PRACTICA 2 - Limpieza y Análisis de datos

Autores: Alfonso Jiménez Hernández y Jorge Martín de la Calle

Objetivos

Los objetivos concretos de esta práctica son:

- Aprender a aplicar los conocimientos adquiridos y su capacidad de resolución de problemas en entornos nuevos o poco conocidos dentro de contextos más amplios o multidisciplinares.
- Saber identificar los datos relevantes y los tratamientos necesarios (integración, limpieza y validación) para llevar a cabo un proyecto analítico.
- Aprender a analizar los datos adecuadamente para abordar la información contenida en los datos.
- Identificar la mejor representación de los resultados para aportar conclusiones sobre el problema planteado en el proceso analítico.
- Actuar con los principios éticos y legales relacionados con la manipulación de datos en función del ámbito de aplicación.
- Desarrollar las habilidades de aprendizaje que les permitan continuar estudiando de un modo que tendrá que ser en gran medida autodirigido o autónomo.
- Desarrollar la capacidad de búsqueda, gestión y uso de información y recursos en el ámbito de la ciencia de datos.

Descripción del dataset

Fuente: https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009 (https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009)

El conjunto de datos elegido, está relacionado con variantes rojas del vino portugués “Vinho Verde”, observando las caracteristicas de este como pueden ser alcohol, acidez,… Es un análisis interesante ya que se puede realizar una clasificación de calidad, en este caso el vino, segun sus componentes y características, pudiendose extrapolar, este estudio, a otros productos.

Los atributos que aparecen son los siguientes:

- **fixed.acidity**: acidez fija. Se produce durante la fermentación por acción de las bacterias lácticas, en la denominada fermentación maloláctica (tras la fermentación alcohólica). El ácido láctico es mucho más abundante en tintos que en blancos, ya que estos últimos habitualmente no se someten a la maloláctica.
- **volatile.acidity**: acidez volátil. Es la proporción de vinagre que tiene el vino. Es un factor claramente indeseable pero inevitable. Todo el esfuerzo del enólogo es que sea mínimo. conjunto de todos los ácidos presentes en el vino.
- **citric.acid**: ácido cítrico. Este ácido desaparece lentamente al ser fermentado por las bacterias. No es muy abundante en la uva.
- **residual.sugar**: azúcar residual. El vino principalmente obtiene su dulzor del azúcar de la uva (glucosa y fructosa) que queda sin fermentar y que llamamos azúcares residuales. Es el dulzor que podemos detectar en la cata de vino.
- **chlorides**: cloruros. Las sales minerales del vino confieren, obviamente, el característico sabor salado del vino. Los principales componentes de las sales del vino son: fosfatos, sulfatos, cloruros, sulfitos, flúor, silicio, yodo, bromo, boro, zinc, calcio, etc…
- **free.sulfur.dioxide**: dióxido de azufre libre. La parte que no combina del dióxido de azufre, juega un papel importante en la preservación del vino de las alteraciones oxidantes y de algunos microorganismos.
- **total.sulfur.dioxide**: dióxido de azufre total. En la elaboración del vino, es importante añadir SO2 es evitar la oxidación.
- **density**: densidad. La densidad aparente de un vino puede indicar su contenido en alcohol.
- **pH**: nivel pH. Es la acidez real del vino.
- **sulphates**: sulfatos. Los sulfatos de sodio y calcio aparecen en el agua y por lo tanto la uva y el vino pueden contenerlos. Lo ideal es que sea lo menor posible.
- **alcohol**: cantidad de alcohol.
- **quality**: calidad. Es la variable objetivo.

Integración y selección de los datos de interés a analizar

En primer lugar realizamos la carga del dataset y analizamos su composición y estructura.

In [1]:

```
# Carga de Librerías
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
import statsmodels.api as sm
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

In [31]:

```
df = pd.read_csv('winequality-red.csv', sep = ',', decimal = '.')
```

Mostramos una muestra de 5 filas, con todos sus atributos, para comprobar que se ha realizado correctamente la carga

In [32]:

```
df.head(5)
```

Out[32]:

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

A continuación obtendremos la información de como está compuesto el dataset, con datos de su estructura:

In [33]:

```
#Forma, tamaño y número de valores del dataset
print('Forma del dataset: \n', df.shape[0], 'filas \n', df.shape[1], 'columnas')
print('\n')
print('Tamaño del dataset: ', df.size, ' registros')
print('\n')
print('Numero de valores del dataset: ')
df.nunique()
```

```
Forma del dataset:
 1599 filas
 12 columnas


Tamaño del dataset:  19188  registros


Numero de valores del dataset:
```

Out[33]:

```
fixed acidity            96
volatile acidity        143
citric acid              80
residual sugar           91
chlorides               153
free sulfur dioxide      60
total sulfur dioxide    144
density                 436
pH                       89
sulphates                96
alcohol                  65
quality                   6
dtype: int64
```

Como última estudio del apartado realizaremos un resumen estadístico básico:

In [34]:

```
#Resumen estadístico
df.describe()
```

Out[34]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.(
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.4
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.8
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.(
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.(
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.(
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.(
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.(

Nota: No realizaremos ningun filtrado previo de la información, ya que se pretende analizar todo el set de datos, con las distintas muestras de vinos

Limpieza de los datos

¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

En primer lugar se identifica si existen valores vacíos

In [35]:

```
# Valores vacíos por columna
print('Valores vacíos por columna: ')
df.isna().sum()
```

```
Valores vacíos por columna:
```

Out[35]:

```
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
```

Puesto que el conjunto de datos está compuesto por variables continuas, se debe analizar si el valor 0 dentro de su rango aceptado de valores.

- El primer paso será ver que variables tienen valor a 0.

In [36]:

```
(df == 0).sum()
```

Out[36]:

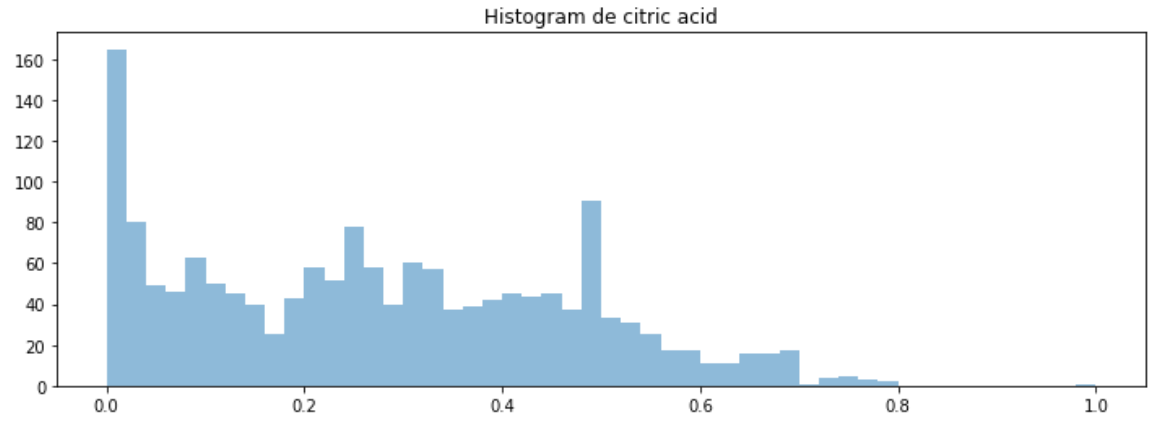
```
fixed acidity             0
volatile acidity          0
citric acid             132
residual sugar            0
chlorides                 0
free sulfur dioxide       0
total sulfur dioxide      0
density                   0
pH                        0
sulphates                 0
alcohol                   0
quality                   0
dtype: int64
```

La única variable que tiene valores 0 es citric acidity.

- El siguiente paso será saber si el valor 0 dentro de la variable **citric acidity** es válido o no, para ello se realizará un histograma para poder asimilar e identificar la distribución de la variable.

In [37]:

```
f, ax = plt.subplots(1, 1, figsize=(12,4))
plt.hist(df['citric acid'], bins = 50, alpha=0.5)
plt.title("Histogram de citric acid")
plt.show()
```



Podemos considerar que sí existen valores vacíos para la variable Citric Acid, y que la variable sigue una distribución más o menos uniforme.

A la hora de tratar los valores nulos, existen diferentes metodos:

- **Completar manualmente los registros faltantes:** Es el método más sencillo de implementar, ya que consiste en recoger el valor perdido y sustituirlo por un valor conocido, lo que hace que el conocimiento de los atributos sea muy importante a la hora de aplicar está técnica. A su vez, si el número de valores perdidos es elevado está técnica puede ser costosa.
- **Reemplazar el conjunto de valores perdidos por una misma constante o etiqueta:** En caso de que el número de valores perdidos sea elevado o no se conozcan los valores por los cuales deben ser sustituidos, y además la perdida de esta información no suponga de importancia para el estudio a realizar, se pueden sustituir, por una etiqueta común, que nos informe del valor perdido.
- **Reemplazan los registros perdidos por una misma medida de tendencia central:** En el caso que la pérdida de esos valores suponga un impacto importante para el estudio a realizar, podemos sustituir los valores, por ejemplo por la media o mediana de la población con las características similares a la muestra que posea el valor perdido, aunque en este caso no será de forma precisa, pero no nos llevará a cometer errores de gran impacto en nuestro análisis.
- **Imputación de los valores perdidos:** En el caso que el valor perdido se quiera precisar con más acierto que por ejemplo el reemplazo de la media o mediana, se pude utilizar un método que reemplace el valor perdido de una muestra, por el valor del de otra muestra de características en sus atributos similares.

En este caso optaremos por **eliminar aquellos registros con valores vacíos** en la variable **Citric Acid**, debido a 2 razones:

- El desconocimiento de un valor adecuado por el cual sustituir los valores vacíos.
- La distribución de la variable es Uniforme, sutituir los valores vacíos por un estadístico cambiaría dicha distribución de la variable, es decir si se aplica por ejemplo la media para sustitución de los valores vacíos la distribución cambiaría de ser uniforme a ser una distribución normal.

In [38]:

```
df = df[df['citric acid'] != 0]
```

Identificación y tratamiento de valores extremos

Antes de empezar en el apartado, debemos separar la variable target del set de datos.

In [39]:

```
df_data = df[list(df.columns)[:-1]]
df_target = df['quality']
```

Para identificar y poder tratar los valores extremos del conjunto de datos, realizaremos una representación de ***boxplot***, analizando la distribución de datos basada en "mínimo", primer cuartil (Q1), mediana, tercer cuartil (Q3) y "máximo".

Es necesario normalizar los datos para poder observar los outliers de todos los atributos de forma visual

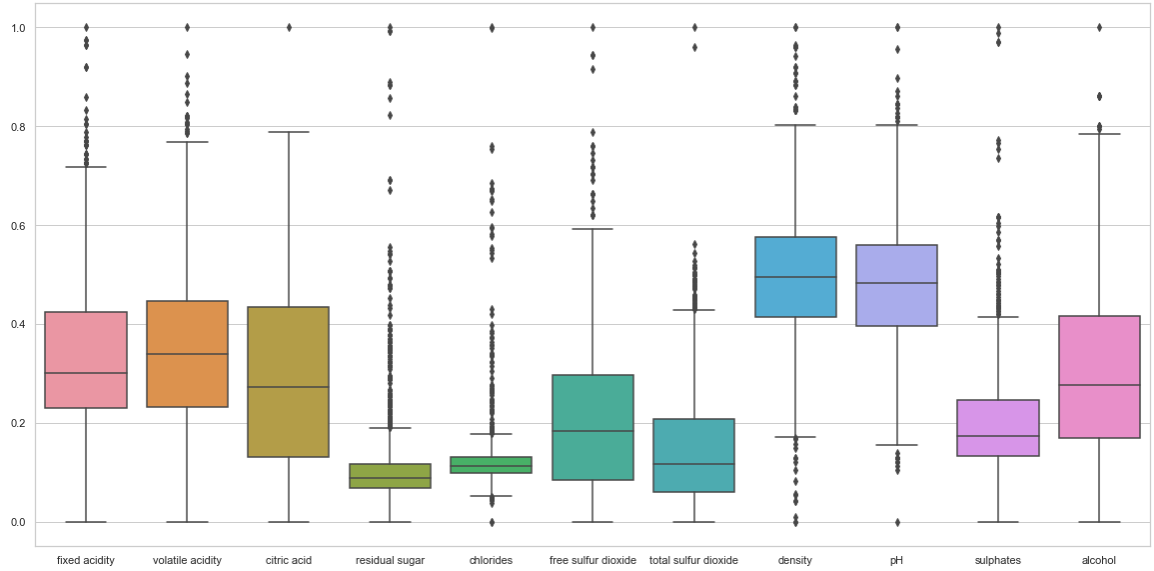
In [40]:

```
min_max_scaler = preprocessing.MinMaxScaler()
np_min_max_scaled = min_max_scaler.fit_transform(df_data)
min_max_scaled_red_df_data = pd.DataFrame(np_min_max_scaled, columns = [name for name i
n list(df_data)])

fig = plt.figure(figsize = (20,10))
sns.set(style="whitegrid")
sns.boxplot(data = min_max_scaled_red_df_data)
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15981b98438>
```



Se dan 2 situaciones para estos tipos de datos:

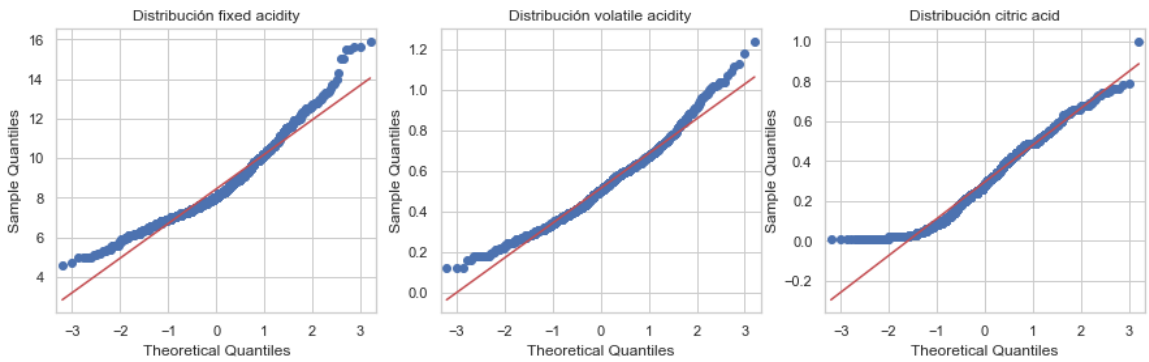
- Los valores extremos no suponen una gran pérdida de información para el conjunto de datos, en cuyo caso se eliminarán.
- En el caso que los valores aporten gran parte de la información del conjunto de datos deberán ser corregidos o tenidos en cuenta.

En primer lugar analizaremos las distribuciones de las variables

Si la distribución **NO** es normal, tendremos en cuenta los outliers

In [45]:

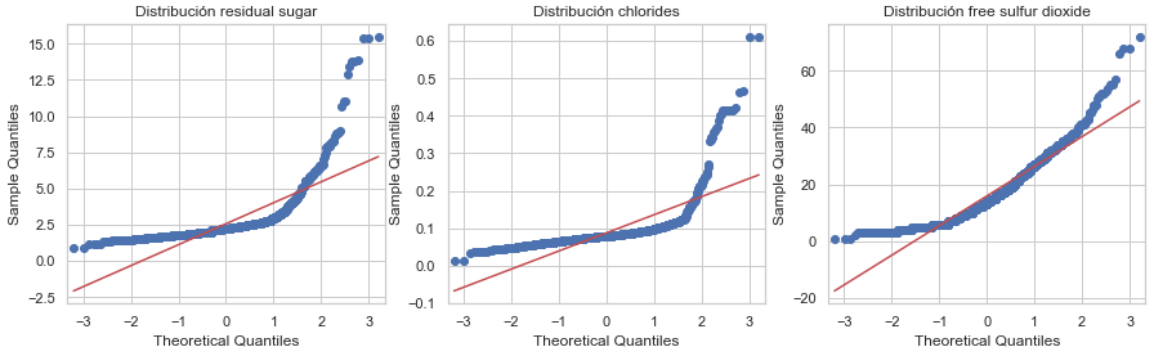
```
f, ax = plt.subplots(1, 3, figsize=(15,4))
sm.qqplot(df['fixed acidity'], line = 's', ax = ax[0])
ax[0].set_title('Distribución fixed acidity')
sm.qqplot(df['volatile acidity'], line = 's', ax = ax[1])
ax[1].set_title('Distribución volatile acidity')
sm.qqplot(df['citric acid'], line = 's', ax = ax[2])
ax[2].set_title('Distribución citric acid')
plt.show()
```



Tras analizar la distribución de las 3 variables observamos que es una distribución normal, por lo tanto procederemos a eliminar los outliers de las 3 variables.

In [46]:

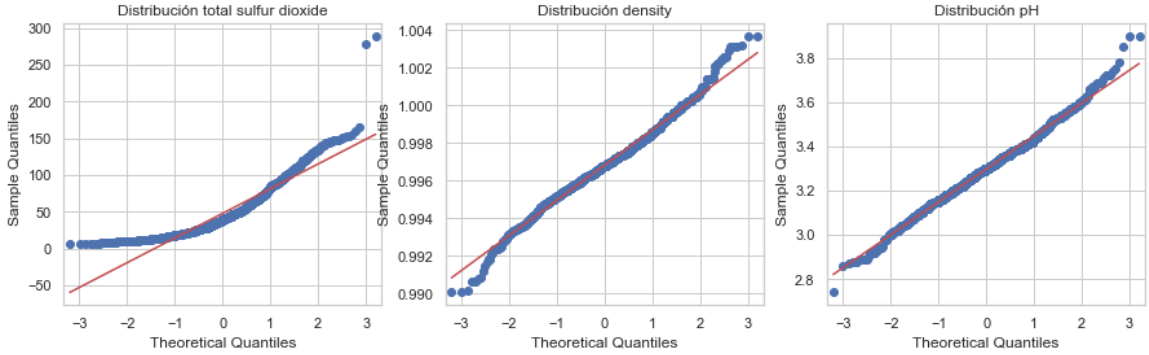
```
f, ax = plt.subplots(1, 3, figsize=(15,4))
sm.qqplot(df['residual sugar'], line = 's', ax = ax[0])
ax[0].set_title('Distribución residual sugar')
sm.qqplot(df['chlorides'], line = 's', ax = ax[1])
ax[1].set_title('Distribución chlorides')
sm.qqplot(df['free sulfur dioxide'], line = 's', ax = ax[2])
ax[2].set_title('Distribución free sulfur dioxide')
plt.show()
```



Tras analizar las distribuciones de las 3 variables, se observa que no siguen una distribución normal, por lo tanto se decide no realizar ningun tratamiento a los outliers pertenecientes a estas 3 variables.

In [47]:

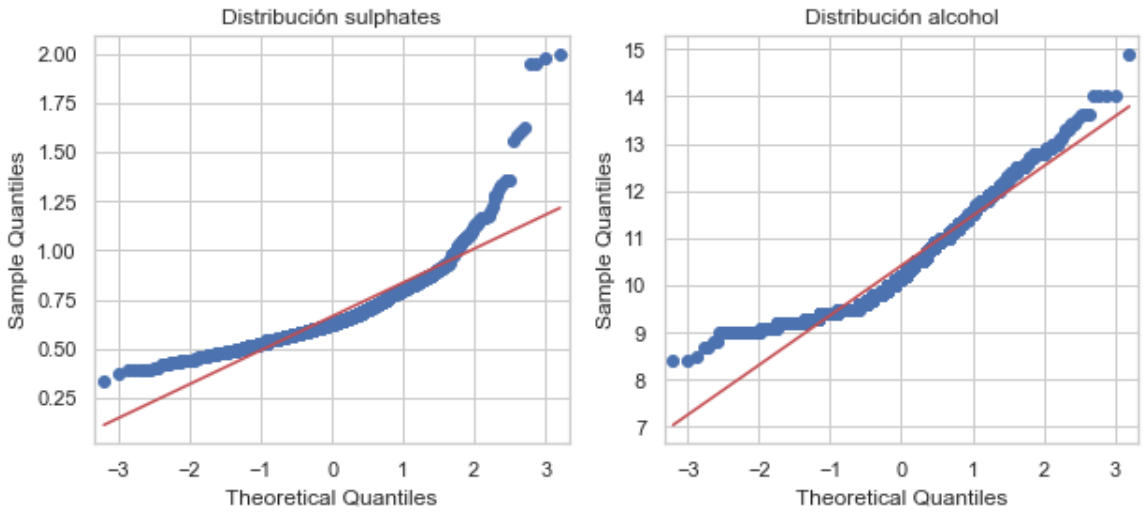
```
f, ax = plt.subplots(1, 3, figsize=(15,4))
sm.qqplot(df['total sulfur dioxide'], line = 's', ax = ax[0])
ax[0].set_title('Distribución total sulfur dioxide')
sm.qqplot(df['density'], line = 's', ax = ax[1])
ax[1].set_title('Distribución density')
sm.qqplot(df['pH'], line = 's', ax = ax[2])
ax[2].set_title('Distribución pH')
plt.show()
```



Se considera una distribución normal de las tres variables, por tanto se realizará tratamiento sobre sus valores extremos.

In [48]:

```
f, ax = plt.subplots(1, 2, figsize=(10,4))
sm.qqplot(df['sulphates'], line = 's', ax = ax[0])
ax[0].set_title('Distribución sulphates')
sm.qqplot(df['alcohol'], line = 's', ax = ax[1])
ax[1].set_title('Distribución alcohol')
plt.show()
```



Al igual que las variables anteriores, se observa que ambas siguen una distribución normal, por lo que se realizará tratamiento sobre los valores extremos de sus variables.

Por tanto las variables sobre los que se realizará la limpieza de valores extremos son:

- fixed acidity
- volatile acidity
- citric acidity
- total sulfur dioxide
- density
- pH
- sulphates
- alcohol

Las variables sobre las que no se realizará tratamiento de outliers son:

- residual sugar
- chlorides
- free sulfur dioxide

En segundo lugar observaremos el número de outliers por cada atributo a tratar.

In [52]:

```
# Solo las variables que queremos limpiar de valores extremos
df_to_clean = df[['fixed acidity','volatile acidity','citric acid',
                  'total sulfur dioxide','density','pH','sulphates','alcohol']]
```

In [53]:

```
# Calculamos los cuantiles y el rango intercuantil
Q1 = df_to_clean.quantile(0.25)
Q3 = df_to_clean.quantile(0.75)
IQR = Q3 - Q1
```

In [54]:

```
# numero de outliers por variable
((df_to_clean < (Q1 - 1.5 * IQR)) | (df_to_clean > (Q3 + 1.5 * IQR))).sum()
```

Out[54]:

```
fixed acidity           30
volatile acidity        18
citric acid              1
total sulfur dioxide    45
density                 43
pH                      26
sulphates               54
alcohol                  9
dtype: int64
```

A continuación lugar, identificamos cuales son outliers

In [55]:

```
# Generamos una funcion que identifique los outliers dentro del dataframe
def outliers_detection(data):

    # Convertimos a array el dataframe para obtener los quantiles
    data = np.array(data)

    # Calculamos los Quantiles 25 y 75 y la zona intercuantil
    Q1 = np.quantile(data, 0.25)
    Q3 = np.quantile(data, 0.75)
    IQR = Q3 - Q1

    limite_inf = Q1 - 1.5 * IQR
    limite_sup = Q3 + 1.5 * IQR

    outliers = [] # Definimos la lista de outliers
    for x in list(data): # Recorremos la lista
        if (x < limite_inf or x > limite_sup):
            outliers.append(x) # Si es otlier, nos quedamos con el valor
        else:
            outliers.append('No outlier') # Si no es outliers le ponemos la etiqueta

    return outliers
```

In [71]:

```
# Aplicamos la función para cada columna
outliers_focused = {}
for name in list(df_to_clean):
    outliers_focused.setdefault(name, outliers_detection(df_to_clean[name]))

# Convertimos a dataframe
red_df_outliers_focused = pd.DataFrame(data = outliers_focused)
red_df_outliers_focused.head(10)
```

Out[71]:

	fixed acidity	volatile acidity	citric acid	total sulfur dioxide	density	pH	sulphates	alcohol
0	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier
1	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier
2	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier
3	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier
4	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier
5	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier
6	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier
7	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	1.56	No outlier
8	No outlier	No outlier	No outlier	145	No outlier	No outlier	No outlier	No outlier
9	No outlier	No outlier	No outlier	148	No outlier	No outlier	No outlier	No outlier

Todos aquellos que tienen almenos un valor extremo en alguno de sus atributos,es considerado outlier

In [57]:

```
# Nos quedamos con aquellas filas en almenos conyengan un valor outlier
series_list = []
for index, row in red_df_outliers_focused.iterrows():
    for name in list(red_df_outliers_focused):
        if type(row[name]) == np.float64: # Comparamos la fila mediante el tipo, numeri
co
            series_list.append(row)
            break

red_df_outliers = pd.DataFrame(series_list, columns=list(red_df_outliers_focused))
red_df_outliers.head()
```

Out[57]:

	fixed acidity	volatile acidity	citric acid	total sulfur dioxide	density	pH	sulphates	alcohol
7	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	1.56	No outlier
8	No outlier	No outlier	No outlier	145	No outlier	No outlier	No outlier	No outlier
9	No outlier	No outlier	No outlier	148	No outlier	No outlier	No outlier	No outlier
11	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	1.28	No outlier
13	No outlier	No outlier	No outlier	No outlier	No outlier	No outlier	1.08	No outlier

Añadimos la calidad correspondiente de cada vino a los registros identificados como outliers

In [58]:

```
# Cocatenamos y añadimos la calidad y las variables faltantes correspondiente a cada fi
la con outliers
# Eliminamos los que no hayan cruzado
outliers_quality = pd.concat((red_df_outliers, df[['residual sugar','chlorides', 'free
 sulfur dioxide']], df['quality']), axis=1).dropna()
```

Observamos la cantidad de vinos que hay por las distintas calidades, y los outliers que hay para cada calidad y el porcentaje que suponen los outliers sobre cada tipo de calidad.

In [67]:

```
'''
El significado de las listas es el siguiente:

1. Columna Quality: informa del tipo de calidad
2. Columna N_outliers: informa del numero de filas con outliers
3. Columna N_total: numero de filas correspondiente por calidad
4. Columna Per_Outliers: El peso que tiene el numero de filas con outliers respecto el
 total de filas por calidad

Con esta información podemos decidir que hacer con los outliers
'''

print('Calidades: ', list(df['quality'].unique()))
print('Nº de vinos por calidad: ', list(df['quality'].value_counts()))
print('\n')
print('Calidades con outliers: ', list(outliers_quality['quality'].unique()))
print('Nº outliers por calidad: ', list(outliers_quality['quality'].value_counts()))
```

```
Calidades:  [5, 6, 7, 4, 8, 3]
Nº de vinos por calidad:  [624, 584, 191, 43, 18, 7]


Calidades con outliers:  [7.0, 5.0, 6.0, 4.0]
Nº outliers por calidad:  [74, 66, 19, 5]
```

Podemos concluir en que la eliminación de estos valores extremos no afectará de forma crítica alconjunto de datos, las unicas muestras que se verán más afectadas serán las de la calidad intermedia.

En el caso de que la pérdida de esta información fuera más crítica, seplantearía la opción de sustituir los outliers por un estadistico como puede ser la media o mediana.

Procedemos a la eliminación de los outliers

In [68]:

```
outliers_indices = red_df_outliers.index.tolist()
```

In [69]:

```
df_cleans = df_data.drop(df_data.index[outliers_indices])
df_cleans = pd.concat((df_cleans, df['quality']), axis=1).dropna()
df_cleans = df_cleans.reset_index()
df_cleans = df_cleans.drop(['index'], axis=1)
df_cleans.head(10)
```

Out[69]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alc
0	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	
1	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	
2	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	
3	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	
4	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	
5	6.7	0.58	0.08	1.8	0.097	15.0	65.0	0.9959	3.28	0.54	
6	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	
7	8.5	0.28	0.56	1.8	0.092	35.0	103.0	0.9969	3.30	0.75	
8	7.4	0.59	0.08	4.4	0.086	6.0	29.0	0.9974	3.38	0.50	
9	8.9	0.22	0.48	1.8	0.077	29.0	60.0	0.9968	3.39	0.53	

Por último como comprobación, comprobaremos que varían, respecto a los datos inicales, los estadisticos básicos sin los outliers

In [70]:

```
df_cleans.describe()
```

Out[70]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total
count	1284.000000	1284.000000	1284.000000	1284.000000	1284.000000	1284.000000	1284.
mean	8.360125	0.507212	0.283964	2.487111	0.083086	15.419393	43.
std	1.564358	0.160528	0.178232	1.160443	0.027382	9.921610	28.
min	5.200000	0.120000	0.010000	1.200000	0.038000	1.000000	6.
25%	7.200000	0.380000	0.127500	1.900000	0.071000	7.000000	22.
50%	8.000000	0.500000	0.270000	2.200000	0.079000	13.000000	37.
75%	9.300000	0.611250	0.420000	2.600000	0.089000	21.000000	60.
max	12.700000	0.980000	0.780000	13.900000	0.387000	68.000000	128.

Análisis de los datos y Representación de los resultados

Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar)

Para los análisis que se van a realizar utilizaremos la información resultante de la limpieza de datos.

A continuación se detallan y planifican los distintos análisis que se realizarán:

- **Análisis estadístico descriptivo**, similar al realizado anteriormente, simplemente se tratará de obtener información de estadísticos básicos, comos son las medidas de tendencia central y las medidas de dispersión. Se propondrá realizar un análisis descriptivo para cada tipo de calidad del vino.
- **Análisis de correlación entre variables**, que consiste en medir la asociación entre dos variables. esto nos permitirá descartar variables que estén altamnente relacionadas, y no aporten variabilidad a los datos, para a la hora de aplicar nuestro modelo de clasificación, simplificar.
- **Aplicación de un modelo supervisado de clasificación**, que permita clasificar nuevos vinos, con caracteristicas similares a las que existen en el set de datos, de forma que en función del entrenamiento del algoritmo y definición de las reglas de clasificación, se puedan predecir la calidad de futuros vinos, en función de sus caracteristicas.

Para simplificar realizaremos una nueva clasificación de las calidades de los vinos:

- quality > 6.5 ==> Bueno
- quality < 6.5 ==> Media

In [72]:

```
new_quality = []
for i in df_cleans['quality']:
    if i > 6.5:
        new_quality.append(1)
    elif i < 6.5:
        new_quality.append(2)

df_cleans['new_quality'] = new_quality
```

Generamos un CSV con la información final que utilizaremos en nuestro análisis:

In [73]:

```
df_cleans.to_csv('red_wine_clean.csv', index = False, header = True)
```

Comprobación de la normalidad y homogeneidad de la varianza

- A continuación realizaremos una observación de la normalidad de cada variable de forma visual.
- Adicionalmente realizaremos una observación de la normalidad de cada variable mediante el test Shapiro.

Un gráfico de Q-Q (quantile-quantile) es un método gráfico para comparar dos distribuciones de probabilidad al trazar sus cuantiles uno contra el otro.En este caso, lo ideal es que los puntos se acerquen a una recta diagonal.

El Shapiro test, nos indica si la variable sigue o no normalidad o no:

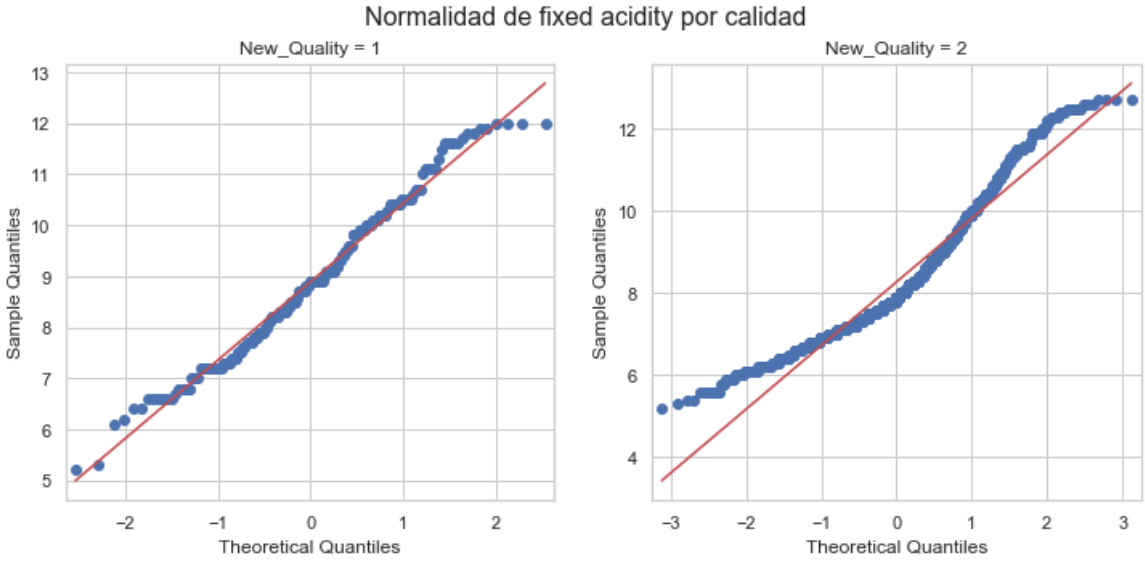
- p-value < 0.05. Rechazamos la hipótesis, es decir no sigue distribución normal.
- p-value >= 0.05. No se puede rezachar la hipótesis, por lo que sigue una distribución normal.

In [85]:

```
from scipy import stats
```

In [74]:

```
f, ax = plt.subplots(1, 2, figsize=(12,5))
f.suptitle('Normalidad de fixed acidity por calidad', fontsize=16)
sm.qqplot(df_cleans[df_cleans['new_quality'] == 1]['fixed acidity'], line = 's', ax = a
x[0])
ax[0].set_title('New_Quality = 1')
sm.qqplot(df_cleans[df_cleans['new_quality'] == 2]['fixed acidity'], line = 's', ax = a
x[1])
ax[1].set_title('New_Quality = 2')
plt.show()
```



In [92]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 1]['fixed acidit
y'])[1])
```

```
p-value:  0.01915414072573185
```

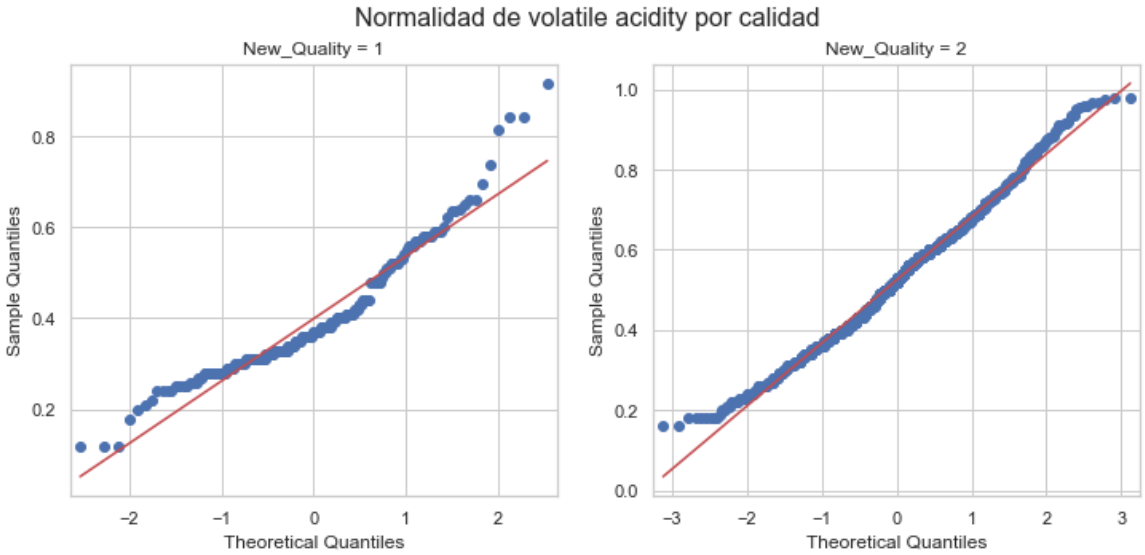
In [111]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 2]['fixed acidit
y'])[1])
```

```
p-value:  6.2340026617564945e-21
```

In [75]:

```
f, ax = plt.subplots(1, 2, figsize=(12,5))
f.suptitle('Normalidad de volatile acidity por calidad', fontsize=16)
sm.qqplot(df_cleans[df_cleans['new_quality'] == 1]['volatile acidity'], line = 's', ax
= ax[0])
ax[0].set_title('New_Quality = 1')
sm.qqplot(df_cleans[df_cleans['new_quality'] == 2]['volatile acidity'], line = 's', ax
= ax[1])
ax[1].set_title('New_Quality = 2')
plt.show()
```



In [93]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 1]['volatile aci
dity'])[1])
```

```
p-value:  1.1012441092361769e-07
```

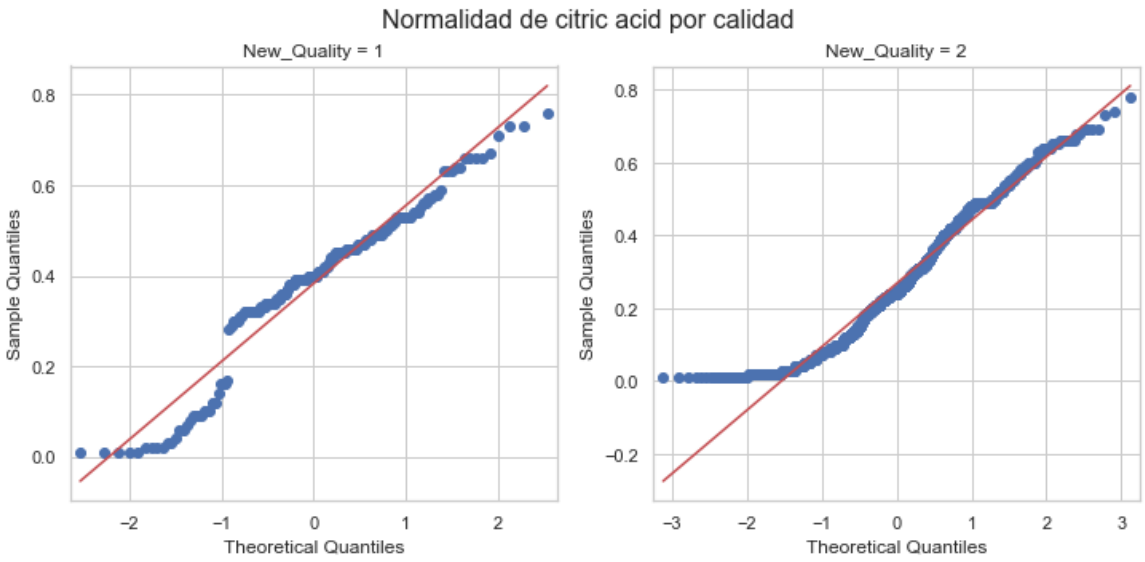
In [112]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 2]['volatile aci
dity'])[1])
```

```
p-value:  6.5028489188989624e-06
```

In [76]:

```
f, ax = plt.subplots(1, 2, figsize=(12,5))
f.suptitle('Normalidad de citric acid por calidad', fontsize=16)
sm.qqplot(df_cleans[df_cleans['new_quality'] == 1]['citric acid'], line = 's', ax = ax[
0])
ax[0].set_title('New_Quality = 1')
sm.qqplot(df_cleans[df_cleans['new_quality'] == 2]['citric acid'], line = 's', ax = ax[
1])
ax[1].set_title('New_Quality = 2')
plt.show()
```



In [95]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 1]['citric acid'
])[1])
```

```
p-value:  9.73129203885037e-07
```

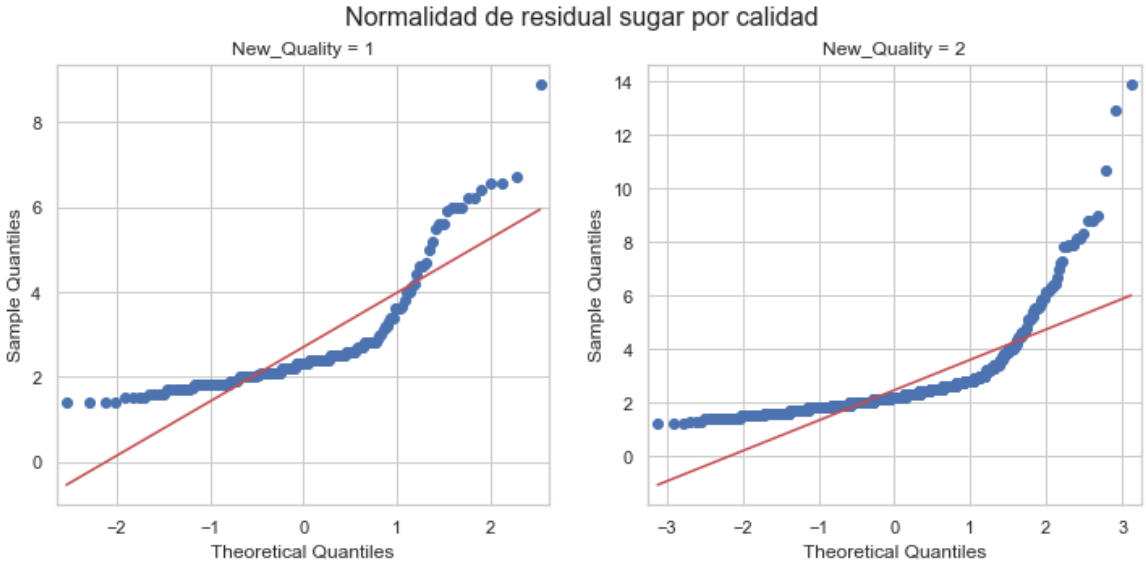
In [113]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 2]['citric acid'
])[1])
```

```
p-value:  1.0772660976131248e-16
```

In [77]:

```
f, ax = plt.subplots(1, 2, figsize=(12,5))
f.suptitle('Normalidad de residual sugar por calidad', fontsize=16)
sm.qqplot(df_cleans[df_cleans['new_quality'] == 1]['residual sugar'], line = 's', ax =
ax[0])
ax[0].set_title('New_Quality = 1')
sm.qqplot(df_cleans[df_cleans['new_quality'] == 2]['residual sugar'], line = 's', ax =
ax[1])
ax[1].set_title('New_Quality = 2')
plt.show()
```



In [98]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 1]['residual sug
ar'])[1])
```

```
p-value:  1.834119748312445e-16
```

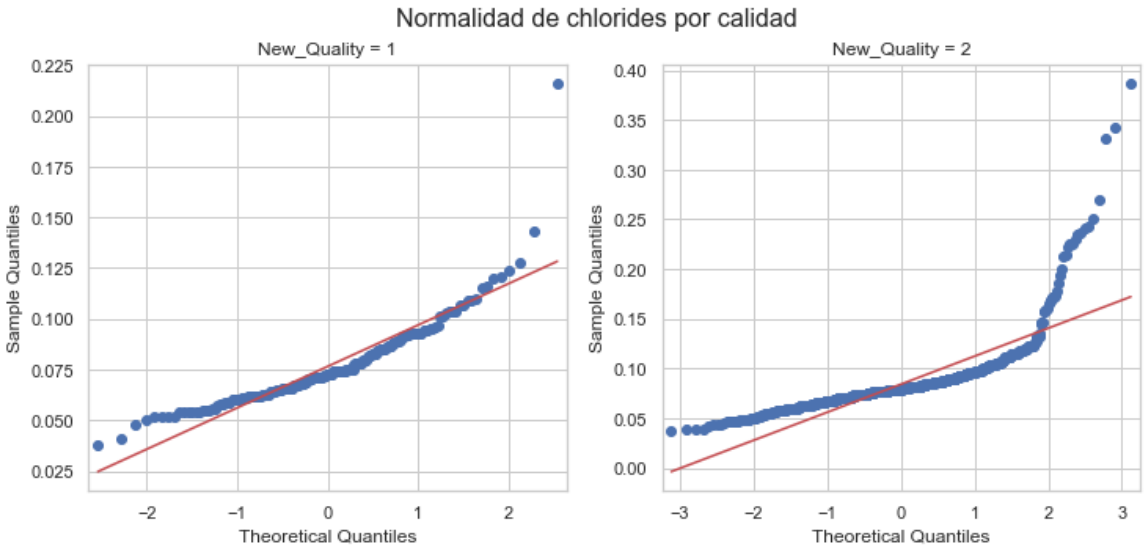
In [114]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 2]['residual sug
ar'])[1])
```

```
p-value:  2.1019476964872256e-44
```

In [78]:

```
f, ax = plt.subplots(1, 2, figsize=(12,5))
f.suptitle('Normalidad de chlorides por calidad', fontsize=16)
sm.qqplot(df_cleans[df_cleans['new_quality'] == 1]['chlorides'], line = 's', ax = ax[0
])
ax[0].set_title('New_Quality = 1')
sm.qqplot(df_cleans[df_cleans['new_quality'] == 2]['chlorides'], line = 's', ax = ax[1
])
ax[1].set_title('New_Quality = 2')
plt.show()
```



In [100]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 1]['chlorides'])
[1])
```

p-value: 3.8706308123914734e-12

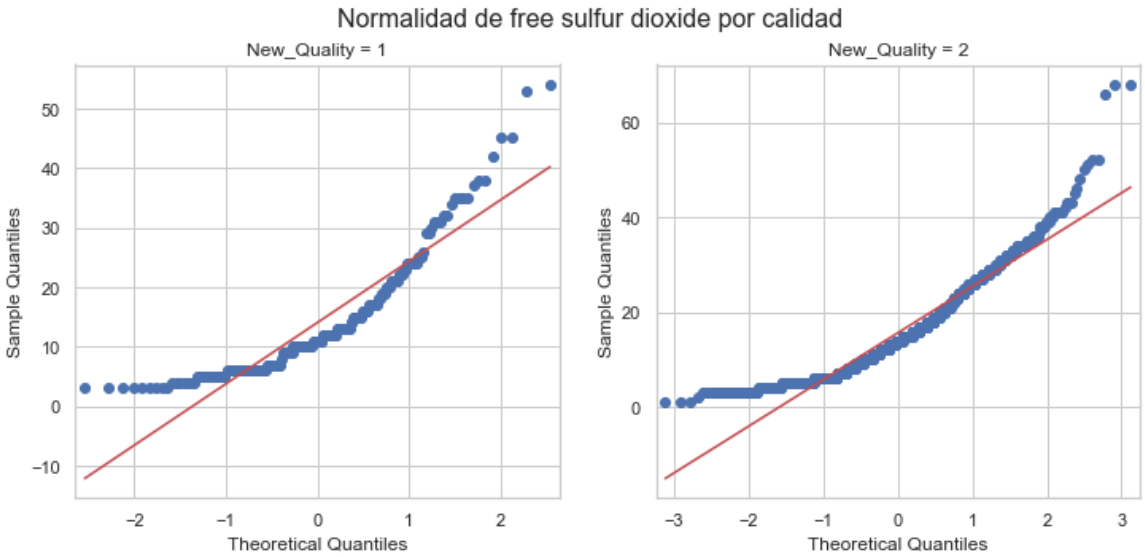
In [115]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 2]['chlorides'])
[1])
```

p-value: 4.834479701920619e-43

In [79]:

```
f, ax = plt.subplots(1, 2, figsize=(12,5))
f.suptitle('Normalidad de free sulfur dioxide por calidad', fontsize=16)
sm.qqplot(df_cleans[df_cleans['new_quality'] == 1]['free sulfur dioxide'], line = 's',
ax = ax[0])
ax[0].set_title('New_Quality = 1')
sm.qqplot(df_cleans[df_cleans['new_quality'] == 2]['free sulfur dioxide'], line = 's',
ax = ax[1])
ax[1].set_title('New_Quality = 2')
plt.show()
```



In [102]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 1]['free sulfur
 dioxide'])[1])
```

p-value: 1.8420369396476843e-12

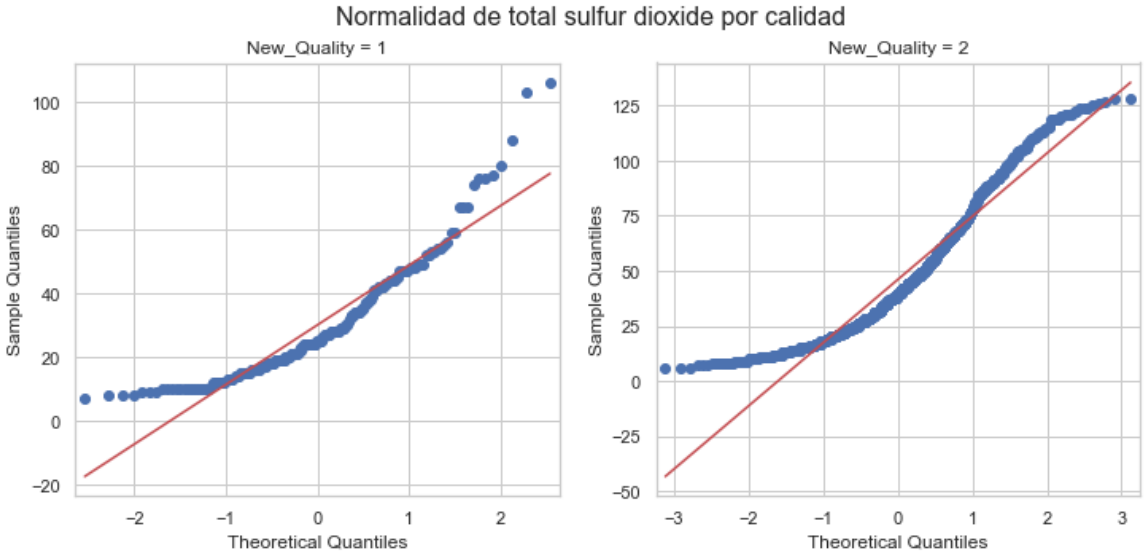
In [116]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 2]['free sulfur
 dioxide'])[1])
```

p-value: 2.716939312283617e-24

In [80]:

```
f, ax = plt.subplots(1, 2, figsize=(12,5))
f.suptitle('Normalidad de total sulfur dioxide por calidad', fontsize=16)
sm.qqplot(df_cleans[df_cleans['new_quality'] == 1]['total sulfur dioxide'], line = 's',
ax = ax[0])
ax[0].set_title('New_Quality = 1')
sm.qqplot(df_cleans[df_cleans['new_quality'] == 2]['total sulfur dioxide'], line = 's',
ax = ax[1])
ax[1].set_title('New_Quality = 2')
plt.show()
```



In [105]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 1]['total sulfur
dioxide'])[1])
```

```
p-value:  1.3759222927678394e-10
```

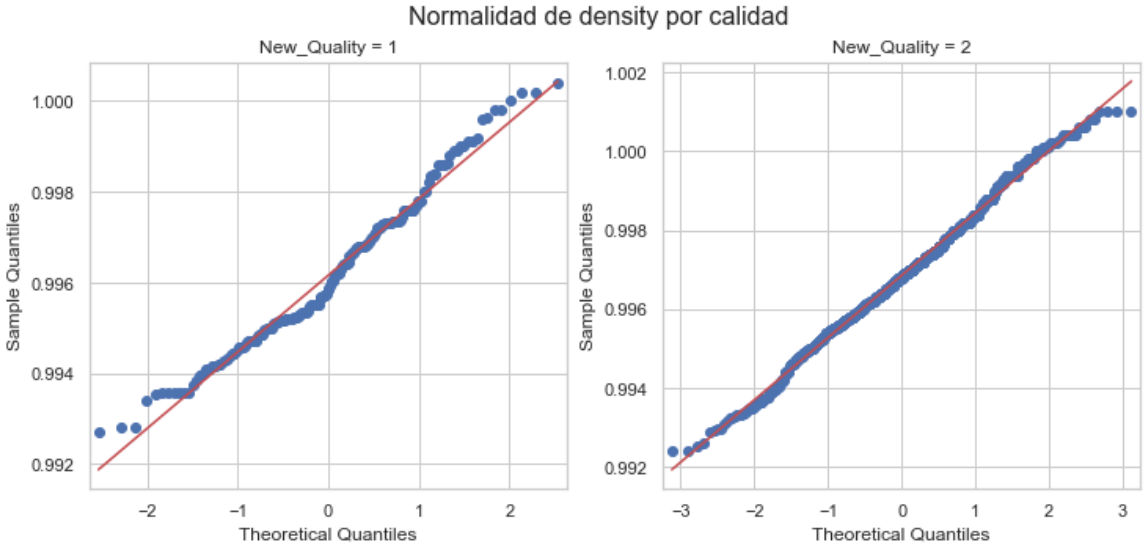
In [117]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 2]['total sulfur
dioxide'])[1])
```

```
p-value:  2.3545754638246664e-23
```

In [81]:

```
f, ax = plt.subplots(1, 2, figsize=(12,5))
f.suptitle('Normalidad de density por calidad', fontsize=16)
sm.qqplot(df_cleans[df_cleans['new_quality'] == 1]['density'], line = 's', ax = ax[0])
ax[0].set_title('New_Quality = 1')
sm.qqplot(df_cleans[df_cleans['new_quality'] == 2]['density'], line = 's', ax = ax[1])
ax[1].set_title('New_Quality = 2')
plt.show()
```



In [107]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 1]['density'])[1
])
```

```
p-value:  0.0036729429848492146
```

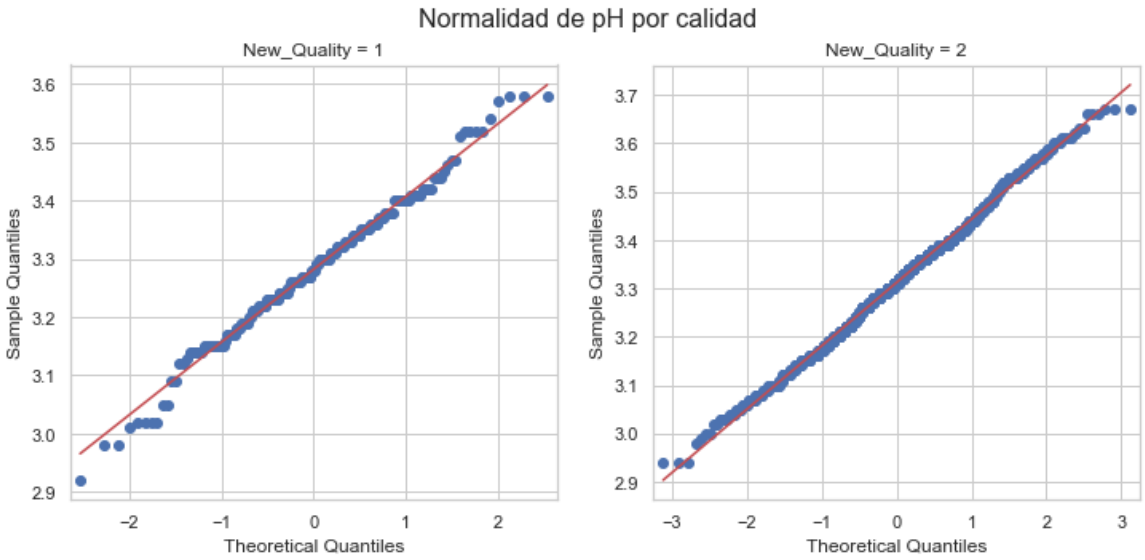
In [118]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 2]['density'])[1
])
```

```
p-value:  0.0025953238364309072
```

In [82]:

```
f, ax = plt.subplots(1, 2, figsize=(12,5))
f.suptitle('Normalidad de pH por calidad', fontsize=16)
sm.qqplot(df_cleans[df_cleans['new_quality'] == 1]['pH'], line = 's', ax = ax[0])
ax[0].set_title('New_Quality = 1')
sm.qqplot(df_cleans[df_cleans['new_quality'] == 2]['pH'], line = 's', ax = ax[1])
ax[1].set_title('New_Quality = 2')
plt.show()
```



In [109]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 1]['pH'])[1])
```

```
p-value:  0.2839526832103729
```

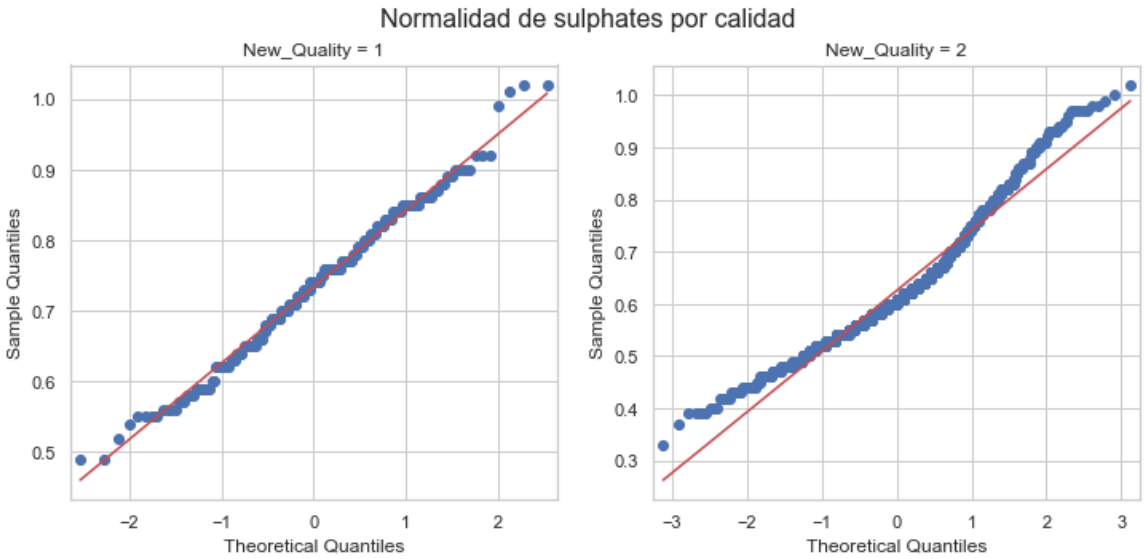
In [119]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 2]['pH'])[1])
```

```
p-value:  0.007029709871858358
```

In [83]:

```
f, ax = plt.subplots(1, 2, figsize=(12,5))
f.suptitle('Normalidad de sulphates por calidad', fontsize=16)
sm.qqplot(df_cleans[df_cleans['new_quality'] == 1]['sulphates'], line = 's', ax = ax[0
])
ax[0].set_title('New_Quality = 1')
sm.qqplot(df_cleans[df_cleans['new_quality'] == 2]['sulphates'], line = 's', ax = ax[1
])
ax[1].set_title('New_Quality = 2')
plt.show()
```



In [120]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 1]['sulphates'])
[1])
```

```
p-value:  0.22605502605438232
```

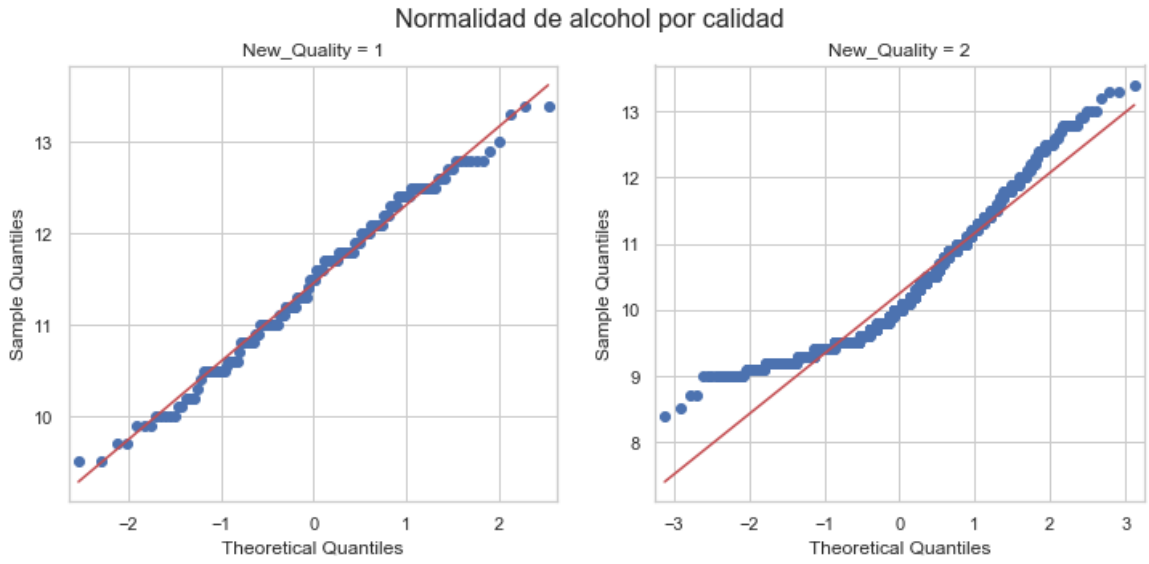
In [121]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 2]['sulphates'])
[1])
```

```
p-value:  3.5561490459112174e-17
```

In [84]:

```
f, ax = plt.subplots(1, 2, figsize=(12,5))
f.suptitle('Normalidad de alcohol por calidad', fontsize=16)
sm.qqplot(df_cleans[df_cleans['new_quality'] == 1]['alcohol'], line = 's', ax = ax[0])
ax[0].set_title('New_Quality = 1')
sm.qqplot(df_cleans[df_cleans['new_quality'] == 2]['alcohol'], line = 's', ax = ax[1])
ax[1].set_title('New_Quality = 2')
plt.show()
```



In [123]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 1]['alcohol'])[1
])
```

```
p-value:  0.09047840535640717
```

In [122]:

```
print('p-value: ', stats.shapiro(df_cleans[df_cleans['new_quality'] == 2]['alcohol'])[1
])
```

```
p-value:  6.313812338004654e-23
```

Conclusión:

Tras analizar las distribuciones de ambos conjuntos mediante los gráficos Q-Q podemos observar que variables cumplen más o menos el principio de normalidad y cuales no:

- **fixed acidity**: Visualmente podemos decir que cumple más normalidad para calidades altas que para bajas, pero mediante el test shapiro, se concluye que **no sigue una distribución normal**.
- **volatile acid**: Cumple más normalidad para calidades bajas que para altas, pero mediante el test shapiro, se concluye que **no sigue una distribución normal**.
- **ciric acid**: No cumple aparentemente de forma visual el principio de normalidad, además existe una pronunciada cola inferior. El test Shapiro confirma esto.
- **residual sugar**: Podemos observar, mediante el test shapiro, que **no sigue una distribución normal**.
- **Chlorides**: La representación visual, cumple principio de normalidad para ambos tipos de calidad, pero mediante el test shapiro, se concluye que **no sigue una distribución normal**.
- **free sulfr dioxide**: No cumple principio de normalidad.
- **total sulfr dioxide**: No cumple principio de normalidad.
- **density**: Cumple principio de normalidad para ambos tipos de calidad, si se observa la gráfica Q-Q, pero si consideramos el test de shapiro no cumple principio de normalidad, por tanto concluimos que **no sigue una distribución normal**.
- **pH**: Cumple principio de normalidad para calidades altas, sin embargo para calidades bajas no cumple distribución normal. Concluimos afirmando que **cumple una distribución de nomal**.
- **sulphates**: Cumple principio de normalidad para calidades altas, sin embargo para calidades bajas no cumple distribución normal. Concluimos afirmando que **cumple una distribución de nomal**.
- **alcohol**: Cumple principio de normalidad para calidades altas, sin embargo para calidades bajas no cumple distribución normal. Concluimos afirmando que **cumple una distribución de nomal**.

- Para continuar procederemos a realizar un análisis de la homogeneidad de la varianza

Para realizar el análisis de la homogeneidad de varianzas entre nuestros 2 grupos de calidades, utilizaremos el test-Levene.

- En caso que la variable no cumpla una distribución normal, se aplicará el test sobre la **mediana**
- En caso que si cumpla una distribución normal se aplicará sobre la **media**

La prueba de Levene es un procedimiento estadístico para evaluar la igualdad de varianzas, entre dos o más poblaciones de muestra. La prueba de Levene a menudo se emplea para probar la suposición de que las varianzas de 2 poblaciones son iguales.

La elección de esta prueba es debido a que la prueba de Levene es menos sensible cuando las muestras de población generalmente no están distribuidas, al contrario que otros métodos como puede ser el caso del test de Bartlett.

En este caso consideramos un $alpha$ = 0.05

Para considerar que ambas vrianzas son homogéneas, se debe cumplir que el p-valor es > 0.05 (alpha)

In [125]:

```
import scipy.stats as test
```

fixed acidity

In [128]:

```
print('p-valor: ',test.levene(df_cleans[df_cleans['new_quality'] == 1]['fixed acidity'
],
            df_cleans[df_cleans['new_quality'] == 2]['fixed acidity'],
            center = 'median')[1])
```

```
p-valor:  0.5015369739049174
```

conclusión: Para la variable *fixed acidity* podemos afirmar que **las varianzas son iguales**, ya que $pvalue > alpha$.

---

volatile acidity

In [130]:

```
print('p-valor: ',test.levene(df_cleans[df_cleans['new_quality'] == 1]['volatile acidit
y'],
            df_cleans[df_cleans['new_quality'] == 2]['volatile acidity'],
            center = 'median')[1])
```

```
p-valor:  0.00029076535932777295
```

conclusión: Para la variable *volatile acidity* podemos afirmar que **las varianzas no son iguales**, ya que $pvalue < alpha$.

---

citric acid

In [131]:

```
print('p-valor: ',test.levene(df_cleans[df_cleans['new_quality'] == 1]['citric acid'],
            df_cleans[df_cleans['new_quality'] == 2]['citric acid'],
            center = 'median')[1])
```

```
p-valor:  0.06626890848436845
```

conclusión: Para la variable *citric acid* podemos afirmar que **las varianzas son iguales**, ya que $pvalue > alpha$.

---

residual sugar

In [132]:

```
print('p-valor: ',test.levene(df_cleans[df_cleans['new_quality'] == 1]['residual sugar'
],
            df_cleans[df_cleans['new_quality'] == 2]['residual sugar'],
            center = 'median')[1])
```

```
p-valor:  0.019497167604893194
```

conclusión: Para la variable *residual sugar* podemos afirmar que **las varianzas no son iguales**, ya que $pvalue < alpha$.

---

chlorides

In [133]:

```
print('p-valor: ',test.levene(df_cleans[df_cleans['new_quality'] == 1]['chlorides'],
            df_cleans[df_cleans['new_quality'] == 2]['chlorides'],
            center = 'median')[1])
```

```
p-valor:  0.7248920048554717
```

conclusión: Para la variable *chlorides* podemos afirmar que **las varianzas son iguales**, ya que $pvalue > alpha$.

---

free sulfur dioxide

In [134]:

```
print('p-valor: ',test.levene(df_cleans[df_cleans['new_quality'] == 1]['free sulfur dio
xide'],
            df_cleans[df_cleans['new_quality'] == 2]['free sulfur dioxide'],
            center = 'median')[1])
```

```
p-valor:  0.5748040655691935
```

conclusión: Para la variable *free sulfur dioxide* podemos afirmar que **las varianzas son iguales**, ya que $pvalue > alpha$.

---

total sulfur dioxide

In [135]:

```
print('p-valor: ',test.levene(df_cleans[df_cleans['new_quality'] == 1]['total sulfur di
oxide'],
            df_cleans[df_cleans['new_quality'] == 2]['total sulfur dioxide'],
            center = 'median')[1])
```

```
p-valor:  2.076651802199037e-09
```

conclusión: Para la variable *total sulfur dioxide* podemos afirmar que **las varianzas no son iguales**, ya que $pvalue < alpha$.

---

density

In [136]:

```
print('p-valor: ',test.levene(df_cleans[df_cleans['new_quality'] == 1]['density'],
            df_cleans[df_cleans['new_quality'] == 2]['density'],
            center = 'median')[1])
```

```
p-valor:  0.05933739858132664
```

conclusión: Para la variable *density* podemos afirmar que **las varianzas son iguales**, ya que $pvalue > alpha$.

---

pH

In [137]:

```
print('p-valor: ',test.levene(df_cleans[df_cleans['new_quality'] == 1]['pH'],
            df_cleans[df_cleans['new_quality'] == 2]['pH'],
            center = 'mean')[1])
```

```
p-valor:  0.27434042129567515
```

conclusión: Para la variable *pH* podemos afirmar que **las varianzas son iguales**, ya que $pvalue > alpha$.

---

volatile acidity

In [138]:

```
print('p-valor: ',test.levene(df_cleans[df_cleans['new_quality'] == 1]['sulphates'],
            df_cleans[df_cleans['new_quality'] == 2]['sulphates'],
            center = 'mean')[1])
```

```
p-valor:  0.6094310846196442
```

conclusión: Para la variable *sulphates* podemos afirmar que **las varianzas son iguales**, ya que $pvalue > alpha$.

---

alcohol

In [139]:

```
print('p-valor: ',test.levene(df_cleans[df_cleans['new_quality'] == 1]['alcohol'],
            df_cleans[df_cleans['new_quality'] == 2]['alcohol'],
            center = 'mean')[1])
```

```
p-valor:  0.4198840533006394
```

conclusión: Para la variable *alcohol* podemos afirmar que **las varianzas son iguales**, ya que $pvalue > alpha$.

Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes

Análisis estadístico descriptivo, similar al realizado anteriormente, simplemente se tratará de obtener información de estadísticos básicos, comos son las medidas de tendencia central y las medidas de dispersión. Se propondrá realizar un análisis descriptivo para cada tipo de calidad del vino.

Calidad Media *(quality 3, 4, 5, 6)*

In [140]:

```
df_cleans[df_cleans['new_quality'] == 2].describe()
```

Out[140]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total s di
count	1106.000000	1106.000000	1106.000000	1106.000000	1106.000000	1106.000000	1106.0(
mean	8.275136	0.524625	0.268065	2.453571	0.084141	15.642405	46.0
std	1.552358	0.157268	0.174051	1.136476	0.028204	9.840438	28.6
min	5.200000	0.160000	0.010000	1.200000	0.038000	1.000000	6.0(
25%	7.100000	0.400000	0.120000	1.900000	0.072000	8.000000	23.0(
50%	7.900000	0.520000	0.240000	2.200000	0.079500	14.000000	39.0(
75%	9.100000	0.630000	0.400000	2.600000	0.089000	21.000000	64.0(
max	12.700000	0.980000	0.780000	13.900000	0.387000	68.000000	128.0(

Calidad Alta *(quality 7, 8)*

In [141]:

```
df_cleans[df_cleans['new_quality'] == 1].describe()
```

Out[141]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000
mean	8.888202	0.399017	0.382753	2.695506	0.076534	14.033708	30.005618
std	1.539141	0.136808	0.172472	1.283437	0.020463	10.332822	18.753681
min	5.200000	0.120000	0.010000	1.400000	0.038000	3.000000	7.000000
25%	7.700000	0.310000	0.320000	2.000000	0.063000	6.000000	16.000000
50%	8.900000	0.370000	0.400000	2.300000	0.073000	11.000000	25.000000
75%	10.075000	0.480000	0.490000	2.800000	0.086750	17.750000	41.750000
max	12.000000	0.915000	0.760000	8.900000	0.216000	54.000000	106.000000

Conclusiones:

Observando los estadísticos básicos podemos tener una primera idea de como se comportarán las calidades en función de las variables y como se podrán clasificar las nuevas muestras que queramos clasificar:

- **Media**: Con la media podemos obtener distintas conclusiones:
  - La cantidad de **volatile acidity**, **total sulfur dioxide**, **free sulfur dioxide** y **pH** desciende cuanto mayor sea la calidad del vino.
  - Respecto al nivel del **citric acid** asciende cuanto mayor sea la calidad, al igual que el **residual sugar**, el **alcohol** y los **sulphates**.

- **Desviación estandard**: Indica la variacion y dispersión de las muestras en cada variables. Podemos observar que dispersión se tiene en cada variables para cada tipo de calidad:
  - Podemos observar que para todas las variables se presenta una dispersión más o menos similiar para ambas calidades, salvo para **total sulfur dioxide** que es el que mas dispersión presenta, siendo mayor para vinos de calidad más baja.
  - La variable que menos dispersión presenta, es el **pH**.

- **Cuartiles**: Con las medidas de los distintos cuartiles, podemos calcular el rango intercuartílico, de forma que nos aporte información sobre la distribución y dispersión de las muestras. Una forma sencilla de poder identificar esta información también puede ser mediante visualización de la información con diagramas de cajas.

Análisis de correlación entre variables, que consiste en medir la asociación entre dos variables. esto nos permitirá descartar variables que estén altamnente relacionadas, y no aporten variabilidad a los datos, para a la hora de aplicar nuestro modelo de clasificación, simplificar.

A continuación generamos la matriz de correlación que nos proporciona información de como están relacionadas las variables entre sí, estando más relacionadas las variables con coeficiente cercano a 1 y -1.

Las variables más correlacionadas podríamos eliminarlas de forma que reduzcamos la dimensionalidad de nuestro set de datos.

En este caso se utilizará la correlación de Spearman, ya que por lo general los datos no siguen una distribución normal

In [142]:

```
df_cleans.corr(method='spearman')
```

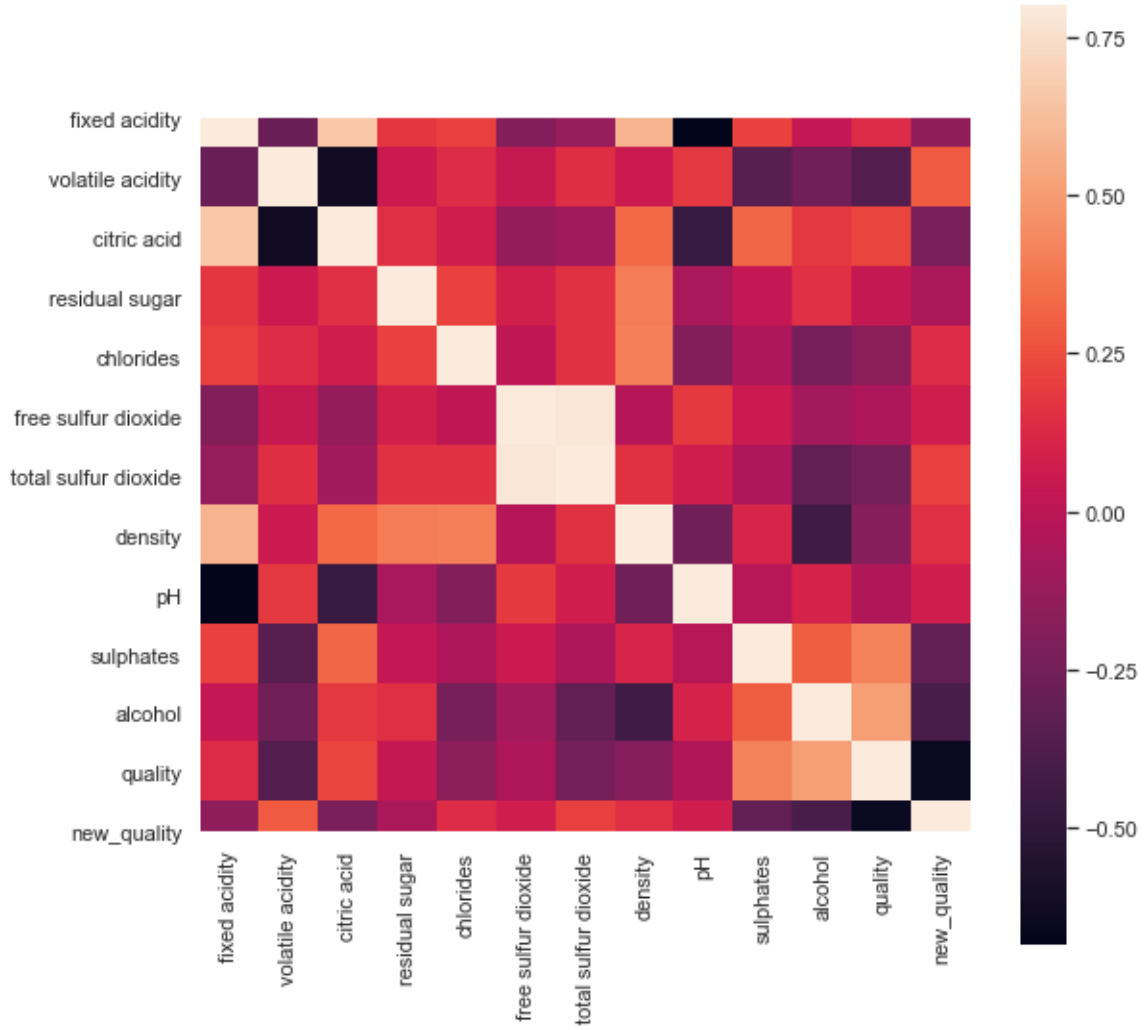
Out[142]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	de
fixed acidity	1.000000	-0.284187	0.659940	0.176005	0.212834	-0.194238	-0.127183	0.58
volatile acidity	-0.284187	1.000000	-0.622359	0.054188	0.138914	0.042231	0.146115	0.05
citric acid	0.659940	-0.622359	1.000000	0.150965	0.075099	-0.130063	-0.091731	0.33
residual sugar	0.176005	0.054188	0.150965	1.000000	0.214388	0.081065	0.159218	0.39
chlorides	0.212834	0.138914	0.075099	0.214388	1.000000	0.013504	0.160777	0.40
free sulfur dioxide	-0.194238	0.042231	-0.130063	0.081065	0.013504	1.000000	0.778561	-0.02
total sulfur dioxide	-0.127183	0.146115	-0.091731	0.159218	0.160777	0.778561	1.000000	0.15
density	0.580145	0.053485	0.330415	0.397794	0.400849	-0.028421	0.157280	1.00
pH	-0.685611	0.183300	-0.462264	-0.067916	-0.193850	0.185814	0.077754	-0.25
sulphates	0.215219	-0.348669	0.322747	0.031671	-0.049097	0.055738	-0.051165	0.11
alcohol	0.029838	-0.256810	0.182999	0.152803	-0.233760	-0.087816	-0.304851	-0.43
quality	0.135396	-0.365652	0.224471	0.038405	-0.162505	-0.049575	-0.240463	-0.18
new_quality	-0.150256	0.286113	-0.221908	-0.064511	0.139973	0.078556	0.204738	0.15

Podemos observar los resultados anteriores en forma de tabla, de forma gráfica donde la escala de colores nos marca la correlación entre variables.

In [144]:

```
corrmat = df_cleans.corr(method='spearman')
f, ax = plt.subplots(figsize=(9, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
```



Conclusión:

- Se puede observar como la variable **fixed acidity**, tiene una alta correlación con las variables **pH**, con una correlación de -0.68, al igual que **citric acid** con un coeficiente de 0.66, y **density** del 0.6.
- Además podemos observar la alta correlación entre **density** y **alcohol** con un coeficiente de correlación del -0.43.
- La correlación más alta se da entre **total sulfur dioxide** y **free sulfur dioxide**, con un coeficiente del 0.77
- Por último podemos observar que variables tendrán más peso a la hora de clasificar un vino en distintas calidades.
  - Las variables que más explican la calidad del vino son **alcohol** y **sulphates** con coeficientes del 0.5 y 0.41, respectivamente.
  - Las variables que menos influyen a la hora de definir una calidad de los vinos, son **free sulfur dioxide** y **residual sugar** con coeficientes 0.07 y 0.06 respectivamente.

- **Acabamos eliminando las variables que menos correlación tengan con nuestra variable objetivo, ya que no aportarán variabilidad a nuestro modelo**

In [145]:

```
df_model = df_cleans.drop(['free sulfur dioxide', 'residual sugar'], axis = 1)
```

In [146]:

```
df_model.head()
```

Out[146]:

	fixed acidity	volatile acidity	citric acid	chlorides	total sulfur dioxide	density	pH	sulphates	alcohol	quality	new
0	7.8	0.76	0.04	0.092	54.0	0.9970	3.26	0.65	9.8	5	
1	11.2	0.28	0.56	0.075	60.0	0.9980	3.16	0.58	9.8	6	
2	7.9	0.60	0.06	0.069	59.0	0.9964	3.30	0.46	9.4	5	
3	7.8	0.58	0.02	0.073	18.0	0.9968	3.36	0.57	9.5	7	
4	7.5	0.50	0.36	0.071	102.0	0.9978	3.35	0.80	10.5	5	

Aplicación de modelos supervisados de clasificación, que permitan clasificar nuevos vinos, con caracteristicas similares a las que existen en el set de datos, de forma que en función del entrenamiento del algoritmo y definición de las reglas de clasificación, se puedan predecir la calidad de futuros vinos, en función de sus caracteristicas.

Antes de empezar simplificaremos el set de datos eliminando la calidad inicial, ya que clasificaremos en función de nuestra nueva calidad definida:

In [147]:

```
df_model = df_model.drop(['quality'], axis = 1)
```

El primer paso será generar 2 conjuntos de datos, uno de entrenamiento y uno de test. Se generará el conjunto de entrenamiento con el 70% de los datos y el conjunto de test con el 30%.

In [148]:

```
# Separamos el target del conjunto de datos
data_target = df_model.iloc[:,-1]
data = df_model.iloc[:,0:9]

# Generamos el conjunto de entrenamiento y de test
X_train, X_test, y_train, y_test = train_test_split(data, data_target, test_size=0.3, r
andom_state=42)
```

Regresión logística

A continuación procedemos a la aplicación de una regresión logística, apartir de los conjuntos de entrenamiento y test definidos anteirormente.

In [149]:

```
# Aplicación de la regresión logística
reg_log = LogisticRegression()
reg_log.fit(X_train, y_train)
reg_log_predict = reg_log.predict(X_test)
```

In [150]:

```
# mostramos la matriz de confusión y la precisión
reg_log_conf_matrix = confusion_matrix(y_test, reg_log_predict)
reg_log_acc_score = accuracy_score(y_test, reg_log_predict)

print('La matriz de confusión:')
print(reg_log_conf_matrix)
print('\n')
print('La precisión de clasificación:', reg_log_acc_score * 100)
```

```
La matriz de confusión:
[[ 15  34]
 [  3 334]]


La precisión de clasificación: 90.41450777202073
```

La regresión logística es útil para modelar la probabilidad de un evento ocurriendo como función de otros factores, en este caso la calidad del vino en función de las variables que más correlación tenia con la variable de la calidad. La precisión con la que nuestro modelo clasificará es de aproximadamente el 90%

Random decision forests

Aplicaremos un modelo de clasificación, basado en un clasificador random forest, es una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos

In [151]:

```
# Aplicación del random forest
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
rf_predict = rf.predict(X_test)

# mostramos la matriz de confusión y la precisión
rf_conf_matrix = confusion_matrix(y_test, rf_predict)
rf_acc_score = accuracy_score(y_test, rf_predict)
print('La matriz de confusión:')
print(rf_conf_matrix)
print('\n')
print('La precisión de clasificación:', rf_acc_score * 100)
```

```
La matriz de confusión:
[[ 26  23]
 [ 16 321]]


La precisión de clasificación: 89.89637305699482
```

Por último, si comparamos ambos clasificadores se puede considerar una precisión practicamente similar, siendo algo mejor el modelo de random forest

Realizaremos una predicción basada en los análisis probabilísticos previos a la implantación del modelo,para comprobar la clasificación realizada por los modelos.

- **fixed acidity**: 8.6
- **volatile acidity**: 0.4
- **citric acid**: 0.35
- **chlorides**: 0.07
- **total sulfur dioxide**: 30.6
- **density**: 0.994
- **pH**: 3.25
- **sulphates**: 0.73
- **alcohol**: 12.47

In [152]:

```
data_new = pd.DataFrame({'fixed acidity': [8.6], 'volatile acidity': [0.4], 'citric acid': [0.35],
                         'chlorides': [0.07], 'total sulfur dioxide': [30.6], 'density': [0.994],
                         'pH': [3.25], 'sulphates': [0.88], 'alcohol': [11.77]})

reg_log_prediction = reg_log.predict(data_new)
```

In [153]:

```
reg_log_prediction
```

Out[153]:

```
array([1], dtype=int64)
```

El vino con las caracteristicas insertadas, estaría clasificado como un vino de **alta calidad**.

La calidad está altamente relacionada con la cantidad de alcohol y sulfatos, cuanto mayor cantidad tenga de alcohol y de sulfatos, mayor calidad poseerá. Aunque los demás parámetros influyan, podemos concluir que estos 2 son los que más peso aportan a la hora de clasificar las muestras

---

Resolución del problema

A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

Podemos terminar afirmando que los distintos análisis realizados han sido útiles para poder entender el comportamiento de los datos, es decir, con los diferentes análisis realizados (y los demás que no se han podido realizar) hemos podido identificar que características de un vino, son másimportantes para poder clasificar los vinos en diferentes calidades.

Los análisis realizados pueden ser más o menos complejos, o se pueden extraer conclusiones, más o menos intuitivas (por lo general, es más fácil interpretar resultados de forma gráfica, que de forma numérica), pero si se combinan todas las técnicas de anáalisis posibles, el potencial a la hora de extraer conocimiento de los datos es elevado.

En el caso de nuestro conjunto de datos, la pregunta/problemática a resolver era que como se podria realizar una clasificación de los distintos vinos, que carácteristicas son más importantes para poder realizar una clasificación de vinos, incluso se ha realizado 2 modelos de clasificación, que predicirán a futuro nuevas muestras de las que se quiera conocer su calidad, con una precisión bastante elevada.

Se ha terminado concluyendo que entre todas las variables que se tenían en el conjunto, para cada vino, aquellas que más explicaban la calidad de cada muestra, eran el nivel de alcohol y la cantidad de sulfatos. Esto no quiere decir que el resto devariables no sean importantes, ya que si en elresto de variables sepresenta una variación elevada, afectará de igual manera a la calidad del vino.

Fuentes adicionales

https://www.uv.es/~mamtnez/IECRC.pdf (https://www.uv.es/~mamtnez/IECRC.pdf)
https://docs.scipy.org/doc/scipy-0.14.0/reference/index.html (https://docs.scipy.org/doc/scipy-0.14.0/reference/index.html)
https://aaronschlegel.me/levenes-test-equality-variances-python.html (https://aaronschlegel.me/levenes-test-equality-variances-python.html)
https://rpubs.com/Joaquin_AR/218466 (https://rpubs.com/Joaquin_AR/218466)
https://relopezbriega.github.io/blog/2016/03/13/analisis-de-datos-cuantitativos-con-python/ (https://relopezbriega.github.io/blog/2016/03/13/analisis-de-datos-cuantitativos-con-python/)
https://rpubs.com/Joaquin_AR/218465 (https://rpubs.com/Joaquin_AR/218465)
https://blog.findemor.es/2017/12/machine-learning-introduccion-estadistica-basica-python/ (https://blog.findemor.es/2017/12/machine-learning-introduccion-estadistica-basica-python/)

Firmas

Contribuciones	Firma
Investigación previa	AJH, JMC
Redacción de las respuestas	AJH, JMC
Desarrollo código	AJH, JMC