UNIVERSIDAD TÉCNICA DEL NORTE
FICA-CIERCOM
SISTEMAS EMBEBIDOS
Análisis de Datos

Benavides Wilmer, Farinango Luis, Velasco Angel

14 de enero de 2020

1. Introducción

Si se analiza una base de datos con algoritmos de clasificación ,en la mayoría de los casos existirán valores redundantes que entorpecerán el rendimiento del algoritmo o modelo, así como métodos más efectivos de acuerdo a como se analice los datos antes de la aplicación de dichos algoritmos .Por estas razones existen métodos que indican cuan exacto es un algoritmo o que devuelven los porcentajes de variables que pueden ser consideradas despreciables .

2. Marco Teórico

2.1. Matriz de Confusión

Una Matriz de Confusión es un método de evaluación de un modelo de clasificación, es decir que ayuda a determinar que tan bien clasifica un algoritmo determinado. Para esto se debe contar con los datos presentados en el Cuadro 1.

Cuadro 1: Matriz de confusión

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Donde se muestran los valores observados en las filas de cada tabla y los valores predichos por el algoritmos en las columnas de la misma para un clasificador binario. Las variables demostradas significan lo siguiente:

Verdaderos Positivos donde obtenemos todos los valores positivos que el modelo pudo clasificar correctamente como positivos.

Falsos Negativos donde obtenemos todos los valores positivos que el modelo clasifica incorrectamente como negativos.

Falsos Positivos donde obtenemos todos los valores negativos que el modelo clasifica incorrectamente como positivos.

Verdaderos Negativos donde obtenemos todos los valores negativos que el modelo pudo clasificar correctamente como negativos.

En un caso ideal obtendríamos

$$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \tag{1}$$

pero, este caso se encuentra bastante alejado de la realidad, es decir que si llega a suceder estaríamos cometiendo un error en nuestro análisis.

2.1.1. Especificidad (Especificity

El porcentaje de elementos clasificados negativos en una base de datos de valores únicamente negativos, sigue la siguiente expresión:

$$Especificidad = \frac{VN}{TotalNegativos} \tag{2}$$

2.1.2. Sensibilidad o exhaustividad (Sensitivity, Recall)

Indica todos los valores que nuestro modelo clasificó como positivos en una base de datos de valores únicamente positivos, como se muestra en la Ecuación 3

$$Sensibilidad = \frac{VP}{TotalPositivos} \tag{3}$$

2.1.3. Rendimiento de los algoritmos de clasificación

Para obtener un valor cuantitativo que indique el rendimiento de los algoritmos de clasificación que se utilicen, se introducen la Formulas 8 y 5.

Exactitud (Accuracy)

En general, es el porcentaje de datos que el algoritmo clasifica correctamente, se puede determinar de la siguiente manera:

$$Exactitud = \frac{VP + VN}{FP + FN} \tag{4}$$

Tasa de error (Misclassification Rate)

En general, es el porcentaje de datos que el algoritmo clasifica incorrectamente y se representa de la siguiente forma:

$$Error = \frac{FP + FN}{Total} \tag{5}$$

2.2. Curva ROC

Una curva ROC proporciona una representación de la sensibilidad y especifidad para cada valor umbral, que es invariante mediante transformaciones monótonas a los datos de la variable de decisión y que permite comparar dos o más clasificadores en función de su función discriminante.
Un análisis ROC de sus siglas en ingles Receiver Operating Characteristics, es una metodología desarrollada para analizar un sistema de decisión. Este análisis se basa en las nociones de sensibilidad y especificidad.
Una curva ROC proporciona una representación de la sensibilidad generalmente ubicada en el eje de las “y” ,es decir se muestra los verdaderos positivos.

$$\frac{(VP)}{(VP) + (FN)} \tag{6}$$

Donde VP es el número de verdaderos positivos,FN es el número de falsos negativos.
La especificidad se ubica en el eje de las “x” y representa los falsos positivos .

$$\frac{(FP)}{(FP) + (VN)} \tag{7}$$

Donde FP es el número de falsos positivos,VN es el número de verdaderos negativos.
La curva basa sus dos valores sen función de cada valor umbral ,que es invariante mediante transformaciones monótonas a los datos de la variable de decisión y que permite comparar dos o más clasificadores en función de su función discriminante. Además, en la curva ROC cada uno de los puntos representa un par de sensibilidad/especificidad correspondiente a un umbral de decisión particular.

2.3. Medición AUC

El área bajo la curva ROC (AUC) permite representar en un único valor el rendimiento del clasificador. Esto puede resultar útil para realizar comparativas entre clasificadores.

$$AUC = \frac{1 + TPR - FPR}{2} \tag{8}$$

Donde TPR ES tasa positiva verdadera, FPR tasa de falsos positivos.

AUC representa el grado o medida de separabilidad.Indica cuánto modelo es capaz de distinguir entre clases.

La curva ROC se traza con TPR contra el FPR donde TPR está en el eje y y FPR está en el eje x.

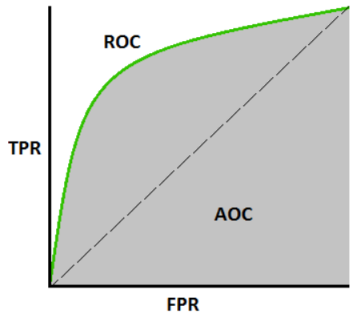


Figura 1: Curva ROC y AUC en funsión de TPR Y FPR

Máximo AUC posible = 1 Un modelo excelente tiene AUC cerca del 1, lo que significa que tiene una buena medida de separabilidad. Un modelo pobre tiene AUC cerca del 0, lo que significa que tiene la peor medida de separabilidad.

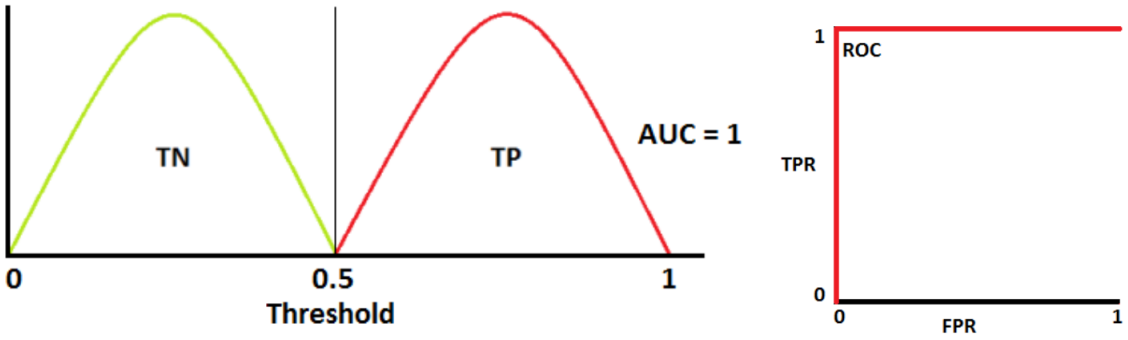


Figura 2: AUC = 1

Cuando AUC es 0.7, significa que hay un 70 por ciento de posibilidades de que el modelo pueda distinguir entre clase positiva y clase negativa.

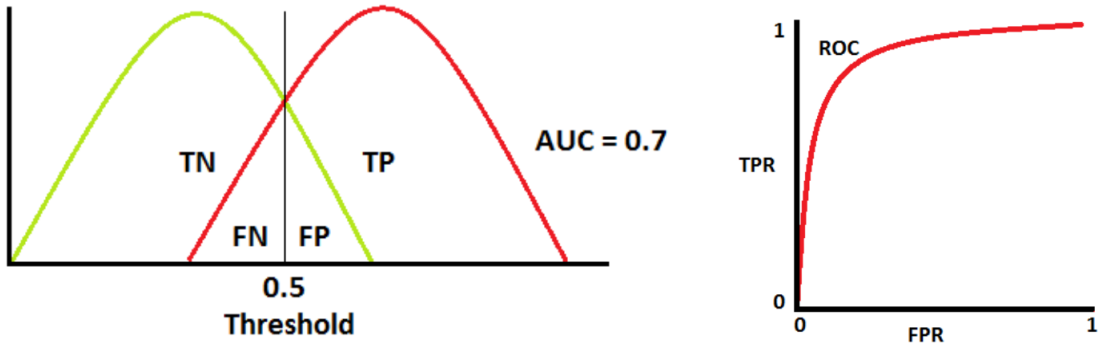
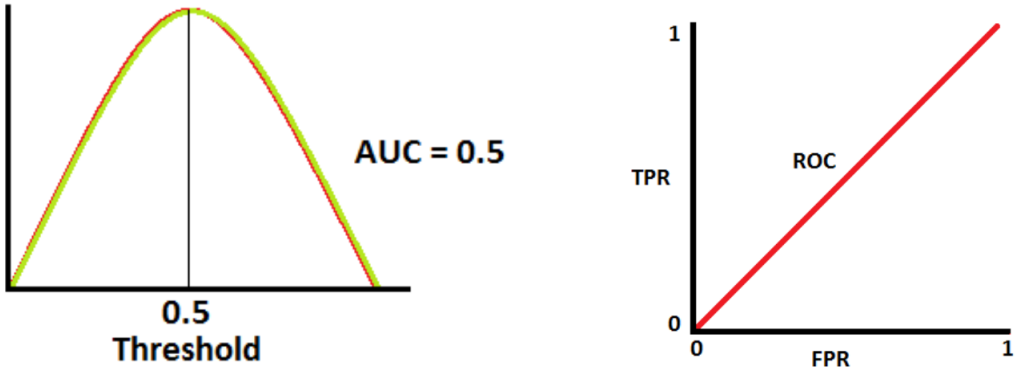


Figura 3: AUC= 0.7

Cuando AUC es 0.5, significa que el modelo no tiene capacidad de separación de clases.

Figura 4: AUC = 0.5



Cuando AUC es aproximadamente 0, el modelo en realidad está correspondiendo las clases. Significa que el modelo predice la clase negativa como una clase positiva y viceversa.

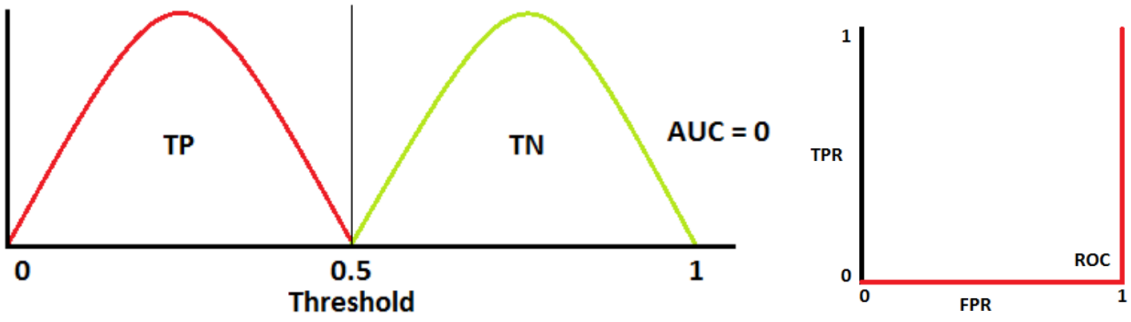


Figura 5: AUC=0

3. Desarrollo

3.1. Si se normaliza la base de datos que rendimiento tiene cada algortimo ?

Para empezare a trabajar debemos preguntarnos lo siguiente:
¿Para qué normalizar o reescalar variables?
Es necesario normalizar las variables ya que en la adquisición de datos estos pueden variar en valores muy altos o muy pequeños, formándose entre ellos unas brechas significativas, por lo cual una solución es ajustar los valores mediante la normalizarlos o reescalar entre (0,1).

A continuación, ser muestra los valores obtenidos mediante el uso de los algoritmos knn y bayesiano usando la base de datos sin normalizar.

```
   y_pred
     1  2  3
  1 39  0  0
  2  0 38  1
  3  0  3 36
> 39+38+36
[1] 113
> 113/117
[1] 0.965812
>
```

Figura 6: Algoritmo K-nn

```
   y_pred
     1  2  3
  1 39  0  0
  2  0 38  1
  3  0  0 39
> 39+38+39
[1] 116
> 116/117
[1] 0.991453
>
```

Figura 7: algoritmo Bayesiano

Donde se puede apreciar que poseen un rendimiento muy significativo, lo que nos indica que los algoritmos funcionan correctamente.
Otra forma de probar el rendimiento es reduciendo los valores de la base de datos, es decir reducir a una matriz con valores de calidad.

```
   y_pred
     1  2  3
  1 39  0  0
  2 11 11 17
  3 12 16 11
> 39+11+11
[1] 61
> 61/117
[1] 0.5213675
>
```

Figura 8: Algoritmo K-nn con matriz reducida

```
   y_pred
     1  2  3
  1 39  0  0
  2  0 38  1
  3  0  0 39
> 39+38+39
[1] 116
> 116/117
[1] 0.991453
>
```

Figura 9: Algoritmo bayesiano con matriz reducida

Donde se puede apreciar que el rendimiento del algoritmo K-nn cae significativamente ya que posee pocos vecinos, por otro lado el bayesiano mantiene su rendimiento.

A continuación, ser muestra los valores obtenidos mediante el uso de los algoritmos knn y bayesiano usando la base de datos normalizada.

```
   y_pred
     1  2  3
  1 39  0  0
  2  0 38  1
  3  0  8 31
> 39+38+31
[1] 108
> 108/117
[1] 0.9230769
>
```

Figura 10: K-nn normalizado

```
   y_pred
     1  2  3
  1 38  0  1
  2  0 39  0
  3  0  4 35
> 38+39+35
[1] 112
> 112/117
[1] 0.957265
>
```

Figura 12: K-nn con matriz reducida y normalizada

```
   y_pred
     1  2  3
  1 39  0  0
  2  2  0 37
  3  0  3 36
> 39+36
[1] 75
> 75/117
[1] 0.6410256
>
```

Figura 11: Bayesiano normalizado

```
   y_pred
     1  2  3
  1 39  0  0
  2 39  0  0
  3 20  5 14
> 39+14
[1] 53
> 53/117
[1] 0.4529915
>
```

Figura 13: bayesiano con matriz reducida y normalizado

Con los datos normalizados existe una mejor en el algoritmo K-nn, donde se mantiene un buen rendimiento. El Bayesiano se reduce significativamente lo cual demuestra que dependiendo de el algoritmo es el rendimiento del modelo.

3.2. ¿Funciona la selección de prototipos?

El uso de prototipos proporciona un entrenamiento al momento de tratar los datos, enfocado al rendimiento que se determina al usar varios algoritmos, los cuales proporcionan los resultados de si la base de datos posee valores útiles.

3.3. Porcentualmente que variable aporta mayor datos al clasificador?

Para el cálculo de las features selection se debe primeramente normalizar la base de datos .Para saber cuál variable aporta un mayor peso en la base de datos se utiliza el paquete boruta ,este paquete devuelve los porcentajes que aportan todas las variables de una base de datos basándose en un factor que en este caso será la etiqueta. La estructura de la función se muestra en la imagen posterior.

```
#Feature Selection
borutaTem=Boruta(Label~ .,data=test_set,doTrace=2)
borutaTem
plot(borutaTem,las=2,cex.axis=0.7)
plotImpHistory(borutaTem)
bor=TentativeRoughFix(borutaTem)
attStats(borutaTem)
```

Figura 14: Codígo en R para el desarrollo de porcentaje de datos

Las variables existentes en la base de datos son 4 ,de acuerdo a los resultados obtenidos ninguna de ellas tiene un peso lo suficientemente bajo para ser eliminada.Los valores a elimiar en este tipo de gráfica deben tener un color rojo ,los valores dudativos un color amarillo y los valores verdes con valores que son importantes para el modelo. Los porcentajes actuales de acuerdo al análisis realizado son :

- PH : 30/100
- TDS: 25/100
- Temperatura:20/100
- Turbidez:15/100
- Valores Sombra:10/100

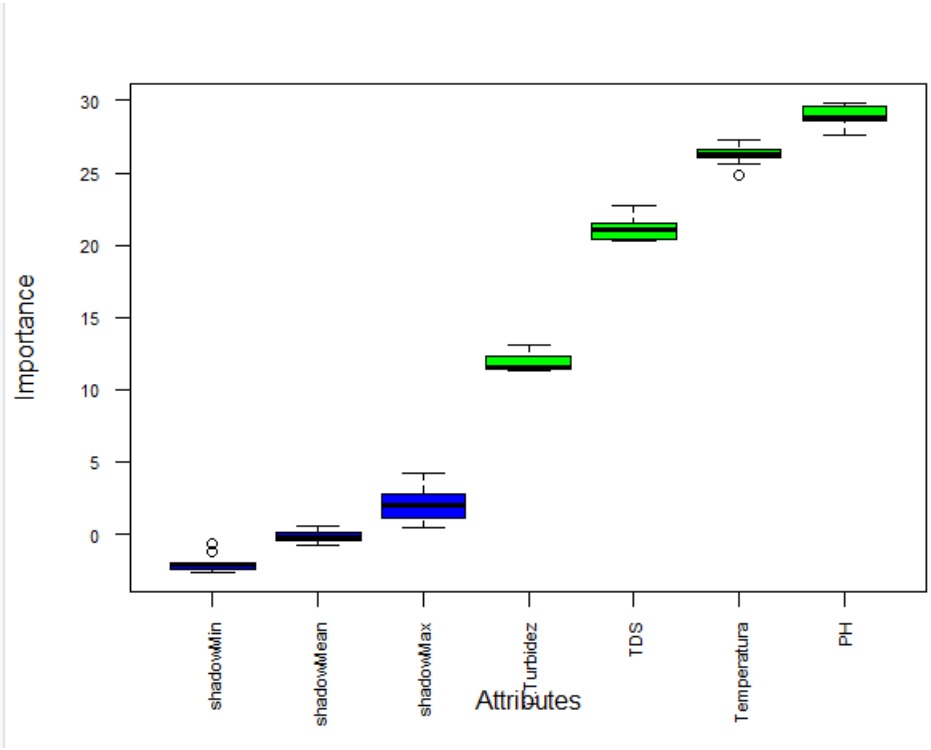


Figura 15: Gráfica obtenida del porcentaje de variables dentro de la base de datos

3.4. Si se elimina la variable de menor importancia qué sucede con el clasificador?

Después de haber obtenido los porcentajes de importancia de cada sensor y determinar que el sensor de Turbidez es el que tiene menor relevancia en la clasificación de los datos, vamos a realizar una prueba del clasificador KNN excluyendo los valores de dicho sensor.
Esto lo realizamos seleccionando las filas utilizadas en el algoritmo, cambiando los corchetes [,-5] por [,2:4], indicando que se tomaran las columnas de la 2 a la 4 como se aprecia en la Figura 16.

```
#Cargar base de datos al entorno de R
datos=read.csv("datos.csv",header =TRUE, sep = ";")
split=sample.split(datos$LABEL,SplitRatio = 0.80)
training_set=subset(datos,split==TRUE)
test_set=subset(datos,split==FALSE)

#########CLASIFICADOR KNN###########

y_pred = knn(train = training_set[,2:4],
             test = test_set[,2:4],
             cl = training_set[,5],
             k = 3)
cm_knn = table(test_set$LABEL,y_pred, dnn = c("Actual", "Predicción"))
cm_knn
```

Figura 16: Código utilizado para aplicar KNN sin valores de turbidez

Como resultados obtenemos que el utilizar o no los valores proporcionados por el sensor de turbidez no hace ninguna diferencia, los resultados obtenidos son exactamente los mismos en ambos casos (Figura 17). Esto quiere decir que es un sensor que bien puede retirarse de nuestro sistema y no influirá en los datos resultantes.

```
> #########CLASIFICADOR KNN CON VALOR DE TURBIDEZ##########
> y_pred = knn(train = training_set[,-5],
+              test = test_set[,-5],
+              cl = training_set[,5],
+              k = 3)
> cm_knn = table(test_set$LABEL,y_pred, dnn = c("Actual", "Predicción"))
> round(prop.table(cm_knn)*100, 2) #Representado en porcentaje
      Predicción
Actual     1     2     3
     1 33.05  0.00  0.00
     2  0.00 28.81  4.24
     3  0.00  3.39 30.51
> #########CLASIFICADOR KNN SIN VALOR DE TURBIDEZ##########
> y_pred = knn(train = training_set[,2:4],
+              test = test_set[,2:4],
+              cl = training_set[,5],
+              k = 3)
> cm_knn = table(test_set$LABEL,y_pred, dnn = c("Actual", "Predicción"))
> round(prop.table(cm_knn)*100, 2) #Representado en porcentaje
      Predicción
Actual     1     2     3
     1 33.05  0.00  0.00
     2  0.00 28.81  4.24
     3  0.00  3.39 30.51
```

Figura 17: Resultados obtenidos con y sin sensor de Turbidez

4. Conclusiones y Recomendaciones

Conclusiones

- El rendimiento de los diferentes algoritmos si se ve afectada si los valores están normalizados especialmente en el algoritmo bayesiano que de 99 por ciento baja a un pendiente del 55 por ciento, lo que permite saber que algoritmo es mejor.
- Según el análisis realizado es bastante difícil la eliminación de algunos datos de la base de datos ,dado que todos presentan un nivel de porcentaje considerable además de que ninguno es representativo por sí mismo.
- Uno de los resultados ha sido que el sensor de Turbidez no afecta en el comportamiento del clasificador, a partir del 15 % de importancia obtenido para este sensor podemos concluir que cualquier tasa de muestras extra con un porcentaje de importancia menor al 15 % no tendrá relevancia en nuestro sistema.

Recomendaciones

* Una buena opción es la de remplazar los datos normalizados en la misma matriz para evitar crear nuevas variables que puedan producir errores debido a una mala interpretación de variables.
* Los algoritmos de clasificación tienden a fallar cuando en la base de datos existe un cambio abrupto de valores, esto puede suceder al haber errores de lectura, por este motivo se recomienda asegurarse de tomar buenas lecturas y, a mayores realizar una revisión de la base de datos y corregir en caso de haber un error.