PRÁCTICA 2. LIMPIEZA Y VALIDACIÓN DE DATOS

Juan Alonso Franco Blanco
Juan Prieto Pena

5 DE DICIEMBRE DE 2020
UNIVERSITAT OBERTA DE CATALUNYA
ASIGNATURA DE TIPOLOGÍA Y CICLO DE VIDA DE LOS DATOS

Contenido

Introducción

Para la realización de este ejercicio se ha decidido emplear el dataset de Kaggle Red Wine Quality. Este dataset contiene diversas propiedades físico-químicas de diferentes vinos tintos de Portugal (que no están identificados más que por su número de fila en la tabla) junto con una variable de calidad obtenida en base a características sensoriales mediante un proceso de cata.

Estos datos se han obtenido a partir del artículo de Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos y José Reis: "Modeling wine preferences by data mining from physicochemical properties" publicado en "Decision Support Systems", Vol. 47 Num. 4, pg. 547-553 Nov 2019.

El objetivo de esta práctica empleando este conjunto de datos es la realización de un modelo que nos permita predecir la calidad de un futuro vino en base a sus características físico-químicas. Para ello, después de realizar un proceso de carga del dataset se realizará una limpieza de los datos y un estudio estadístico para determinar cuáles son las mejores cualidades a incorporar al modelo.

Finalmente, se evaluarán dos modelos, de clasificación y de regresión lineal para la conclusión de validez de cada modelo.

Descripción del dataset

El dataset contiene datos de 1599 vinos diferentes y proporciona 12 características diferentes para cada uno. Estas características son 11 propiedades físico-químicas diferentes de cada uno de los 1599 vinos tintos portugueses junto con una propiedad de puntuación de calidad, entre 0 (vino muy malo) y 10 (vino perfecto) obtenida en base a catas realizadas por expertos. Los diferentes atributos del dataset pueden verse a continuación:

1. Fixed acidity: Se expresa en g/l. La acidez fija es el conjunto de los ácidos naturales procedentes de la uva (tartárico, málico, cítrico y succínico) o formados en la fermentación maloláctica (láctico). (No se evaporan fácilmente).
2. Volatile acidity: Se expresa en g/l. La acidez volátil es una parte de la acidez total de un vino, formada por los ácidos primarios que ya están presentes en el mosto de uva (málico y tartárico) y los secundarios que son los generados durante los procesos de fermentación (acético, succínico, málico, …). También añaden acidez algunos gases y sustancias disueltas como dióxido de carbono, dióxido de azufre y sulfitos (añadidos como conservantes). La cantidad de ácido acético en el vino, que en niveles demasiado altos puede provocar un sabor desagradable parecido al del vinagre.
3. Citric acid: Se expresa en g/l. Es normal que no supere la cantidad de 1 g/l. El ácido cítrico es un ácido débil, con un nivel de pH entre 3 y 6. Encontrado en pequeñas cantidades, el ácido cítrico puede agregar 'frescura' y sabor a los vinos.
4. Residual sugar: Se expresa en g/l. La cantidad de azúcar que queda después de que se detiene la fermentación, es raro encontrar vinos con menos de 1 g/l.
5. Chlorides: Se expresa en g/l. Es la cantidad de sal en el vino. Se pretende asegurar la salubridad del vino.
6. Free sulfur dioxide: Se expresa en mg/l. La forma libre del SO2 existe en equilibrio entre el SO2 molecular (como gas disuelto) y el ion bisulfito; es un compuesto químico de

azufre y oxígeno. Es el aditivo más ampliamente utilizado en vinificación, y también el más indispensable.

7. Total sulfur dioxide: Se expresan en mg/l. Cantidad de formas libres y unidas de SO2; en bajas concentraciones, el SO2 es mayormente indetectable en el vino, pero en SO2 libre.
8. Density: Se expresa con cuatro decimales y es adimensional. Se expresa en g/mL. La densidad del vino es cercana a la del agua dependiendo del porcentaje de alcohol y contenido de azúcar.
9. pH: Describe qué tan ácido o básico es un vino en una escala de 0 (muy ácido) a 14 (muy básico); Es el grado de acidez o alcalinidad y es medido en una escala que va de 0 a 14. Las informaciones cuantitativas dadas por el valor del pH expresan el grado de acidez de un ácido o de una base en términos de la actividad de los iones de hidrógeno.
10. Sulphates: Se expresa en g/l. Es un aditivo para el vino que puede contribuir a los niveles de dióxido de azufre (SO2), que actúa como antimicrobiano.
11. Alcohol: el porcentaje de contenido de alcohol del vino. Para convertir el volumen (por ejemplo, ml) en masa (por ejemplo, g) y viceversa es necesario saber la densidad. Densidad de alcohol d=0,8 g/ml. Por tanto: un vino al 12,5 % vol contiene 12,5 ml de alcohol/100 ml de vino x 0,8 g/ml = 10 g de alcohol/100 ml de vino. Esto equivale a 1 unidad de bebida (= 10 g).

Finalmente, la variable de salida:

12. Quality (puntuación entre 0 y 10): Basada en datos sensoriales, puntuación entre 0 y 10 Antes de la fermentación o durante su proceso, se añade el azúcar necesario para corregir la carencia de azúcares en el mosto. Las levaduras actúan y transforman parte del azúcar en alcohol y CO2, superando una determinada graduación alcohólica, las levaduras mueren y finaliza la fermentación.

Limpieza de los datos

El código se escribirá en Python. El código puede verse adjunto a este informe como un notebook interactivo de Python. Las librerías empleadas han sido las siguientes:

- Pandas: Definimos las estructuras de datos
- Matplotlib: Herramienta de diseño de gráficos.
- Seaborn: Herramienta de diseño de gráficos.
- Numpy: Da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas.
- Scipy.stats Se basa en el objeto de matriz NumPy y es parte del conjunto NumPy, que incluye herramientas como Matplotlib, pandas y SymPy, y un conjunto en expansión de bibliotecas de computación científica. Este conjunto está dirigido al mismo tipo de usuarios que los de aplicaciones como MATLAB, GNU Octave, y Scilab. Este subpaquete de SciPy contiene librerías de cálculo estadístico.

- Statsmodels.api: Otro módulo estadístico que contiene información sobre clases y funciones para la estimación de diferentes modelos estadísticos.

- Scikitlearn: Paquete de aprendizaje automático para Python. Se emplean diversas funciones de ese paquete, como modelos, funciones de escala y funciones de evaluación de los modelos.

Comenzamos cargando los datos:

```
In [2]: df_wine=pd.read_csv('winequality-red.csv')
```

Comprobación de que la carga es correcta:

```
In [5]: df_wine.head()
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

Estudio preliminar de los datos

Comprobamos que el tamaño del dataset es el adecuado:

```
In [9]: print('El tamaño del dataset es: ')
        df_wine.shape

        El tamaño del dataset es:

Out[9]: (1599, 12)
```

El siguiente paso consiste en comprobar el tipo de datos que hemos cargado:

```
In [5]: df_wine.dtypes

Out[5]: fixed acidity           float64
        volatile acidity        float64
        citric acid             float64
        residual sugar          float64
        chlorides               float64
        free sulfur dioxide     float64
        total sulfur dioxide    float64
        density                 float64
        pH                      float64
        sulphates               float64
        alcohol                 float64
        quality                   int64
        dtype: object
```

Podemos ver que todos los atributos físico-químicos son números de coma flotante de 64 bits, mientras que la variable de calidad del vino es un número entero (también de 64 bits). Esto es

consecuente con las variables explicadas en el apartado anterior, y no se realizará ningún cambio en el tipo de los datos.

Valores nulos

Existe un atributo que contiene elementos con valor nulo, como podemos ver cuando se corre el siguiente código:

```
In [10]: df_wine.eq(0).any()

Out[10]: fixed acidity           False
         volatile acidity        False
         citric acid              True
         residual sugar          False
         chlorides               False
         free sulfur dioxide     False
         total sulfur dioxide    False
         density                 False
         pH                      False
         sulphates               False
         alcohol                 False
         quality                 False
         dtype: bool
```

```
In [8]: df_wine[df_wine['citric acid']==0]
```

Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.0	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.0	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
4	7.4	0.700	0.0	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
5	7.4	0.660	0.0	1.8	0.075	13.0	40.0	0.99780	3.51	0.56	9.4	5
7	7.3	0.650	0.0	1.2	0.065	15.0	21.0	0.99460	3.39	0.47	10.0	7
...	...	...	...	...	...	...	...	...	...	...	...	...
1455	6.5	0.900	0.0	1.6	0.052	9.0	17.0	0.99467	3.50	0.63	10.9	6
1461	6.2	0.785	0.0	2.1	0.060	6.0	13.0	0.99664	3.59	0.61	10.0	4
1550	7.1	0.680	0.0	2.3	0.087	17.0	26.0	0.99783	3.45	0.53	9.5	5
1551	7.1	0.670	0.0	2.3	0.083	18.0	27.0	0.99768	3.44	0.54	9.4	5
1553	7.3	0.735	0.0	2.2	0.080	18.0	28.0	0.99765	3.41	0.60	9.4	5

132 rows × 12 columns

Concretamente, existen 132 vinos con un contenido de ácido cítrico nulo. Esto es una ocurrencia completamente normal, dado que generalmente el ácido cítrico está presente en cantidades muy pequeñas o en trazas, y al no tratarse de una anomalía, no vamos a realizar ningún tipo de cambio en estos datos. Además, tratándose de vinos tintos los matices cítricos que proporciona este ácido no son buscados para los sabores de este tipo de caldos.

Adicionalmente, la Unión Europea no permite la adición de ácido cítrico al vino excepto en casos muy puntuales, por lo que un bajo nivel de este ácido es esperado en vinos europeos, como los de Portugal.

Como puede verse en la siguiente imagen, no existen datos nulos (Null o NaN) en este dataset:

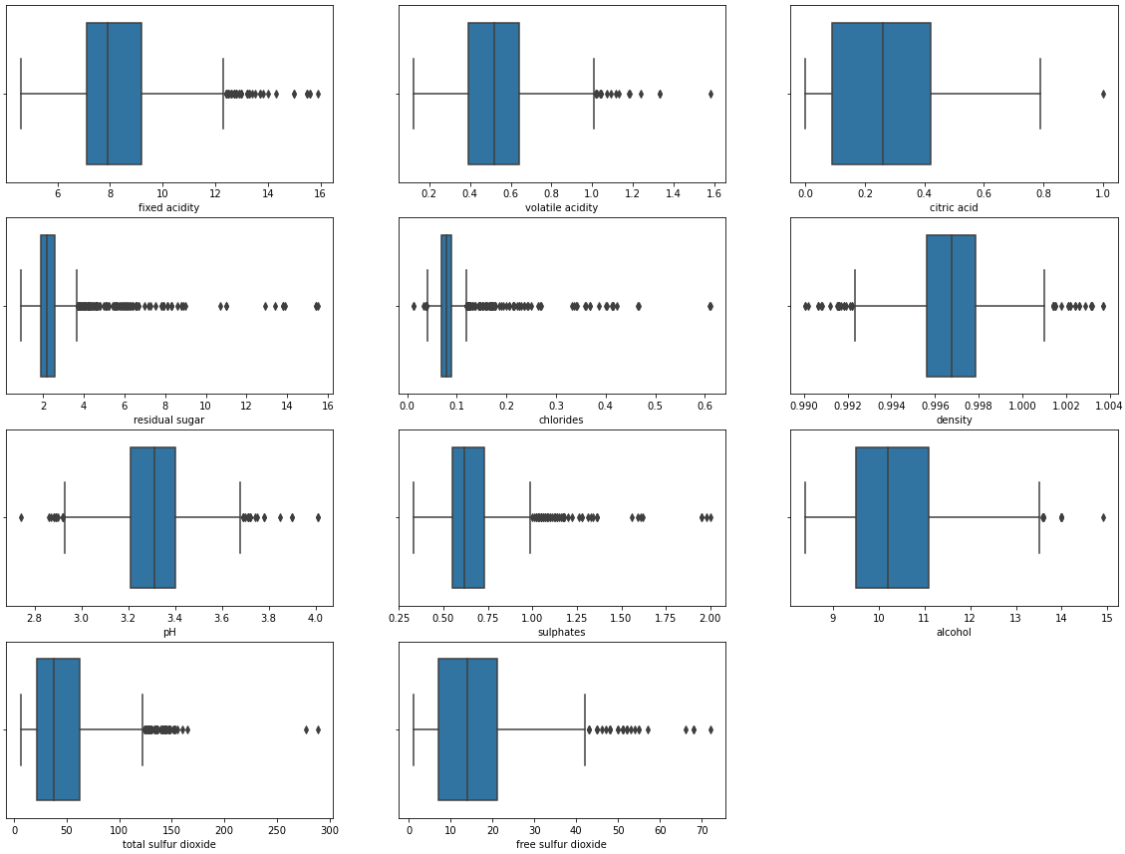
```
In [12]: print(f'El dataset contiene {df_wine.shape[0]} entradas.\nEl dataset contiene los siguientes valores nulos:')
         df_wine.isnull().sum()

         El dataset contiene 1599 entradas.
         El dataset contiene los siguientes valores nulos:

Out[12]: fixed acidity           0
         volatile acidity        0
         citric acid             0
         residual sugar          0
         chlorides               0
         free sulfur dioxide     0
         total sulfur dioxide    0
         density                 0
         pH                      0
         sulphates               0
         alcohol                 0
         quality                 0
         dtype: int64
```

Finalmente, podemos ver en la siguiente tabla un resumen de los datos, con sus valores medios, desviaciones típicas, cuartiles y valores mínimos y máximos:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1451.000000	1451.000000	1451.000000	1451.000000	1451.000000	1451.000000	1451.000000	1451.000000	1451.000000	1451.000000	1451.000000	1451.000000
mean	8.310062	0.522950	0.265382	2.387285	0.081425	15.104755	43.735355	0.996710	3.315934	0.642584	10.421089	5.659545
std	1.646458	0.168531	0.190934	0.862078	0.020966	9.309768	29.441284	0.001716	0.141096	0.129801	1.021588	0.781605
min	5.000000	0.120000	0.000000	1.200000	0.038000	1.000000	6.000000	0.991500	2.880000	0.330000	8.500000	4.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	21.000000	0.995600	3.220000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.250000	2.200000	0.079000	13.000000	36.000000	0.996700	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.630000	0.420000	2.600000	0.089000	21.000000	58.000000	0.997800	3.400000	0.720000	11.100000	6.000000
max	13.500000	1.040000	0.790000	6.700000	0.226000	47.000000	145.000000	1.002200	3.750000	1.160000	13.600000	8.000000

Identificación de valores extremos

La última parte de este apartado consistirá en analizar los datos para buscar posibles outliers. Para ello, emplearemos dos métodos. En el primero visualizamos los datos mediante boxplots para estimar el posible número de outliers en nuestras medidas, como puede verse en la siguiente figura:



Procederemos a contar el número de outliers que existen en los datos con el siguiente código:

```
In [29]: df_wine_iqr=df_wine.drop(['quality'], axis=1)

         Q_1 = df_wine_iqr.quantile(0.25)
         Q_3 = df_wine_iqr.quantile(0.75)
         IQR = Q_3 - Q_1

         mask = (df_wine_iqr < (Q_1 - 1.5 * IQR)) | (df_wine_iqr > (Q_3 + 1.5 * IQR))
```

```
In [30]: print('El número de outliers para cada una de las variables es:')
         print(mask.sum())
         print('\nSuma de los outliers:', mask.sum().sum())
```

```
El número de outliers para cada una de las variables es:
fixed acidity            49
volatile acidity         19
citric acid               1
residual sugar          155
chlorides               112
free sulfur dioxide      30
total sulfur dioxide     55
density                  45
pH                       35
sulphates                59
alcohol                  13
dtype: int64

Suma de los outliers: 573
```

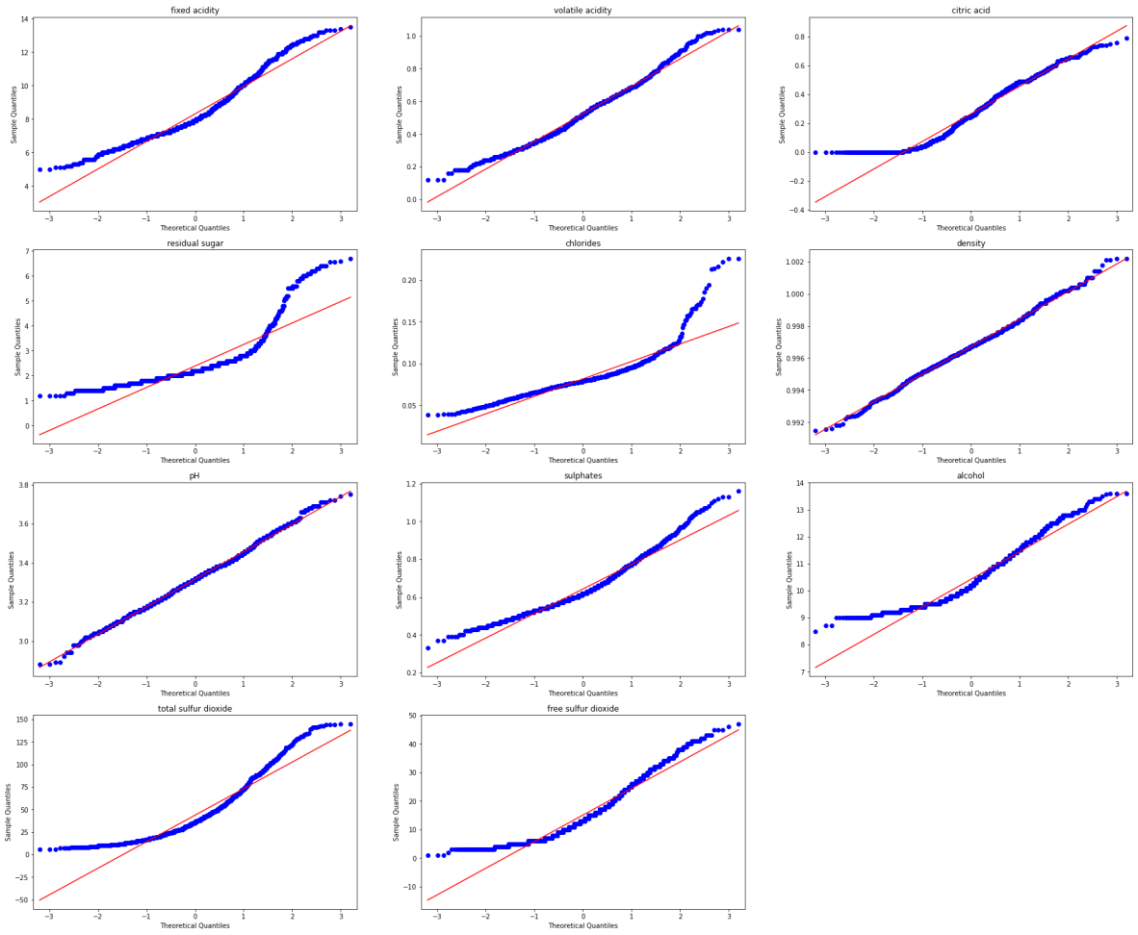
Podemos ver que prácticamente todos los parámetros poseen valores extremos. En el artículo citado anteriormente, no consideran que los datos extremos sean incorrectos en el caso del vino blanco y son empleados para la realización de su modelo, por lo que asumimos que no ha habido

errores de transcripción en la base de datos y los valores extremos son las medidas correctas para cada uno de los vinos analizados.

Análisis de los datos

Test de normalidad

Si nos fijamos en los diagramas de cajas de cantidades como la densidad o el pH, aparentemente sus valores están centrados y sus medias coinciden aproximadamente con sus medianas. ¿Siguen los datos una distribución normal? Primero, estudiaremos sus distribuciones de cuantiles comparándolos con distribuciones normales, como se puede ver en la siguiente gráfica:



Podemos ver que muchos de los parámetros no siguen una distribución normal, pero datos como la densidad y el pH se aproximan mucho a un comportamiento normal. Para cerciorarnos, realizamos tests de Shapiro-Wilk y Kolmogorov-Smirnov a nuestros datos para ver si siguen dicha distribución.

```
Test de Shapiro-Wilk para la columna fixed acidity
ShapiroResult(statistic=0.9462423324584961, pvalue=1.38919314177183e-22)
Test de Kolmogorov-Smirnov para la columna fixed acidity
KstestResult(statistic=0.9999997133484281, pvalue=0.0)

Test de Shapiro-Wilk para la columna volatile acidity
ShapiroResult(statistic=0.986117422580719, pvalue=1.4341020038166619e-10)
Test de Kolmogorov-Smirnov para la columna volatile acidity
KstestResult(statistic=0.5748486552953148, pvalue=0.0)

Test de Shapiro-Wilk para la columna citric acid
ShapiroResult(statistic=0.953080415725708, pvalue=3.752339251165586e-21)
Test de Kolmogorov-Smirnov para la columna citric acid
KstestResult(statistic=0.5, pvalue=0.0)

Test de Shapiro-Wilk para la columna residual sugar
ShapiroResult(statistic=0.7502002716064453, pvalue=3.050626756835127e-42)
Test de Kolmogorov-Smirnov para la columna residual sugar
KstestResult(statistic=0.9130407218827004, pvalue=0.0)

Test de Shapiro-Wilk para la columna chlorides
ShapiroResult(statistic=0.8417074084281921, pvalue=9.856487128263449e-36)
Test de Kolmogorov-Smirnov para la columna chlorides
KstestResult(statistic=0.51515615898524, pvalue=0.0)

Test de Shapiro-Wilk para la columna free sulfur dioxide
ShapiroResult(statistic=0.9254329204559326, pvalue=3.118045142851958e-26)
Test de Kolmogorov-Smirnov para la columna free sulfur dioxide
KstestResult(statistic=0.9958933824645794, pvalue=0.0)

Test de Shapiro-Wilk para la columna total sulfur dioxide
ShapiroResult(statistic=0.8920235633850098, pvalue=1.0336284515198528e-30)
Test de Kolmogorov-Smirnov para la columna total sulfur dioxide
KstestResult(statistic=0.9999999990134123, pvalue=0.0)

Test de Shapiro-Wilk para la columna density
ShapiroResult(statistic=0.9970580339431763, pvalue=0.008058983832597733)
Test de Kolmogorov-Smirnov para la columna density
KstestResult(statistic=0.8392792538231437, pvalue=0.0)

Test de Shapiro-Wilk para la columna pH
ShapiroResult(statistic=0.9972261190414429, pvalue=0.01196026336401701)
Test de Kolmogorov-Smirnov para la columna pH
KstestResult(statistic=0.9980116241451057, pvalue=0.0)

Test de Shapiro-Wilk para la columna sulphates
ShapiroResult(statistic=0.9515525698661804, pvalue=1.7444439403112077e-21)
Test de Kolmogorov-Smirnov para la columna sulphates
KstestResult(statistic=0.6560644812653279, pvalue=0.0)

Test de Shapiro-Wilk para la columna alcohol
ShapiroResult(statistic=0.9325990676879883, pvalue=4.484160993554993e-25)
Test de Kolmogorov-Smirnov para la columna alcohol
KstestResult(statistic=1.0, pvalue=0.0)

Test de Shapiro-Wilk para la columna quality
ShapiroResult(statistic=0.8471745848655701, pvalue=2.962721586736633e-35)
Test de Kolmogorov-Smirnov para la columna quality
KstestResult(statistic=0.9999683287581669, pvalue=0.0)
```

Los resultados muestran que ninguna de las columnas sigue una distribución normal, obteniendo los test de Shapiro-Wilk y Kolmogorov-Smirnov realizados p-valores muy inferiores a 0,05, pudiendo rechazar la hipótesis nula de que las columnas de datos siguen una distribución normal.

Normalización de los datos mediante transformaciones de Yeo-Johnson

Procederemos a normalizar los datos mediante una transformación de Yeo-Johnson. Emplearemos esta transformación en vez de la de Box-Cox debido a que en nuestros datos existen valores nulos, que la transformación de Box-Cox no acepta.

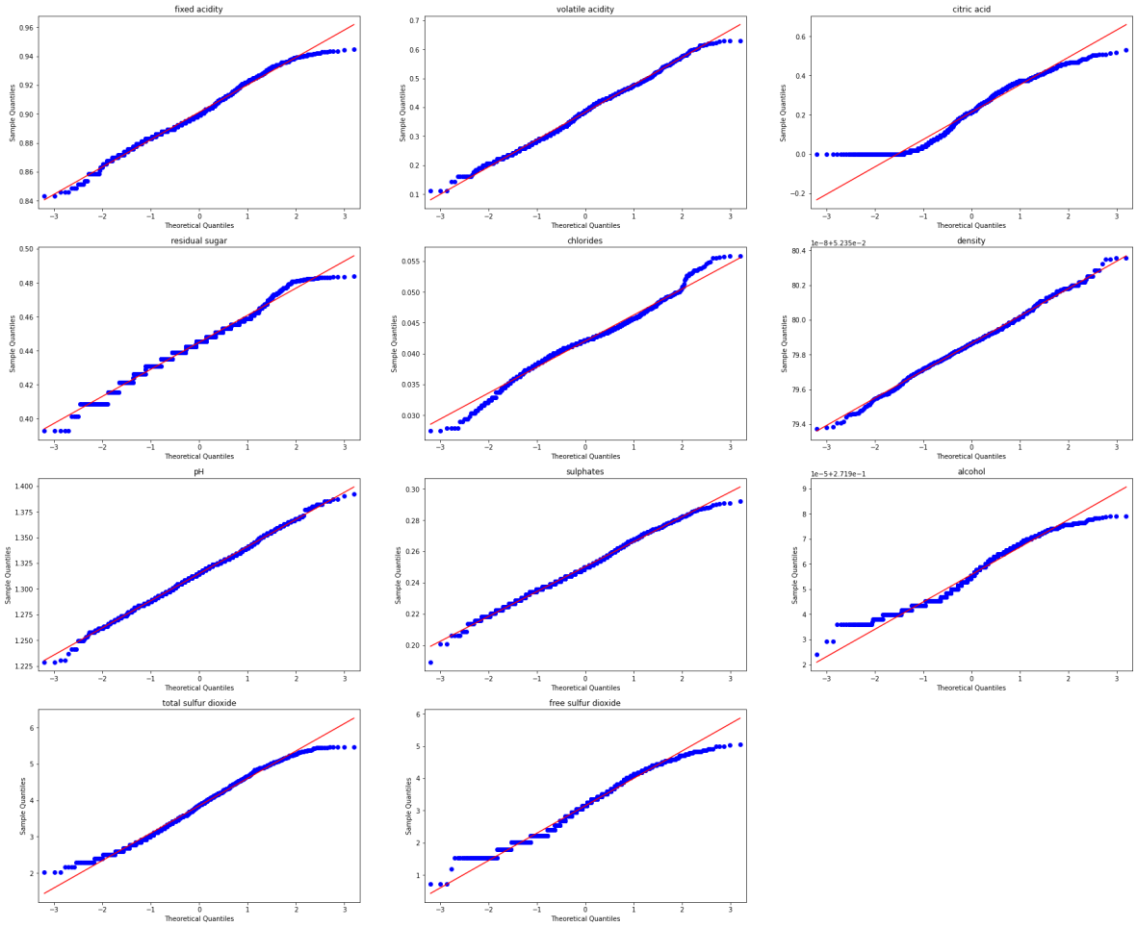
Esta transformación está incluida en scipy.stats. No realizaremos la transformación en la variable de salida (calidad) puesto que la transformaremos de otra manera para la realización del modelo.

También guardaremos los valores del parámetro de transformación para poder usarlo con posibles nuevos datos que puedan aparecer.

```
df_wine_norm=pd.DataFrame()
lambda_transf=[]

for col in df_wine.columns[:-1]:
    transf, lam_par = stats.yeojohnson(df_wine[col])
    lambda_transf.append(lam_par)
    df_wine_norm[col]=transf

df_wine_norm.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	0.892415	0.482834	-0.000000	0.435227	0.041431	2.943864	3.799481	0.052351	1.350593	0.240859	0.271944
1	0.898041	0.564416	-0.000000	0.455284	0.046209	4.076388	4.566264	0.052351	1.293276	0.257455	0.271950
2	0.898041	0.511286	0.038977	0.448254	0.045077	3.351725	4.319256	0.052351	1.304741	0.253764	0.271950
3	0.931072	0.236165	0.414603	0.435227	0.041169	3.523259	4.439553	0.052351	1.285528	0.243990	0.271950
4	0.892415	0.482834	-0.000000	0.435227	0.041431	2.943864	3.799481	0.052351	1.350593	0.240859	0.271944

Comprobaremos la transformación viendo los diagramas de cuantil-cuantil comparándolos con una distribución normal:



Puede verse que la transformación no parece arreglar el problema de la falta de normalidad en los datos. Por ello, y para evitar posibles complicaciones, no utilizaremos los datos transformados para realizar el modelo de predicción de la calidad del vino.

Transformación de la variable de salida

En la información del dataset en Kaggle se recomienda que la variable de salida se transforme a una variable dicotómica con el fin de poder usar un modelo de clasificación sobre los datos. Para ello, ejecutaremos el siguiente código, que añadirá una nueva columna con el valor 0 si el vino tiene una puntuación de calidad de 6 o inferior y el valor 1 si la puntuación de calidad es de 7 o superior.

```
[19] bins = (0, 6.5, 10)
     group_names = ['bad', 'good']
     df_wine['quality_dic'] = pd.cut(df_wine['quality'], bins = bins, labels = group_names)
```

Número de vinos pertenecientes a cada categoría:

```
[20] df_wine['quality_dic'].value_counts()

     bad     1250
     good     201
     Name: quality_dic, dtype: int64
```

Generación de csv con datos finales

Se procede a generar un csv con los datos finales usando el siguiente código:

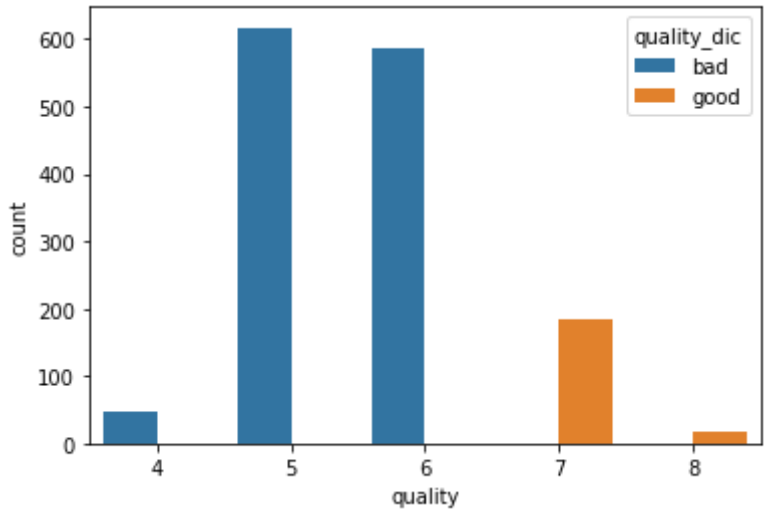
```
[64] df_wine.to_csv("df_wine_final.csv", index=True)
```

El fichero puede verse en GitHub. Añadimos también en este repositorio el fichero con los datos ya filtrados para el análisis con nombre df_wine_final.csv.

Planificación de pruebas estadísticas

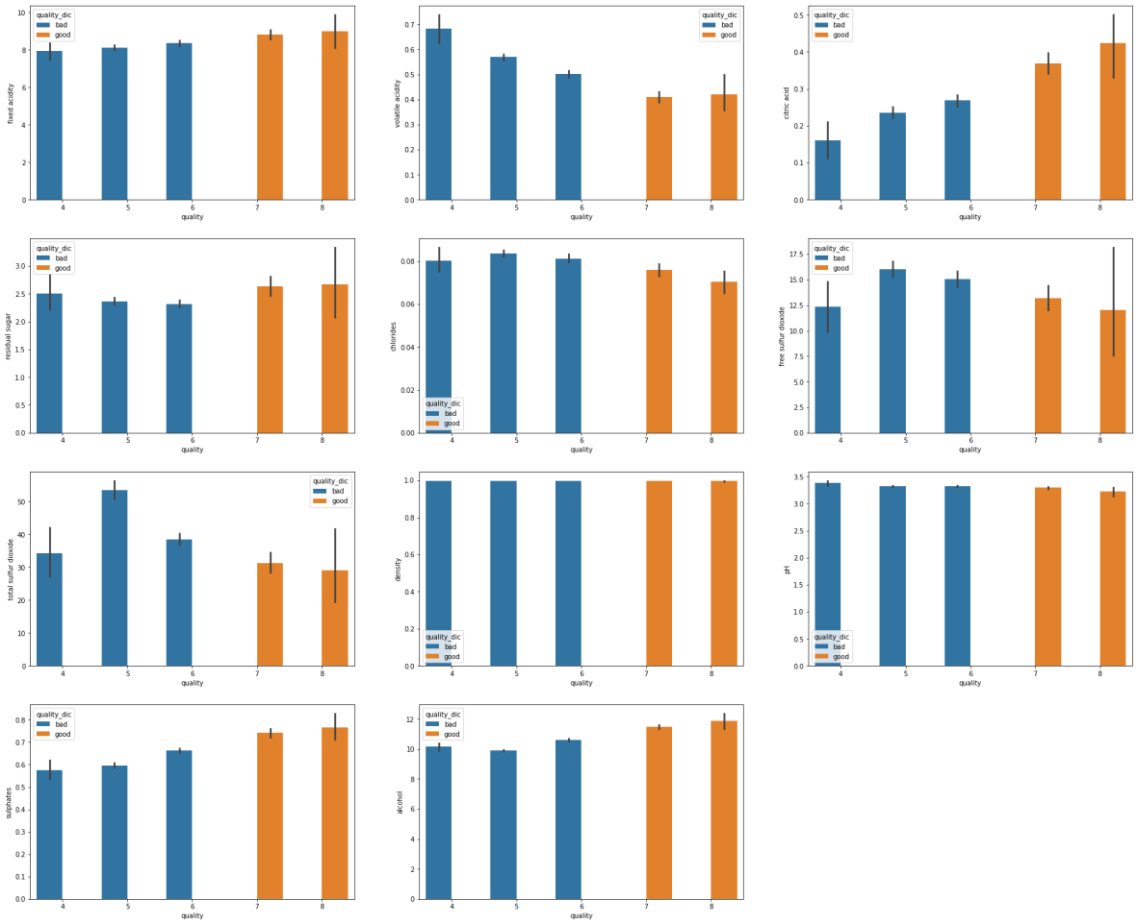
En esta sección mostraremos cómo varían los diferentes valores medios de los datos en función de la calidad del vino. Usaremos para ello ambas variables de calidad (la escala de 0-10 y la variable dicotómica).

Hemos decidido mostrar el valor medio agrupado en función de la calidad del vino para mostrar de una manera rápida las posibles variaciones de las diferentes variables de entrada en función de la calidad. Posteriormente estudiaremos de un modo más cuantitativo la correlación entre las variables con una matriz de correlaciones.



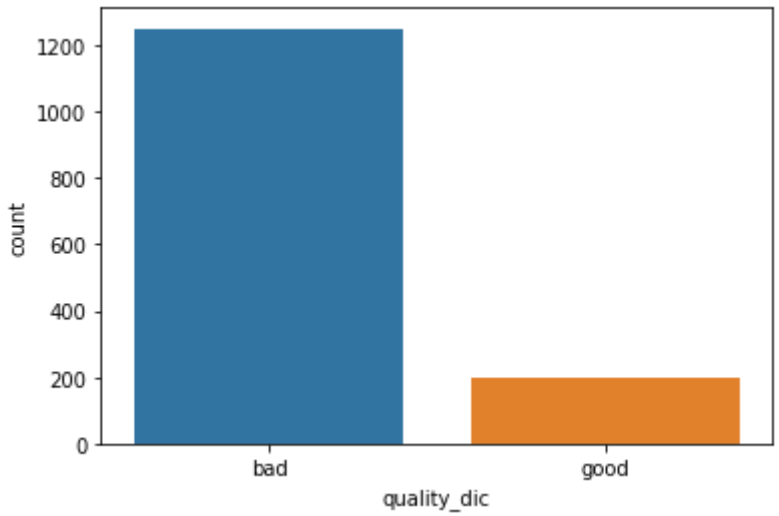
Podemos ver que existe un desequilibrio en el número de vinos con cada nota. Gran parte del conjunto se compone de vinos con notas de 5 o 6, y existen muy pocos vinos con notas de 4, 7 y 8. Este desequilibrio tendrá importancia a la hora de construir los conjuntos de entrenamiento y prueba en los modelos de regresión y clasificación.

A continuación se muestra la distribución de los valores medios de las variables de entrada en función de la escala de calidad del vino (escala de 0 a 10).

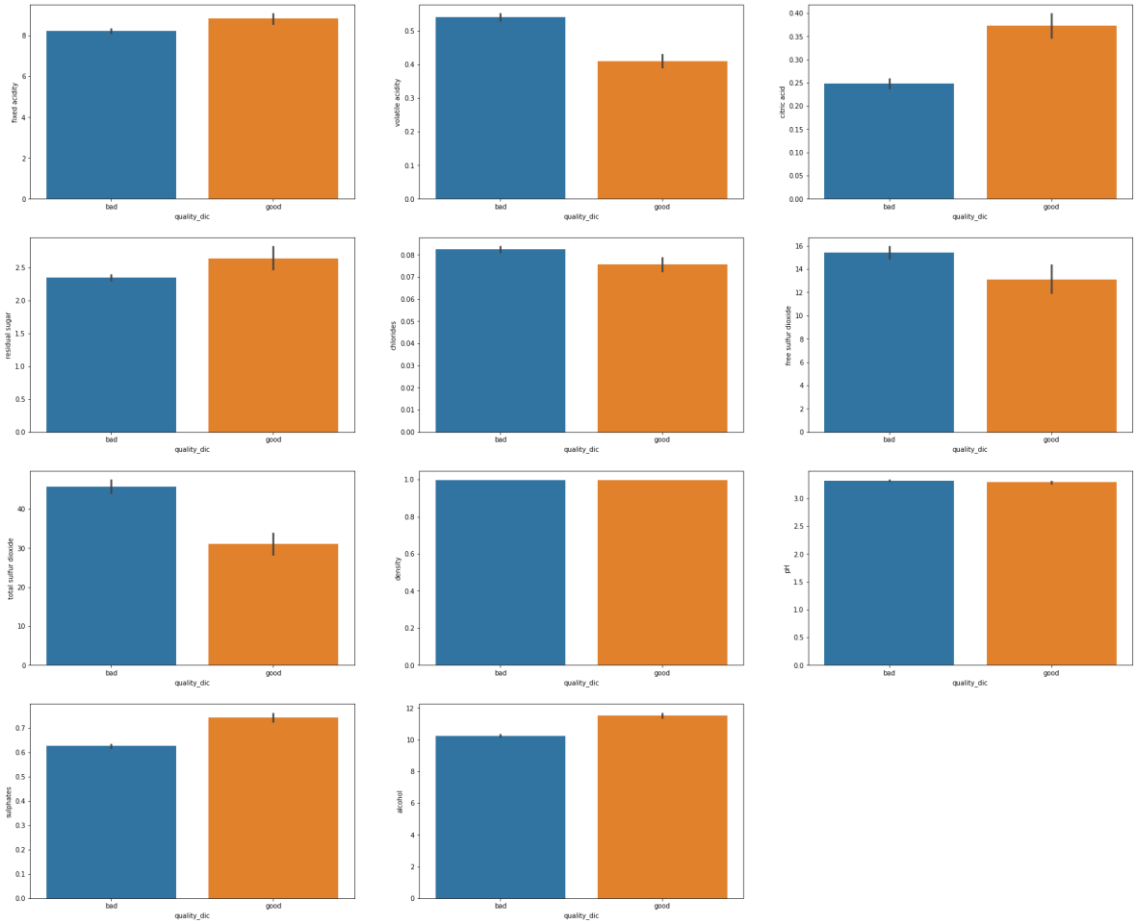


Puede verse en la imagen que existen ciertas variables muy interesantes que podrían mostrar una correlación fuerte entre ellas y la calidad del vino, como pueden ser los parámetros de “volatile acidity”, “citric acid” y “sulphates”. El resto de cantidades no muestran, a priori, unas variaciones distinguibles del parámetro en función de la calidad que podrían ser empleadas en un modelo de correlación de la calidad en función de alguno de estos parámetros.

Por otra parte, si hacemos las mismas representaciones con la variable dicotómica de calidad obtenemos los siguientes gráficos:



Podemos ver aquí que existe un gran desequilibrio entre vinos “buenos” y “malos”, como se ha visto en el apartado anterior.



Los grupos mencionados anteriormente siguen mostrando una diferencia en los valores medios. Adicionalmente, otros parámetros muestran una diferencia que podría ser explotada a la hora de realizar un modelo de clasificación.

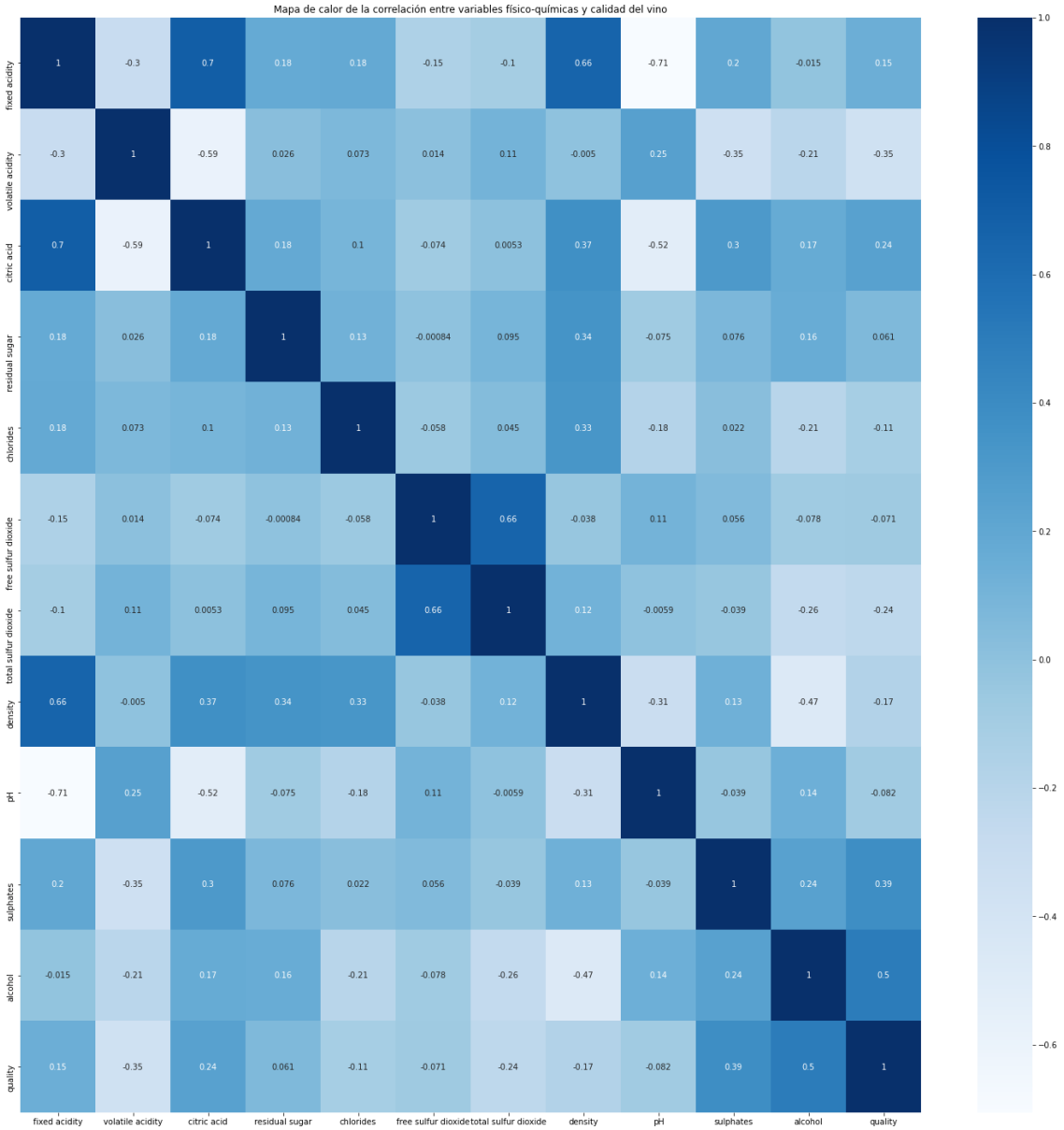
Se dispone de evidencias cualitativas de que existen diferencias que podrían explotarse para desarrollar modelos de correlación y clasificación que permitan predecir si un vino será bueno o no (o incluso predecir su nota en una cata) en función de sus características físico-químicas, pero, ¿son esas evidencias una coincidencia o hay signos robustos de correlación?

Para determinarlo, desarrollaremos una matriz de correlaciones en el caso del modelo de regresión lineal (usando el parámetro calidad) y realizaremos pruebas de t de Student en el caso de la variable dicotómica de calidad para determinar si esto es realmente así o simplemente es una coincidencia de nuestro conjunto de datos.

Aplicación de pruebas estadísticas

Matriz de correlación entre la variable “calidad” y las variables de entrada

Para determinar de una manera visual la correlación entre las diferentes variables de entrada y la variable de salida generaremos una matriz de correlaciones que proporcionará esa información de manera gráfica:



Si nos fijamos en los valores que se muestran en la columna de la variable de salida “quality” obtenemos lo siguiente:

```
corr_full['quality'].abs().sort_values(ascending=False)
```

```
quality                 1.000000
alcohol                 0.501501
sulphates               0.386567
volatile acidity        0.353443
citric acid             0.243999
total sulfur dioxide    0.237745
density                 0.167568
fixed acidity           0.145163
chlorides               0.108787
pH                      0.082164
free sulfur dioxide     0.071202
residual sugar          0.061482
Name: quality, dtype: float64
```

Podemos ver que no existen correlaciones fuertes entre las variables físico-químicas y la variable de salida, siendo la correlación más fuerte la del alcohol con la calidad, pero obteniendo un valor de 0,501501 en el coeficiente de correlación de Pearson. Los siguientes valores se sitúan solamente por encima de 0.35 (sulfatos y acidez volátil), por lo que probablemente el modelo de correlación no sea muy efectivo para poder predecir la calidad de los vinos.

Pruebas de t de Student para estimar la diferencia de valores medios con la variable de salida dicotómica

El siguiente paso es realizar una serie de pruebas de t de Student para comprobar si la diferencia en los valores medios observados en las gráficas en las que se empleaba la variable de calidad dicotómica son estadísticamente significativos, es decir, que no se deben a la casualidad o a la selección de datos empleados. En este caso emplearemos el test t de Student para medias independientes, puesto que suponemos que no existe dependencia entre ambos grupos.

El test t de Student asume un comportamiento normal en la población. Aunque en un apartado anterior hayamos determinado que esto no se cumple, es posible emplear este test teniendo en cuenta que debido al teorema del límite central, y como los conjuntos de datos son superiores a 30 elementos, el comportamiento puede aproximarse a una distribución normal.

Debemos estudiar también si sus varianzas son iguales para poder escoger el test de t de Student que mejor se adapte a cada caso. Para ello realizaremos el test de Levene empleando las medianas para cada una de las variables físico-químicas agrupadas según la variable de calidad dicotómica. En este test se prueba la hipótesis nula de que ambas poblaciones tienen varianzas iguales. Un resultado de $p < 0,05$ rechaza la hipótesis nula, demostrando que las varianzas no son iguales.

```
for col in df_wine.columns[:-2]:
  a=df_wine[col][df_wine['quality_dic']=='good']
  b=df_wine[col][df_wine['quality_dic']=='bad']
  stat, p = stats.levene(a, b)
  print('Test de levene para la variable',col,':\n',round(p,4))
```

```
Test de levene para la variable fixed acidity :
 0.0042
Test de levene para la variable volatile acidity :
 0.0028
Test de levene para la variable citric acid :
 0.3461
Test de levene para la variable residual sugar :
 0.0
Test de levene para la variable chlorides :
 0.4447
Test de levene para la variable free sulfur dioxide :
 0.0885
Test de levene para la variable total sulfur dioxide :
 0.0
Test de levene para la variable density :
 0.0003
Test de levene para la variable pH :
 0.9166
Test de levene para la variable sulphates :
 0.5996
Test de levene para la variable alcohol :
 0.3021
```

Tan solo en el caso de las variables “citric acid”, “chlorides”, “free sulfur dioxide”, “pH”, “sulphates” y “alcohol” no es posible rechazar la hipótesis nula de que las varianzas son iguales por mostrar un p valor superior a 0,05 en el test de Levene. Estas variables serán testadas empleando el test t de Student para dos grupos independientes con varianzas iguales, mientras que las demás serán testadas usando el test t de Student para varianzas desiguales.

El test t de Student nos muestra si la diferencia entre las medias es estadísticamente significativa si el p-valor obtenido es inferior a 0,05. De no ser así, no nos es posible rechazar la hipótesis nula de que la diferencia de medias no es estadísticamente significativa. Los resultados obtenidos han sido los siguientes:

```
[69] columns_var=['citric acid', 'chlorides', 'free sulfur dioxide', 'pH', 'sulphates', 'alcohol']
     columns_novar=['fixed acidity','volatile acidity','residual sugar','total sulfur dioxide','density']
```

```
[67] for col in columns_var:
       a=df_wine[col][df_wine['quality_dic']=='good']
       b=df_wine[col][df_wine['quality_dic']=='bad']
       stat, p = stats.ttest_ind(a,b,equal_var=True)
       print('Test de t de Student para la variable',col,':\n',p)
```

```
Test de t de Student para la variable citric acid :
 1.6491736739559251e-18
Test de t de Student para la variable chlorides :
 1.875862936235615e-05
Test de t de Student para la variable free sulfur dioxide :
 0.000985812934457707
Test de t de Student para la variable pH :
 0.0025664061191861607
Test de t de Student para la variable sulphates :
 8.759726091547653e-34
Test de t de Student para la variable alcohol :
 1.1393984848102585e-66
```

```
[68] for col in columns_novar:
       a=df_wine[col][df_wine['quality_dic']=='good']
       b=df_wine[col][df_wine['quality_dic']=='bad']
       stat, p = stats.ttest_ind(a,b,equal_var=False)
       print('Test de t de Student para la variable',col,':\n',p)
```

```
Test de t de Student para la variable fixed acidity :
 1.4257260388444856e-05
Test de t de Student para la variable volatile acidity :
 1.0267517575636705e-25
Test de t de Student para la variable residual sugar :
 0.0010976521828344153
Test de t de Student para la variable total sulfur dioxide :
 1.3527726673438106e-16
Test de t de Student para la variable density :
 4.874915523560399e-07
```

Podemos ver que los resultados de los test de Student establecen que existe una diferencia significativa entre los valores medios de todas las variables físico-químicas empleadas entre los vinos con buena calidad y mala calidad. Por ello, emplearemos todas las variables en el modelo.

Construcción de los modelos

Los modelos se han construido empleando diferentes funciones de la librería sklearn. Para el modelo de regresión lineal se ha empleado la función LinearRegression, mientras que para el modelo de clasificación se ha empleado el modelo de regresión logística LogisticRegression.

Adicionalmente se han empleado también la función train_test_split para dividir nuestro conjunto original de datos en subconjuntos de entrenamiento y prueba, manteniendo la proporción de calidades igual en todos los subconjuntos para evitar sesgos en el modelo (mediante el argumento stratify), y la función MinMaxScaler para normalizar los datos de entrada a un rango similar de valores para evitar problemas con los modelos.

Se ha decidido emplear un número de semilla aleatorio específico para ayudar con la repetibilidad de los resultados. El código se muestra a continuación.

```
[31] from sklearn.model_selection import train_test_split

     X=df_wine.drop(['quality', 'quality_dic'], axis=1)
     y_reg=df_wine['quality']
     y_clas=df_wine['quality_dic'].map({'good':1,'bad':0})

     SEED = 2 # por reproducibilidad

     #Stratified train/test split for visualization purposes
     X_tr_reg, X_te_reg, y_tr_reg, y_te_reg = train_test_split(X, y_reg, test_size=0.2, stratify=y_reg, random_state=SEED)
     X_tr_clas, X_te_clas, y_tr_clas, y_te_clas = train_test_split(X, y_clas, test_size=0.2, stratify=y_clas, random_state=SEED)
```

```
[32] from sklearn.linear_model import LinearRegression, LogisticRegression

     Reg_model= LinearRegression()
     Clas_model = LogisticRegression(random_state = SEED)
```

```
[33] from sklearn.preprocessing import MinMaxScaler

     min_max_scaler_reg = MinMaxScaler()
     min_max_scaler_clas = MinMaxScaler()

     X_tr_reg_minmax = min_max_scaler_reg.fit_transform(X_tr_reg)
     X_te_reg_minmax = min_max_scaler_reg.transform(X_te_reg)

     X_tr_clas_minmax = min_max_scaler_clas.fit_transform(X_tr_clas)
     X_te_clas_minmax = min_max_scaler_clas.transform(X_te_clas)
```

Resultados de los modelos

Regresión

El modelo de regresión lineal se entrenó con el subconjunto de datos de entrenamiento para regresión lineal. A continuación, se probó en el subconjunto de prueba. El código y los resultados obtenidos se muestran a continuación.

```
[41] from sklearn.metrics import mean_squared_error
```

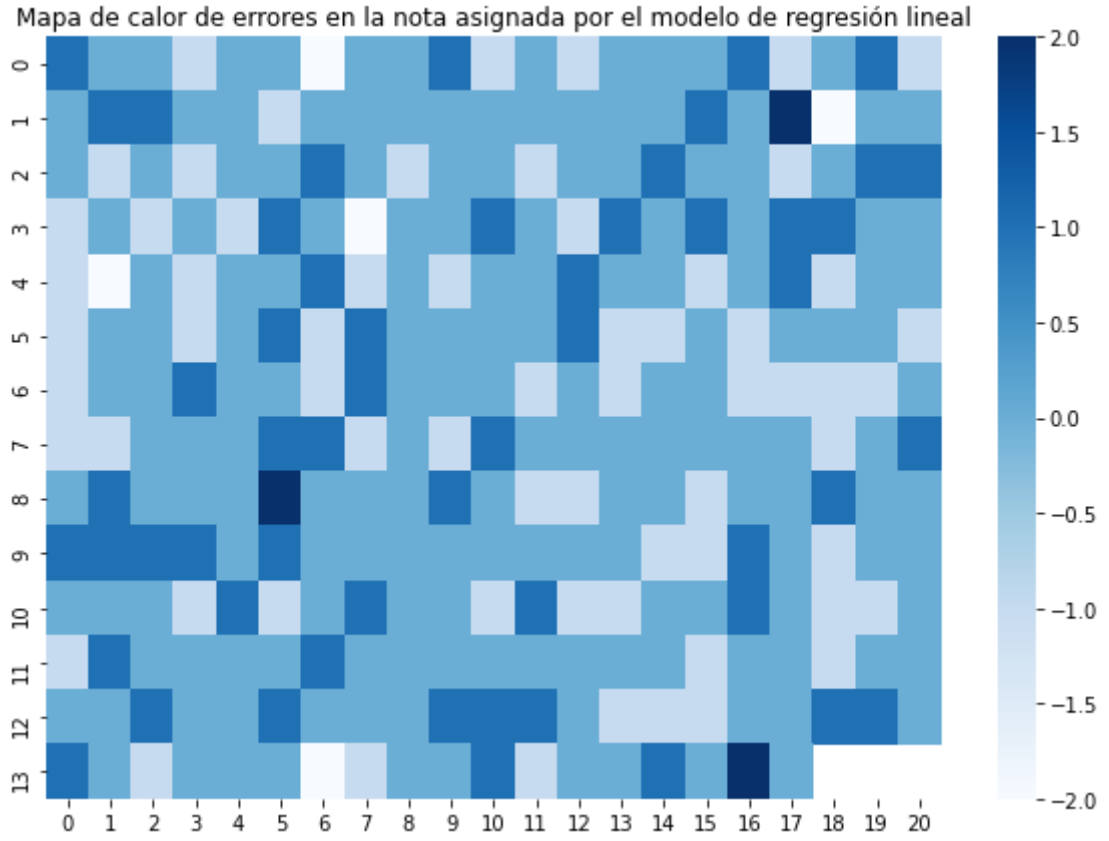
```
[47] Reg_model.fit(X_tr_reg_minmax,y_tr_reg)
     y_pred_reg=Reg_model.predict(X_te_reg_minmax)
```

```
[50] print('Puntuación R2 para el conjunto de entrenamiento:', Reg_model.score(X_tr_reg_minmax,y_tr_reg))

     Puntuación R2 para el conjunto de entrenamiento: 0.39346048157900454
```

```
[49] print('Puntuación R2 para el conjunto de prueba:', Reg_model.score(X_te_reg_minmax,y_te_reg))
     print('MSE para el conjunto de entrenamiento:', mean_squared_error(y_te_reg,y_pred_reg))

     Puntuación R2 para el conjunto de prueba: 0.31244638476846176
     MSE para el conjunto de entrenamiento: 0.41423235840250605
```

Se calculó el coeficiente de determinación R2 de la predicción, junto con el error cuadrático medio, obteniéndose puntuaciones R2 de 0,393 para el conjunto de entrenamiento y 0,312 para el conjunto de prueba. La máxima puntuación obtenible es 1, aunque el resultado ideal (una predicción perfecta) es igual a 0. Es posible obtener resultados negativos si el modelo da resultados muy malos.

Se calculó la diferencia entre la nota predicha y la nota real del conjunto de prueba, obteniéndose el siguiente mapa de calor, donde cada elemento es uno de los vinos del conjunto de prueba (las últimas tres casillas corresponden a NaN empleados para rellenar la matriz cuadrada:



El modelo ha logrado puntuar correctamente a 166 vinos de 291, correspondiendo al 57.04 % de vinos del subconjunto.

El modelo predice más o menos bien la puntuación de los vinos, con un margen de error de aproximadamente un punto, por lo que podría ser usado para predecir un margen en una nota de cata a partir de sus características físico-químicas.

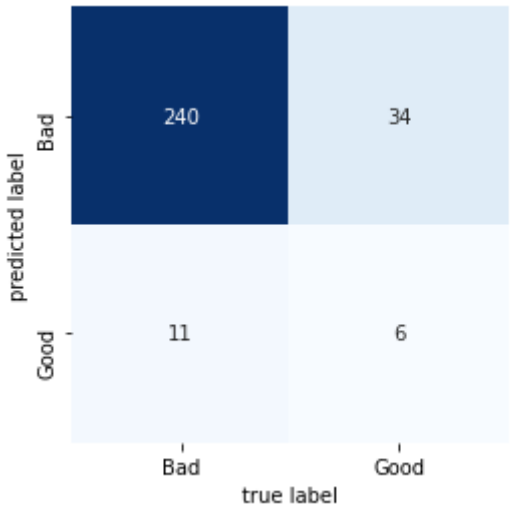
Clasificación

Se puede simplificar el modelo para poder distinguir un vino bueno (nota de 7 o más en la cata) de uno malo. Para ello, haremos uso de un algoritmo de clasificación basado en la regresión logística. El modelo, la predicción y la construcción de la matriz de confusión, un elemento usado para ver lo bien que se comporta el modelo clasificando los vinos, puede verse a continuación:

```
Clas_model.fit(X_tr_clas_minmax, y_tr_clas)
y_pred_clas = Clas_model.predict(X_te_clas_minmax)
```

```
from sklearn.metrics import confusion_matrix, classification_report

mat=confusion_matrix(y_te_clas, y_pred_clas)

sns.heatmap(mat.T, cmap=("Blues"), square=True, annot=True, fmt='d', cbar=False,
            xticklabels=['Bad','Good'],
            yticklabels=['Bad','Good'])
plt.xlabel('true label')
plt.ylabel('predicted label');
print(classification_report(y_te_clas, y_pred_clas))
```

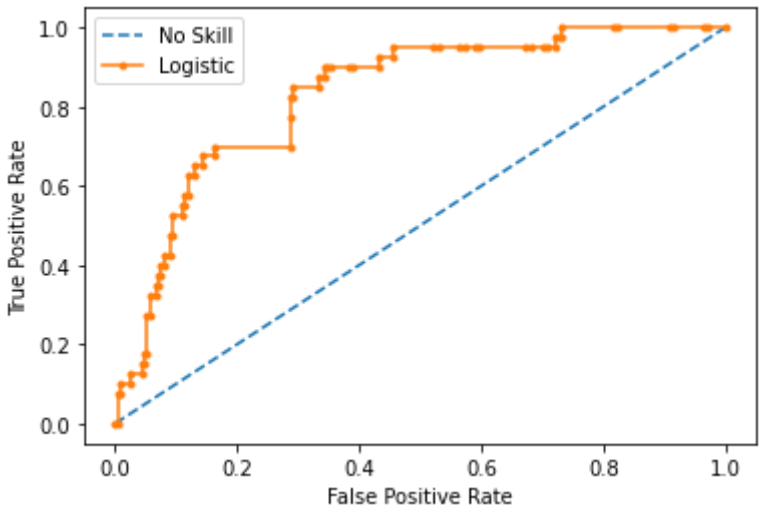
El resultado de la matriz de confusión es el siguiente:



Podemos ver que de los 291 vinos del conjunto, se clasifican correctamente 246, un 84,5%. Este resultado es mucho mejor que con el modelo anterior, aunque se ha realizado una simplificación de las notas, pasando de una escala entre 0 y 10 a una escala dicotómica (bueno o malo). Se muestra a continuación una tabla con algunas métricas generadas, tales como la precisión, especificidad y sensibilidad.

	precision	recall	f1-score	support
0	0.88	0.96	0.91	251
1	0.35	0.15	0.21	40
accuracy			0.85	291
macro avg	0.61	0.55	0.56	291
weighted avg	0.80	0.85	0.82	291

Por último, se muestra también la curva ROC del modelo. El modelo ha obtenido un área bajo la curva de 0.83.



Es posible ver que al disminuir la cantidad de información (pasando de una escala de 0 a 10 a una dicotómica) el modelo mejora su respuesta.

Conclusiones

El objetivo de esta práctica consistía en, a partir de un conjunto de datos que mostraban características físico-químicas de diferentes vinos tintos portugueses junto con su nota de calidad en una cata, y tras realizar una limpieza y análisis exploratorio de dichos datos, construir un modelo que permitiese predecir la calidad de un vino en base a sus características físico-químicas.

Para ello, se construyeron dos modelos diferentes. Primero, un modelo de regresión lineal que nos devuelve una puntuación entre 0 y 10 de los vinos. El segundo modelo consistió en un modelo de clasificación basado en una regresión logística en el que la variable de calidad se dicotomizó entre vinos buenos (aquellos con una nota de 7 o superior) y vinos malos (vinos con nota inferior a 7).

En el modelo de regresión lineal obtenemos un aproximadamente un 60% de acierto para predecir la calidad de un futuro vino en base a sus características físico-químicas. Los errores nunca superaron los dos puntos de diferencia entre el valor predicho y el valor real de los caldos del conjunto de prueba. Las puntuaciones R2 obtenidas se situaban en torno a 0,30. Es posible que el modelo sea mejorable aplicando selección de atributos u optimizando los modelos de regresión.

El modelo dicotómico obtiene aproximadamente un 80% de precisión con el mismo conjunto de prueba. A costa de perder precisión en la puntuación de calidad del vino se ha aumentado la tasa de éxito en la predicción, obteniendo un modelo con una precisión alta. El área bajo la curva ROC es de 0,83, lo que indica un modelo razonablemente bueno.

Código y enlace a GitHub

El código de esta práctica se puede encontrar en el mismo repositorio de GitHub en el que está alojado este informe. EL nombre es PRAC_2.ipynb.

Como ya se ha mencionado anteriormente, el código está escrito en una libreta de Python.

Se puede consultar en Github https://github.com/jpripen/UOC_TCVD_PRAC_2

Tabla de contribuciones

Contribuciones	Firma
Investigación previa	JAFB, JPP
Redacción de las respuestas	JAFB, JPP
Desarrollo código	JAFB, JPP