SVM

Daniela Cuesta - Paola Peralta Flores - Mariuxi Tenesaca

Paso 1 - Recopilar Datos y Transformar

```
libraries <- c("reshape2", "ggplot2", "kernlab" ,"caret")
check.libraries <- is.element(libraries, installed.packages()[, 1])==FALSE
libraries.to.install <- libraries[check.libraries]
if (length(libraries.to.install!=0)) {
  install.packages(libraries.to.install)
}

success <- sapply(libraries,require, quietly = FALSE,  character.only = TRUE)
```

```
Loading required package: reshape2

Loading required package: ggplot2

Loading required package: kernlab


Attaching package: 'kernlab'

The following object is masked from 'package:ggplot2':

    alpha

Loading required package: caret

Loading required package: lattice
```

```
if(length(success) != length(libraries)) {stop("A package failed to return a
                                                success in require() function.")}
```

- Se carga un archivo CSV en el objeto de datos "datos" utilizando la función read.csv().

```
datos<-read.csv("./datos/cancer.csv")
```

- Se convierte la columna "diagnostico" en la variable "datos" en un factor utilizando la función "as.factor()" y guarda los resultados en la misma variable "datos".
- Se crea una nueva variable llamada "clase" y almacena los valores de la columna "diagnostico" de la variable "datos":

```
datos$diagnostico <- as.factor(datos$diagnostico)
clase <- datos$diagnostico
```

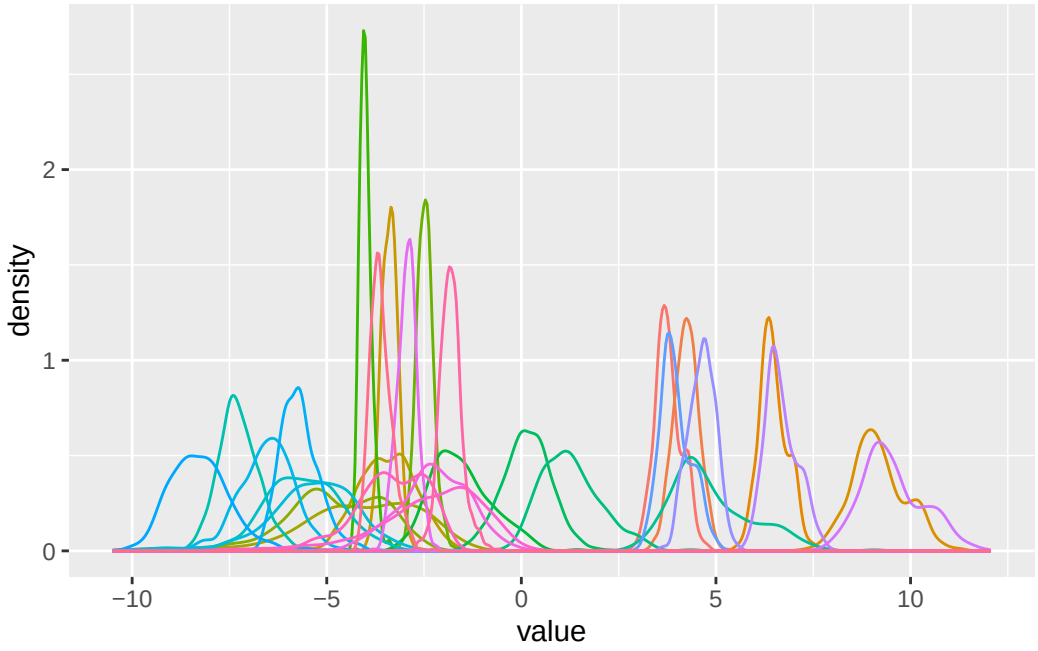
- Selecciona las columnas de "datos" que van desde la columna 2 hasta la columna 11 y las almacena en la variable "Columnas", se crea una nueva variable llamada "X" que contiene solo las columnas seleccionadas.
- Se aplica la función "log2()" a los valores de "X" y realiza el derretimiento de los datos utilizando la función "melt()". El resultado se almacena en la variable "X.melt".
- Se crea un objeto de gráfico utilizando "ggplot()" y establece "value" en el eje x y "variable" en el color de los datos. Los datos utilizados son los de "X.melt".

```
Columnas<-colnames(datos)[2:31]
X<-datos[,Columnas]
X.melt<-melt((log2(X)))
```

```
No id variables; using all as measure variables
```

```
p <- ggplot(aes(x=value,colour=variable), data=X.melt)
p + geom_density(show.legend = F)
```

```
Warning: Removed 78 rows containing non-finite values (`stat_density()`).
```



Interpretación

El eje x del gráfico está representado por "value", lo que significa que muestra los valores de las variables seleccionadas en el eje x. El eje y del gráfico muestra la densidad de probabilidad, ya que se utiliza la función geométrica "geom_density()" para representar las curvas de densidad.

Cada curva de densidad representa la distribución de valores para una variable específica. La variable correspondiente a cada curva se identifica por el color de la curva, ya que se utiliza `"colour = variable"` en la estética del gráfico.

Al superponer las curvas de densidad, se puede visualizar la distribución y la forma de las variables seleccionadas. Las áreas más altas y más anchas en las curvas de densidad indican una mayor concentración de valores en esos rangos. Mientras que las áreas más bajas y más estrechas indican una menor concentración de valores.

En la primera línea, se realizará la transformación logarítmica a cada valor en "X".

Luego, en la segunda línea, se combina "X.log" y "clase" utilizando "cbind()" para crear una nueva matriz o data frame llamado "datos.log". En "datos.log", "X.log" se ubicará en las primeras columnas, seguido por la columna "clase".

```
X.log<-log2(X)
datos.log<-cbind(X,clase)
```

```r
class(datos.log)
```

```
[1] "data.frame"
```

Paso 2 - Dividir los datos en Entrenamiento y Prueba

- Se selecciona un subconjunto de columnas del conjunto de datos **datos**, asigna esos nombres de columna a una variable y luego crea un nuevo conjunto de datos solo con esas columnas seleccionadas. Además, convierte la columna "diagnostico" en un factor

```r
DatoscnDiagnostico<-colnames(datos[1:31])
DatoscnDiagnostico<-datos[,DatoscnDiagnostico]
DatoscnDiagnostico$diagnostico<-as.factor(DatoscnDiagnostico$diagnostico)
```

- **set.seed(123456)**: Esta línea establece una semilla para garantizar la reproducibilidad de los resultados aleatorios.
- **DatoscnDiagnostico[,2:31] <-as.data.frame(scale(DatoscnDiagnostico[,2:31]))**: Las columnas 2 a 31 del conjunto de datos se seleccionan utilizando **[,2:31]** y luego se escalan utilizando la función **scale()**.
- **levels(DatoscnDiagnostico) <- c("B","M")**: En esta línea se establecen los niveles de la variable "diagnostico" en **DatoscnDiagnostico** como "B" y "M".
- **Entrenamiento <- sample(nrow(DatoscnDiagnostico), size = nrow(DatoscnDiagnostico) * 0.7)**: Se realiza una muestra aleatoria de filas del conjunto de datos **DatoscnDiagnostico** para crear un conjunto de entrenamiento. **size = nrow(DatoscnDiagnostico) * 0.7** indica el tamaño deseado del conjunto de entrenamiento (70% del total de filas).
- **Entrenamiento.data <- DatoscnDiagnostico[Entrenamiento,]** y **Prueba.data <- DatoscnDiagnostico[-Entrenamiento,]**: Estas líneas crean conjuntos de datos separados para entrenamiento y prueba.

```r
set.seed(123456)

DatoscnDiagnostico[,2:31]<-as.data.frame(scale(DatoscnDiagnostico[,2:31]))

levels(DatoscnDiagnostico)<-c("B","M")
Entrenamiento<-sample(nrow(DatoscnDiagnostico),size=nrow(DatoscnDiagnostico)*0.7)

Entrenamiento.data<-DatoscnDiagnostico[Entrenamiento,]
```

```
Prueba.data<-DatoscnDiagnostico[-Entrenamiento,]
```

Paso 3 - Entrenamiento

El código crea dos clasificadores SVM, uno con un kernel lineal y otro con un kernel Gaussiano, utilizando los datos de entrenamiento en **`Entrenamiento.data`**.

- Estos clasificadores pueden usarse para predecir la variable "diagnostico" en nuevos datos basándose en las características del conjunto de datos de entrenamiento.

```
clasifier.lineal <- ksvm(diagnostico ~ ., data = Entrenamiento.data,
                         kernel = "vanilladot")
```

```
 Setting default kernel parameters
```

```
clasifier.lineal
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Linear (vanilla) kernel function.

Number of Support Vectors : 33

Objective Function Value : -21.2999
Training error : 0.012563
```

Interpretación

El modelo tiene un total de 33 vectores de soporte, que son los puntos de datos más cercanos a la frontera de decisión. Estos vectores de soporte son los puntos clave para la clasificación y se utilizan para definir la frontera de decisión.

El valor de la función objetivo del modelo es de -21.2999 y el error de entrenamiento obtenido es de 0.012563, lo que indica que el modelo ha logrado una buena capacidad de clasificación en los datos de entrenamiento.

```r
clasifier.gauss <- ksvm(diagnostico ~ ., data = Entrenamiento.data, kernel = "rbfdot")

clasifier.gauss
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.0405393061667693

Number of Support Vectors : 107

Objective Function Value : -47.3373
Training error : 0.015075
```

Interpretación

El modelo tiene un total de 107 vectores de soporte, que son los puntos de datos más cercanos a la frontera de decisión. Estos vectores de soporte son los puntos clave para la clasificación y se utilizan para definir la frontera de decisión.

El hiperparámetro sigma del kernel Gaussiano tiene un valor de 0.0405393061667693. Este parámetro controla la flexibilidad del modelo y afecta la forma de la frontera de decisión.

El valor de la función objetivo del modelo es de -47.3373 y el error de entrenamiento obtenido es de 0.015075, lo que indica que el modelo ha logrado una buena capacidad de clasificación en los datos de entrenamiento.

Paso 4 - Evaluación del Rendimiento del Modelo

Se utilizan dos clasificadores, uno de tipo linear y otro gaussiano.

- El **`clasifier.lineal`** realiza una predicción en **`Entrenamiento.data`**. La función **`predict()`** devuelve las predicciones realizadas por el clasificador en base a los datos de entrenamiento.
- Creación de la tabla de contingencia para la predicción lineal comparando las predicciones de **`prediction.linear`** con **`Entrenamiento.data$diagnostico`**.

```r
prediction.linear<-predict(clasifier.lineal,Entrenamiento.data)
res.linear<-table(prediction.linear,Entrenamiento.data$diagnostico)

(cmatrix1 <- confusionMatrix(res.linear, diagnostico="t"))
```

```
Confusion Matrix and Statistics


prediction.linear   B   M
                B 248   5
                M   0 145

               Accuracy : 0.9874
                 95% CI : (0.9709, 0.9959)
    No Information Rate : 0.6231
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9731

 Mcnemar's Test P-Value : 0.07364

            Sensitivity : 1.0000
            Specificity : 0.9667
         Pos Pred Value : 0.9802
         Neg Pred Value : 1.0000
             Prevalence : 0.6231
         Detection Rate : 0.6231
   Detection Prevalence : 0.6357
      Balanced Accuracy : 0.9833

       'Positive' Class : B
```

Interpretación

- Se guarda la matriz de confusión en la variable **`cmatrix1.`** Se llama a la función **`confusionMatrix()`** con dos argumentos. El primero, **`res.linear`**, es la tabla de contingencia que contiene las predicciones. El segundo, **`diagnostico = "t"`**, indica que la clase "t" se considera la clase positiva en la matriz de confusión.

Es útil para evaluar la precisión, sensibilidad, especificidad y otros métricas relacionadas con la clasificación.

- Accuracy: La proporción de instancias clasificadas correctamente sobre el total de instancias. En este caso, el valor es 98%, lo que indica una alta exactitud.
- Sensitivity: La proporción de instancias positivas (clase "M") clasificadas correctamente es de 1.0000, lo que significa que todas las instancias de la clase "M" se clasificaron correctamente.
- Specificity: La proporción de instancias negativas (clase "B") clasificadas correctamente es de 0.9667, lo que indica un alto nivel de precisión en la clasificación de la clase "B".

La matriz de confusión y las estadísticas muestran que el clasificador lineal tiene un rendimiento muy bueno en la clasificación de las clases "B" y "M", con una alta exactitud y valores altos tanto en sensibilidad como en especificidad.

- **`clasifier.gauss`** para realizar una predicción en **`Entrenamiento.data`**. La función **`predict()`** devuelve las predicciones realizadas por el clasificador en base a los datos de entrenamiento.
- Creación de la tabla de contingencia para la predicción gaussiana comparando las predicciones realizadas por el clasificador gaussiano (**`prediction.gauss`**) con las clases reales en los datos de entrenamiento (**`Entrenamiento.data$diagnostico`**).

```
prediction.gauss<-predict(clasifier.gauss,Entrenamiento.data)
res.gauss<-table(prediction.gauss,Entrenamiento.data$diagnostico)

(cmatrix2<-confusionMatrix(res.gauss,diagnostico = "t"))
```

```
Confusion Matrix and Statistics


prediction.gauss   B   M
               B 248   6
               M   0 144

               Accuracy : 0.9849
                 95% CI : (0.9675, 0.9944)
    No Information Rate : 0.6231
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9676

 Mcnemar's Test P-Value : 0.04123
```

```
            Sensitivity : 1.0000
            Specificity : 0.9600
         Pos Pred Value : 0.9764
         Neg Pred Value : 1.0000
             Prevalence : 0.6231
         Detection Rate : 0.6231
   Detection Prevalence : 0.6382
      Balanced Accuracy : 0.9800

       'Positive' Class : B
```

Interpretación

De igual manera, tomamos en cuenta los tres puntos anteriores:

- Accuracy: La exactitud global con el clasificador gaussiano fue de un 98.49 % (predicciones realizadas de forma correcta), que es bastante buena.
- Specificity: Representa el porcentaje de casos negativos reales que se identificaron correctamente como casos negativos. El porcentaje que identificó de manera correcta fue de 96,00 %.
- Sensitivity: Indica el procentaje de casos positivos que fueron identificados de manera correcta como tal. Es decir, tuvo un 1000 de aciertos en casos positivos.

La matriz de confusión y las estadísticas muestran que el clasificador gaussiano tiene un rendimiento muy bueno, sin embargo, fue menor que el clasificador lineal.

Paso 5 - Validación Cruzada 5 veces

- Se entrena el modelo SVM lineal (`model.5v.linear`). Se utiliza la función `train()` para entrenar el modelo. El argumento `method='svmLinear'` indica que se utilizará un SVM lineal.
- El argumento `trControl` especifica que se realizará una validación cruzada con 5 pliegues utilizando `trainControl()`.
- Los argumentos `tuneGrid` y `tuneLength` se establecen en `NULL` y `10` respectivamente, lo que significa que no se realizará una búsqueda de hiperparámetros y se probarán 10 combinaciones de parámetros.
- El argumento `trace` se establece en `FALSE` para desactivar la salida de información de seguimiento durante el entrenamiento.

- Se utiliza la función **summary()** para mostrar un resumen del modelo entrenado..
- Se utiliza la función **predict()** para realizar predicciones utilizando el modelo entrenado.
- Se genera una tabla de contingencia que muestra la cantidad de predicciones correctas e incorrectas para cada clase.
- Se realiza una **confusionMatrix()** a partir de la tabla de contingencia generada en el paso anterior. El argumento **diagnostico="t"** indica que la clase "t" se considera la clase positiva en la matriz de confusión.

```
# Modelo 5-validación cruzada
model.5v.linear <- train(diagnostico ~ ., Entrenamiento.data,
                         method='svmLinear',
                         trControl=trainControl(method='cv', number=5),
                         tuneGrid= NULL, tuneLength=10,
                         trace = FALSE)

# plot(model.5v, alpha=0.6)
summary(model.5v.linear)
```

```
Length  Class   Mode
     1   ksvm     S4
```

```
prediction <- predict(model.5v.linear, Entrenamiento.data) #predict
res.linear.2<-table(prediction, Entrenamiento.data$diagnostico) #compare

# predecir también puede devolver la probabilidad para cada clase:
confusionMatrix(res.linear.2, diagnostico="t")
```

```
Confusion Matrix and Statistics


prediction   B   M
         B 248   5
         M   0 145

               Accuracy : 0.9874
                 95% CI : (0.9709, 0.9959)
    No Information Rate : 0.6231
    P-Value [Acc > NIR] : < 2e-16
```

```
                 Kappa : 0.9731

 Mcnemar's Test P-Value : 0.07364

            Sensitivity : 1.0000
            Specificity : 0.9667
         Pos Pred Value : 0.9802
         Neg Pred Value : 1.0000
             Prevalence : 0.6231
         Detection Rate : 0.6231
   Detection Prevalence : 0.6357
      Balanced Accuracy : 0.9833

       'Positive' Class : B
```

- Se realiza el mismo proceso anterior con diferencia de qque ahora se entrena el modelo SVM radial. El argumento **`method='svmRadial'`** indica que se utilizará un SVM radial.
- Se calcula una matriz de confusión para sus respectivos valores de accuracy, sensitivity y specificity.

```r
# Modelo 5-validación cruzada
model.5v.radial <- train(diagnostico ~ ., Entrenamiento.data,
                         method='svmRadial',
                         trControl=trainControl(method='cv', number=5),
                         tuneGrid= NULL, tuneLength=10 ,trace = FALSE)

# plot(model.5v, alpha=0.6)
summary(model.5v.radial)
```

```
Length  Class   Mode
     1   ksvm     S4
```

```r
prediction <- predict(model.5v.radial, Entrenamiento.data)
res.radial2<-table(prediction, Entrenamiento.data$diagnostico)

# predecir también puede devolver la probabilidad para cada clase:
confusionMatrix(res.radial2, diagnostico="t")
```

```
Confusion Matrix and Statistics


prediction   B   M
         B 248   6
         M   0 144

               Accuracy : 0.9849
                 95% CI : (0.9675, 0.9944)
    No Information Rate : 0.6231
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9676

 Mcnemar's Test P-Value : 0.04123

            Sensitivity : 1.0000
            Specificity : 0.9600
         Pos Pred Value : 0.9764
         Neg Pred Value : 1.0000
             Prevalence : 0.6231
         Detection Rate : 0.6231
   Detection Prevalence : 0.6382
      Balanced Accuracy : 0.9800

       'Positive' Class : B
```

- Se realiza un proceso de entrenamiento y evaluación SVM lineal utilizando el método de remuestreo Bootstrap.
- El método de remuestreo utilizado por defecto es Bootstrap, que implica tomar muestras aleatorias con reemplazo del conjunto de datos original. El número de repeticiones se establece en 25 por defecto.

```r
### BOPTSTRAP

# Por defecto es Bootstrap, con 25 repeticiones para 3 posibles decay
# y 3 posibles sizes
model.bootstrap.linear <- train(diagnostico ~ .,
                                Entrenamiento.data, method='svmLinear',
                                trace = FALSE) # train
```

```r
# agregamos el parámetro 'preProc = c("center", "scale"))'
# en train() para centrar y escalar los datos
summary(model.bootstrap.linear)
```

```
Length  Class   Mode
     1   ksvm     S4
```

```r
prediction <- predict(model.bootstrap.linear, Entrenamiento.data)
res.gauss.2<-table(prediction, Entrenamiento.data$diagnostico)

confusionMatrix(res.gauss.2,diagnostico="t")
```

```
Confusion Matrix and Statistics


prediction   B   M
         B 248   5
         M   0 145

               Accuracy : 0.9874
                 95% CI : (0.9709, 0.9959)
    No Information Rate : 0.6231
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9731

 Mcnemar's Test P-Value : 0.07364

            Sensitivity : 1.0000
            Specificity : 0.9667
         Pos Pred Value : 0.9802
         Neg Pred Value : 1.0000
             Prevalence : 0.6231
         Detection Rate : 0.6231
   Detection Prevalence : 0.6357
      Balanced Accuracy : 0.9833

       'Positive' Class : B
```

Interpretación

En los tres casos de este último paso, se oberva un **accuracy** por encima del 90% de aciertos en su clasificación. De igual manera se obtienen valores bastante altos para el caso de especificidad y sensibilidad, lo que quiere decir que el SVM es muy bueno para realizar estos trabajos.

NOTA: *Se trabajo con un conjuntos de datos ya utilizado con anterioridad y cabe recalcar que en este caso, la Máquina de Vectores tuvo valores mucho más confiables, tanto en el Accuracy, como en los otros dos datos que se tienen en cuenta al obtener la Matriz de Confusión.*