Actividad 1: Evaluación del método de aprendizaje

Carmen Witsman García | 09/11/2024

📌 El objetivo de este trabajo será evaluar correctamente un modelo de aprendizaje automático en función de si es un problema de clasificación o regresión. Para ello, usaré el software estadístico **RStudio**.

Caso 1: Modelado de número de partidos ganados (Regresión)

1.1. Conjunto de datos

Tenemos un conjunto de datos en formato .sav que importaremos a RStudio con la librería "haven".

- `fotbal.sav` - Contiene datos acerca de la liga de fútbol americano y los resultados proporcionados. Cada observación (28 observaciones) corresponde a un equipo distinto. Las distintas variables con las que trabajaremos son:
  - `y` - Partidos ganados
  - `x2` - Yardas de pase (temporada)
  - `x7` - Porcentaje de jugadas de carrera (jugadas de carrera/jugadas totales)
  - `x8` - Yardas de carrera de los oponentes (temporada)

1.2. Ajuste de modelo de regresión

A partir de la variable `y` (partidos ganados), ajustaremos un modelo de regresión que la relacione con las variables `x2`, `x7` y `x8`. Teniendo en cuenta que el output es numérico (número entero y finito de partidos ganados), estamos ante un problema de regresión.

Lo que buscamos con esto es un modelo con el que podamos predecir el número de partidos ganados a partir de las variables x2,x7 y x8.

> 📚 **Librerías necesarias**: haven, caret

Seguiremos los siguientes pasos:

1. **División del conjunto de prueba**. Para optimizar este modelo, 22 de las 28 observaciones (aproximadamente un 80%) Las usaremos para entrenar al modelo. Las 6 restantes, las almacenaremos en un conjunto de datos de prueba para evaluar qué tan efectivas son nuestras predicciones.
2. **Estandarización**. Como tenemos los datos en diferentes escalas, vamos a aplicar la estandarización a los datos de entrenamiento con la función `scale()`, es decir, restamos a cada observación la media y la dividimos por la desviación estándar. Almacenaremos dichas medidas estadísticas para luego estandarizar con ellas los datos de prueba.
3. **Validación cruzada k-folds**. El conjunto de entrenamiento se divide en 10 subconjuntos, y el modelo se entrenará y evaluará 10 veces, cada vez usando un subconjunto diferente como conjunto de prueba, mientras que los otros 9 subconjuntos se utilizarán para entrenar el modelo. Para ello, usamos la función `trainControl()`.
4. **Entrenamiento del modelo de regresión lineal**. Primero, con `set.seed(123)` aseguramos que cada vez que se ejecute el código, los pliegues se generen de la misma manera. Luego, con la función `train()` de la librería "caret", entrenamos al modelo, pasándole los datos de entrenamiento, especificando en `trControl` la validación cruzada definida anteriormente.

1.3. Evaluación del modelo

1. **Realización de predicciones y RMSE.** Una vez entrenado el modelo, con la función `predict()` hacemos una predicción con el conjunto de prueba, es decir, con las nuevas observaciones que el modelo no ha visto durante el entrenamiento. Para medir el error promedio de las predicciones, usamos la raíz del error cuadrático medio (RMSE), cuya formula matemática es la siguiente:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Figura 1. Fórmula matemática de la raíz del error cuadrático medio

```
# SALIDA
> RMSE en el conjunto de entrenamiento: 0.5056543
> RMSE en el conjunto de prueba para la regresión lineal: 0.438693
```

Como podemos observar, nuestro **RMSE** en el conjunto de prueba es de **0.438**, un mejor resultado que en el conjunto de entrenamiento, lo que indica un **buen ajuste del modelo**. Si este valor fuera muy cercano a 0, indicaría un sobreajuste, y si fuera un valor muy alto, un subajuste.

2. **Interpretación de coeficientes**. Con la función `coef()`, extraemos los coeficientes del modelo de regresión lineal, que indican la contribución de cada variable independiente (x2,x7 y x8) a la predicción de la variable dependiente (y).

```
# SALIDA
> (Intercept)          x2          x7          x8
-2.515652e-16   0.4918794   0.2131023 -0.5288693
```

- **Intercepto (**-2.515652e-16 **):** Es el valor esperado de `y` cuando `x2`, `x7` y `x8` son cero.
- **x2 (**0.4918794**):** Un aumento de una unidad en `x2` se asocia con un aumento de 0.49 unidades en `y`, manteniendo las demás variables constantes. Esto podría indicar que, según nuestro modelo, cuantas más yardas de pase tenga un equipo, más aumenta la probabilidad de ganar el partido.
- **x7 (**0.2131023**):** Un aumento de una unidad en `x7` se asocia con un aumento de 0.21 unidades en `y`, manteniendo las demás variables constantes. Esto podría indicar que, según nuestro modelo, si un equipo hace más jugadas de carrera, tiene alguna posibilidad de ganar el partido, pero no es determinante.
- **x8 (**-0.5288693**):** Un aumento de una unidad en `x8` se asocia con una disminución de 0.53 unidades en `y`, manteniendo las demás variables constantes. Esto podría indicar que, según nuestro modelo, cuantas más yardas de carrera tenga el oponente de un equipo, más probable es que el oponente gane.

De los coeficientes, podemos concluir que las variables de yardas de pase y yardas de carrera del oponente influyen significativamente en los partidos ganados de un equipo. Es decir, cuantas más yardas de pase y de carrera tenga un equipo, más favorable es que dicho equipo gane un partido.

Si realizamos un resumen de nuestro modelo con la función `summary()`, podemos ver a través del p-valor los niveles de significancia de cada variable predictora:

```
# SALIDA
Coefficients:
             Estimate          Std. Error       t value       Pr(>|t|)
(Intercept)    -2.516e-16       0.1192           0.000       1.00000
x2              0.491879        0.1273           3.863       0.00114 **
x7              0.213102        0.1544           1.381       0.18433
x8             -0.528869        0.1541          -3.433       0.00297 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Por lo tanto, finalmente concluimos que las **variables estadísticamente significativas** en nuestro modelo son `x2` y `x8`, con un nivel de significancia de 0.01.

3. **Evaluación de las métricas**. Para ver lo alejados que están los valores reales de los valores ajustados, primero calcularemos el error residual estándar (RSE), que mide la cantidad promedio que los puntos de nuestros datos se desvían de la línea de regresión ajustada. Hacemos uso nuevamente de la función `summary()`.

```
# SALIDA
Residual standard error: 0.559 on 18 degrees of freedom
Multiple R-squared: 0.7321, Adjusted R-squared: 0.6875
F-statistic: 16.4 on 3 and 18 DF,  p-value: 2.179e-05
```

Obtenemos un RSE de 0.559, lo que indica un **buen ajuste** con errores residuales relativamente pequeños. Además, el R-cuadrado múltiple es de 0.7321, lo sugiere que **el modelo explica bien la variabilidad en los datos**, así como el R-cuadrado ajustado por el número de predictores. Finalmente, la estadística F y su valor p indican que el modelo tiene predictores que son significativamente útiles en la predicción de la variable dependiente.

Para ilustrar el RSE, creamos una gráfica Q-Q que muestra la línea de regresión ajustada de nuestro modelo y los valores reales:

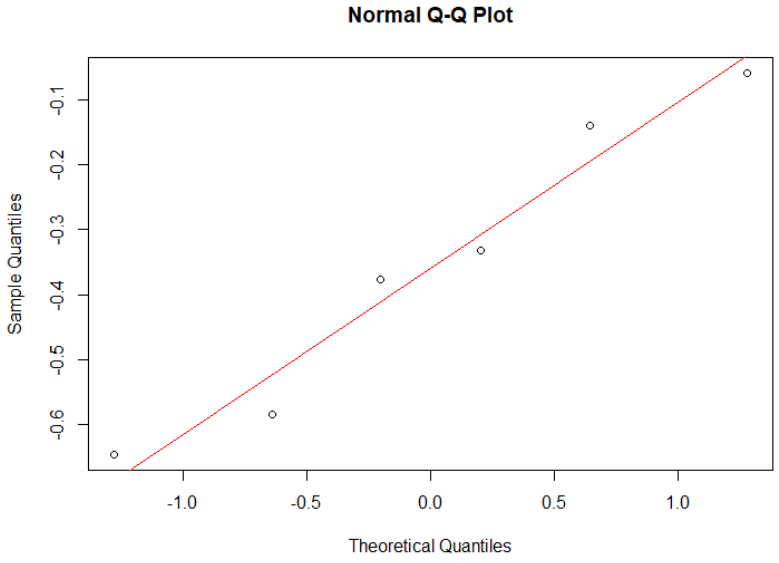

Figura 2. Gráfica Q-Q de los residuos del modelo de regresión lineal.

Esta gráfica nos indica que los residuos del modelo siguen aproximadamente una distribución normal, cumpliendo uno de los supuestos importantes de la regresión lineal. los valores reales están bastante cerca de la línea de regresión ajustada.

> Por lo tanto, podemos concluir que nuestro modelo está bien ajustado, con predicciones precisas y con significancia estadística. Aun así, la variable x7 no es lo suficientemente significativa, por lo que no contribuye de manera útil al modelo.

1.4. Predicción de n° de juegos

Se nos pide que predigamos el número medio de partidos ganados a partir de las variables predictoras:

```
datos_nuevos = data.frame("x2" = 2300, "x7"=56, "x8"=2100)
```

Para ello, las incluiré en el conjunto de datos principal para volver a hacer la estandarización. Pero antes, almacenaré la media y desviación estándar original de la variable `y` para poder deshacer el cambio al obtener nuestra predicción.

Luego de que las variables estén estandarizadas con la media y desviación estándar de los datos del conjunto de entrenamiento, usaré la función `predict()` con estos nuevos datos.

Como el resultado que obtenemos está estandarizado, deshacemos la estandarización, es decir, multiplicamos por la desviación estándar y restamos la media de la variable `y` :

```
num_juegos <- prediccion_nueva * desviacion_estandar_y + media_y

> 7.01397
```

> Conclusión: cuando el equipo tiene 2300 yardas de pase, 56 jugadas de carrera y su oponente tiene 2100 yardas de carrera, el número medio de juegos ganados, según nuestro modelo de regresión lineal, será de unos 7 partidos.

1.5. Nuevo modelo sin x7

Como vimos antes que la variable `x7` no contribuía significativamente al modelo, haremos un nuevo modelo de regresión lineal en el que solo trabajemos con las variables predictoras `x2` y `x8` para predecir `y`.

Volvemos a repetir el proceso descrito anteriormente, y obtenemos los siguientes resultados:

- **Coeficientes** - `x2 = 0.44`, `x8 = -0.63`

  Aumenta la contribución de x8 en el modelo en 0.12 puntos. En este modelo parece tener significativamente más peso la variable x8, mientras que la contribución de x2 solo aumentó 0.04 puntos.

- **Significancia** - `x2 < 0.01 (** )` , `x8 < 0.001 (***)`

  Se confirma que x8 tiene mayor significancia estadística.

- **RSE** - 0.5722

  Aumenta el RSE en este modelo en más de 0.01 puntos, es decir, los residuos se encuentran algo más alejados de la recta de regresión ajustada.

- **RMSE** - 0.576225

  Aumenta el RMSE en 0.14 puntos. Es decir, las predicciones de este modelo son menos precisas.

- **R^2** - 0.7038

  Disminuye R^2 en 0.03 puntos, lo que indica que este modelo explica algo peor la variabilidad en los datos.

> En general, este nuevo modelo es menos preciso que el anterior, pero las variables contribuyen más al modelo, especialmente la de “yardas de carrera del oponente” (x8), que parece significativamente más decisiva a la hora de predecir los partidos ganados de un equipo.

Caso 2: Modelado de concesión de créditos (Clasificación)

2.1. Conjunto de datos

- `tarea5.sav` - Contiene datos de 700 clientes a los que se les negó o no se les negó un crédito. Para esta decisión, se tomaron las siguientes variables socioeconómicas:
  - `age` - Edad en años
  - `ed` - Nivel de educación (1=No completó la educación secundaria, 2=Educación secundaria, 3=Algo de universidad, 4=Universidad, 5=Posgrado)
  - `employ` - Años con el empleador actual
  - `address` - Años en la dirección actual
  - `income` - Ingreso familiar en miles
  - `debtinc` - Relación deuda-ingreso (x100)
  - `creddebt` - Deuda de tarjetas de crédito en miles
  - `othdebt` - Otras deudas en miles
  - `default` - Incumplimiento previo o historial de impago (1=Sí / crédito denegado al cliente, 2= No / crédito concedido al cliente)

2.2. Ajuste de modelo de clasificación 1

En este caso, crearemos un modelo que nos permita clasificar a nuevos clientes para saber si se les puede conceder un crédito o no. Tomaremos la variable `default` como el output del modelo y, por lo tanto, se tratará de un modelo de clasificación, ya que la variable de salida es binaria. En este caso, ajustaremos un modelo de regresión logística.

> 📚 **Librerías necesarias**: tidyverse, broom, caret, haven

Primero, ajustaremos un modelo con todas las variables predictoras que se proporcionan en el conjunto de datos (modelo 1), luego, trabajaremos con las variables más significativas para admitir o denegar concesión del crédito (modelo 2).

Seguiremos los siguientes pasos:

1. **Conversión de output a factor**. Para poder ajustar el modelo de regresión logística en R, debemos asegurarnos de que la variable objetivo sea del tipo factor. La transformaremos usando la función `mutate(default=as.factor(default))`.
2. **División del conjunto de prueba**. Para optimizar este modelo, usaremos un 70% de las observaciones para entrenar al modelo con la función `createDataPartition()`, especificando `p = 0.7`. El 30% restante, lo almacenaremos en un conjunto de datos de prueba para

evaluar qué tan efectivas son nuestras predicciones. Antes, debemos especificar `set.seed(42)` para asegurar la reproducibilidad.

3. **Entrenamiento del modelo con CV**. Ajustamos el modelo con validación cruzada con la función `train()`, usando 10 pliegues:

```
# CÓDIGO
log_model <- train(default ~ ., data = train_data, method = "glm",
                   family = binomial, trControl = trainControl(method = "cv", number = 10))
```

4. **Selección de variables significativas**. Hacemos un filtrado de los p-valores menores a 0.05 para escoger las variables más significativas a la hora de predecir la variable objetivo.

```
 # SALIDA
  term     estimate std.error statistic  p.value
<chr><dbl><dbl><dbl><dbl>
1 employ    -0.250     0.0393     -6.37 1.86e-10
2 creddebt   0.626     0.139       4.51 6.46e- 6
3 address   -0.105     0.0283     -3.72 1.98e- 4
4 debtinc    0.0819    0.0352      2.32 2.01e- 2
```

   Ordenadas de mayor a menor significancia, trabajaremos con las 4 variables más significativas: `employ`, `creddebt`, `address` y `debtinc`.

2.3. Ajuste de modelo de clasificación 2

Teniendo en cuenta el punto anterior, ajustaremos un nuevo modelo de regresión lineal (modelo 2) siguiendo los pasos previamente expuestos, pero esta vez trabajaremos únicamente sobre las variables `employ`, `creddebt`, `address` y `debtinc` para predecir el output `default`.

2.4. Evaluación de modelo 2

Evaluamos las métricas de nuestro modelo de regresión logística tanto para el conjunto de entrenamiento como para el conjunto de prueba.

- **Predicciones y métricas.**

```r
train_preds <- predict(log_model2, newdata = train_data, type = "prob")[,2]
train_preds_class <- ifelse(train_preds > 0.5, 1, 0)
confusionMatrix(as.factor(train_preds_class), train_data$default)
```

Con la función `predict()`, especificando `type="prob"`, se evalúa la probabilidad de que un cliente pertenezca a una clase u otra. Si la probabilidad es mayor que 0.5, se considera que el cliente pertenece a la clase 1, y 0 en el caso contrario.

Elaboramos la matriz de confusión con la función `confusionMatrix()` para obtener las siguientes métricas:

Datos de entrenamiento

```
Accuracy : 0.8147
Sensitivity : 0.9199   # TP / (TP+FN)
Specificity : 0.5194
Precision : 0.843038   # TP / (TP+FP)
F1 Score : 0.8797886   # 2*TP/(TP+(FP+FN)/2)
```

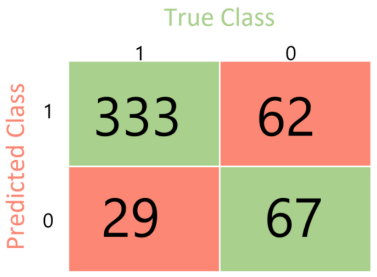

Figura 3. Matriz de confusión del modelo de regresión logística para datos de entrenamiento

Datos de prueba

```
Accuracy : 0.8278
Sensitivity : 0.9548   # TP / (TP+FN)
Specificity : 0.4630
Precision : 0.8361582  # TP / (TP+FP)
F1 Score : 0.8915663   # 2*TP/(TP+(FP+FN)
```

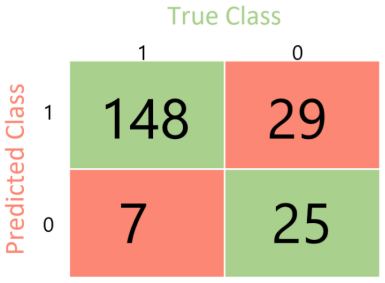

Figura 4. Matriz de confusión del modelo de regresión logística para datos de prueba

- **Conclusiones**.
  1. **Alta sensibilidad (sensitivity)**: El modelo tiene una alta sensibilidad tanto en el conjunto de prueba (0.9548) como en el de entrenamiento (0.9199). Esto significa que el modelo es muy eficaz en la detección de casos de default.
  2. **Especificidad moderada (specificity)**: La especificidad es moderada en ambos conjuntos, con valores de 0.4630 (prueba) y 0.5194 (entrenamiento). Esto indica que

nuestro modelo tiene dificultades para identificar correctamente los casos en los que no hay default, lo que puede llevar a un número significativo de falsos positivos.

3. **Buena precisión y F1 score**: El modelo tiene una buena precisión y un alto F1 score en ambos conjuntos, lo que sugiere un buen equilibrio entre la capacidad del modelo para detectar defaults y la exactitud de sus predicciones.
4. **Exactitud consistente (accuracy)**: La exactitud de nuestro modelo es bastante consistente entre el conjunto de entrenamiento (0.8147) y el de prueba (0.8278), lo que sugiere que el modelo no está sobreajustado (overfitted).

2.5. Predicción de asignación de créditos

A partir de los datos de un nuevo cliente, usaremos nuestro modelo para predecir si, según las variables predictoras usadas en el modelo 2, se le concederá o no un crédito.

Estos datos son:

```
nuevo_cliente <- tibble(
age = 37,
ed = 1,
employ = 20,
address = 13,
income = 41,
debtinc = 12.9,
creddebt = 0.9,
othdebt = 4.49
)
```

Aplicando nuestro modelo:

```
# CÓDIGO
# - predicción
nuevo_cliente_pred <- predict(log_model2, newdata = nuevo_cliente, type = "prob")[,2]
nuevo_cliente_pred_class <- ifelse(nuevo_cliente_pred > 0.5, 1, 0)
# - clasificación
if (nuevo_cliente_pred_class == 1) {
print("No se aprueba el crédito (default)")
} else {
print("Se aprueba el crédito (no default)")
}

# SALIDA
> "Se aprueba el crédito (no default)"
```

Como podemos ver, según nuestro modelo de regresión logística, el nuevo cliente con las características anteriormente descritas es apto para la concesión del crédito.