Classification and diagnostic prediction of cancers using gene expression profiling and Support Vector Machine

PEC 3: Machine Learning

Jessica Vera

18 de December, 2021

Contents

El presente documento utiliza la potencia de la herramientas de *machine learning*, máquinas de soporte vectorial o *support vector machine* (SVM), en la predicción de tipos de cáncer, dada la expresión génica de ciertos genes. Para ello probaremos un modelo lineal y varios no lineales, para identificar cuál es la mejor manera de distinguir a los tipos de cáncer, de modo que se obtenga la más alta precisión.

1 Algoritmo Support Vector Machine

La máquina de soporte vectorial es un algoritmo de aprendizaje supervisado que pretende encontrar un hiperplano multidimensional (dependiendo del número de características que se tengan en el conjunto de datos) que distinga los grupos de entrenamiento entre los puntos de datos, de modo que los grupos que se formen sean homógeneos dentro de ellos. El algoritmo busca el hiperplano que maximice la distancia entre los grupos encontrados.

Es una de las técnicas preferidas en machine learning porque produce alta precisión a un bajo coste computacional; así como, porque puede ser usado tanto como técnica de clasificación como de regresión. Además, es uno de los métodos que maneja muy bien los conjuntos de datos en que existe mayor número de características que ejemplos.

Este algoritmo se basa en técnicas de aprendizaje del vecino más cercano y regresión lineal. No obstante, su uso puede extenderse a grupos que no son linealmente separables. Para estos casos puntuales se emplea una función Kernel, que pudieran ser lineales, polinomiales, gausianas y sigmoides.

1.1 Fortalezas y debilidades

El algoritmo de máquina de soporte vectorial tiene características que lop hacenun método ventajoso, pero también presenta desventajas, las cuales detallaremos a continuación.

Table 1: Fortalezas y debilidades del algoritmo Support Vector Machine.

Fortalezas	**Debilidades**
Útil en problemas de predicción numérica o de clasificación	Llegar al mejor modelo requiere de varias combinaciones de los parámetros
Es robusto a la sobreestimación	La fase de entrenamiento puede llegar a ser lenta
Puede resultar más fácil que las redes neuronales	Los resultados pueden llegar a ser complejos y de difícil comprensión
Alta precisión a bajo costo computacional	
Funciona también con conjuntos de datos donde hay más características que ejemplos	

2 Ejemplo de aplicacion

Para evaluar la bondad del algoritmo de máquinas de soporte vectorial realizaremos un ejemplo de aplicación en el diagnóstico de cáncer, usando información del perfil de expresión génica obtenida mediante técnicas de microarrays.

En este experimento, se clasifican los tipos de cáncer como: B-like, ERBB2p, Nrm, Lum-B.C

2.1 Lectura de los datos

El conjunto de datos se encuentra condensado en un archivo `.csv` que hemos denominado datacancer.csv. En este conjunto de datos se encuentran las características predictoras y la clase target. Para iniciar, importaremos el conjunto de datos, cuyo separador es una coma `","`.

```
# importando la data
dfcancer <- read.table(params$myfile, stringsAsFactors = FALSE, sep = ",", header = TRUE)
dim(dfcancer)
```

```
[1]  102 5564
```

Este dataset contiene 102 ejemplos y 5564 características. La variable target `Y` representa el tipo de cáncer y se codifica como:

- 1: B-like
- 2: ERBB2p
- 3: Nrm
- 4: Lum-B.C

2.2 Exploracion de los datos

Verificamos que las características sean numéricas, aplicando a cada columna del dataset `dfcancer` la función `is.numeric()` y retornando aquellas que devuelven el valor `FALSE`

```
# ¿cuales de las caracteristicas no son numericas?
which(apply(dfcancer,2,is.numeric)==FALSE)
```

```
named integer(0)
```

Podemos observar que ninguna de las características resultó distinta de numérica, esto incluye a la variable target, por lo que, debemos asegurarnos de que sea de tipo factor

```
# Definiendo como factor a la variable target
dfcancer$Y <- as.factor(dfcancer$Y)
```

Ahora consultemos si existen valores faltantes en el conjunto de datos. Esto lo realizaremos mediante la función `is.na()`. De acuerdo con los resultados de la consulta, hay ausencia de valores faltantes en la muestra.

```
# ¿qu$\'{e}$ elementos est$\'{a}$n faltantes?
which(is.na(dfcancer))
```

```
integer(0)
```

Ahora, mostraremos la distribución de frecuencias de los tipos de tumor en este experimento.

```
# Frecuencias en la variable target
yabs <- table(dfcancer$Y)
yrel <- round(table(dfcancer$Y)/dim(dfcancer)[1],3)
frecuenciay <- data.frame(Tipo = c("B-like", "ERBB2p", "Nrm", "Lum-B.C"),
                          Freq.Absoluta = as.numeric(yabs),
                          Freq.Relativa = as.numeric(yrel))
```

De acuerdo con los resultados, la distribución de los tipos de tumor es similar.

```
kbl(frecuenciay, caption = "Frecuencia de los tipos de tumor en la muestra",
    booktabs = T) %>%
  kable_styling(latex_options = "hold_position")
```

Table 2: Frecuencia de los tipos de tumor en la muestra

Tipo	Freq.Absoluta	Freq.Relativa
B-like	25	0.245
ERBB2p	26	0.255
Nrm	28	0.275
Lum-B.C	23	0.225

Dado que contamos con varios genes, tomaremos una muestra aleatoria de ellos y obtendremos su resumen estadístico por tipo de cáncer. Tomaremos 10 genes aleatorios

```
# establecemos una semilla para la selecci$\'{o}$n aleatoria de caracter$\'{i}$sticas
set.seed(params$seed.train)
# 20 indices de columnas aleatorios
RandomGenes<- sample(c(2:ncol(dfcancer)),10)
```

Ahora identificaremos los ejemplos que pertenecen a cada tipo de muestra.

```
# muestras del tipo de tumor 1
indices1 <- which(dfcancer$Y=="1")
# muestras del tipo de tumor 2
indices2 <- which(dfcancer$Y=="2")
# muestras del tipo de tumor 3
indices3 <- which(dfcancer$Y=="3")
# muestras del tipo de tumor 4
indices4 <- which(dfcancer$Y=="4")
```

Creamos los nuevos subconjuntos de datos por clase de cáncer

```
# Datasets de las 10 variables seleccionadas aleatoriamente
dataset1 <- dfcancer[indices1, RandomGenes]
dataset2 <- dfcancer[indices2, RandomGenes]
dataset3 <- dfcancer[indices3, RandomGenes]
dataset4 <- dfcancer[indices4, RandomGenes]
```

Si ahora realizamos gráficos de cajas para observar la distribución de los genes seleccionados aleatoriamente por cada grupo, y ajustamos para que presenten la misma escala, podemos observar que la distribución de los genes es similar entre los tipos de cáncer.

```
par(mfrow=c(2,2))
boxplot(dataset1, cex.axis=1, ylab='Expresion',main="1: B-like",las=2,
        col = "plum3", ylim = c(-5,5), cex.lab = 1.5)
boxplot(dataset2,cex.axis=1, ylab='', main="2: ERBB2p", las=2,
        col = "wheat3", ylim = c(-5,5))
```

```
boxplot(dataset3, cex.axis=1, ylab='Expresion',main="3: Nrm",las=2,
        col = "lightblue", ylim = c(-5,5), cex.lab = 1.5)
boxplot(dataset4,cex.axis=1, ylab='', main="4: Lum-B.C", las=2,
        col = "palegreen", ylim = c(-5,5))
```

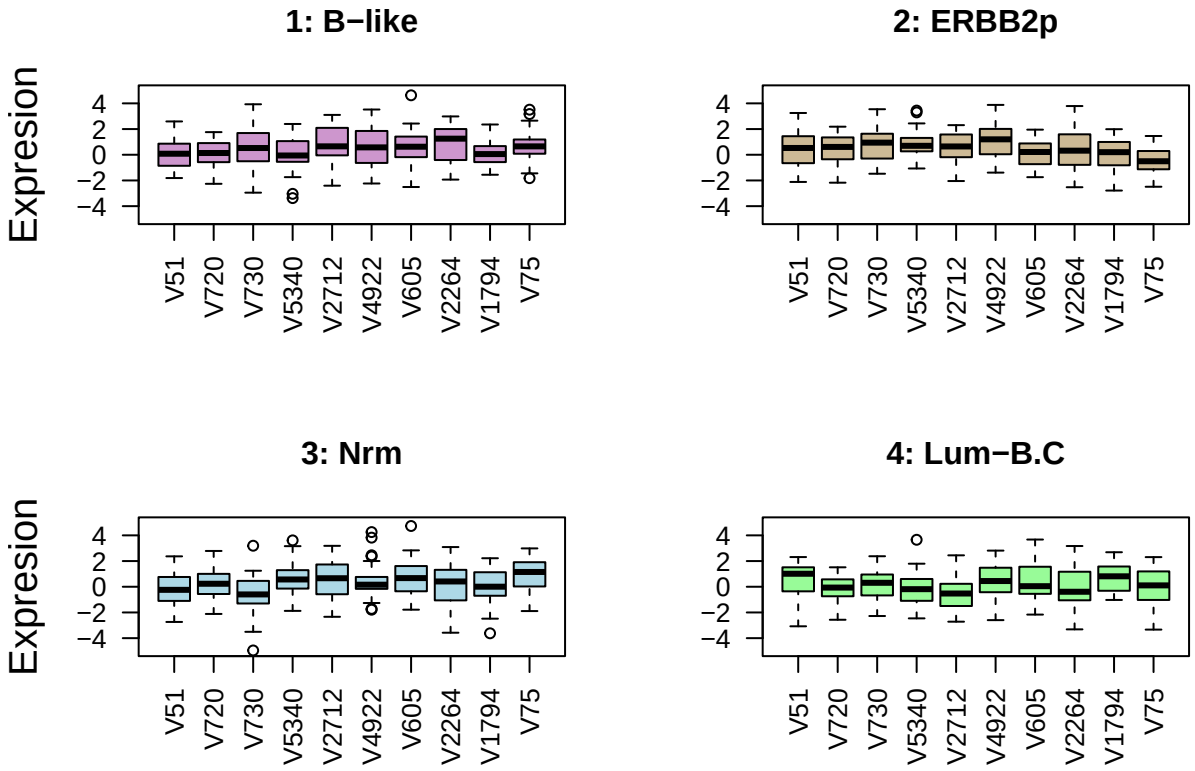

Figure 1: Boxplots de la expresi′on g′enica de un subconjunto de genes por cada tipo de c′ancer

Ahora trataremos de obtener cierta información de cómo las clases se agrupan, considerando todos los genes, mediante análisis de componentes principales. Las dimensiones mostradas nos permiten observar que hay una alta distinción entre el tipo de cáncer 2: ERBB2p y el resto de los tipos. Los tipos de cáncer 1: B-like y 4: Lum-B.C también presenta distinción entre ellos. No obstante, el gráfico mostrado explica menos del 10% de la variabilidad total de los datos.

```
# Analisis de componentes principales - pca
res.pca <- PCA(dfcancer[,-1], graph = FALSE)
```

```
# grafico de las muestras en dos dimensiones
fviz_pca_ind(res.pca, # pca
             geom.ind = "point", # solo los puntos, no etiquetas de las muestras
             col.ind = dfcancer$Y, # tipos de cancer
             palette = c("yellow3", "tomato3", "royalblue", "seagreen"),
             legend.title = "Tipo Cancer")
```

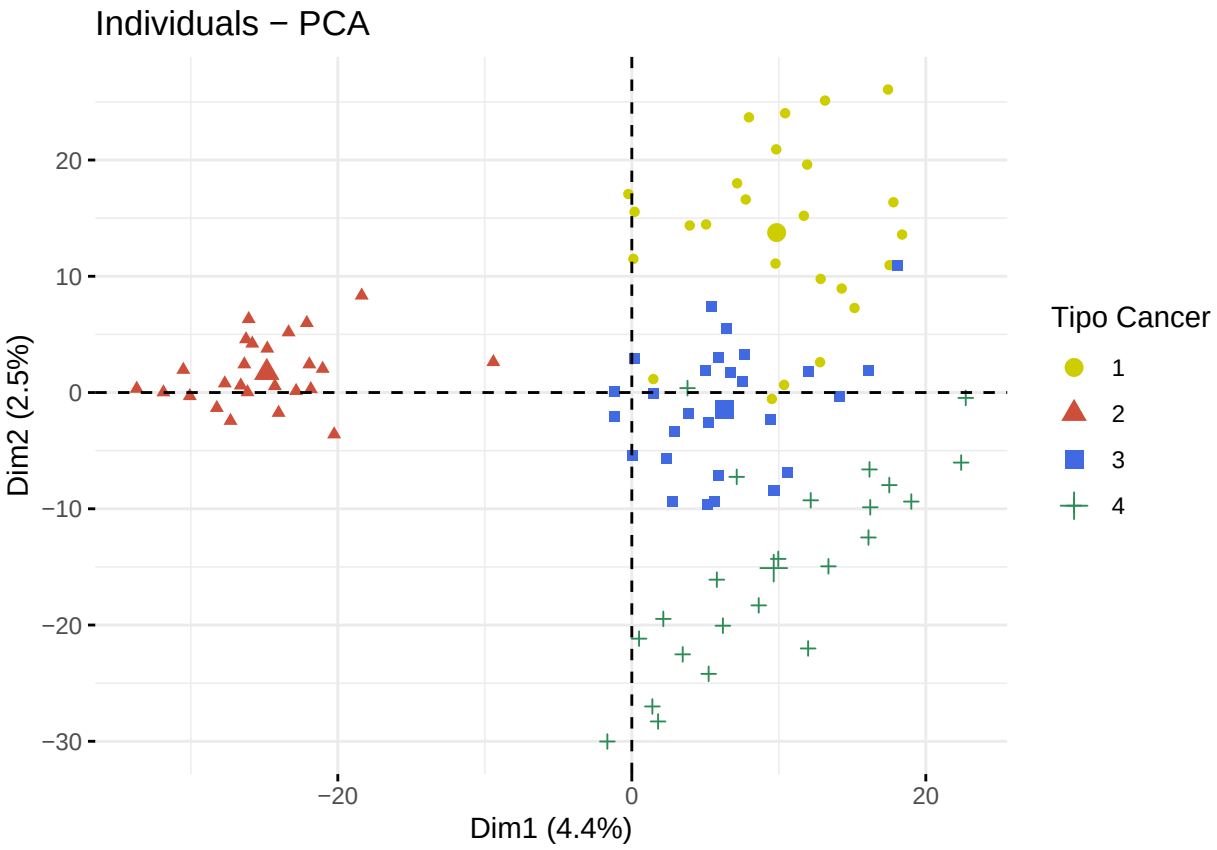

Figure 2: Heatmap de los genes diferencialmente expresados

2.2.1 Division de la muestra

Con el fin de modelar el algoritmo, divideremos los datos en una muestra de entrenamiento (`training`) y una de prueba (`test`), las cuales contendrán el 67% y 33% de los ejemplos, respectivamente.

```
## Seleccion de muestra de entrenamiento
# Tamano de la muestra de entrenamiento
ntrain <- dim(dfcancer)[1] * params$p.train
# semilla para la aleatorizacion
set.seed(params$seed.train)
# indices aleatorios que formaran parte de la muestra de entrenamiento
trainpos <- sample(c(1:dim(dfcancer)[1]), size = ntrain)
# muestra de entrenamiento
dftrain <- dfcancer[trainpos,]

## Seleccion de muestra de prueba
# muestra de prueba
dftest <- dfcancer[-trainpos,]
# Tamano de la muestra de prueba
ntest <- dim(dftest)[1]
```

2.3 Entrenamiento del modelo

Ahora construiremos dos tipos de modelos lineal y no-lineal, mediante el kernel radial (RBF). La construcción de estos modelos se realizará mediante la función `train()` del paquete `caret`

2.3.1 Modelo Lineal svmLinear

Para este modelo usaremos como parámetros al método `svmLinear` para realizar una clasificación lineal sobre el conjunto de datos `dftrain`. Así también, escalaremos las variables predictoras, mediante el argumento `preProcess`; y definiremos el control del entrenamiento de 3-fold crossvalidation, en el parámetro `trControl`.

```
# definir semilla
set.seed(params$seed.otro)
# entrenamiento del modelo
modelolineal <- caret::train(Y ~ ., dftrain, method = 'svmLinear',
                trControl= trainControl("cv", number = 3),
                preProcess = c("center","scale"),
                tuneGrid= NULL, trace = FALSE)
modelolineal
```

```
Support Vector Machines with Linear Kernel

  68 samples
5563 predictors
   4 classes: '1', '2', '3', '4'

Pre-processing: centered (5563), scaled (5563)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 43, 46, 47
Resampling results:
```

```
  Accuracy   Kappa
  0.9404906  0.9203634

Tuning parameter 'C' was held constant at a value of 1
```

La información que devuelve el modelo nos menciona que la precisión de la predicción realizada es 0.94 en el remuestreo; así como una tasa de coincidencias no debidas al azar (kappa) de 0.92. Así también, podemos verificar que el costo es el definido por defecto: $C = 1$.

2.3.2 Modelo no-lineal svmRadial

Para este modelo usaremos como parámetros al método `svmRadial` para realizar una clasificación no lineal sobre el conjunto de datos `dftrain`. Así también, escalaremos las variables predictoras, mediante el argumento `preProcess`.

2.3.2.1 3-fold crossvalidation Definiremos el control del entrenamiento de 3-fold crossvalidation, en el parámetro `trControl`.

```
# definir semilla
set.seed(params$seed.otro)
# entrenamiento del modelo
modelonolineal3CV <- caret::train(Y ~ ., dftrain, method = 'svmRadial',
               trControl= trainControl("cv", number = 3),
               preProcess = c("center","scale"),
               tuneGrid= NULL, trace = FALSE)
modelonolineal3CV
```

```
Support Vector Machines with Radial Basis Function Kernel

  68 samples
5563 predictors
   4 classes: '1', '2', '3', '4'

Pre-processing: centered (5563), scaled (5563)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 43, 46, 47
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.2794805  0.0000000
  0.50  0.2794805  0.0000000
  1.00  0.5752092  0.4173508

Tuning parameter 'sigma' was held constant at a value of 8.87513e-05
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 8.87513e-05 and C = 1.
```

La información que devuelve el modelo nos menciona que la precisión máxima alcanzada de la predicción realizada es 0.575 para los parámetros $sigma = 8.87513e-05$ y $C = 1$. De acuerdo con los resultados y el gráfico 3, para valores inferiores de $C = 1$, la precisión es inferior

```r
# Grafico del costo vs precision
plot(modelonolineal3CV)
```

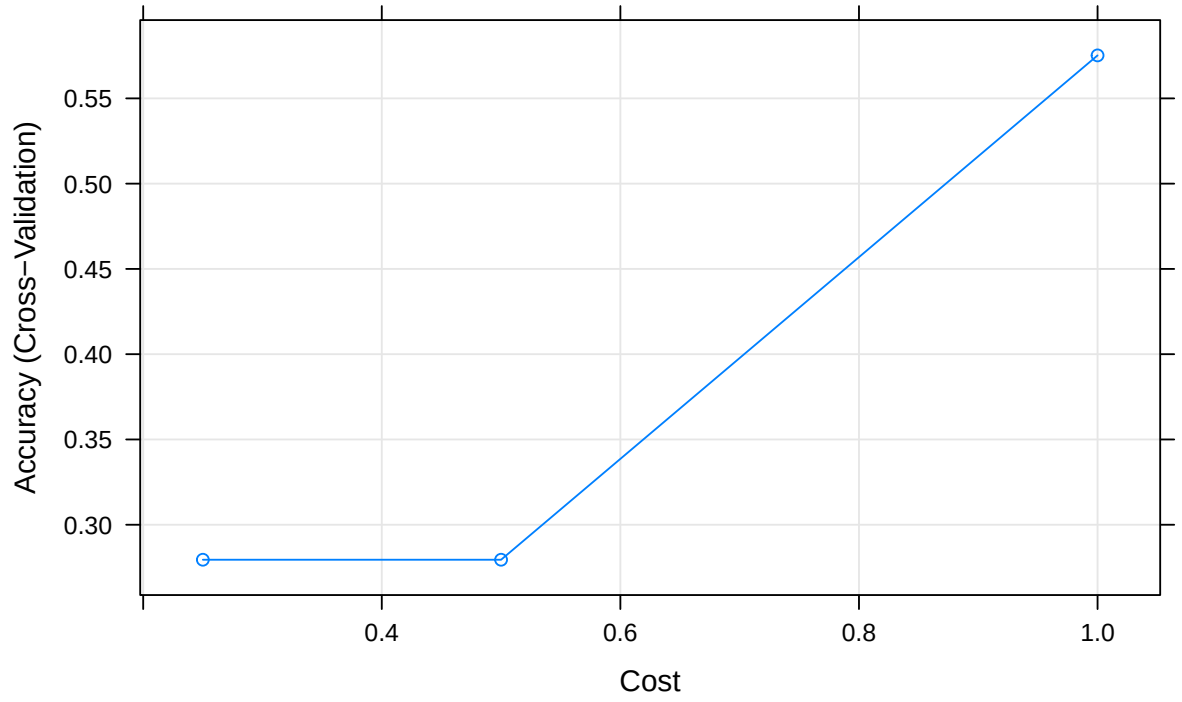

Figure 3: Precision vs. Costo del modelo no lineal 3-fold crossvalidation

2.3.2.2 5-fold crossvalidation Definiremos el control del entrenamiento de 5-fold crossvalidation, en el parámetro `trControl`.

```r
# definir semilla
set.seed(params$seed.otro)
# entrenamiento del modelo
modelonolineal5CV <- caret::train(Y ~ ., dftrain, method = 'svmRadial',
                trControl= trainControl("cv", number = 5),
                preProcess = c("center","scale"),
                tuneGrid= NULL, trace = FALSE)
modelonolineal5CV
```

```
Support Vector Machines with Radial Basis Function Kernel

  68 samples
5563 predictors
   4 classes: '1', '2', '3', '4'

Pre-processing: centered (5563), scaled (5563)
Resampling: Cross-Validated (5 fold)
```

```
Summary of sample sizes: 54, 56, 54, 54, 54
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.2785714  0.0000000
  0.50  0.2785714  0.0000000
  1.00  0.6142857  0.4753055

Tuning parameter 'sigma' was held constant at a value of 8.87513e-05
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 8.87513e-05 and C = 1.
```

La información que devuelve el modelo nos menciona que la precisión máxima alcanzada de la predicción realizada es 0.614 para los parámetros $sigma = 8.87513e-05$ y $C = 1$. De acuerdo con los resultados y el gráfico 4, para valores inferiores de $C = 1$, la precisión es inferior

```
# Grafico del costo vs precision
plot(modelonolineal5CV)
```

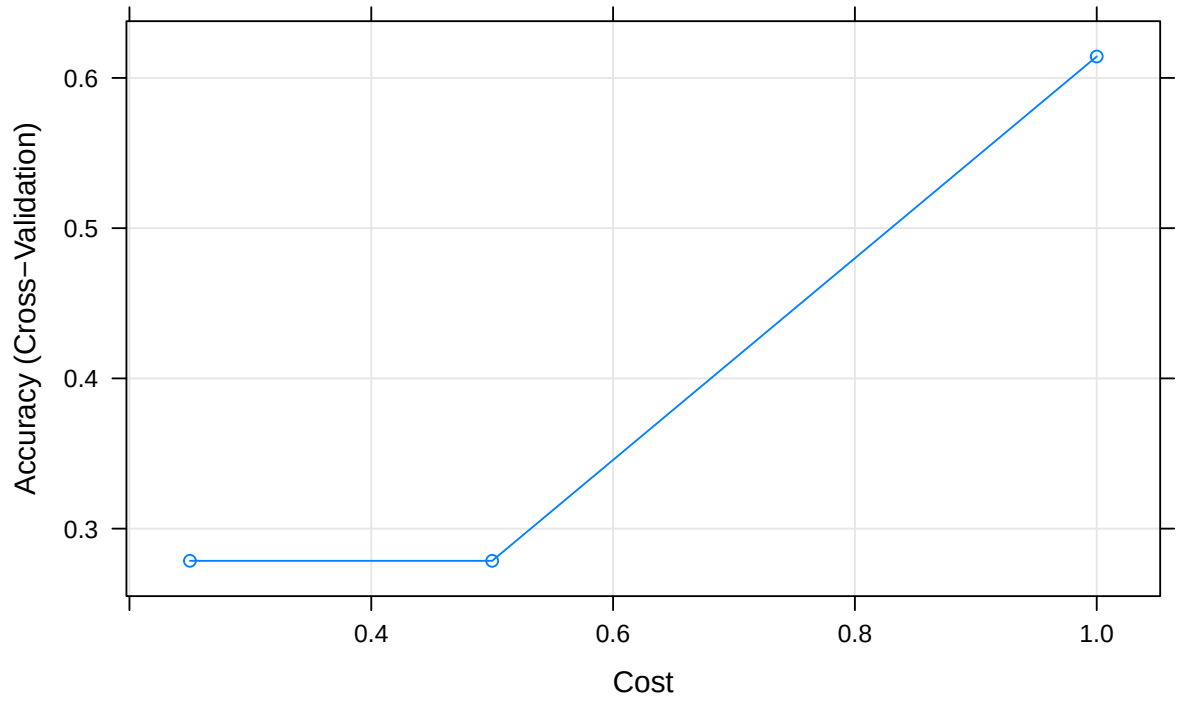

Figure 4: Precision vs. Costo del modelo no lineal 10-fold crossvalidation

2.3.2.3 10-fold crossvalidation Definiremos el control del entrenamiento de 10-fold crossvalidation, en el parámetro `trControl`.

```r
# definir semilla
set.seed(params$seed.otro)
# entrenamiento del modelo
modelonolinealCV <- caret::train(Y ~ ., dftrain, method = 'svmRadial',
               trControl= trainControl("cv", number = 10),
               preProcess = c("center","scale"),
               tuneGrid= NULL, trace = FALSE)
modelonolinealCV
```

```
Support Vector Machines with Radial Basis Function Kernel

  68 samples
5563 predictors
   4 classes: '1', '2', '3', '4'

Pre-processing: centered (5563), scaled (5563)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 60, 61, 61, 60, 62, 62, ...
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.2840476  0.0000000
  0.50  0.2840476  0.0000000
  1.00  0.7050000  0.5981914

Tuning parameter 'sigma' was held constant at a value of 8.87513e-05
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 8.87513e-05 and C = 1.
```

La información que devuelve el modelo nos menciona que la precisión máxima alcanzada de la predicción realizada es 0.705 para los parámetros $sigma = 8.87513e-05$ y $C = 1$. De acuerdo con los resultados y el gráfico 5, para valores inferiores de $C = 1$, la precisión es inferior

```r
# Grafico del costo vs precision
plot(modelonolinealCV)
```

2.3.2.4 Boostrap El método Bootstrap con 25 repeticiones para 3 posibles decay es el que se usa por defecto, por lo que omitiremos el parámetro `trControl`.

```r
# definir semilla
set.seed(params$seed.otro)
# entrenamiento del modelo
modelonolinealBS <- caret::train(Y ~ ., dftrain, method = 'svmRadial',
               preProcess = c("center","scale"),
               tuneGrid= NULL, trace = FALSE)
modelonolinealBS
```

```
Support Vector Machines with Radial Basis Function Kernel

  68 samples
5563 predictors
```

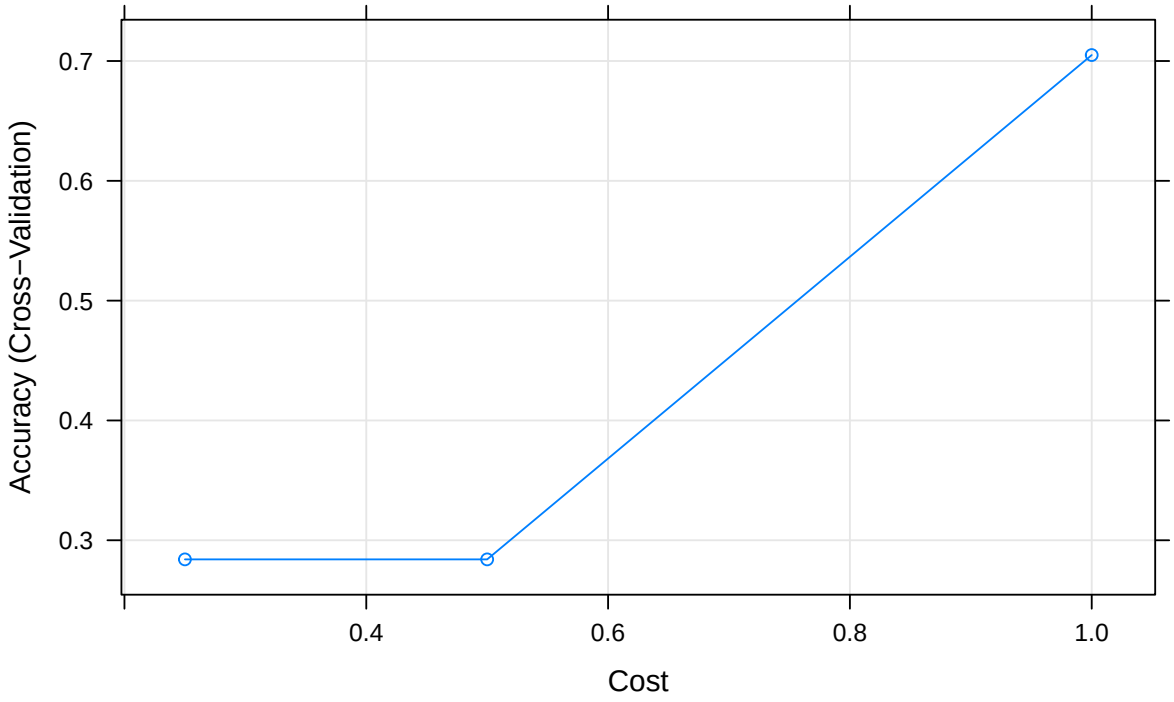

Figure 5: Precision vs. Costo del modelo no lineal 10-fold crossvalidation

```
   4 classes: '1', '2', '3', '4'

Pre-processing: centered (5563), scaled (5563)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 68, 68, 68, 68, 68, 68, ...
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.2128311  0.009842932
  0.50  0.3778613  0.214866185
  1.00  0.6381423  0.544279866

Tuning parameter 'sigma' was held constant at a value of 8.87513e-05
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 8.87513e-05 and C = 1.
```

La información que devuelve el modelo nos menciona que la precisión máxima alcanzada de la predicción realizada es 0.638 para los parámetros $sigma = 8.87513e-05$ y $C = 1$. De acuerdo con los resultados y el gráfico a continuación, para valores inferiores de $C = 1$, la precisión es inferior

```
# Grafico del costo vs precision
plot(modelonolinealBS)
```

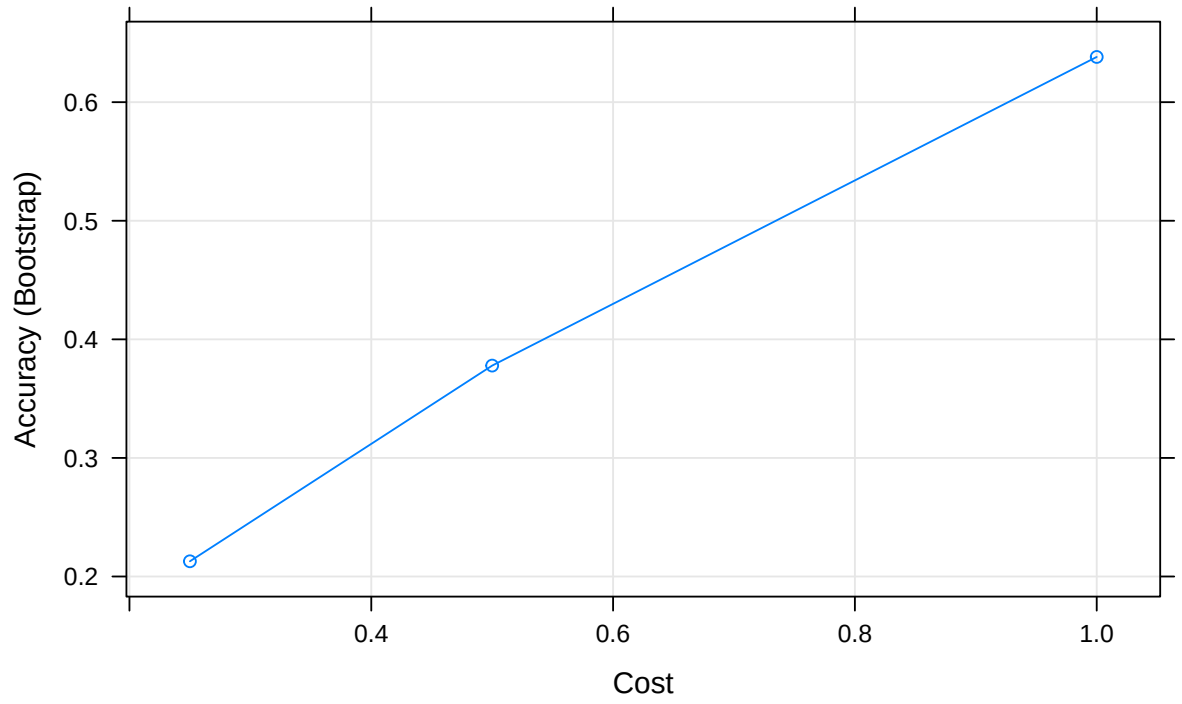

Figure 6: Precision vs. Costo del modelo no lineal Bootstrap

Podemos observar, entonces, que el ajuste realizado mediante 10-fold crossvalidation devuelve una mayor precisión, por lo que será el modelo que consideraremos para las secciones posteriores.

2.4 Evaluacion del modelo

Para evaluar el desempeño de los modelos, analizaremos las predicciones de cada uno y evaluaremos mediante matrices de confusión.

2.4.1 Modelo lineal

Para evaluar la bondad del modelo lineal ajustado, usaremos los resultados de la predicción que este realizaría en el conjunto de datos de prueba. Utilizaremos la función `predict()` sobre los datos `dftest`.

```
predmodelolineal <- predict(modelolineal, dftest)
```

A partir de la clasificación predicha por el modelo realizado, obtendremos la matriz de confusión, usando la función `confusionMatrix()` del paquete `caret`.

```
evaltestlineal <- table(predmodelolineal, dftest$Y)
(confmatrizlineal <- confusionMatrix(evaltestlineal))
```

```
Confusion Matrix and Statistics


predmodelolineal  1  2  3  4
               1  6  0  0  0
               2  0  9  0  0
               3  0  0 11  0
               4  0  0  1  7

Overall Statistics

               Accuracy : 0.9706
                 95% CI : (0.8467, 0.9993)
    No Information Rate : 0.3529
    P-Value [Acc > NIR] : 2.652e-14

                  Kappa : 0.96

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            1.0000   1.0000   0.9167   1.0000
Specificity            1.0000   1.0000   1.0000   0.9630
Pos Pred Value         1.0000   1.0000   1.0000   0.8750
Neg Pred Value         1.0000   1.0000   0.9565   1.0000
Prevalence             0.1765   0.2647   0.3529   0.2059
Detection Rate         0.1765   0.2647   0.3235   0.2059
Detection Prevalence   0.1765   0.2647   0.3235   0.2353
Balanced Accuracy      1.0000   1.0000   0.9583   0.9815
```

Sin especificar una categoría como positiva, podemos observar que la precisión fue alta (0.97) y observamos una clasificación errónea para el tipo de tumor 3: Nrm, el cual fue confundido con el tipo 4: Lum-B.C. Esto provoca que disminuyan la sensibilidad y especificidad del modelo en las clases 3 y 4, a 0.916 y 0.963, respectivamente.

2.4.2 Modelo no lineal - metodo RBF

Para evaluar la bondad del modelo no lineal seleccionado, usaremos los resultados de la predicción que este realizaría en el conjunto de datos de prueba. Utilizaremos la función `predict()` sobre los datos `dftest`.

```
predmodelonolineal10CV <- predict(modelonolinealCV, dftest)
```

A partir de la clasificación predicha por el modelo realizado, obtendremos la matriz de confusión, usando la función `confusionMatrix()` del paquete `caret`.

```
evaltestnolineal10CV <- table(predmodelonolineal10CV, dftest$Y)
(confmatriznolineal <- confusionMatrix(evaltestnolineal10CV))
```

```
Confusion Matrix and Statistics


predmodelonolineal10CV 1 2 3 4
                     1 6 0 9 1
                     2 0 9 0 0
                     3 0 0 3 0
                     4 0 0 0 6

Overall Statistics

               Accuracy : 0.7059
                 95% CI : (0.5252, 0.849)
    No Information Rate : 0.3529
    P-Value [Acc > NIR] : 2.984e-05

                  Kappa : 0.6226

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            1.0000   1.0000  0.25000   0.8571
Specificity            0.6429   1.0000  1.00000   1.0000
Pos Pred Value         0.3750   1.0000  1.00000   1.0000
Neg Pred Value         1.0000   1.0000  0.70968   0.9643
Prevalence             0.1765   0.2647  0.35294   0.2059
Detection Rate         0.1765   0.2647  0.08824   0.1765
Detection Prevalence   0.4706   0.2647  0.08824   0.1765
Balanced Accuracy      0.8214   1.0000  0.62500   0.9286
```

Podemos observar que la precisión global fue 0.70, como se mostró en el resumen del modelo; y, además, observamos una alta tasa de error en la clasificación del tipo de tumor 3: Nrm, el cual fue mayormente confundido con el tipo 1: B-like. Esto afecta a la sensibilidad del modelo en la predicción del tipo de tumor 3, disminuyéndola a 0.25. En menor medida, el modelo confunde características del tumor tipo 4: Lum-B.C, con las del tumor tipo 1: B-like.
Si analizamos cómo serían las matrices de confusión para los modelos en que usamos 3, 5-fold crossvalidation y boostrap, podremos notar que son las mismas y que, por tanto, la precisión máxima alcanzada es la del modelo con 10-fold crossvalidation, 0.70.

```r
# modelo no lineal 3-fold cv
confmatriz3CV <- confusionMatrix(table(predict(modelonolineal3CV, dftest), dftest$Y))$table
# modelo no lineal 5-fold cv
confmatriz5CV <- confusionMatrix(table(predict(modelonolineal5CV, dftest), dftest$Y))$table
# modelo no lineal bootstrap
confmatrizBS <- confusionMatrix(table(predict(modelonolinealBS, dftest), dftest$Y))$table
# Mostrando las matrices de confusi$\'{o}$n
list(print("3-fold CV"),confmatriz3CV,
     print("5-fold CV"),confmatriz5CV,
     print("Bootstrap"),confmatrizBS)
```

```
[1] "3-fold CV"
[1] "5-fold CV"
[1] "Bootstrap"

[[1]]
[1] "3-fold CV"

[[2]]

   1 2 3 4
 1 6 0 9 1
 2 0 9 0 0
 3 0 0 3 0
 4 0 0 0 6

[[3]]
[1] "5-fold CV"

[[4]]

   1 2 3 4
 1 6 0 9 1
 2 0 9 0 0
 3 0 0 3 0
 4 0 0 0 6

[[5]]
[1] "Bootstrap"

[[6]]

   1 2 3 4
 1 6 0 9 1
 2 0 9 0 0
 3 0 0 3 0
 4 0 0 0 6
```

3 Discusion

Los resultados nos muestran que los tipos de cáncer presentan un tipo de distinción lineal, es decir, el hiperplano que separa a los grupos es lineal, por lo que, el modelo lineal resultó con mayor precisión, 0.97.

Así también, las medidas de sensibilidad y especificidad para todos los tipos de cáncer estuvieron por encima de 0.9. Por su parte, el modelo no lineal, indistintamente del tipo de control empleado, crossvalidation o Bootstrap presentó los mismos resultados al momento de la predicción.

El modelo lineal resultó muy útil, pese a que el tipo de tumor 3: Nrm, presenta características que hacen que se confunda con el tipo de cáncer 4: Lum-B.C o 1: B-like, como pudo observarse en el gráfico 2; no obstante el modelo no lineal, predijo de forma deficiente al tipo de cáncer 3: Nmr, obteniéndose que a más de la mitad de los casos con este tipo de tumor los clasificó como 1: B-like (Ver sección).

References

Lantz, Brett. 2015. *Machine Learning with r Discover How to Build Machine Learning Algorithms, Prepare Data, and Dig Deep into Data Prediction Techniques with r.* Packt publishing ltd.