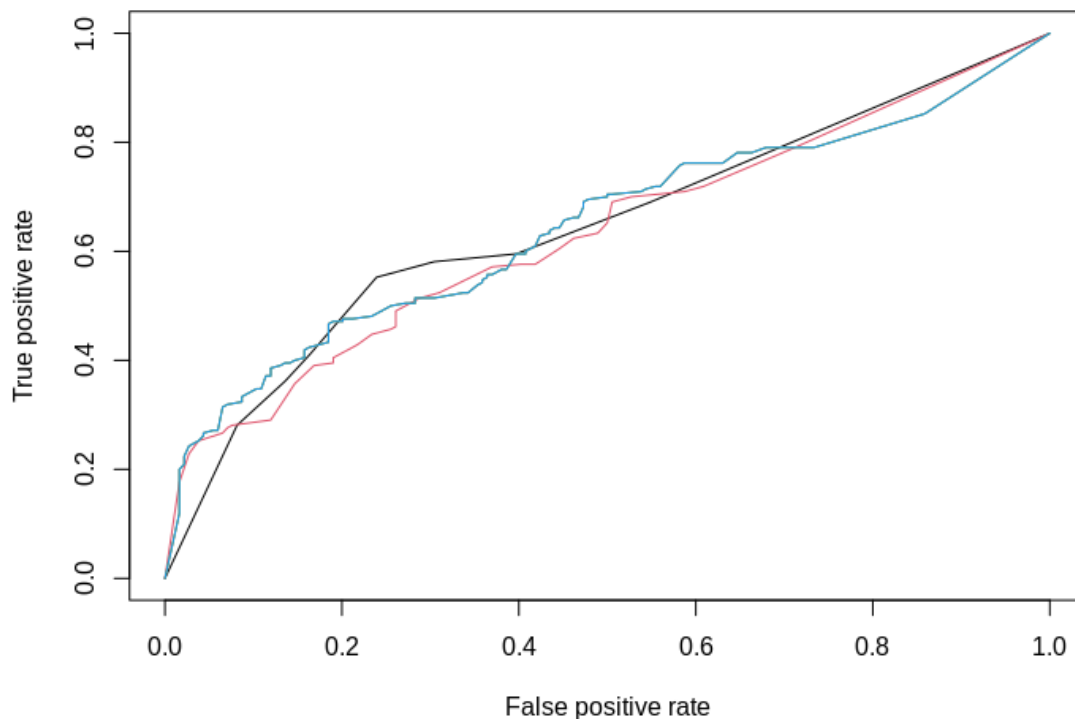


Figura 31: Comparación de curvas ROC

```
> # AUC
> as.numeric(performance(pred1, "auc")@y.values)
[1] 0.6463898
> # AUC
> as.numeric(performance(pred2, "auc")@y.values)
[1] 0.6331134
> # AUC
> as.numeric(performance(pred3, "auc")@y.values)
[1] 0.6447334
```

Figura 32: Área debajo de cada curva ROC

3.3. Clasificación Naïve-Bayes

Este tipo de clasificación permite tratar de forma directa la incertidumbre. Hasta ahora una subasta es competitiva o no es al 100 %, pero es interesante dar una probabilidad para saber la probabilidad de que una subasta sea competitiva o no.

Esta clasificación se ha hecho para cada una de las tres particiones que se hicieron en la Sección 3.1. El código usado para esta clasificación se muestra en la Listing 10. Se ha usado la librería **e1071** (se muestra el código para la partición 1, el código para las demás particiones es similar).

```
1 bayes1=naiveBayes(Competitive ~ .,entrenamiento)
2 print(bayes1)
3 predict(bayes1,test)
4 table1=table(predict(bayes1,test),test$Competitive)
5 table1
6 #Genero el vector diagonal y lo sumo
7 diag1=diag(table1)
8 bien1=sum(diag1)
9 bien_clasificados1=(bien1/nrow(test))*100
10 #Bien y mal clasificados
11 bien_clasificados1
12 100-bien_clasificados1
13 #Calculo de la curva ROC
14 m1=predict(bayes1, newdata=test,type = "raw")[,2]
15 pred1=prediction(m1,test$Competitive)
16 perf1=performance(pred1,"tpr","fpr")
17 plot(perf1, main="Naïve Bayes")
18 # AUC
19 as.numeric(performance(pred1, "auc")@y.values)
```

Listing 10: Clasificación con Naïve Bayes

3.3.1. Naïve Bayes con la partición 1

En la Figura 33 se muestra el **porcentaje de errores y aciertos**. En la Figura 34 se muestra la matriz de confusión. Por último, en la Figura 35 se muestra la **curva ROC**.

```
> bien_clasificados1  
[1] 63.32454  
> 100-bien_clasificados1  
[1] 36.67546
```

Figura 33: Porcentaje de errores y aciertos con la partición 1 con Naïve Bayes

```
> table1  
  
      0    1  
0 113  70  
1  69 127
```

Figura 34: Matriz de confusión con la partición 1 con Naïve Bayes

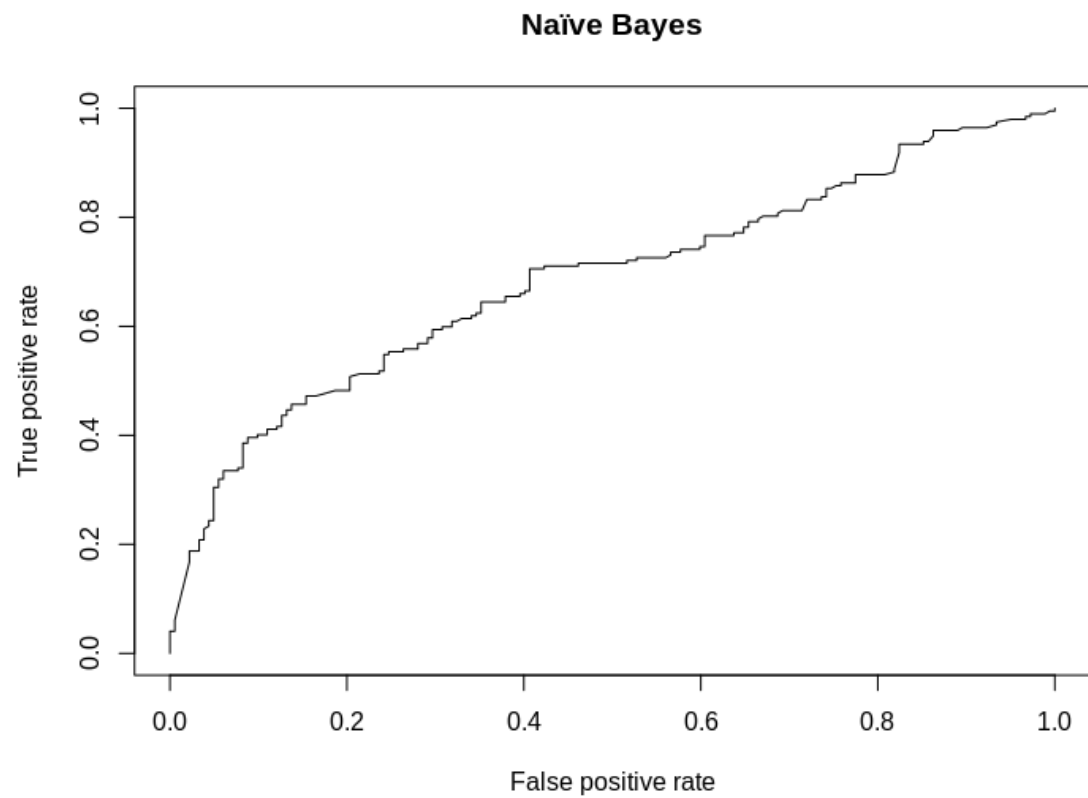


Figura 35: Curva ROC con la partición 1 con Naïve Bayes

3.3.2. Naïve Bayes con la partición 2

En la Figura 36 se muestra el **porcentaje de errores y aciertos**. En la Figura 37 se muestra la matriz de confusión. Por último, en la Figura 38 se muestra la **curva ROC**.

```
> bien_clasificados2  
[1] 61.3941  
> 100-bien_clasificados2  
[1] 38.6059
```

Figura 36: Porcentaje de errores y aciertos con la partición 2 con Naïve Bayes

```
> table2  
  
      0    1  
0 117  89  
1  55 112
```

Figura 37: Matriz de confusión con la partición 2 con Naïve Bayes

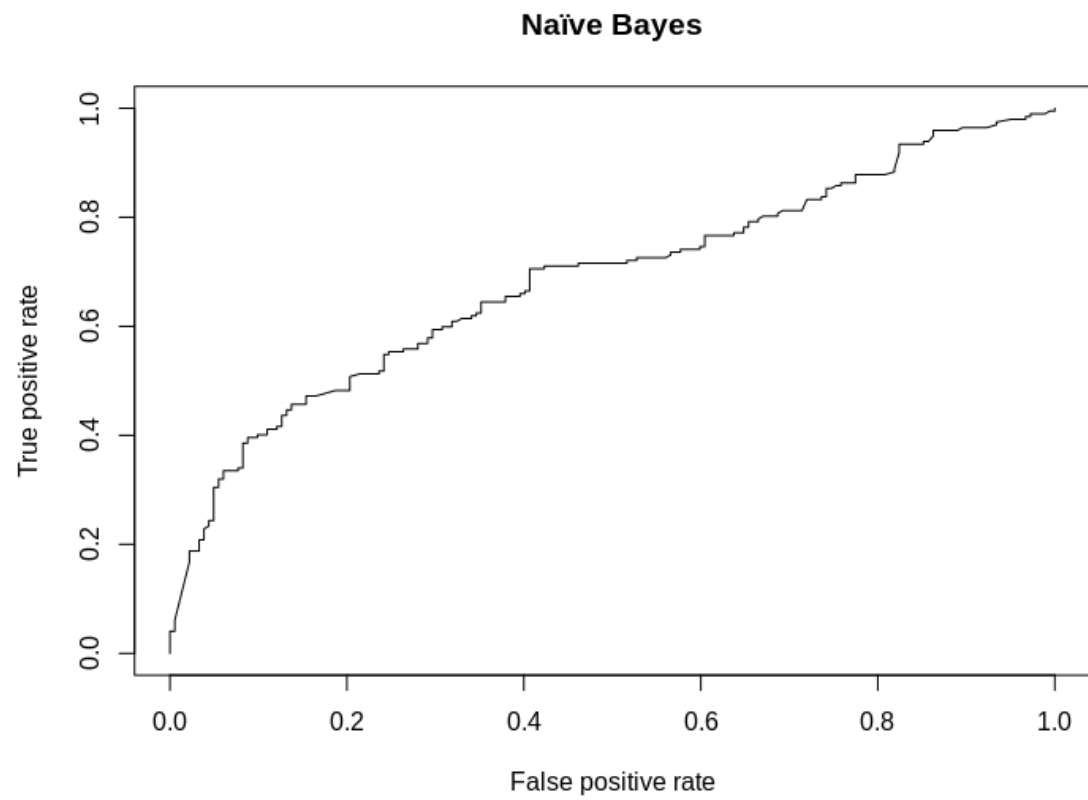


Figura 38: Curva ROC con la partición 2 con Naïve Bayes

3.3.3. Naïve Bayes con la partición 3

En la Figura 39 se muestra el **porcentaje de errores y aciertos**. En la Figura 36 se muestra la matriz de confusión. Por último, en la Figura 41 se muestra la **curva ROC**.

```
> bien_clasificados3  
[1] 64.02116  
> 100-bien_clasificados3  
[1] 35.97884
```

Figura 39: Porcentaje de errores y aciertos con la partición 3 con Naïve Bayes

```
> table3  
  
      0    1  
0 108  88  
1  48 134
```

Figura 40: Matriz de confusión con la partición 3 con Naïve Bayes

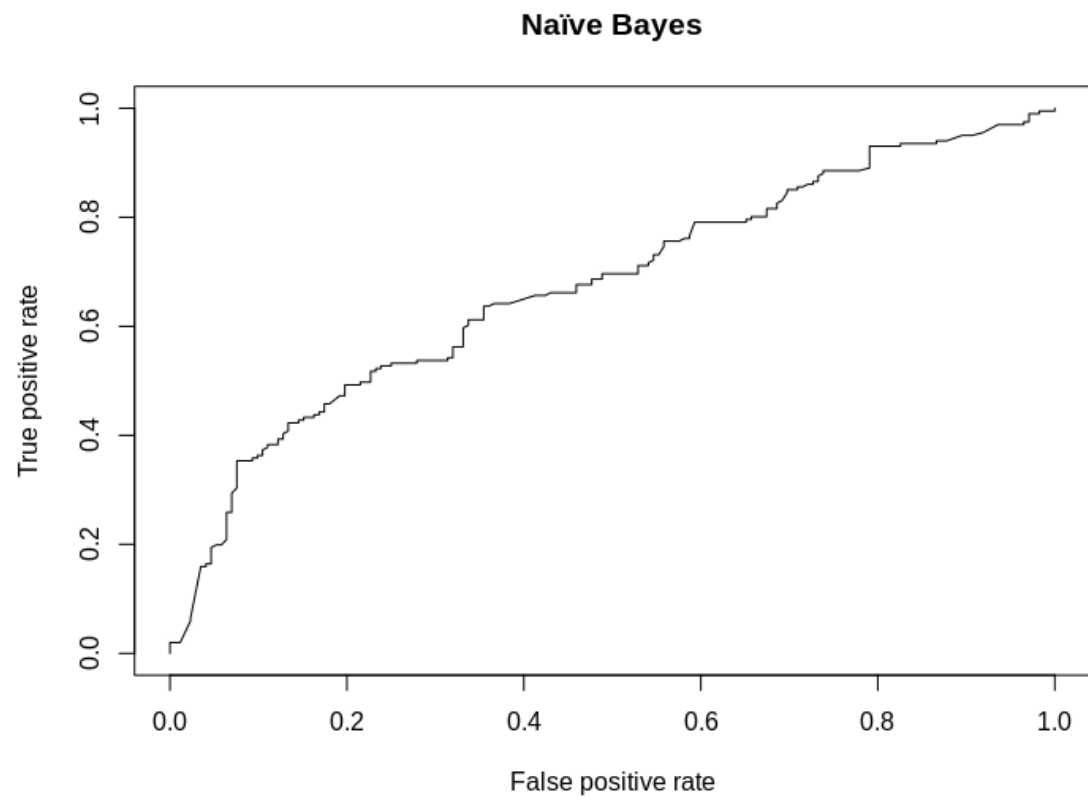


Figura 41: Curva ROC con la partición 3 con Naïve Bayes

3.3.4. Conclusiones

La mejor clasificación se ha obtenido con la **partición 3**, con un 67.02% de aciertos. Es la misma partición con la que se obtuvo un mejor resultado en el árbol de decisión.

En la Figura 42 se muestra una gráfica con las **3 curvas ROC** generadas en esta Sección. Son curvas bastante similares pero no son todo lo buenas que podríamos esperar. En color **negro** se muestra la curva ROC con la partición 1, en **rojo** se muestra la curva ROC con la partición 2 y en **azul** se muestra la curva ROC con la partición 3.

En la Figura 43 se muestra el área por debajo de cada curva. La partición 1, a pesar de no ser la que mejor resultado da, es la que obtiene mayor área debajo de la curva.

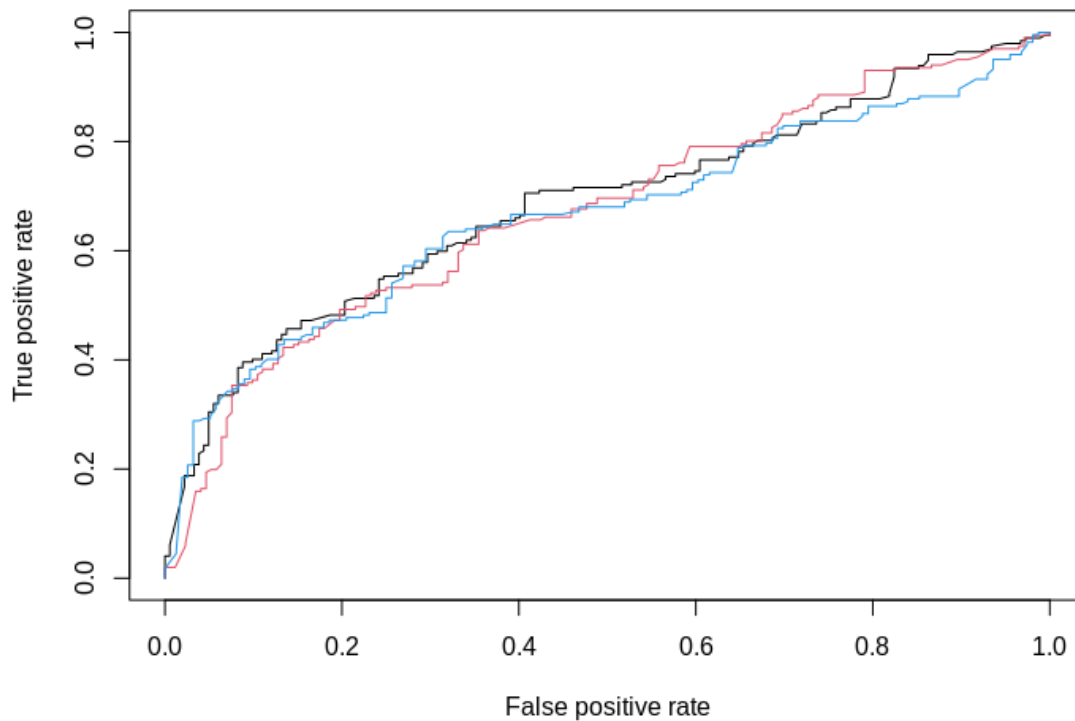


Figura 42: Comparación de curvas ROC

```
> # AUC
> as.numeric(performance(pred1, "auc")@y.values)
[1] 0.6858091
> # AUC
> as.numeric(performance(pred2, "auc")@y.values)
[1] 0.6738401
> # AUC
> as.numeric(performance(pred3, "auc")@y.values)
[1] 0.6662047
```

Figura 43: Área de cada curva ROC

4. Conclusiones

En la Tabla 1 se recogen los mejores resultados obtenidos en las técnicas de clasificación empleadas en este documento. Tal y como se puede ver, el **árbol decisión es el que mejor resultado ha dado con un 76.98 % de aciertos usando la partición 3.**

Técnica de clasificación	Porcentaje de aciertos
Árbol de decisión	76.98
k-NN	67.57
Naïve Bayes	64.02

Tabla 1: Comparación de resultados de las técnicas de clasificación

En la Tabla 2 se muestran las diferentes áreas debajo de las curvas ROC obtenidas en cada una de las técnicas usadas. **La curva ROC obtenida en el árbol de decisión con la partición 1 es la curva con mayor área.**

Técnica de clasificación	Porcentaje de aciertos
Árbol de decisión, partición 1	0.79
Árbol de decisión, partición 2	0.75
Árbol de decisión, partición 3	0.78
k-NN, k=3	0.65
k-NN, k=5	0.63
k-NN, k=7	0.64
Naïve Bayes, partición 1	0.69
Naïve Bayes, partición 2	0.67
Naïve Bayes, partición 3	0.66

Tabla 2: Comparación de resultados del área de las curvas ROC

En cuanto a las predicciones, los árboles de decisión en la Sección 3.1 (que es la técnica de clasificación que mejores resultados ha obtenido) lo que más tienen en cuenta es el precio de inicio de la subasta (**OpenPriceUS**) y el rating del vendedor (**sellerRating**). Me ha sorprendido ya que inicialmente pensaba que la variable `endDay` influiría mucho en las decisiones, pero ha resultado ser todo lo contrario.