PRAC2
LIMPIEZA Y ANÁLISIS

Tutora: Meritxell Figueres Boquera
Profesor: José Moreira Sánchez
Alumno: Javier López Calderón y José Maria Cano Hernández
Asignatura: Tipología y ciclo de vida de los datos

Nota Importante. Se aporta un fichero HTML complementario con el detalle del estudio en R. Este documento pretende resumir lo más relevante del estudio realizado, respondiendo a las cuestiones planteadas en el ejercicio práctico.

1.- Descripción del dataset. ¿Por qué es tan importante y qué pregunta/problema pretente responder.

Esta práctica pretende analizar la Calidad del vino tinto a partir de una serie de atributos. Toda la información relativa al DataSet que se ha utilizado, se ha extraído de la siguiente URL: https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009.

La variable objetivo (calidad del vino) forma parte del DataSet.

La descripción de las variables contenidas en el fichero es la siguiente:

- fixed acidity: Acidez fija. Cantidad de ácidos fijos o no volátiles.
- volatile acidity: Acidez volátil. Cantidad de ácido acético en el vino.
- citric acid: ácido cítrico. Encontrado en pequeñas cantidades aporta frescura y sabor a los vinos.
- residual sugar: azúcar residual. Cantidad de azúcar presente tras el proceso de fermentación.
- chlorides: Clorides. Cantidad de sal en el vino.
- free sulfur dioxide: dióxido de sulfuro libre. Cantidad.
- total sulfur dioxide: total de dioxido de sulfuro. Cantidad.
- density: densidad. Densidad de agua dependiente del porcentaje de alcohol y contenido en azúcar.
- pH: describe el nivel de acidez o basal que es un vino en la escala de 0 (muy ácido) a 14 (muy básico).
- sulfatos: sulfatos, aditivo del vino que puede contribuir a los niveles de dioxido de sulfuro.
- alcohol: porcentaje de alcohol en el vino.

Variable de Salida: * quality: Calidad. Variable objetivo. Rango de valores de 0 a 10.

La pregunta que pretende responder es determinar la calidad del vino a partir de los atributos que se disponen. Se trata, por tanto, de un problema supervisado de clasificación en el que se construirán modelos cuyo objetivo sea predecir la calidad del vino tino.

2.-Integración y selección de los datos de interés a analizar.

Se realiza un análisis preliminar del dataset destacando lo siguiente:

- Todos los atributos son numéricos.
- No se identifican valores nulos o NA.
- Se generan las distribuciones de los distintos atributos con respecto a la variable objetivo “quality”, así como la distribución de la variable objetivo con respecto a las distintas variables.
- Como conclusión, nos encontramos que entre las calidades 5 y 6, se concentran la gran mayoría de muestras para los distintos atributos. No se distinguen diferencias importantes en lo que respecta a la distribución de las calidades en función de un atributo u otro.
- En cuanto a la distribución de los valores de los atributos, sí que se identifican valores extremos que será necesario tratar en la gran mayoría de los atributos.

En lo que respecta a integración y selección de los datos destacamos lo siguiente:

- No es necesario llevar a cabo integración alguna.
- Seleccionaremos todos los atributos del conjunto de datos para el estudio.
- En cuanto a la variable objetivo, se opta por añadir una nueva variable “revisionCalidad”, discretizando los valores para simplificar el estudio y mejorar la interpretación de los resultados.
    - Entre 0 y 4, se asigna Calidad Baja.
    - Entre 5 y 6, se asigna Calidad Normal.
    - Entre 7 y 10, se asigna Calidad Excelente.

3.- Limpieza de los datos.

En lo que respecta a la limpieza de los datos, resaltar que no hay que manipular datos nulos o NA; pero es necesario identificar los valores exactos de los outliers que se identificaron visualmente en el análisis preliminar.
Para ello, se construye esta función que nos proporcionará dicha información teniendo en cuenta que son valores con más de 3 desviaciones estándar con respecto a la media del atributo.

> Calculo Outliers Se considera outlier todo valor de un atributo que esté a más de 3 desviaciones estándar de la media.

```
# Función que calcula Outliers a partir de un data frame.
zscore_outlier <- function(df) {
  output <- NULL
  media <- mean(df)
  desvStd <- sd(df)

  for (i in 1:length(df)) {
    if (abs((df[i]-media)/desvStd)>3) {
      output <- c(output,df[i])
    }
  }
  return(output)
}

# Se recorren las columnas relevantes para enumerar los outliers.
for (columna in colnames(dataWine[1:11])) {
  print(sprintf("Atributo: %s",columna))
  list_outlier <- zscore_outlier(dataWine[,columna])
  print(sort(list_outlier))
}
```

La lista de Outliers que se identifican son:

```
## [1] "Atributo: fixed.acidity"
##  [1] 13.7 13.7 13.8 14.0 14.3 15.0 15.0 15.5 15.5 15.6 15.6 15.9
## [1] "Atributo: volatile.acidity"
##  [1] 1.070 1.090 1.115 1.130 1.180 1.185 1.240 1.330 1.330 1.580
## [1] "Atributo: citric.acid"
## [1] 1
## [1] "Atributo: residual.sugar"
##  [1]  7.0  7.2  7.3  7.5  7.8  7.8  7.9  7.9  7.9  8.1  8.1  8.3  8.3  8.3  8.6
## [16]  8.8  8.8  8.9  9.0 10.7 11.0 11.0 12.9 13.4 13.8 13.8 13.9 15.4 15.4 15.5
## [1] "Atributo: chlorides"
##  [1] 0.230 0.235 0.236 0.241 0.243 0.250 0.263 0.267 0.270 0.332 0.337 0.341
## [13] 0.343 0.358 0.360 0.368 0.369 0.387 0.401 0.403 0.413 0.414 0.414 0.415
## [25] 0.415 0.415 0.422 0.464 0.467 0.610 0.611
## [1] "Atributo: free.sulfur.dioxide"
##  [1] 48 48 48 48 50 50 51 51 51 51 52 52 52 53 54 55 55 57 66 68 68 72
## [1] "Atributo: total.sulfur.dioxide"
##  [1] 147 147 147 148 148 149 151 151 152 153 155 160 165 278 289
## [1] "Atributo: density"
##  [1] 0.99007 0.99007 0.99020 0.99064 0.99064 0.99080 0.99084 1.00242 1.00242
## [10] 1.00260 1.00260 1.00289 1.00315 1.00315 1.00315 1.00320 1.00369 1.00369
## [1] "Atributo: pH"
## [1] 2.74 3.78 3.78 3.85 3.90 3.90 4.01 4.01
## [1] "Atributo: sulphates"
##  [1] 1.17 1.17 1.17 1.17 1.17 1.18 1.18 1.18 1.20 1.22 1.26 1.28 1.28 1.31 1.33
## [16] 1.34 1.36 1.36 1.36 1.56 1.59 1.61 1.62 1.95 1.95 1.98 2.00
## [1] "Atributo: alcohol"
## [1] 14.0 14.0 14.0 14.0 14.0 14.0 14.0 14.9
```

Optamos **por eliminar los Outliers obtenidos previamente**, asumiendo que son valores perdidos o tomados erróneamente en las mediciones, con el objeto de que no condicione la solución.
Hay que tener en cuenta que es necesario considerar outliers particulares, así como outliers que están por encima o por debajo de 3 desviaciones estándar en términos absolutos. Así por ejemplo, en densidad y pH, hay valores con más de 3 desviaciones estándar superiores e inferiores a la media, por lo que es necesario contemplarlos ambos.

En cualquier caso, el código de los valores que se eliminan se muestra a continuación:

> Tratamiento Outliers.

Optamos por eliminar los Outliers obtenido previamente, asumiendo que son valores perdidos o tomados erróneamente en las mediciones, con el objeto de que no condicione la solución.

```
dataWineAux <- dataWine
# Eliminamos Outliers.
# Nos basamos en los valores obtenidos y confirmados en apartado anterior.
dataWineAux$fixed.acidity[dataWineAux$fixed.acidity>=13.7] <- NA
dataWineAux$volatile.acidity[dataWineAux$volatile.acidity>=1.07] <- NA
dataWineAux$citric.acid[dataWineAux$citric.acid==1.0] <- NA
dataWineAux$residual.sugar[dataWineAux$residual.sugar>=7.0] <- NA
dataWineAux$chlorides[dataWineAux$chlorides>=0.23] <- NA
dataWineAux$free.sulfur.dioxide[dataWineAux$free.sulfur.dioxide>=48.0] <- NA
dataWineAux$total.sulfur.dioxide[dataWineAux$total.sulfur.dioxide>=147.0] <- NA
dataWineAux$density[dataWineAux$density<=0.99084] <- NA
dataWineAux$density[dataWineAux$density>=1.00242] <- NA
dataWineAux$pH[dataWineAux$pH<=2.74] <- NA
dataWineAux$pH[dataWineAux$pH>=3.78] <- NA
dataWineAux$sulphates[dataWineAux$sulphates>=1.17] <- NA
dataWineAux$alcohol[dataWineAux$alcohol>=14.0] <- NA

# Se omite los NA asignados previamente
dataWineFinal=na.omit(dataWineAux)
dataWineAux <- NULL
```

4.-Análisis de los datos.

Se llevan a cabo las siguientes tareas:

- Chequeo de la Normalidad y Homocedasticidad.

  Según test de Shapiro-Wilk no se puede garantizar que los distintos atributos del DataSet sigan una distribución Normal. Sin embargo, **aplicando el Teorema central del límite** que establece que la media de una muestra de cualquier conjunto de datos es cada vez más normal a medida que aumenta la cantidad de observaciones; y teniendo en cuenta que tenemos un número reducido de muestras, concluimos que los datos sí siguen una distribución normal.

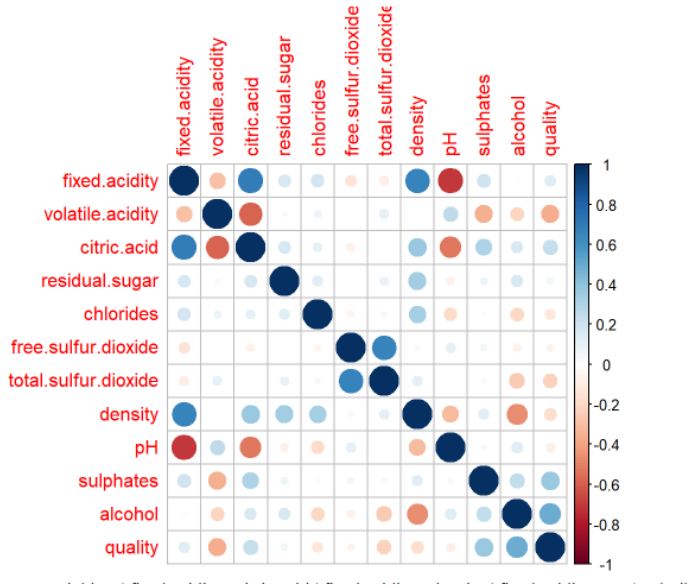
  En lo que respecta a la homogeneidad de la varianza (Levene test), asumiendo que sí es normal aplicando el teorema central del límite, encontramos que las siguientes variables sí tienen la misma homogeneidad de la varianza que la nueva variable discretizada:
  - citric.acid,
  - chlorides,
  - free.sulfur.dioxide,
  - total.sulfur.dioxide,
  - pH,
  - sulphates,
  - alcohol.

- Correlación de las variables o atributos.
  A continuación, se muestra la gráfica que muestra la correlación entre variables.

Correlación de los atributos.

Examinamos la matriz de correlación de las variables númericas del conjunto de datos.

```
library(corrplot)
corr.res<-cor(dataWineFinal[1:12])
corrplot(corr.res,method="circle")
```



Se observa una correlación relevante entre las siguientes variables:

- fixed.acidity y citric.acid,
- fixed.acidity y density,
- fixed.acidity y pH,
- volatile.acidity y citric.acid,
- density y citric.acid,
- citric.acid y pH,
- free.sulfur.dioxide y total.sulfur.dioxide

Sin embargo, no existe una correlación tan significativa que aconseje una reducción de la dimensionalidad, eliminando algún atributo dependiente de otro. Esto implicaría seguramente pérdida de información. Por tanto, continua el análisis manteniendo el mismo número de atributos.

- Preparación del conjunto de datos de entrenamiento y pruebas.
  Se construye un conjunto de datos de entrenamiento formado por los 2/3 del conjunto original una vez que se eliminan outliers; mientras que el conjunto de datos de test corresponde al 1/3 de las muestras restante.

En lo que respecta a algoritmos de clasificación, se consideran los siguientes algoritmos:

- Clustering – K-nearest Neighbour (knn).

Para el algoritmo KNN, emplearemos el paquete Caret.

```
library(caret)

# Se empleará Cross Validation
ctrl <- trainControl(method="repeatedcv",repeats = 3)
knnFit <- train(trainX,trainy, method = "knn", trControl = ctrl, preProcess = c("center","scale"), tuneLength = 20)

knnFit
```

El resultado del algoritmo es el siguiente:

```
## k-Nearest Neighbors
##
## 972 samples
##  11 predictor
##   3 classes: '0-Baja', '1-Normal', '2-Excelente'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 875, 874, 874, 875, 875, 875, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8330439  0.3407574
##    7  0.8364670  0.3374844
##    9  0.8388551  0.3390454
##   11  0.8371542  0.3226681
##   13  0.8372278  0.3070074
##   15  0.8392968  0.2937392
##   17  0.8399597  0.2930574
##   19  0.8361966  0.2754027
##   21  0.8358319  0.2592719
##   23  0.8361967  0.2537311
##   25  0.8330966  0.2308206
##   27  0.8334469  0.2302331
##   29  0.8334506  0.2254532
##   31  0.8313747  0.2145026
##   33  0.8347762  0.2229030
##   35  0.8358214  0.2277380
##   37  0.8375220  0.2215352
##   39  0.8361437  0.2141604
##   41  0.8385461  0.2222324
##   43  0.8375256  0.2105929
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 17.
```

```
predicted_model <- predict( knnFit, testX, type="raw" )
print(sprintf("La precisión de KNN es: %.2f %%",100*sum(predicted_model == testy) / length(predicted_model)))
```

```
## [1] "La precisión de KNN es: 85.42 %"
```

Este algoritmo encuentra la mayor precisión con 17 clusters.

- Árbol de decisión (incluyendo variante de Boosting adaptativo).

> Análisis Arbol Decision

Para el árbol de decisión, obtendremos también las reglas de decisión. Para ello, utilizaremos el paquete C50.

```
trainy = as.factor(trainy)
model_rules <- C50::C5.0(trainX, trainy,rules=TRUE )
summary(model_rules)
```

Los resultados son los siguientes:

```
## Evaluation on training data (972 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##      28   87( 9.0%)   <<
##
##
##     (a)   (b)   (c)    <-classified as
##    ----  ----  ----
##       7    29     1    (a): class 0-Baja
##       1   782    19    (b): class 1-Normal
##            37    96    (c): class 2-Excelente
##
##
##  Attribute usage:
##
##   94.55% alcohol
##   81.48% volatile.acidity
##   80.56% sulphates
##   75.10% free.sulfur.dioxide
##   72.53% residual.sugar
##   71.60% citric.acid
##   56.48% total.sulfur.dioxide
##   55.76% pH
##   24.18% density
##   13.99% chlorides
##    0.72% fixed.acidity
##
```

El árbol de decisión genera 28 reglas de decisión, con lo que consiguen clasificar correctamente el conjunto de entrenamiento salvo en 87 casos (9% de los casos).

Adicionalmente, se indica el uso de los atributos más relevantes en las reglas.

A modo de ejemplo, se muestra una regla que indica los valores de los atributos para conseguir un vino excelente con una probabilidad elevada del 92%

```
## Rule 16: (11, lift 6.7)
##  volatile.acidity > 0.27
##  volatile.acidity <= 0.38
##  free.sulfur.dioxide <= 36
##  total.sulfur.dioxide > 17
##  density <= 0.9975
##  pH <= 3.26
##  sulphates > 0.68
##  alcohol <= 11.6
##  ->  class 2-Excelente  [0.923]
```

La precisión del árbol de decisión es aproximadamente del 86% de los casos del conjunto de prueba.

Adicionalmente, se lleva a cabo un Boost adaptativo para medir la mejora en la predicción del árbol de decisión. Se basa en aplicar diferentes iteraciones de clasificación sobre el mismo conjunto de datos de entrenamiento. El primer clasificador, tal y como hicimos con las reglas de decisión, cometió una serie de

errores sobre los que trabajará el siguiente clasificador en la siguiente iteración, y así sucesivamente hasta que ya no haya mejora significativa o se llegue al número de iteraciones o trial indicado. Cada clasificador tiene su aportación a la hora de determinar la clase que se debe predecir. La gran diferencia, por tanto, es que no es un clasificador sino un conjunto de ellos.

Se realizarán 10 iteraciones del árbol de decisión.

> Tunning: Iteración (Boosting adaptativo) del árbol del decisión.

A continuación, utilizaremos el Boosting adaptativo para intentar mejorar el modelo del árbol de decisión.

Adicionalmente, cargaremos el paquete gmodels, para utilizar Crosstable y obtener más información en la matriz de confusión, concretamente los porcentajes en cada casilla, así como los totales por fila y columna.

```
library(gmodels)

# Generamos modelo Boosted con 10 trial (lo habitual es indicar 10)
model_tree_boosted <- C50::C5.0(trainX, trainy,trial=10)
summary(model_tree_boosted)
```

Resultado

```
## Evaluation on training data (972 cases):
##
## Trial        Decision Tree
## -----      ----------------
##    Size      Errors
##
##    0     62   48( 4.9%)
##    1     36  107(11.0%)
##    2     49  120(12.3%)
##    3     52  107(11.0%)
##    4     54  101(10.4%)
##    5     55  113(11.6%)
##    6     59  108(11.1%)
##    7     55   76( 7.8%)
##    8     53  133(13.7%)
##    9     44  135(13.9%)
## boost              0( 0.0%)   <<
##
##
##     (a)   (b)   (c)    <-classified as
##    ----  ----  ----
##      37                (a): class 0-Baja
##           802          (b): class 1-Normal
##                 133    (c): class 2-Excelente
##
##
##  Attribute usage:
##
##  100.00% volatile.acidity
##  100.00% sulphates
##  100.00% alcohol
##   99.90% residual.sugar
##   99.38% citric.acid
##   93.00% free.sulfur.dioxide
##   91.56% total.sulfur.dioxide
##   91.05% density
##   83.54% chlorides
##   83.44% pH
##   72.33% fixed.acidity
##
```

Con esta variante se consigue un modelo que logra clasificar correctamente las muestras de entrenamiento.

Con respecto al conjunto de test, este modelo logra mejor resultados que los anteriores, ofreciendo porcentaje de precisión del 87.47%.

```
porcentaje_correct<-100 * sum(diag(mat_conf)) / sum(mat_conf)
print(sprintf("El %% de registros correctamente clasificados es: %.4f %%",porcentaje_correct))
```

```
## [1] "El % de registros correctamente clasificados es: 87.4743 %"
```

La matriz de confusión se muestra a continuación (obtenido mediando uso de Crosstable):

```
##
##              | Prediction
##      Reality |      0-Baja |    1-Normal | 2-Excelente |   Row Total |
## -------------|-------------|-------------|-------------|-------------|
##       0-Baja |           1 |          16 |           0 |          17 |
##              |       0.002 |       0.033 |       0.000 |             |
## -------------|-------------|-------------|-------------|-------------|
##     1-Normal |           1 |         388 |          13 |         402 |
##              |       0.002 |       0.797 |       0.027 |             |
## -------------|-------------|-------------|-------------|-------------|
##  2-Excelente |           0 |          31 |          37 |          68 |
##              |       0.000 |       0.064 |       0.076 |             |
## -------------|-------------|-------------|-------------|-------------|
## Column Total |           2 |         435 |          50 |         487 |
## -------------|-------------|-------------|-------------|-------------|
##
```

- Random Forest.
  Se emplea el paquete Caret para la ejecución de este algoritmo.

```
library(caret)

# Se empleará Cross Validation
ctrl <- trainControl(method="cv",repeats = 3)
rfFit <- train(trainX,trainy, method = "rf", trControl = ctrl, preProcess = c("center","scale"), tuneLength = 20)
```

Random Forest utiliza 4 candidatos para alimentar el algoritmo en nuestro caso particular.

```
## Random Forest
##
## 972 samples
##  11 predictor
##   3 classes: '0-Baja', '1-Normal', '2-Excelente'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 875, 874, 875, 874, 875, 874, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8704588  0.4336767
##    3    0.8673868  0.4392480
##    4    0.8704691  0.4586604
##    5    0.8642833  0.4388390
##    6    0.8601806  0.4216908
##    7    0.8632629  0.4480780
##    8    0.8642728  0.4510819
##    9    0.8663346  0.4673163
##   10    0.8601701  0.4433055
##   11    0.8591497  0.4439614
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

Matriz de confusión:

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0-Baja 1-Normal 2-Excelente
##   0-Baja           0        1           0
##   1-Normal        17      390          30
##   2-Excelente      0       11          38
##
## Overall Statistics
##
##                Accuracy : 0.8789
##                  95% CI : (0.8465, 0.9065)
##     No Information Rate : 0.8255
##     P-Value [Acc > NIR] : 0.0007617
##
##                   Kappa : 0.5058
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0-Baja Class: 1-Normal Class: 2-Excelente
## Sensitivity               0.000000          0.9701            0.55882
## Specificity               0.997872          0.4471            0.97375
## Pos Pred Value            0.000000          0.8924            0.77551
## Neg Pred Value            0.965021          0.7600            0.93151
## Prevalence                0.034908          0.8255            0.13963
## Detection Rate            0.000000          0.8008            0.07803
## Detection Prevalence      0.002053          0.8973            0.10062
## Balanced Accuracy         0.498936          0.7086            0.76629
```

```
#Precisión del algoritmo
print(sprintf("La precisión de RANDOM FOREST es: %.2f %%",100*sum(predicted_rf == testy) / length(predicted_rf)))
```

```
## [1] "La precisión de RANDOM FOREST es: 87.89 %"
```

Se obtiene una precisión ligeramente superior al conseguido con el Boosting Adaptativo previamente.

5.-Representación de los resultados a partir de tablas y gráficas.

En caso de que no se haya incluido en otros apartados de este documento, el detalle del análisis preliminar y análisis propiamente dicho se incluye en el fichero HTML que se adjunta en esta entrega.

6.-Resolución del problema. A partir de los resultados obtenidos, ¿Cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

Las conclusiones son las siguientes:

- Se hace un análisis preliminar en el que principalmente se identifican una serie de Outliers que se decide eliminar para asegurarnos que no afecta al modelo que se pretende construir.
- Se escogen distintos algoritmos de clasificación, diferentes entre ellos, logrando en el caso de K-neastest neighbour y árbol de decisión, porcentajes de precisión con el conjunto de prueba similares.
- Se lleva a cabo una variante del árbol de decisión (Boost adaptativo) que acomete una serie de iteraciones que mejoran ligeramente la precisión del modelo en su conjunto.
- El algoritmo que mejor comportamiento tiene es el Random Forest, aunque es muy similar al Boost del árbol del decisión.

En definitiva, el resultado obtenido permite resolver el problema de predicción de la calidad del vino, en base a los atributos mencionados, que pretendíamos resolver con una ratio de precisión aceptable.

7.-Código.

Se aporta código en R, así como HTML generado a partir de dicho código.

CONTRIBUCIONES

Contribuciones	Firma
Investigación Previa	Javier López Calderón, José M Cano Hernández
Redacción de las respuestas	Javier López Calderón, José M Cano Hernández
Desarrollo de Código	Javier López Calderón, José M Cano Hernández