Predicción de interacción entre péptido y el complejo mayor de histocompatibilidad tipo I

Marc Bañuls Tornero

25/10/2019

Contents


```
data<-read.delim("peptidos.csv", header = TRUE, sep = ";")
secuencia<-data$sequence
```

1. Algoritmo k-NN

El algoritmo k-NN es el que clasifica registros que no están previamente identificados mediante la comparación de sus características con otros registros previamente clasificados correctamente. Por ello, este algoritmo se llama el de clasificación por "los vecinos más cercanos" o "nearest neighbor".

Este algoritmo tiene unas fortalezas y debilidades:

Fortalezas	Debilidades
· Simple y efectivo	· No produce un modelo, limitando el conocimiento de cómo están relacionadas las características a la clase
· No hace suposiciones subyacentes sobre la distribución de los datos	· Requiere de la selección de una k apropiada
· Fase de entrenamiento rápida	· Fase de clasificación lenta · Las características nominales y datos perdidos requieren de un procesamiento adicional

2. Desarrollar dos funciones de R:

Una que implemente la codificación *one-hot encoding* de las secuencias:

```
onehot_f<-function(x){
  x<-lapply(x,as.character)
  x<-gsub("A", "10000000000000000000", x)
  x<-gsub("R", "01000000000000000000", x)
  x<-gsub("N", "00100000000000000000", x)
  x<-gsub("D", "00010000000000000000", x)
  x<-gsub("C", "00001000000000000000", x)
  x<-gsub("Q", "00000100000000000000", x)
  x<-gsub("E", "00000010000000000000", x)
  x<-gsub("G", "00000001000000000000", x)
  x<-gsub("H", "00000000100000000000", x)
  x<-gsub("I", "00000000010000000000", x)
  x<-gsub("L", "00000000001000000000", x)
  x<-gsub("K", "00000000000100000000", x)
  x<-gsub("M", "00000000000010000000", x)
  x<-gsub("F", "00000000000001000000", x)
  x<-gsub("P", "00000000000000100000", x)
  x<-gsub("S", "00000000000000010000", x)
  x<-gsub("T", "00000000000000001000", x)
  x<-gsub("W", "00000000000000000100", x)
  x<-gsub("Y", "00000000000000000010", x)
  x<-gsub("V", "00000000000000000001", x)
  x<-gsub("1", "1 ", x)
  x<-gsub("0", "0 ", x)
  peptidos<-lapply(strsplit(x, " "), as.numeric)
  peptidos<-matrix(unlist(peptidos), nrow = 15840, byrow = TRUE)
  return(peptidos)
}
```

Otra que implemente la *transformación basada en una matriz de substitución BLOSUM62* de las secuencias

```
blosum_f<-function(x){
  x<-lapply(x,as.character)
  x<-gsub("A", "0.29015 0.03104 0.02564 0.02969 0.02159 0.02564 0.04049 0.07827 0.01484 0.04318 0.05938
  x<-gsub("R", "0.04457 0.34496 0.03876 0.03101 0.00775 0.04845 0.05233 0.03295 0.02326 0.02326 0.04651
  x<-gsub("N", "0.0427 0.04494 0.31685 0.08315 0.00899 0.03371 0.04944 0.06517 0.03146 0.02247 0.03146
  x<-gsub("D", "0.04104 0.02985 0.06903 0.39739 0.00746 0.02985 0.09142 0.04664 0.01866 0.02239 0.02799
  x<-gsub("C", "0.06504 0.01626 0.01626 0.01626 0.48374 0.0122 0.01626 0.03252 0.00813 0.04472 0.06504
  x<-gsub("Q", "0.05588 0.07353 0.04412 0.04706 0.00882 0.21471 0.10294 0.04118 0.02941 0.02647 0.04706
  x<-gsub("E", "0.05525 0.04972 0.04052 0.09024 0.00737 0.06446 0.2965 0.03499 0.02578 0.0221 0.03683 0
  x<-gsub("G", "0.07827 0.02294 0.03914 0.03374 0.0108 0.01889 0.02564 0.51012 0.0135 0.01889 0.02834 0
  x<-gsub("H", "0.04198 0.0458 0.05344 0.03817 0.00763 0.03817 0.05344 0.03817 0.35496 0.0229 0.03817 0
  x<-gsub("I", "0.04713 0.01767 0.01473 0.01767 0.0162 0.01325 0.01767 0.02062 0.00884 0.27099 0.16789
  x<-gsub("L", "0.04453 0.02429 0.01417 0.01518 0.01619 0.01619 0.02024 0.02126 0.01012 0.11538 0.37551
  x<-gsub("K", "0.05699 0.10708 0.04145 0.04145 0.00864 0.05354 0.07081 0.04318 0.02073 0.02763 0.04318
  x<-gsub("M", "0.05221 0.03213 0.02008 0.02008 0.01606 0.02811 0.02811 0.02811 0.01606 0.1004 0.19679
  x<-gsub("F", "0.03383 0.01903 0.01691 0.01691 0.01057 0.01057 0.01903 0.02537 0.01691 0.06342 0.11416
  x<-gsub("P", "0.05685 0.02584 0.02326 0.03101 0.01034 0.02067 0.03618 0.03618 0.01292 0.02584 0.03618
```

```
  x<-gsub("S", "0.10995 0.04014 0.0541 0.04887 0.01745 0.03316 0.05236 0.06632 0.0192 0.02967 0.04188 0
  x<-gsub("T", "0.07298 0.0355 0.04339 0.03748 0.01775 0.02761 0.03945 0.04339 0.01381 0.05325 0.06509 0
  x<-gsub("W", "0.0303 0.02273 0.01515 0.01515 0.00758 0.01515 0.02273 0.0303 0.01515 0.0303 0.05303 0.0
  x<-gsub("Y", "0.0405 0.02804 0.02181 0.01869 0.00935 0.02181 0.02804 0.02492 0.04673 0.04361 0.06854 0
  x<-gsub("V", "0.06996 0.02195 0.01646 0.01783 0.0192 0.01646 0.02332 0.02469 0.00823 0.16461 0.13032 0
  peptidos<-lapply(strsplit(x, " "), as.numeric)
  peptidos<-matrix(unlist(peptidos), nrow = 15840, byrow = TRUE)
  return(peptidos)
}
```

3. Desarrollar un script en R que implemente un clasificador knn. El script debe realizar los siguientes apartados:

(a) Leer los datos peptidos.csv y hacer una breve descripción de ellos. Incluir en esta descripción el patrón de cada clase de péptido mediante la representación de su secuencia logo.

Leemos los datos:

```
datos<-read.delim("peptidos.csv", header = TRUE, sep = ";")
secuencia<-datos$sequence
```

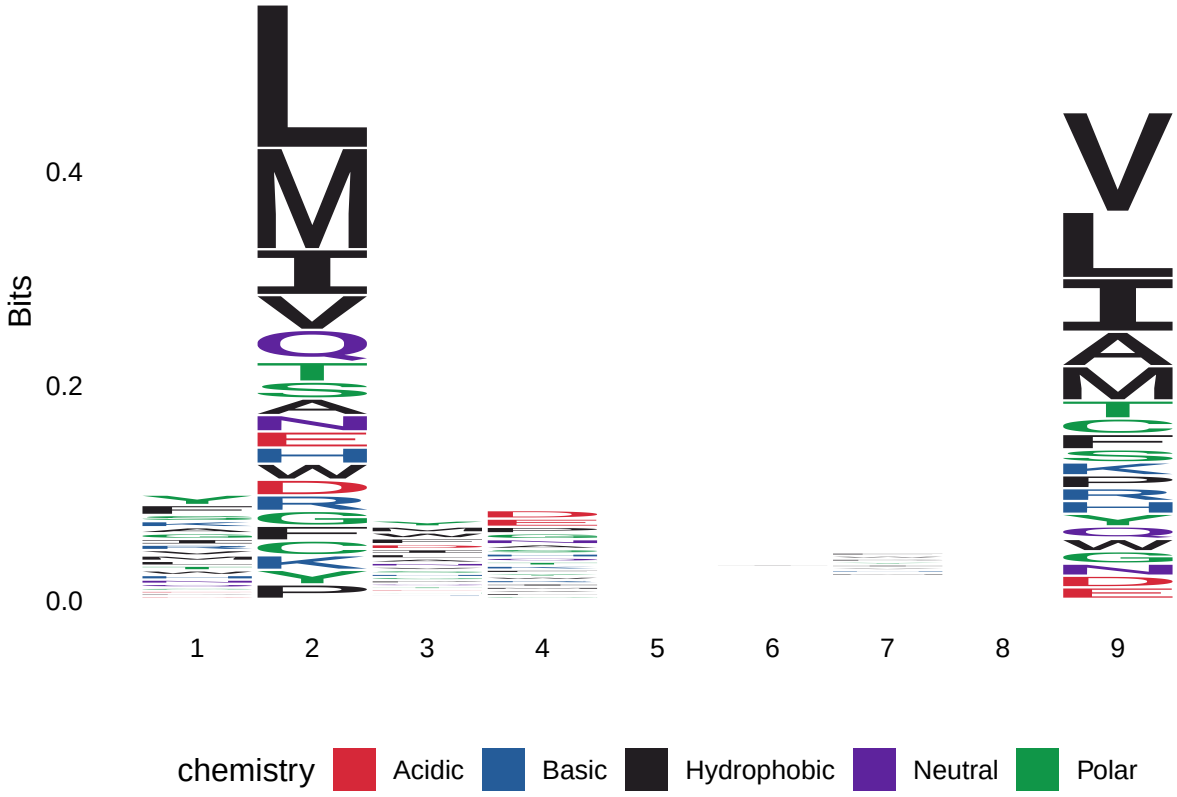
```
str(datos)
```

```
## 'data.frame':    15840 obs. of  2 variables:
##  $ sequence: Factor w/ 15840 levels "AAAPSACSV","AACNCGQTV",..: 7469 5105 3169 13842 8498 13106 13741
##  $ label   : Factor w/ 2 levels "NB","SB": 2 1 2 1 1 2 2 1 1 1 ...
```

Observamos una dos factores, uno de ellos contiene un string de 9 aminoácidos formando un 9ámero. El segundo factor indica sí hay una interacción entre el complejo MHC I y éste péptido.

Realizamos la secuencia logo de toda la secuencia de aminoácidos. Para ello cambiamos antes el formato de los datos a vector:

```
logo<-ggseqlogo(as.vector(secuencia))
print(logo)
```



En la secuencia logo observamos una elevada frecuencia de aparición de leucina, metionina e isoleucina en la posición 2 del conjunto de péptidos. También observamos en la posición 9 de los péptidos una elevada frecuencia de valina, leucina, isoleucina, alanina y metionina. Las altas frecuencias de estos aminoácidos en estas posiciones aumentan la hidrofobicidad de estos péptidos.
Aparte de estas frecuencias destacables, en las otras posiciones no se encuentran aminoácidos con una frecuencia suficientemente alta para ser significativa.

(b) Para cada forma de representar los datos: *one-hot encoding* o *transformación basada en una matriz de substitución BLOSUM62*, realizar la implementación del algoritmo knn.

One-hot encoding:

Transformar las secuencias de aminoácidos en vectores numéricos usando la función de transformación desarrollada anteriormente.

Asignamos las secuencias de aminoácidos transformadas a la variable *onehot_peptidos*:

```
onehot_peptidos<-onehot_f(secuencia)
```

Utilizando la semilla aleatoria 123, separar los datos en dos partes, una parte para training (67%) y una parte para test (33%).

Antes de realizar la obtención de datos para el training y test datasets, hemos de procesar los datos para que puedan ser leídos en el formato adecuado, y para ser leídos más fácilmente en resultados posteriores:

Asignamos la variable *etiquetas* a la columna que identifica si ha habido interacción o no con el péptido. Además, cambiamos los nombres de las etiquetas por “Si union” y “No union” respectivamente.

```
etiquetas_onehot<-factor(data$label, levels = c("NB","SB"),labels = c("No union", "Si union"))
```

Pasamos los péptidos a formato dataframe:

```
dfpeptid_onehot<-as.data.frame(onehot_peptidos)
```

Unimos las etiquetas al nuevo dataframe:

```
dfpeptidos_onehot<-cbind(etiquetas_onehot,dfpeptid_onehot)
```

Hacemos la selección de datos, 67% para train dataset y 33% para el test dataset:

```
set.seed(123)
recogida_datos_onehot<-sample(1:nrow(dfpeptidos_onehot), nrow(dfpeptidos_onehot)*0.67,replace = FALSE)
```

Creamos dos variables para guardar los datos seleccionados para el train dataset y test dataset:

```
train_onehot<-dfpeptidos_onehot[recogida_datos_onehot,2:181]
test_onehot<-dfpeptidos_onehot[-recogida_datos_onehot,2:181]
```

Guardamos las etiquetas del train dataset y test dataset:

```
train_label_onehot<-dfpeptidos_onehot[recogida_datos_onehot,1]
test_label_onehot<-dfpeptidos_onehot[-recogida_datos_onehot,1]
```

Utilizar un knn (*k*=3, 5, 7, 11) basado en el training para predecir que péptidos del test interaccionan o no con el MHCI. Además, realizar una curva ROC para cada *k*.

El K-nn nos devuelve un vector factor de etiquetas predecidas para cada uno de los ejemplos en el test dataset. Realizamos un kNN para k=3, k=5, k=7 y k=11:

```
knn3_onehot<- knn(train= train_onehot, test= test_onehot, cl= train_label_onehot, k= 3, prob= TRUE)
knn5_onehot<- knn(train= train_onehot, test= test_onehot, cl= train_label_onehot, k= 5, prob= TRUE)
knn7_onehot<- knn(train= train_onehot, test= test_onehot, cl= train_label_onehot, k= 7, prob= TRUE)
knn11_onehot<- knn(train= train_onehot, test= test_onehot, cl= train_label_onehot, k= 11, prob= TRUE)
```

Para realizar un plot con la curva ROC, reescalamos los valores de los kNN realizados anteriormente:

```
knn3_z_onehot<-attr(knn3_onehot, "prob")
knn3_Z_onehot<- 2 * ifelse(knn3_onehot =="-1", 1-knn3_z_onehot, knn3_z_onehot) -1

knn5_z_onehot<-attr(knn5_onehot, "prob")
knn5_Z_onehot<- 2 * ifelse(knn5_onehot =="-1", 1-knn5_z_onehot, knn5_z_onehot) -1

knn7_z_onehot<-attr(knn7_onehot, "prob")
knn7_Z_onehot<- 2 * ifelse(knn7_onehot =="-1", 1-knn7_z_onehot, knn7_z_onehot) -1

knn11_z_onehot<-attr(knn11_onehot, "prob")
knn11_Z_onehot<- 2 * ifelse(knn11_onehot =="-1", 1-knn11_z_onehot, knn11_z_onehot) -1
```

Convertimos estos valores reescalados y las etiquetas a formato dataframe para que estén en un formato que pueda leer el paquete ROCR:

```
knn3_z_onehot<-as.data.frame(knn3_z_onehot)
knn5_z_onehot<-as.data.frame(knn5_z_onehot)
knn7_z_onehot<-as.data.frame(knn7_z_onehot)
knn11_z_onehot<-as.data.frame(knn11_z_onehot)
test_label_z_onehot<-as.data.frame(test_label_onehot)
```

Realizamos la predicción de cada kNN para estandarizar el formato:

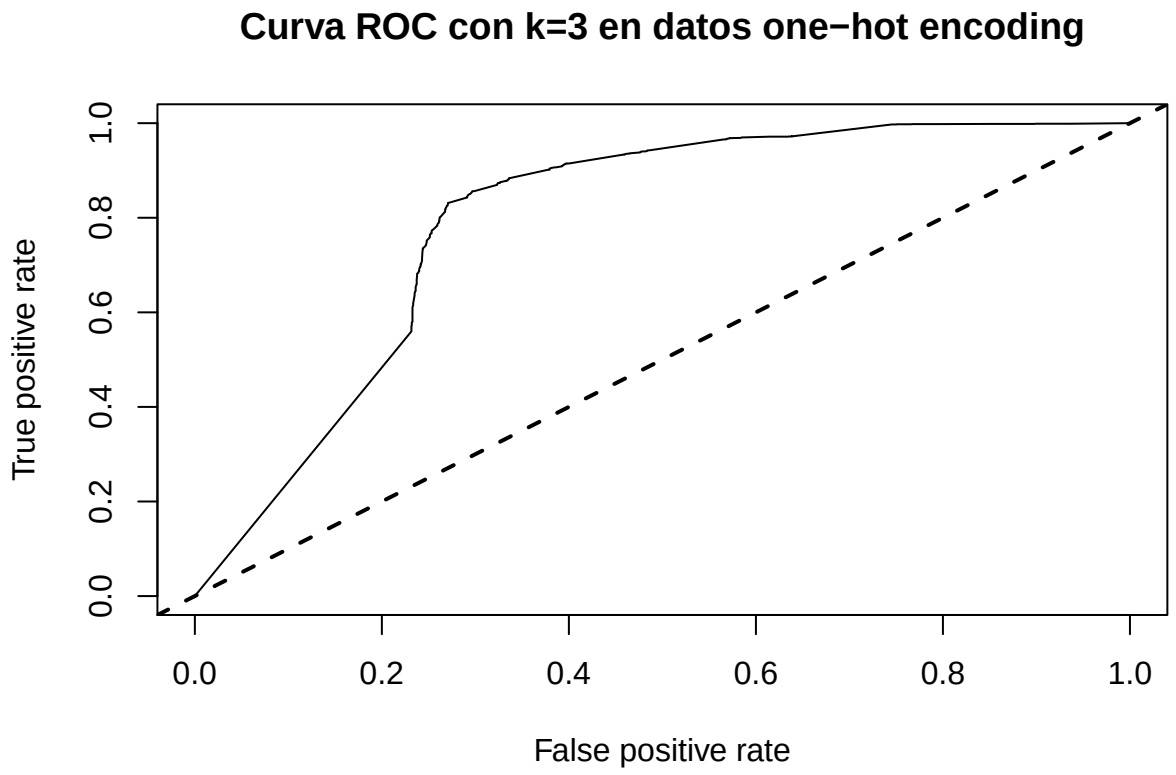
```
knn3_pred_onehot<-prediction(predictions = knn3_z_onehot, labels = test_label_z_onehot)

knn5_pred_onehot<-prediction(predictions = knn5_z_onehot, labels = test_label_z_onehot)

knn7_pred_onehot<-prediction(predictions = knn7_z_onehot, labels = test_label_z_onehot)

knn11_pred_onehot<-prediction(predictions = knn11_z_onehot, labels = test_label_z_onehot)
```

A partir de los valores estandarizados anteriormente realizamos la evaluación de los datos para hacer la curva ROC, midiendo el ratio "False positive" contra el "True positive":

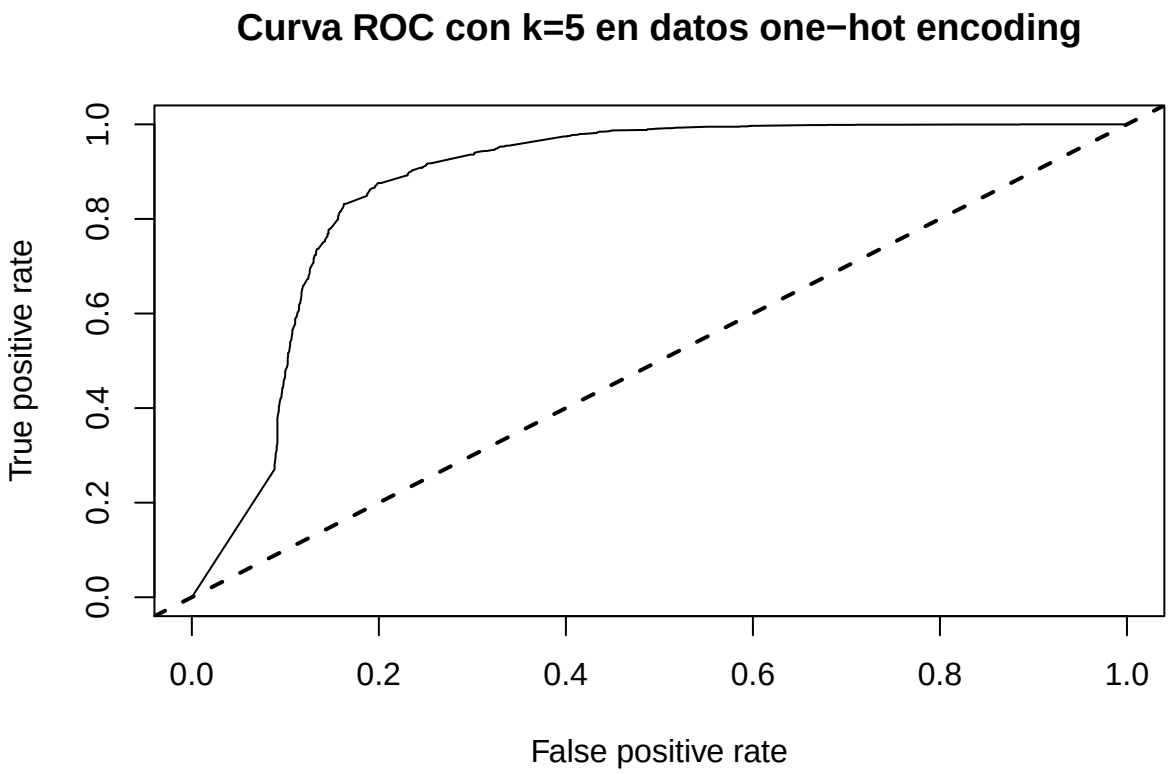
```
perf_k3_onehot<-performance(knn3_pred_onehot, measure =  "tpr", x.measure = "fpr")

perf_k5_onehot<-performance(knn5_pred_onehot, measure =  "tpr", x.measure = "fpr")

perf_k7_onehot<-performance(knn7_pred_onehot, measure =  "tpr", x.measure = "fpr")

perf_k11_onehot<-performance(knn11_pred_onehot, measure =  "tpr", x.measure = "fpr")
```

Hacemos un plot para cada valor de k para visualizar la curva ROC:

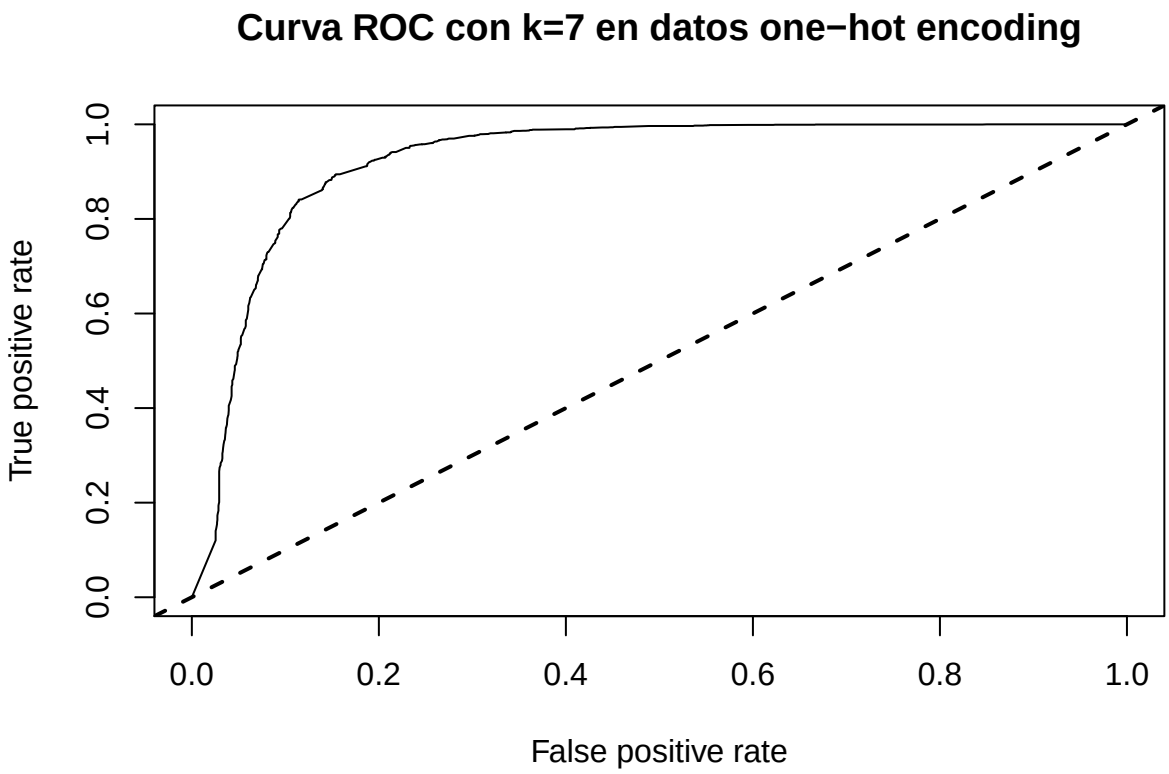
```
plot(perf_k3_onehot, main= "Curva ROC con k=3 en datos one-hot encoding")
abline(a=0,b=1,lwd=2,lty=2)
```



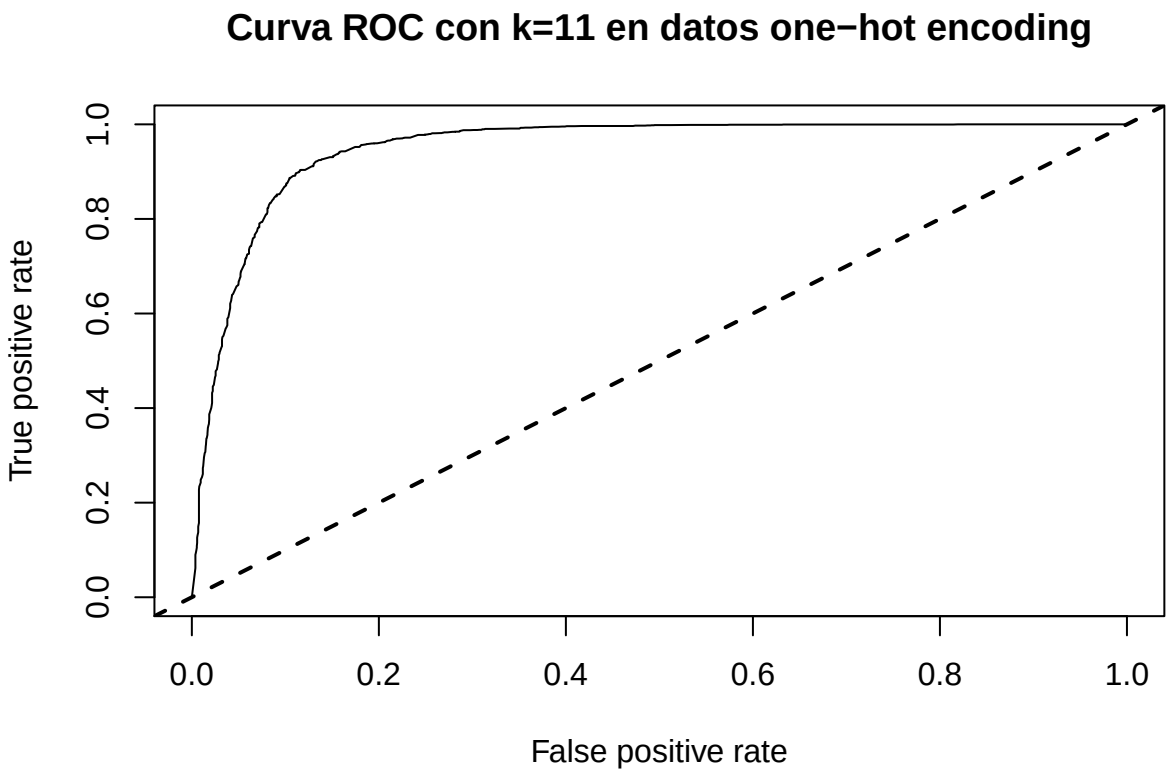

```
plot(perf_k5_onehot, main= "Curva ROC con k=5 en datos one-hot encoding")
abline(a=0,b=1,lwd=2,lty=2)
```




```
plot(perf_k7_onehot, main= "Curva ROC con k=7 en datos one-hot encoding")
abline(a=0,b=1,lwd=2,lty=2)
```




```
plot(perf_k11_onehot, main= "Curva ROC con k=11 en datos one-hot encoding")
abline(a=0,b=1,lwd=2,lty=2)
```



También podemos observar los falsos positivos y negativos, y el error de clasificación para los diferentes de k mediante una matriz de confusión para cada valor de k (consideramos la si interacción como la clase positiva):

Matriz de confusión de una knn con k=3

```
confusionMatrix(knn3_onehot, test_label_onehot,positive = "Si union")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No union Si union
##   No union     1818        2
##   Si union      816     2592
##
##                Accuracy : 0.8435
##                  95% CI : (0.8334, 0.8533)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6878
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9992
##             Specificity : 0.6902
##          Pos Pred Value : 0.7606
##          Neg Pred Value : 0.9989
```

```
##              Prevalence : 0.4962
##          Detection Rate : 0.4958
##    Detection Prevalence : 0.6519
##       Balanced Accuracy : 0.8447
##
##        'Positive' Class : Si union
##
```

Matriz de confusión de una knn con k=5

```
confusionMatrix(knn5_onehot, test_label_onehot,positive = "Si union")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No union Si union
##   No union     1866        0
##   Si union      768     2594
##
##                Accuracy : 0.8531
##                  95% CI : (0.8432, 0.8626)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7068
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.7084
##          Pos Pred Value : 0.7716
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4962
##          Detection Rate : 0.4962
##    Detection Prevalence : 0.6431
##       Balanced Accuracy : 0.8542
##
##        'Positive' Class : Si union
##
```

Matriz de confusión de una knn con k=7

```
confusionMatrix(knn7_onehot, test_label_onehot,positive = "Si union")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No union Si union
##   No union     1858        0
##   Si union      776     2594
##
##                Accuracy : 0.8516
##                  95% CI : (0.8416, 0.8611)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.7038
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.7054
##          Pos Pred Value : 0.7697
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4962
##          Detection Rate : 0.4962
##    Detection Prevalence : 0.6446
##       Balanced Accuracy : 0.8527
##
##        'Positive' Class : Si union
##
```

Matriz de confusión de una knn con k=11

```
confusionMatrix(knn11_onehot, test_label_onehot,positive = "Si union")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No union Si union
##   No union     1822        0
##   Si union      812     2594
##
##                Accuracy : 0.8447
##                  95% CI : (0.8346, 0.8544)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6901
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.6917
##          Pos Pred Value : 0.7616
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4962
##          Detection Rate : 0.4962
##    Detection Prevalence : 0.6515
##       Balanced Accuracy : 0.8459
##
##        'Positive' Class : Si union
##
```

Comentar los resultados de la clasificación en función de la curva ROC y del número de falsos positivos, falsos negativos y error de clasificación obtenidos para los diferentes valores de *k*. La clase que será asignada como positiva es la SB.

En las curvas ROC observamos como cuanto mayor es el valor de k dado, mejor es el clasificador, ya que se acerca más rápido a el valor máximo de la ratio de verdaderos positivos. Sin embargo, visualizando las matrices de confusión respectivas para cada valor de k, se puede observar qué valor de k es mejor para nuestro modelo **knn**.

Con un valor de k=3 nos encontramos con una sensibilidad casi perfecta con solo 2 predicciones falsas positivas. Sin embargo, hay una elevada cantidad de falsos positivos, dando una especificidad del 69%. En el caso de k=5 observamos una sensibilidad perfecta y un aumento de la especificidad al 70%, indicando que la knn predice correctamente gran cantidad de los casos. Con el valor de k=7 ya empieza a observarse una leve disminución en la especificidad, afectando la precisión del modelo. Finalmente, con una k=11 se observa de manera más significativa el "overfitting" del modelo knn, ya que empiezan a predecirse demasiados valores como falsos positivos, disminuyendo la especificidad y consecuentemente, la precisión del modelo.

En conclusión, para este "dataset" de péptidos y utilizando estos porcentajes de "train dataset" y "test dataset", el valor de k para conseguir el modelo óptimo de una **knn** es k=5.

Transformación basada en una matriz de substitución BLOSUM62

Transformar las secuencias de aminoácidos en vectores numéricos usando la función de transformación desarrollada anteriormente.

Asignamos las secuencias de aminoácidos transformadas a la variable *blosum_peptidos*:

```
blosum_peptidos<-blosum_f(secuencia)
```

Utilizando la semilla aleatoria 123, separar los datos en dos partes, una parte para training (67%) y una parte para test (33%).

Antes de realizar la obtención de datos para el training y test datasets, hemos de procesar los datos para que puedan ser leídos en el formato adecuado, y para ser leídos más fácilmente en resultados posteriores:

Asignamos la variable *etiquetas* a la columna que identifica si ha habido interacción o no con el péptido. Además, cambiamos los nombres de las etiquetas por "Si union" y "No union" respectivamente.

```
etiquetas_blosum<-factor(data$label, levels = c("NB","SB"),labels = c("No union", "Si union"))
```

Pasamos los péptidos a formato dataframe:

```
dfpeptid_blosum<-as.data.frame(blosum_peptidos)
```

Unimos las etiquetas al nuevo dataframe:

```
dfpeptidos_blosum<-cbind(etiquetas_blosum,dfpeptid_blosum)
```

Hacemos la selección de datos, 67% para train dataset y 33% para el test dataset:

```
set.seed(123)
recogida_datos_blosum<-sample(1:nrow(dfpeptidos_blosum), nrow(dfpeptidos_blosum)*0.67,replace = FALSE)
```

Creamos dos variables para guardar los datos seleccionados para el train dataset y test dataset:

```
train_blosum<-dfpeptidos_blosum[recogida_datos_blosum,2:181]
test_blosum<-dfpeptidos_blosum[-recogida_datos_blosum,2:181]
```

Guardamos las etiquetas del train dataset y test dataset:

```
train_label_blosum<-dfpeptidos_onehot[recogida_datos_blosum,1]
test_label_blosum<-dfpeptidos_onehot[-recogida_datos_blosum,1]
```

Utilizar un knn (*k*=3, 5, 7, 11) basado en el training para predecir que péptidos del test interaccionan o no con el MHCI. Además, realizar una curva ROC para cada *k*.

El K-nn nos devuelve un vector factor de etiquetas predecidas para cada uno de los ejemplos en el test dataset. Realizamos un kNN para k=3, k=5, k=7 y k=11:

```
knn3_blosum<- knn(train= train_blosum, test= test_blosum, cl= train_label_blosum, k= 3, prob= TRUE)
knn5_blosum<- knn(train= train_blosum, test= test_blosum, cl= train_label_blosum, k= 5, prob= TRUE)
knn7_blosum<- knn(train= train_blosum, test= test_blosum, cl= train_label_blosum, k= 7, prob= TRUE)
knn11_blosum<- knn(train= train_blosum, test= test_blosum, cl= train_label_blosum, k= 11, prob= TRUE)
```

Para realizar un plot con la curva ROC, reescalamos los valores de los kNN realizados anteriormente:

```
knn3_z_blosum<-attr(knn3_blosum, "prob")
knn3_Z_blosum<- 2 * ifelse(knn3_blosum =="-1", 1-knn3_z_blosum, knn3_z_blosum) -1

knn5_z_blosum<-attr(knn5_blosum, "prob")
knn5_Z_blosum<- 2 * ifelse(knn5_blosum =="-1", 1-knn5_z_blosum, knn5_z_blosum) -1

knn7_z_blosum<-attr(knn7_blosum, "prob")
knn7_Z_blosum<- 2 * ifelse(knn7_blosum =="-1", 1-knn7_z_blosum, knn7_z_blosum) -1

knn11_z_blosum<-attr(knn11_blosum, "prob")
knn11_Z_blosum<- 2 * ifelse(knn11_blosum =="-1", 1-knn11_z_blosum, knn11_z_blosum) -1
```

Convertimos estos valores reescalados y las etiquetas a formato dataframe para que estén en un formato que pueda leer el paquete ROCR:

```
knn3_z_blosum<-as.data.frame(knn3_z_blosum)
knn5_z_blosum<-as.data.frame(knn5_z_blosum)
knn7_z_blosum<-as.data.frame(knn7_z_blosum)
knn11_z_blosum<-as.data.frame(knn11_z_blosum)
test_label_z_blosum<-as.data.frame(test_label_blosum)
```

Realizamos la predicción de cada kNN para estandarizar el formato:

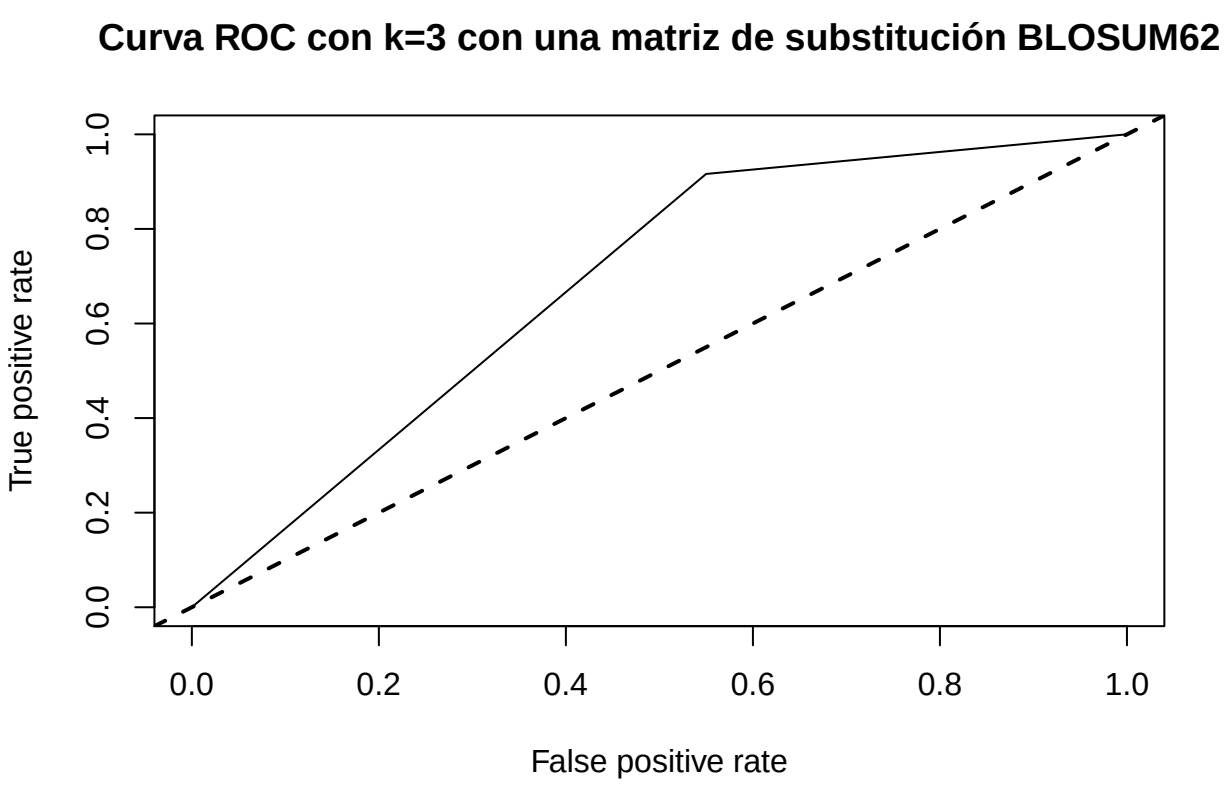
```
knn3_pred_blosum<-prediction(predictions = knn3_z_blosum, labels = test_label_z_blosum)

knn5_pred_blosum<-prediction(predictions = knn5_z_blosum, labels = test_label_z_blosum)

knn7_pred_blosum<-prediction(predictions = knn7_z_blosum, labels = test_label_z_blosum)

knn11_pred_blosum<-prediction(predictions = knn11_z_blosum, labels = test_label_z_blosum)
```

A partir de los valores estandarizados anteriormente realizamos la evaluación de los datos para hacer la curva ROC, midiendo el ratio "False positive" contra el "True positive":

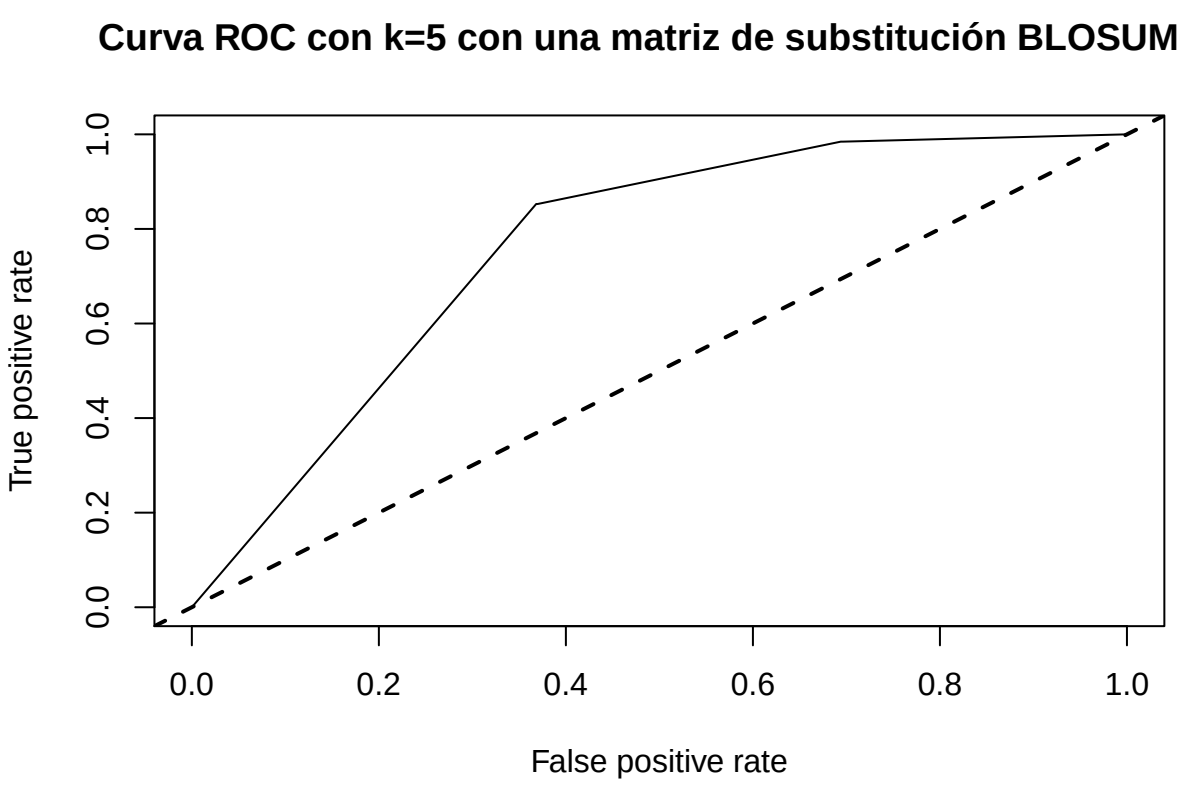
```
perf_k3_blosum<-performance(knn3_pred_blosum, measure =  "tpr", x.measure = "fpr")

perf_k5_blosum<-performance(knn5_pred_blosum, measure =  "tpr", x.measure = "fpr")

perf_k7_blosum<-performance(knn7_pred_blosum, measure =  "tpr", x.measure = "fpr")

perf_k11_blosum<-performance(knn11_pred_blosum, measure =  "tpr", x.measure = "fpr")
```

Hacemos un plot para cada valor de k para visualizar la curva ROC:

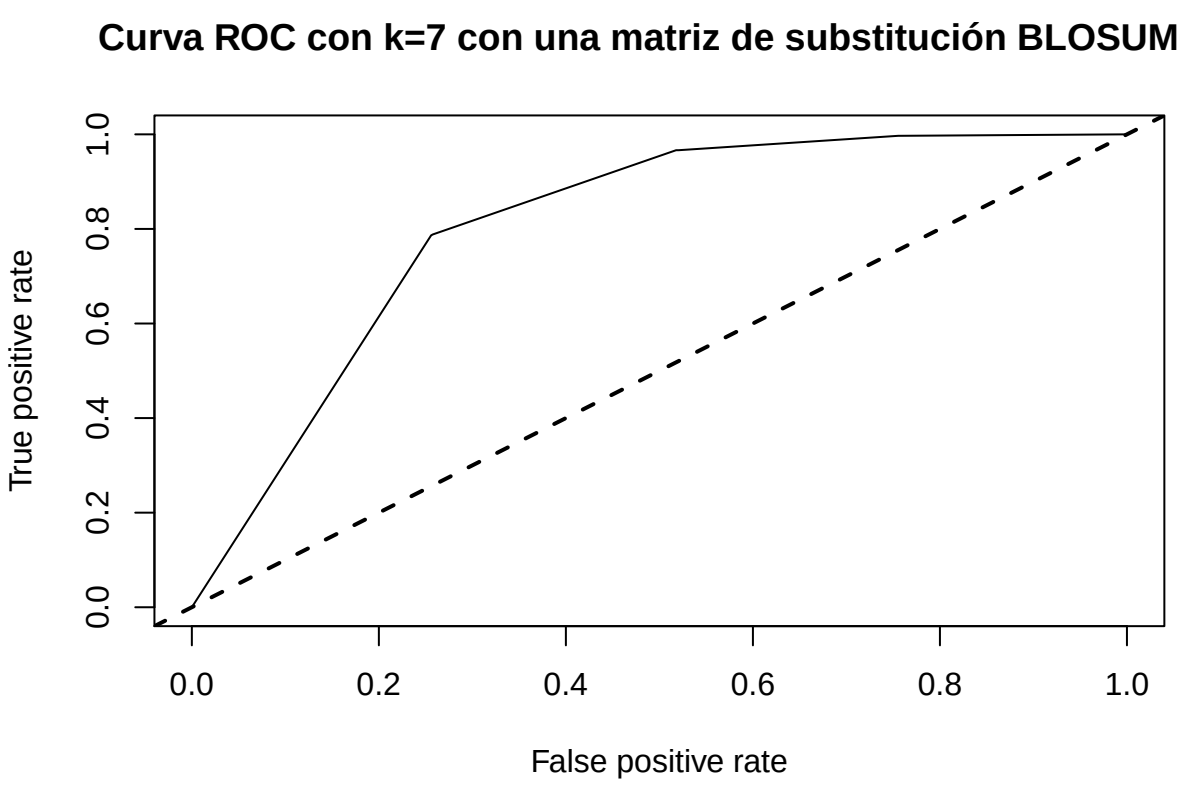
```
plot(perf_k3_blosum, main= "Curva ROC con k=3 con una matriz de substitución BLOSUM62")
abline(a=0,b=1,lwd=2,lty=2)
```



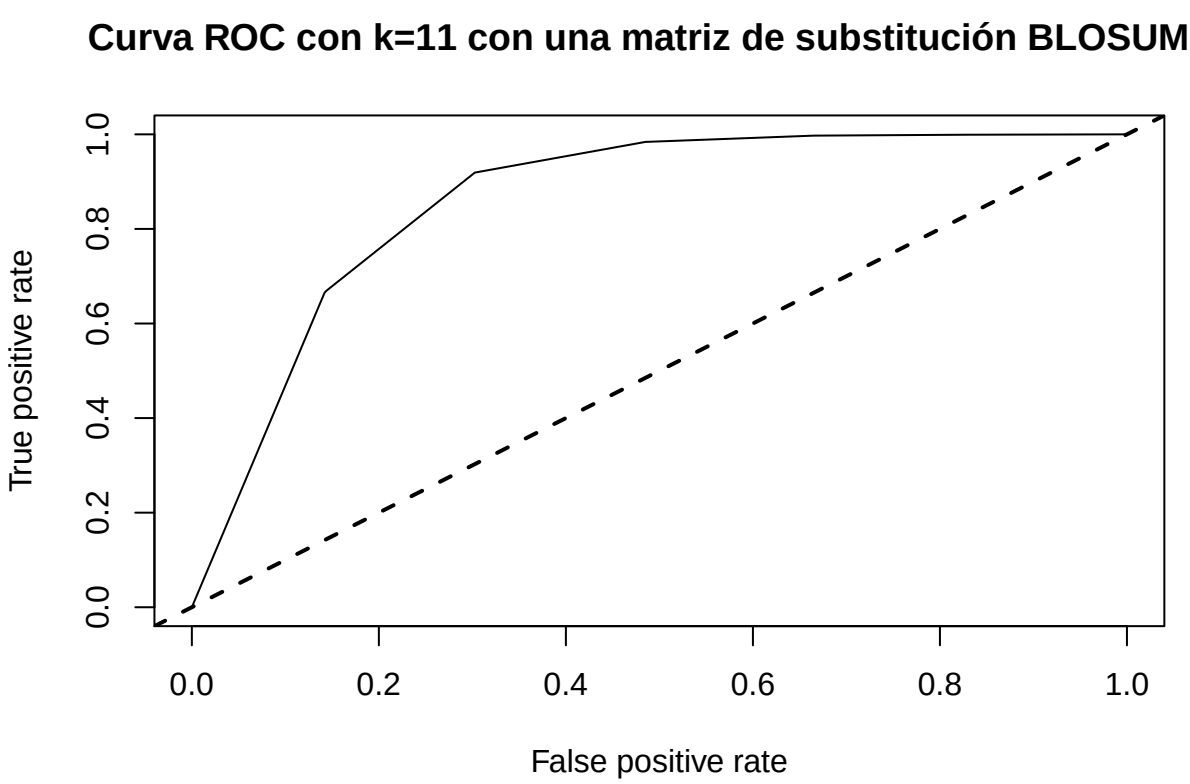

```
plot(perf_k5_blosum, main= "Curva ROC con k=5 con una matriz de substitución BLOSUM62")
abline(a=0,b=1,lwd=2,lty=2)
```




```
plot(perf_k7_blosum, main= "Curva ROC con k=7 con una matriz de substitución BLOSUM62")
abline(a=0,b=1,lwd=2,lty=2)
```




```
plot(perf_k11_blosum, main= "Curva ROC con k=11 con una matriz de substitución BLOSUM62")
abline(a=0,b=1,lwd=2,lty=2)
```



También podemos observar los falsos positivos y negativos, y el error de clasificación para los diferentes de k mediante una matriz de confusión para cada valor de k (consideramos la si interacción como la clase positiva):

Matriz de confusión de una knn con k=3

```
confusionMatrix(knn3_blosum, test_label_blosum,positive = "Si union")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No union Si union
##   No union     1871       11
##   Si union      763     2583
##
##                Accuracy : 0.852
##                  95% CI : (0.842, 0.8615)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7045
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9958
##             Specificity : 0.7103
##          Pos Pred Value : 0.7720
##          Neg Pred Value : 0.9942
```

```
##              Prevalence : 0.4962
##          Detection Rate : 0.4941
##    Detection Prevalence : 0.6400
##       Balanced Accuracy : 0.8530
##
##        'Positive' Class : Si union
##
```

Matriz de confusión de una knn con k=5

```
confusionMatrix(knn5_blosum, test_label_blosum,positive = "Si union")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No union Si union
##   No union     1835        2
##   Si union      799     2592
##
##                Accuracy : 0.8468
##                  95% CI : (0.8367, 0.8565)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6943
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9992
##             Specificity : 0.6967
##          Pos Pred Value : 0.7644
##          Neg Pred Value : 0.9989
##              Prevalence : 0.4962
##          Detection Rate : 0.4958
##    Detection Prevalence : 0.6486
##       Balanced Accuracy : 0.8479
##
##        'Positive' Class : Si union
##
```

Matriz de confusión de una knn con k=7

```
confusionMatrix(knn7_blosum, test_label_blosum,positive = "Si union")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No union Si union
##   No union     1794        0
##   Si union      840     2594
##
##                Accuracy : 0.8393
##                  95% CI : (0.8291, 0.8492)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.6794
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.6811
##          Pos Pred Value : 0.7554
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4962
##          Detection Rate : 0.4962
##    Detection Prevalence : 0.6568
##       Balanced Accuracy : 0.8405
##
##        'Positive' Class : Si union
##
```

Matriz de confusión de una knn con k=11

```
confusionMatrix(knn11_blosum, test_label_blosum,positive = "Si union")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No union Si union
##   No union     1769        0
##   Si union      865     2594
##
##                Accuracy : 0.8345
##                  95% CI : (0.8242, 0.8445)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6699
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.6716
##          Pos Pred Value : 0.7499
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4962
##          Detection Rate : 0.4962
##    Detection Prevalence : 0.6616
##       Balanced Accuracy : 0.8358
##
##        'Positive' Class : Si union
##
```

Comentar los resultados de la clasificación en función de la curva ROC y del número de falsos positivos, falsos negativos y error de clasificación obtenidos para los diferentes valores de *k*. La clase que será asignada como positiva es la SB.

En este caso también observamos en las gráficas de las curvas ROC para cada valor de k en el modelo de **knn** que a mayor el valor de k, mejor parece ser el modelo. Esto se observa en que según se va aumentando el valor de k se va obteniendo una pendiente de la curva más elevada llegando más rápidamente a una sensibilidad

del 100%. Además, podríamos calcular el valor de la área debajo de la curva (AUC) para observar cómo aumenta esta área conforme aumenta el valor de k. Por ejemplo, podemos analizar el valor AUC para k=11:

```
auc_k11_blosum<-performance(knn11_pred_blosum, measure="auc")
unlist(auc_k11_blosum@y.values)
```

```
## [1] 0.8610717
```

Con este valor vemos que el 86% del área se encuentra bajo la curva ROC, indicando que predice los valores con una alta precisión.
Sin embargo, visualizando las matrices de confusión,se observa una pérdida de precisión debido al aumento de contaje de falsos positivos conforme aumenta el valor de k. Si queremos un modelo **knn** con la máxima sensibilidad posible a costa de una pérdida de precisión, optaremos por elegir k=7, ya que éste tiene un 100% de sensibilidad. A partir de este valor de k, aumenta la cantidad de falsos positivos sin ninguna variación en la sensibilidad, así que no tiene mayor utilidad. Si queremos obtener una mayor especificidad, podemos elegir k=3 o k=5, pero veremos consecuentemente una disminución de la sensibilidad.

(c) Comparar los resultados de clasificación obtenidos con las dos técnicas de representación de péptidos.

Comparando las dos técnicas utilizadas en este trabajo, observamos que las curvas ROC que representan una mayor eficacia de la predicción del modelo **knn** son las realizadas con la técnica one-hot. Además, esto se sustenta con las matrices de confusión, donde obtenemos que la precisión del modelo **knn** para valor k=5 es del 85%, la mayor precisión de todas las matrices realizadas en este trabajo.

Por lo tanto, este análisis del modelo **knn** demuestra que el procesado mediante la técnica *one-hot encoding* da mejores resultados que la realizada con la *matriz de substitución BLOSUM62* para este conjunto de datos.