UADY
UNIVERSIDAD AUTÓNOMA DE YUCATÁN
"Luz, Ciencia y Verdad"

FACULTAD DE MATEMÁTICAS

Minería de Datos

Unidad III: Métodos de Clasificación

ADA 9: *Vecinos más cercanos.*

Licenciatura en Actuaria.

Integrantes:
- Álvarez Herrera Samantha
- Ciau Puga Abigail
- Colonia Espinosa Cindy
- Fernández Caro Frida
- Padilla Jiménez Meybor
- Sobrino Bermejo Samantha

M.C. Ernesto Guerrero Lara

Fecha de entrega: Lunes 11 de mayo de 2020

Los datos se encuentran en la base ppg2008.

La base cuenta con 50 observaciones y 21 variables, pero solo trabajaremos con algunas de ellas.

Nuestra variable de clasificación será llamada Nivel y estará definida de la siguiente manera: Se obtiene dividiendo la columna "PTS" entre la columna "MIN": si el valor es mayor que 0.68 entonces la clase del nivel es "Muy Bueno", si el valor está entre 0.55 y 0.68 inclusive entonces el nivel es "Bueno" y se está por debajo de 0.55 es "Regular"

Y de las 21 variables solo usaremos 4 como predictoras: "FGP" Field goals percentage, "FTP" Free throws percentage, "3PP" 3-points percentage, "G" Games.

Antes de trabajar con los datos, normalizaremos nuestras variables predictoras para que todos los valores queden entre 0 y 1 , pues al ser tres de ellas porcentajes y una de ellas el número de juegos nos puede afectar la escala diferente que tienen, y es preferible trabajar con variables del mismo rango.

Para el tamaño de la muestra de entrenamiento se utiliza 40 datos, por lo que el grupo de prueba será de 10 datos.

Distancia Euclidiana.

Lo primero que haremos es usar la función de R llamada “train.kknn” para encontrar el valor de k óptimo, es decir, el número de vecinos óptimos. Lo anterior se realizará con el siguiente comando de R:

```
train.kknn(Nivel~FGP+FTP+X3PP+G,traindata,distance =2,kernel="optimal")
```

La función anterior arroja la siguiente salida de R:

```
Type of response variable: nominal

Minimal misclassification: 0.5

Best kernel: optimal

Best k: 6
```

Por lo que el valor k óptimo es 6.

Después ajustamos el modelo con k=6 con el siguiente comando.

```
kknn(Nivel~FGP+FTP+X3PP+G, train=traindata, test=testdata,k=6,
distance = 2, kernel = "optimal")
```

La matriz de confusión es:

		Valores Reales		
		Bueno	**Muy Bueno**	**Regular**
Predicción	**Bueno**	**3**	**0**	**0**
	Muy Bueno	**2**	**0**	**0**
	Regular	**0**	**0**	**5**

Obteniendo una exactitud del 80%.

Considerando cada clase como positiva, y las otras dos como negativas, los indicadores de exactitud del modelo quedan como sigue:

Clase positiva / Indicador	Bueno	Muy Bueno	Regular
Sensibilidad	0.6	NA	1.0
Especificidad:	1.0	0.8	1.0
Precisión	1.0	NA	1.0

Se decidió comparar el modelo presentado con los modelos de 1 y 7 vecinos cercanos.

La matriz de confusión del **modelo de 1 vecinos cercanos** es la siguiente:

		Valores Reales		
		Bueno	**Muy Bueno**	**Regular**
Predicción	**Bueno**	**3**	**0**	**0**
	Muy Bueno	**1**	**1**	**0**
	Regular	**1**	**1**	**3**

Se obtuvo una exactitud del 70%. Considerando cada clase como positiva y las otras dos como negativas, los indicadores de exactitud fueron los siguientes:

Clase positiva / Indicador	Bueno	Muy Bueno	Regular
Sensibilidad	0.6000	0.5000	1.0000
Especificidad:	1.0000	0.8750	0.7143
Precisión	1.0000	0.5000	0.6000

La matriz de confusión del **modelo de 7 vecinos cercanos** es la siguiente:

		Valores Reales		
		Bueno	**Muy Bueno**	**Regular**
Predicción	**Bueno**	**3**	**0**	**0**
	Muy Bueno	**2**	**0**	**0**
	Regular	**1**	**0**	**4**

Se obtuvo una exactitud del 70%. Considerando cada clase como positiva y las otras dos como negativas, los indicadores de exactitud fueron los siguientes:

Clase positiva / Indicador	Bueno	Muy Bueno	Regular
Sensibilidad	0.500	NA	1.000
Especificidad:	1.000	0.8	0.833
Precisión	1.000	NA	0.800

Se puede notar que la exactitud con un k=6 es de 80% el cual es mayor a los otros modelos los cuales tuvieron una exactitud de 70%.

Distancia Manhattan.

Lo primero que haremos es usar la función de R llamada "train.kknn" para encontrar el valor de k óptimo, es decir el número de vecinos óptimos. Lo anterior se realizará con el siguiente comando de R.

```
train.kknn(Nivel~FGP+FTP+X3PP+G,traindata,distance = 1,kernel="optimal")
```

La función anterior arroja la siguiente salida deR.

```
Type of response variable: nominal

Minimal misclassification: 0.5

Best kernel: optimal

Best k: 8
```

Por lo que el valor k óptimo es 8.

Después ajustamos el modelo con k=8 con el siguiente comando.

```
kknn(Nivel~FGP+FTP+X3PP+G, train=traindata, test=testdata,k=8,
distance = 1, kernel = "optimal")
```

La matriz de confusión fue la siguiente:

		Valores Reales		
		Bueno	**Muy Bueno**	**Regular**
Predicción	**Bueno**	**3**	**0**	**0**
	Muy Bueno	**1**	**0**	**1**
	Regular	**1**	**0**	**4**

Se obtuvo una exactitud del 70%.

Considerando cada clase como positiva y las otras dos como negativas, los indicadores de exactitud del modelo quedan como sigue:

Clase positiva / Indicador	Bueno	Muy Bueno	Regular
Sensibilidad	0.6	NA	0.8
Especificidad:	1.0	0.8	0.8
Precisión	1.0	NA	0.8

Se decidió comparar el modelo presentado con los modelos de 7 y 9 vecinos cercanos.

La matriz de confusión del **modelo de 7 vecinos cercanos** es la siguiente:

		Valores Reales		
		Bueno	**Muy Bueno**	**Regular**
Predicción	**Bueno**	**3**	**0**	**0**
	Muy Bueno	**1**	**0**	**1**
	Regular	**2**	**0**	**3**

Se obtuvo una exactitud del 60%. Considerando cada clase como positiva y las otras dos como negativas, los indicadores de exactitud fueron los siguientes:

Clase positiva / Indicador	Bueno	Muy Bueno	Regular
Sensibilidad	0.50	NA	0.7500
Especificidad:	1.00	0.8	0.6667
Precisión	1.00	NA	0.6000

La matriz de confusión del **modelo de 9 vecinos cercanos** es la siguiente:

		Valores Reales		
		Bueno	**Muy Bueno**	**Regular**
Predicción	**Bueno**	**3**	**0**	**0**
	Muy Bueno	**1**	**0**	**1**
	Regular	**1**	**0**	**4**

Se obtuvo una exactitud del 70%. Considerando cada clase como positiva y las otras dos como negativas, los indicadores de exactitud fueron los siguientes:

Clase positiva / Indicador	Bueno	Muy Bueno	Regular
Sensibilidad	0.60	NA	0.8
Especificidad:	1.00	0.80	0.8
Precisión	1.00	NA	0.8

En el modelo óptimo, el de 8 vecinos cercanos, y el modelo de 9 vecinos cercanos se obtuvo una exactitud del 70%, el modelo de 7 vecinos se obtuvo un porcentaje menor de exactitud siendo este de 60%.

En la siguiente tabla se comparan las exactitudes de los modelos tanto en la distancia Manhattan como la Euclidiana.

	Distancia Euclidiana			**Distancia Manhattan**		
	k=6	**k=1**	**k=7**	**k=8**	**k=7**	**k=9**
Exactitud	0.80	0.70	0.70	0.70	0.60	0.70
Sensibilidad (Bueno)	0.60	0.60	0.50	0.60	0.50	0.60
Sensibilidad (Muy Bueno)	NA	0.50	NA	NA	NA	NA
Sensibilidad (Regular)	1.00	1.00	1.00	0.80	0.75	0.80
Especificidad (Bueno)	1.00	1.00	1.00	1.00	1.00	1.00
Especificidad (Muy Bueno)	0.80	0.875	0.80	0.80	0.80	0.80
Especificidad (Regular)	1.00	0.7143	0.833	0.80	0.6667	0.80
Precisión (Bueno)	1.00	1.00	1.00	1.00	1.00	1.00
Precisión (Muy Bueno)	NA	0.50	NA	NA	NA	NA
Precisión (Regular)	100	0.60	0.80	0.80	0.60	0.80

Podemos notar que el modelo de 6 vecinos cercanos en distancia Euclidiana es el que tiene el mayor indicador de exactitud siendo este de 80%, mientras que todos los demás modelos tienen en su mayoría un 70% y solo uno tiene 60%. Comparando los demás indicadores se puede observar que en la mayoría, este modelo tiene los indicadores más grandes, por tanto, consideramos que este modelo, **k=6 de distancia Euclidiana**, es el mejor modelo para clasificar.