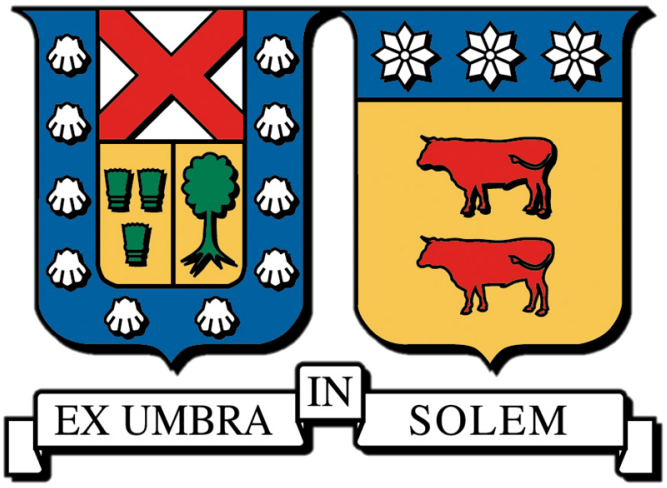


Universidad Técnica Federico Santa María

MAT281: Aplicaciones de la Matemática en Ingeniería

Tarea 2

Profesor:
Alfredo Alegría
Departamento de Matemática

Alumno:
Alejandro Villazón G.
201910009-2

08 de Noviembre del 2021

Problema 1

La siguiente tabla muestra un conjunto de datos que contiene 6 observaciones, 1 covariable y 1 variable respuesta categórica:

x	1	5	10	20	40	80
y	a	b	b	b	b	a

a. Muestre que este conjunto de datos no se puede clasificar de manera perfecta con un clasificador lineal.

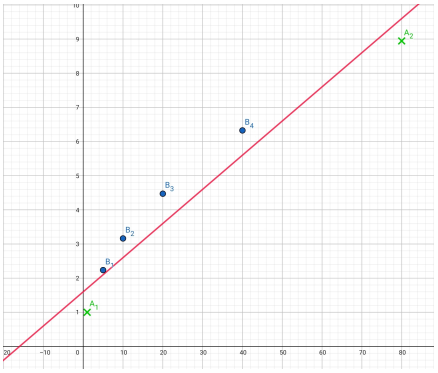
Solución: Supongamos por contradicción que existe un clasificador lineal perfecto, entonces existen constantes $c, d \in \mathbb{R}$ tal que $H(x) = cx + d$ es un clasificador lineal perfecto, si consideramos sin pérdida de generalidad que $a = 1$ y $b = -1$ tenemos las siguientes desigualdades:

$$\begin{array}{rcl} c + d & > & 0 \\ 5c + d & < & 0 \\ 10c + d & < & 0 \\ 20c + d & < & 0 \\ 40c + d & < & 0 \\ 80c + d & > & 0 \end{array}$$

Si $c = 0$ tendremos que $0 < d < 0$ lo cual es una contradicción, si $c > 0$ de la primera y segunda desigualdad obtenemos que $5 < -d/c < 1$ lo cual es una contradicción, por último si $c < 0$ de la segunda y última desigualdad obtenemos que $80 < -d/c < 5$ lo cual es una contradicción. Por lo tanto, no se puede clasificar de manera perfecta con un clasificador lineal a este conjunto de datos.

b. Pruebe que al expandir el espacio de la covariable a través del mapeo $\phi : \mathbb{R} \to \mathbb{R}^2$ dado por $\phi(x) = (x, \sqrt{x})$, si se puede hallar un clasificador lineal perfecto.

Solución: Dado que la raíz cuadrada es una función cóncava, por definición de concavidad podemos encontrar un clasificador lineal perfecto. Es más, si consideramos cualquier recta que pase por los puntos $(x_1, \sqrt{x_1}), (x_2, \sqrt{x_2})$ con $1 < x_1 < 5$ y $40 < x_2 < 80$ obtendremos un clasificador lineal perfecto.
Por ejemplo, con $x_1 = 4, x_2 = 64$ obtenemos el clasificador $H(\boldsymbol{x}) = (0.1, -1)\boldsymbol{x} + 1.6$, el cual es lineal perfecto pues $H(\phi(1)), H(\phi(80)) > 0$ y $H(\phi(5)), H(\phi(10)), H(\phi(20)), H(\phi(40)) < 0$. Gráficamente:



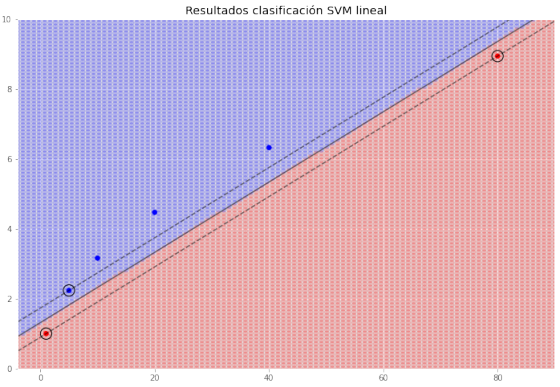
c. Plantee el problema de optimización que permite hallar el clasificador lineal perfecto de margen máximo. Escriba explícitamente la función objetivo y todas las restricciones involucradas.

Solución: Considerando a = 1 y b=−1, buscamos el clasificador lineal perfecto de margen máximo $H(\boldsymbol{x}) = a_0 + \boldsymbol{a}^T\boldsymbol{x}$ con $a_0 \in \mathbb{R}$ y $\boldsymbol{a} \in \mathbb{R}^2$ que resuelve el siguiente problema de optimización:

$$\begin{cases} \min\limits_{\boldsymbol{a}} & \frac{1}{2}\|\boldsymbol{a}\|^2 \\ \text{s.a.} & a_0 + \boldsymbol{a}^T\phi(1) \geq 1 \\ & -a_0 - \boldsymbol{a}^T\phi(5) \geq 1 \\ & -a_0 - \boldsymbol{a}^T\phi(10) \geq 1 \\ & -a_0 - \boldsymbol{a}^T\phi(20) \geq 1 \\ & -a_0 - \boldsymbol{a}^T\phi(40) \geq 1 \\ & a_0 + \boldsymbol{a}^T\phi(80) \geq 1 \end{cases}$$

d. Identifique los vectores soporte en el problema de clasificación del inciso **c.**.

Solución: Gracias a que tenemos herramientas como Python o R que resuelven el problema de clasificación, podemos identificar directamente cuales son los vectores soporte. Al implementar el código adjuntado en el Anexo obtenemos el siguiente gráfico:



de donde podemos identificar que los vectores soporte son $\phi(1), \phi(5)$ y $\phi(80)$.

Problema 2

En un problema de clasificación binario se utilizó la regla de Bayes con 3 umbrales de decisión diferentes, obteniendo las matrices de confusión de más abajo:

	actual	
predicted	0	1
0	22	3
1	22	15

	actual	
predicted	0	1
0	32	6
1	12	12

	actual	
predicted	0	1
0	43	11
1	1	7

a. Para cada uno de estos umbrales, calcule la sensibilidad y la especificidad.

Solución: Considerando positivos a la etiqueta 0, para la primera tabla tenemos que:

$$1 - \text{Especificidad} = \frac{3}{3+15} \implies \text{Especificidad} = \frac{5}{6}, \qquad \text{Sensibilidad} = \frac{22}{22+22} = \frac{1}{2}$$

Para la segunda tabla:

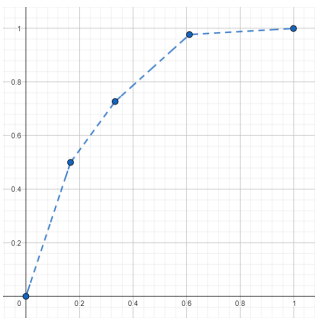
$$1 - \text{Especificidad} = \frac{6}{6+12} \implies \text{Especificidad} = \frac{2}{3}, \qquad \text{Sensibilidad} = \frac{32}{32+12} = \frac{8}{11}$$

Y para la tercera tabla:

$$1 - \text{Especificidad} = \frac{11}{11+7} \implies \text{Especificidad} = \frac{7}{18}, \qquad \text{Sensibilidad} = \frac{43}{43+1} = \frac{43}{44}$$

b. Haga un bosquejo de la curva ROC correspondiente y obtenga el área bajo la curva.

Solución: Dados los umbrales, podemos aproximar la curva ROC por:



con área bajo la curva $A = \frac{101}{132} \approx 0.7652$.

c. ¿Qué se puede decir sobre el desempeño del clasificador?

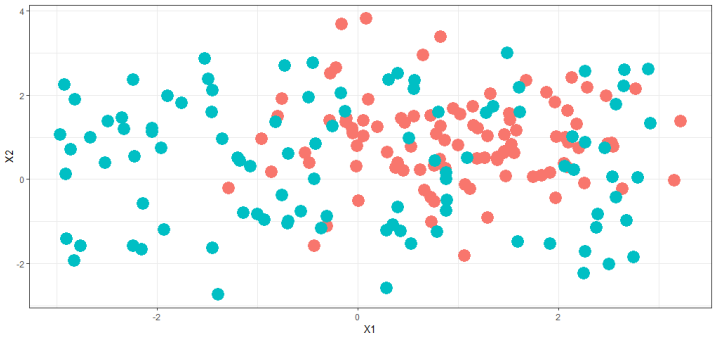
Solución: Dada la aproximación y área obtenida en la parte **b.**, podemos decir que el clasificador es bueno, pues tiene un área bajo la curva superior a 0.75, (criterio encontrado en la web).

Problema 3

Considere el conjunto de datos del archivo `datos_T2.txt` (disponible en AULA), el cual contiene 200 observaciones, 2 covariables (`X1` y `X2`) y 1 variable respuesta categórica (`X3`) . Aplique el método de K vecinos más cercanos para abordar el problema de clasificación. Específicamente, use validación cruzada para evaluar el desempeño del método en términos de K (considere al menos 3 valores distintos de K). Explique claramente el procedimiento usado y sus conclusiones.

Solución:

1. Graficamos el conjunto de datos para tener una idea visual del problema de clasificación al que nos enfrentamos.



con color celeste se muestran las observaciones con `X3`=1. Podemos observar que los puntos naranjos se concentran en una zona, en cambio, los puntos celestes se distribuyen por el rectángulo.

2. Calculamos el(los) mejor(es) valor(es) de K tal que minimice(n) la tasa de error global al aplicar validación cruzada de 10 iteraciones usando el siguiente código en R.

```
library(class)
library(caret)
path = "/Users/Aleja/Documents/6to Sem/MAT281/T2/datos_T2.txt"
datos = read.table(file = path, header = TRUE, sep = " ")
attach(datos)

#Validación cruzada de 10 iteraciones de 20 elementos usando KNN:
folds <- createFolds(X3, k = 10) #creamos 10 subconjuntos de los datos.
K = c()

for(kk in 1:200){
  L_kk <- lapply(folds, function(x){
    training_fold <- datos[-x,]
    test_fold <- datos[x,]
    prediccion <- knn(training_fold[,-3],
                      test_fold[,-3],
                      cl = training_fold[,3],
                      k = kk)
    tab_conf <- table(test_fold[,3], prediccion)
    return((tab_conf[1,2]+tab_conf[2,1])/20) #Tasa de error
  })
  K[kk] = mean(as.numeric(L_kk))#Tasa de error global con K = k
}

order(K)[1:10] #los 10 valores de K con la menor tasa de error global
print(K[order(K)[1:10]]) #los 10 menores valores de tasa de error global
K
```

```
> order(K)[1:10]
 [1] 29 28 37 27 30 33  9 34 35 41
> print(K[order(K)[1:10]])
 [1] 0.245 0.250 0.250 0.255 0.255 0.255 0.260 0.260 0.260 0.260

> K
  [1] 0.350 0.325 0.325 0.340 0.315 0.315 0.280 0.280 0.260 0.295 0.280 0.270 0.265
 [14] 0.265 0.290 0.280 0.290 0.270 0.275 0.265 0.280 0.280 0.275 0.265 0.265 0.270
 [27] 0.255 0.250 0.245 0.255 0.265 0.270 0.255 0.260 0.260 0.275 0.250 0.280 0.265
 [40] 0.265 0.260 0.260 0.265 0.260 0.295 0.280 0.285 0.290 0.310 0.310 0.305 0.320
 [53] 0.315 0.315 0.315 0.310 0.315 0.315 0.320 0.315 0.315 0.315 0.315 0.320 0.310
 [66] 0.310 0.310 0.300 0.315 0.315 0.315 0.310 0.310 0.310 0.310 0.310 0.310 0.305
 [79] 0.315 0.315 0.320 0.315 0.320 0.320 0.325 0.320 0.325 0.325 0.310 0.310 0.320
 [92] 0.325 0.315 0.310 0.315 0.320 0.330 0.325 0.330 0.325 0.325 0.335 0.330 0.340
[105] 0.340 0.340 0.345 0.345 0.350 0.350 0.350 0.350 0.350 0.350 0.355 0.360 0.355
[118] 0.365 0.360 0.370 0.365 0.370 0.365 0.365 0.375 0.380 0.365 0.380 0.370 0.385
[131] 0.375 0.395 0.405 0.415 0.410 0.430 0.435 0.430 0.440 0.440 0.435 0.440 0.435
[144] 0.425 0.425 0.425 0.425 0.425 0.425 0.435 0.425 0.425 0.425 0.420 0.425 0.430
[157] 0.430 0.425 0.420 0.420 0.415 0.430 0.430 0.430 0.430 0.430 0.435 0.460 0.465
[170] 0.460 0.465 0.480 0.470 0.445 0.440 0.445 0.485 0.455 0.480 0.430 0.460 0.560
[183] 0.515 0.465 0.555 0.520 0.540 0.545 0.515 0.510 0.485 0.460 0.490 0.540 0.485
[196] 0.480 0.500 0.450 0.500 0.495
```

De los resultados podemos ver que si elegimos usar este método de clasificación lo mejor es elegir $K = 29$ para este conjunto de datos, ya que al aplicar validación cruzada nos entrega la menor tasa de error global (24.5 %). Al proceder de esta forma, también nos permitió ver como cambia el desempeño del método al variar el valor de K, podemos notar que un valor de K muy grande (mayor a 130) nos eleva considerablemente la tasa de error global a un mínimo de 40 % llegando incluso a 56 %.

Note que en el mejor de los casos la tasa de error global es alta, lo cual nos dice que para este conjunto de datos no es recomendable usar el método de los K vecinos más cercanos pues no tiene un buen desempeño. No obstante, si realmente queremos usar este método, se recomienda considerar $K = 29$ como se mencionó anteriormente o usar $7 \leq K \leq 48$ para obtener una tasa de error global menor al 30 %.

3. Dado los resultados anteriores como extra calcularemos la tasa de error global al aplicar validación cruzada de 10 iteraciones usando otros métodos de clasificación.

   El código utilizado se adjunta en el anexo. Consideramos el método de Regresión Logística el cual nos entregó una tasa de error global del 34.5 %, Naive Bayes con una tasa de error global del 27 % y consideramos también el método SVM con kernel radial el cual nos entregó una tasa de error global del 6 %.

4. Comparemos los diferentes métodos de clasificación basándonos en la tasa de error global para evaluar el desempeño del método KNN en general.

   Note que la Regresión Logística y Naive Bayes no son mejores que el método de los K vecinos más cercanos al usar el mejor valor de K encontrado anteriormente. En cambio, el método SVM con kernel radial reduce considerablemente la tasa de error global obtenida con el método KNN para este conjunto de datos. Por lo tanto, el método KNN no tiene un buen desempeño para este conjunto de datos considerando el desempeño de los otros métodos de clasificación.

Anexo

1.d Puede revisar y ejecutar el código utilizado en el siguiente link de fácil acceso:

`https://colab.research.google.com/drive/1Go1nftZSuIoCylOQW_O6g2nxNu6dsixu?usp=sharing`

3.3

```r
#Código del gráfico
library(ggplot2)
ggplot(data = datos, aes(x = X1, y = X2, color = as.factor(X3))) +
  geom_point(size = 6) +
  theme_bw() +
  theme(legend.position = "none")

#Regresion Logistica
RL <- lapply(folds, function(x){
  training_fold <- datos[-x, ]
  test_fold <- datos[x, ]
  clasificador <- glm(X3 ~ ., family = binomial(link = "logit"),
                      data = training_fold)
  prediccion <- ifelse(predict(clasificador,
                               type = 'response',
                               newdata = test_fold) > 0.5, 1, 0)
  tab_conf <- table(test_fold[,3], prediccion)
  return((tab_conf[1,2]+tab_conf[2,1])/20)#Tasa de error
})
TEG_RL <- mean(as.numeric(RL))
print(TEG_RL)
```

```
> TEG_RL
[1] 0.345
```

```r
#NAIVE BAYES
library(e1071)
NB <- lapply(folds, function(x){
  training_fold <- datos[-x, ]
  test_fold <- datos[x, ]
  clasificador <- naiveBayes(X3 ~ ., data = training_fold)
  prediccion <- predict(clasificador, newdata = test_fold)
  tab_conf <- table(test_fold[,3], prediccion)
  return((tab_conf[1,2]+tab_conf[2,1])/20)#Tasa de error
})
TEG_NB <- mean(as.numeric(NB))
print(TEG_NB)
```

```
> TEG_NB
[1] 0.27
```

```r
#SVM
SVM <- lapply(folds, function(x){
  training_fold <- datos[-x, ]
  test_fold <- datos[x, ]
  clasificador <- svm(X3 ~ ., data = training_fold,
                      kernel = "radial", scale =FALSE)
  prediccion <- predict(clasificador, newdata = test_fold)
  tab_conf <- table(test_fold[,3], prediccion)
  return((tab_conf[1,2]+tab_conf[2,1])/20)#Tasa de error
})
TEG_SVM <- mean(as.numeric(SVM))
print(TEG_SVM)
```

```
> TEG_SVM
[1] 0.06
```