Práctica 2: Limpieza y análisis de datos

En esta práctica se elabora un caso práctico orientado a aprender a identificar los datos relevantes para un proyecto analítico y usar las herramientas de integración, limpieza, validación y análisis de las mismas. Para hacer esta práctica tendréis que trabajar en grupos de 2 personas. Tendréis que entregar un solo archivo con el enlace Github (https://github.com) donde se encuentren las soluciones incluyendo los nombres de los componentes del equipo. Podéis utilizar laWiki de Github para describir vuestro equipo y los diferentes archivos que corresponden a vuestra entrega. Cada miembro del equipo tendrá que contribuir con su usuario Github. Aunque no se trata del mismo enunciado, los siguientes ejemplos de ediciones anteriores os pueden servir como guía:

- Ejemplo:https://github.com/Bengis/nba-gap-cleaning
- Ejemplo complejo (archivo adjunto).

Competencias

En esta práctica se desarrollan las siguientes competencias del Máster de Data Science:

- Capacidad de analizar un problema en el nivel de abstracción adecuado a cada situación y aplicar las habilidades y conocimientos adquiridos para abordarlo y resolverlo.
- Capacidad para aplicar las técnicas específicas de tratamiento de datos (integración, transformación, limpieza y validación) para su posterior análisis.

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 Tipología y ciclo de vida de los datosPráctica 2pág 2función 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 de la Práctica a realizar

El objetivo de esta actividad será el tratamiento de un dataset, que puede ser el creado en la práctica 1 o bien cualquier dataset libre disponible en Kaggle (https://www.kaggle.com). Algunos ejemplos de dataset con los que podéis trabajar son:

- Red Wine Quality (https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009)
- Titanic: Machine Learning from Disaster (https://www.kaggle.com/c/titanic)

El último ejemplo corresponde a una competición activa de Kaggle de manera que, opcionalmente, podéis aprovechar el trabajo realizado durante la práctica para entrar en esta competición.

Siguiendo las principales etapas de un proyecto analítico, las diferentes tareas a realizar (y justificar) son las siguientes:

1. Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

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

3. Limpieza de los datos.

```
 - 3.1.¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos
casos?
 - 3.2.Identificación y tratamiento de valores extremos.
```

4. Análisis de los datos.

- 4.1.Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).
- 4.2.Comprobación de la normalidad y homogeneidad de la varianza.
- 4.3.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.

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

6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema? **7.** Código: Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferís, también podéis trabajar en Python.

Nombre y apellidos:

LUIS ALBERTO PICO

Solución

Librerías y Configuración Inicial

In [552]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from statsmodels.graphics.gofplots import qqplot #Normalidad
from scipy.stats import shapiro #Normalidad
from scipy import stats #Pruebas Estadísticas
from sklearn.preprocessing import StandardScaler #Estandarización
from sklearn.model_selection import train_test_split, GridSearchCV
import umap
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix

#Configuración de estilos y colores
sns.set_style("darkgrid")
custom_palette = ["red", "blue", "orange", "green","yellow", "purple"]
#custom_palette = ['#FBB4AE','#B3CDE3','#CCEBC5','#DECBE4','#FED9A6','#FFFFCC','#E5D8BD','#FDDAEC','#F2F.
sns.set_palette(custom_palette)
sns.set_context("notebook")
hue_colors = {0: "yellow",1: "red",2: "cyan"}
```

1. Descripción del conjunto de datos

Dataset

Red Wine Quality

Autor

Creado por: Paulo Cortez (Univ. Minho), Antonio Cerdeira, Fernando Almeida, Telmo Matos and Jose Reis (CVRVV) @ 2009

Descripción

El conjunto de datos se creó utilizando muestras de vino tinto. Los atributos de entrada o variables independientes incluyen pruebas objetivas (basado en tests fisicoquímicos) y la variable de salida u objetivo se basa en datos sensoriales (mediana de al menos 3 evaluaciones realizadas por expertos .en vino). Cada experto calificó la calidad del vino entre 0 (muy malo) y 10 (muy excelente).

Dimensiones

El Dataset esta compuesto de 1,599 registros (observaciones) y 12 atributos (variables)

Atributos

Atributos de Entrada (basado en tests fisicoquímicos):
1 - fixed acidity. Acidez fija
2 - volatile acidity. Acidez volátil
3 - citric acid. Ácido cítrico
4 - residual sugar. Azucar residual
5 - chlorides. Cloruros
6 - free sulfur dioxide. Dióxido de azufre libre
7 - total sulfur dioxide. Dióxido de azufre total
8 - density. Densidad
9 - pH. pH
10 - sulphates. Sulfatos
11 - alcohol. Alcohol

Atributo de salida (basado en data sensorial):
12 - quality. Calidad (score entre 0 y 10)

Importancia

Hoy en día el vino es disfrutado cada vez más por una gama más amplia de consumidores. Portugal es uno de los diez principales países exportadores de vino. Las exportaciones de su vino verdewine (de la región noroeste) han aumentado. Para respaldar su crecimiento, la industria del vino está invirtiendo en nuevas tecnologías tanto para los procesos de elaboración como para los procesos de venta. La certificación del vino y la evaluación de la calidad son elementos clave en este contexto. La certificación evita la alteración ilegal de los vinos (para salvaguardar la salud humana) y asegura la calidad para el mercado del vino. La evaluación de la calidad es a menudo parte del proceso de certificación y se puede utilizar para mejorar la elaboración del vino (identificando los factores más influyentes) y para estratificar vinos como las marcas premium (útil para fijar precios). La certificación del vino generalmente se evalúa mediante pruebas físico-químicas y sensoriales. Las pruebas de laboratorio fisicoquímicas que se utilizan habitualmente para caracterizar el vino incluyen la determinación de la densidad, el alcohol o los valores de pH, mientras que las pruebas sensoriales se basan principalmente en expertos humanos. Cabe destacar que el gusto es el menos comprendido de los sentidos humanos, por lo que la clasificación del vino es una tarea difícil.

En función del antecendente indicado este dataset se utilizará para predecir la calidad del vino basado en datos fisicoquímicos.

Cita

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236.

2. Integración y selección de Datos

2.1 Carga del Conjunto de Datos

El conjunto de datos seleccionado para esta práctica es el dataset `Red Wine Quality` disponible en Kaggle (https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009).

Procedemos a realizar la lectura del fichero en formato CSV `winequality-red.csv` previamente descargado desde Kaggle que lo almacenaremos en un objeto DataFrame **RedWine_data** y visuaizamos una muestra de los datos que contiene.

In [553]:

```
RedWine_data=pd.read_csv('winequality-red.csv')
RedWine_data.head()
```

Out[553]:

	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

Procedemos a revisar la estructura y tipos de datos que contiene el conjunto de datos.

In [554]:

```
RedWine_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

El conjunto de datos está conformado por 12 atributos o columas y 1,599 registros o filas, todas los datos son **numéricos**.

2.2 Análisis estadístico básico

Procedemos a visualizar estadístos básicos de los atributos del conjunto de datos a través de la función `describe` del dataframe.

In [555]:

```
RedWine_data.describe()
```

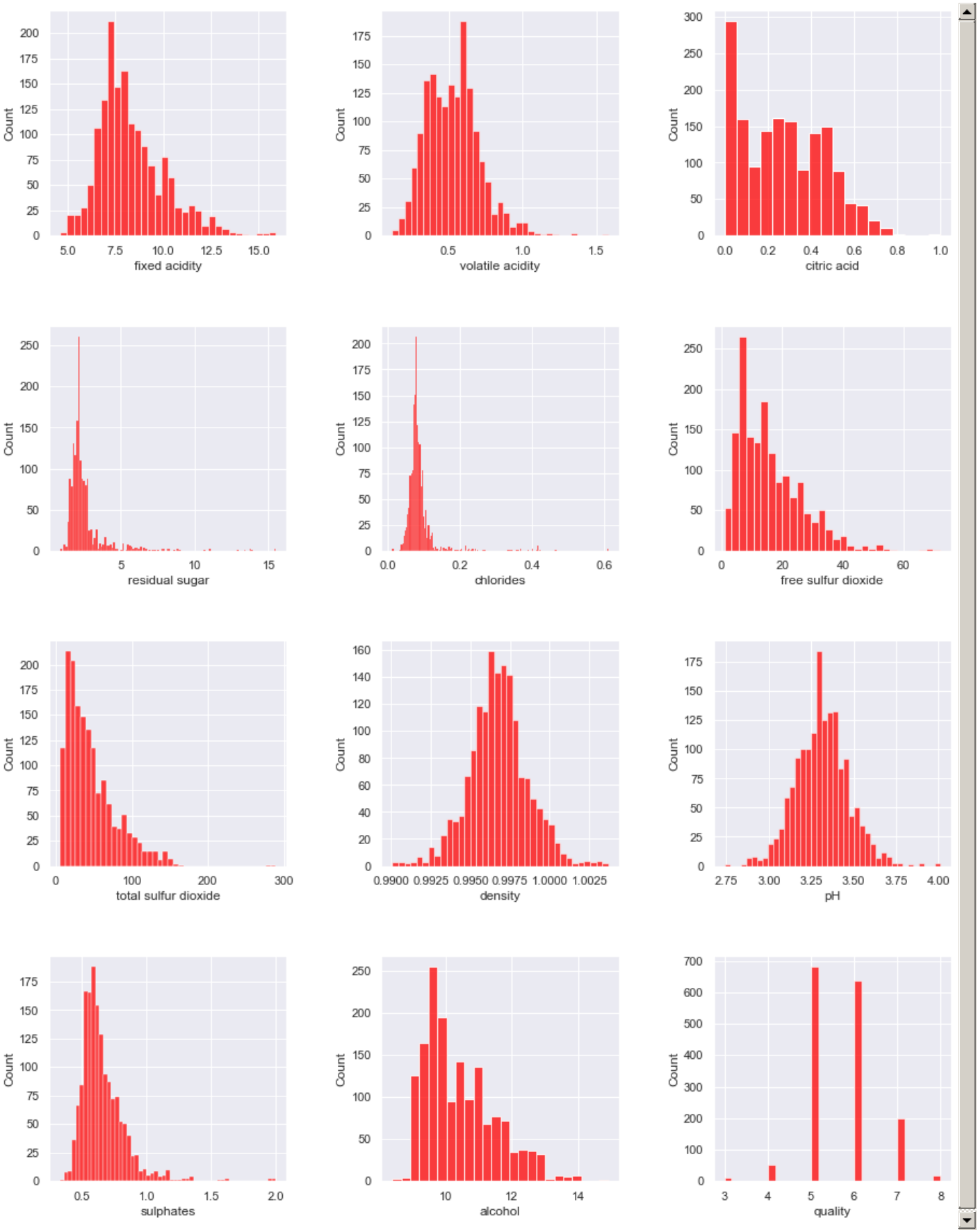
Out[555]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000

Procedemos a visualizar las gráficas de distribución de los atributos del conjunto de datos a través de la función `hisplot` .

In [556]:

```
fig = plt.figure(figsize=(15,20))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
i=1
for variable in RedWine_data:
    ax = fig.add_subplot(4, 3, i)
    sns.histplot(data=RedWine_data,x=variable,ax=ax)
    i=i+1
plt.show()
```



Como se puede visualizar la variable `quality` toma valores entre 3 y 8, además podemos observar tanto en sus estadísticos básicos como en sus histogramas que en las variables `residual sugar` y `chlorides` existe mucha dispersión en sus datos.

2.3 Selección de Datos

En función de que el objetivo del análisis será generar una modelo que permita clasificar la calidad de los vinos en función de datos fisicoquímicos, se utilizará como variables independientes aquellos atributos basados en las pruebas de laboratorio fisicoquímicas ya que estas se utilizan habitualmente para caracterizar el vino y las almacenaremos en el dataframe **X**, mientras que la variable objetivo o independiente que resulta de las pruebas sensoriales del vino realizada por expertos humanos para determinar su calidad se almacenará en el dataframe **y** y será una variable dicotómica (0,1) resultante de discretizar la variable `quality` en dos grupos:

- **Alta Calidad (1)**. vinos catálogados con calidad 7 o 8.
- **Baja Calidad (0)**. vinos catálogados con calidad 3,4,5 o 6.

In [557]:

```
X=RedWine_data.drop(columns=['quality']).copy()
y=np.where(RedWine_data['quality']>=7,1,0) #Alta Calidad (1) Baja Calidad(0)
RedWine_data['y']=y.copy()
RedWine_data.head()
```

Out[557]:

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

3. Limpieza de Datos

Procederemos en este apartado a determinar si los datos contienen ceros o elementos vacíos a gestionarlos en caso de existir alguno y luego identificaremos y trataremos en la medida de los posible los valores extremos.

3.1 Ceros o Elementos Vacíos

Para deteminar si los datos contienen ceros o elementos vacíos utilizaremos la función `isna` del dataframe asociado a nuestro conjunto de datos.

In [558]:

```
RedWine_data.isna().sum()
```

Out[558]:

```
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
y                       0
dtype: int64
```

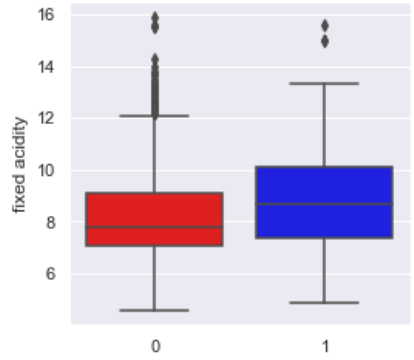
Como se puede evidenciar no existe valores nulos o elementos vacíos en ninguna variable del conjunto de datos, por lo que no se utilizará ningún método (imputación de valor medio o mediana o eliminación del registro,etc) para depurar los mismos.

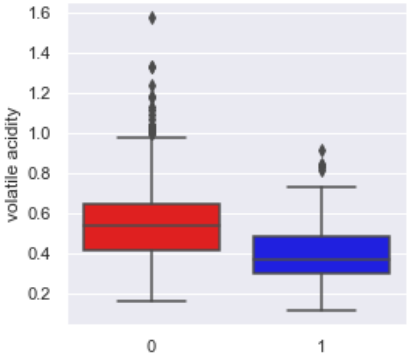
3.2 Valores Extremos

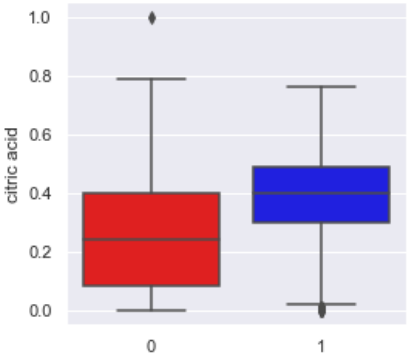
Procedemos a identificar y dar tratamiento en la medidad de lo posible a los valores extremos que se identifiquen en el conjunto de datos, para esto se utilizará el diagrama de cajas para cada una de las variables en función de su calidad.

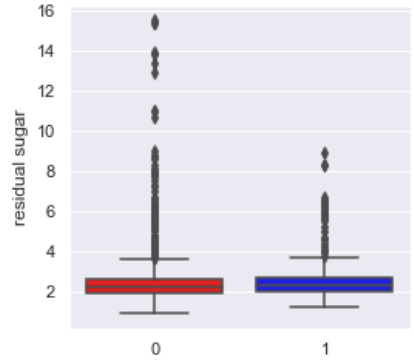
In [559]:

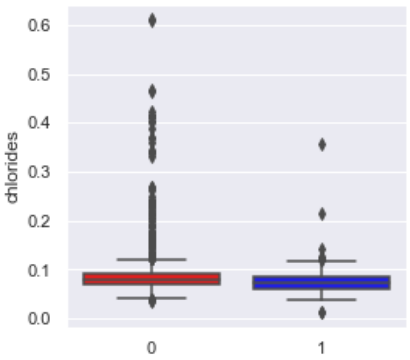
```
fig = plt.figure(figsize=(15,20))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
i=1
for variable in X:
    ax = fig.add_subplot(4, 3, i)
    sns.boxplot(data=X,x=y,y=variable,ax=ax)
    i=i+1
plt.show()
```

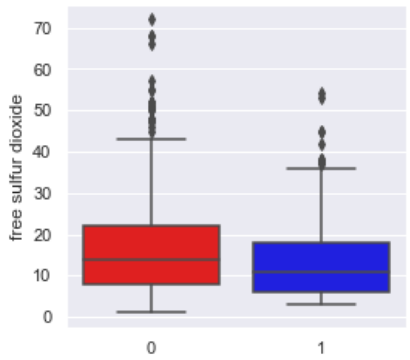

fixed acidity
16
14
12
10
8
6
0
1

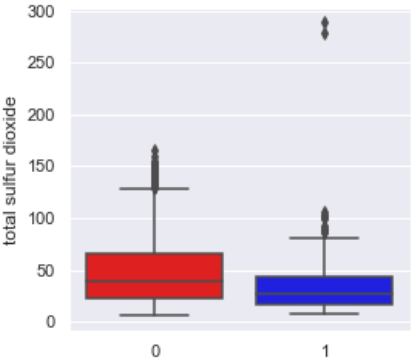

volatile acidity
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
1

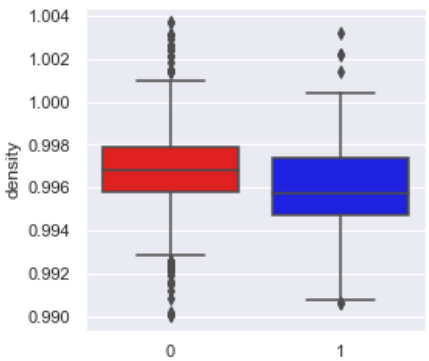

citric acid
1.0
0.8
0.6
0.4
0.2
0.0
0
1

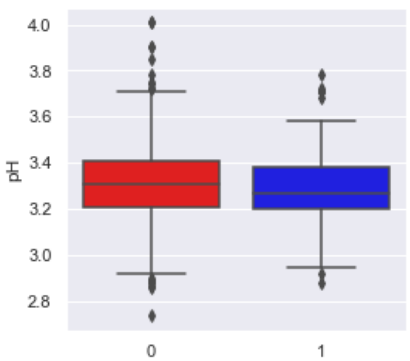

residual sugar
16
14
12
10
8
6
4
2
0
1

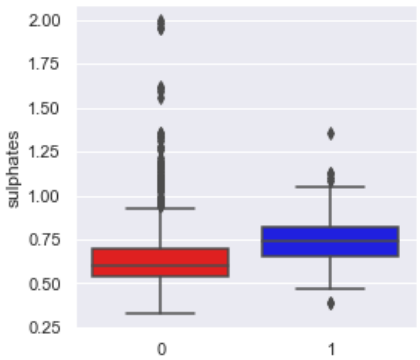

chlorides
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
1

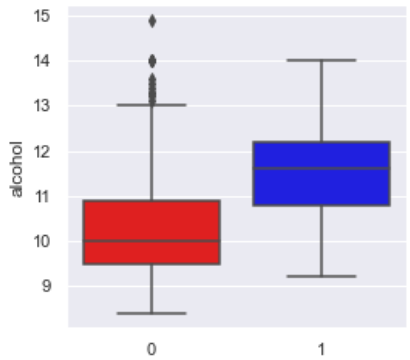

free sulfur dioxide
70
60
50
40
30
20
10
0
0
1


total sulfur dioxide
300
250
200
150
100
50
0
0
1


density
1.004
1.002
1.000
0.998
0.996
0.994
0.992
0.990
0
1


pH
4.0
3.8
3.6
3.4
3.2
3.0
2.8
0
1


sulphates
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
1


alcohol
15
14
13
12
11
10
9
0
1

En función de lo observado se identifica que las variables `residual sugar` y `chlorides` tienen una gran número de observaciones que se pueden considerar como valores extremos, pero al revisar sus estadísticos básicos observamos que el 25% (percentil 75) de las obserciones se definirian como atípicos lo que nos hace suponer que en realidad no lo son y además debemos recordar que los datos son registrados por pruebas fisícoquímicas que son menos propénsas al error, por lo que no se realizará ningún método de tratamiento de atípicos.

4. Análisis de los Datos

Dado que nuestro objetivo de análisis será generar una modelo que permita clasificar la calidad de los vinos en función de datos fisico químicos procedremos a seleccionar los grupos de datos que se quieren realizar el análisis y aplicaremos algunas pruebas estadísticas para comparar estos grupos.

4.1 Selección de los grupos de Datos

Seleccionamos los grupos definidos inicialmente:

- **Alta Calidad (1)**. vinos catálogados con calidad 7 o 8.
- **Baja Calidad (0)**. vinos catálogados con calidad 3,4,5 o 6.

In [560]:

```
alta_calidad=RedWine_data[RedWine_data['y']==1]
baja_calidad=RedWine_data[RedWine_data['y']==0]
```

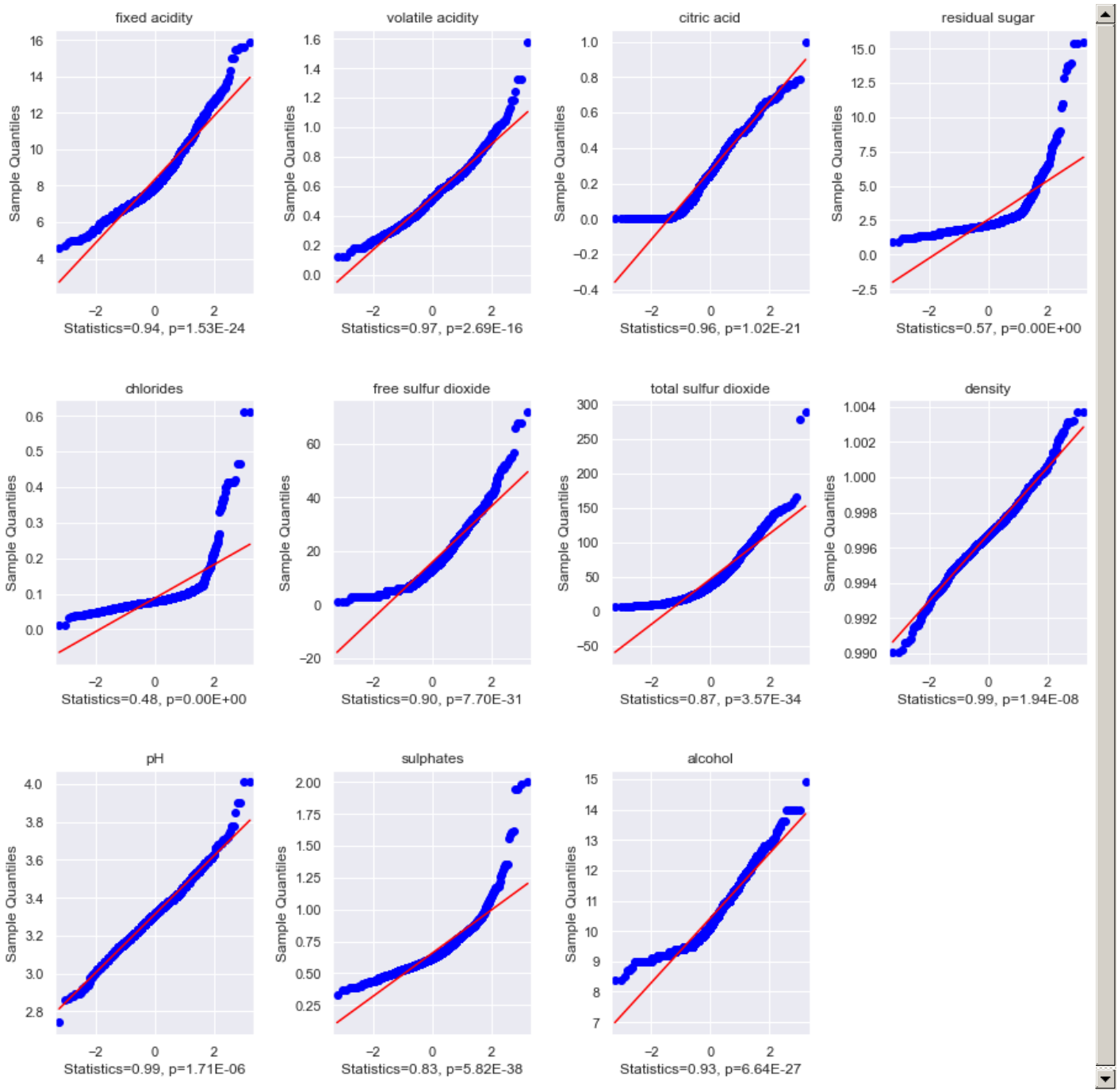
4.2 Normalidad y Homogeneidad de la Varianza de los Datos

4.2.1 Normalidad

Determinamos la normalidad de las variables dependientes para esto se utilizará el método visual `q-q plot` y el método estadístico `Shapiro-Wilk` test.

In [561]:

```
fig = plt.figure(figsize=(15,15))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
i=1
for variable in list(X):
    ax = fig.add_subplot(3, 4, i)
    ax.set_title(variable)
    qqplot(X[variable], line='s',ax=ax) # q-q plot
    stat, p = shapiro(X[variable])
    ax.set_xlabel('Statistics=%.2f, p=%.2E' % (stat, p))
    i=i+1
plt.show()
```



Dado que para todas las variable en el test de `Shapiro-Wilk` se obtiene un *p-valor* **inferior** al nivel de significancia **α = 0.05**, entonces se determina que ninguna variable analizada sigue una distribución normal.

4.2.2 Homogeneidad de la Varianza de los Datos

Se realizará el test de homogeneidad de la varianza de todas las variables dependientes con relación a la calidad del vino.

In [562]:

```
varianza_igual=[]
for variable in list(X):
    statistic,pvalue = stats.levene(X.loc[alta_calidad.index,variable],X.loc[baja_calidad.index,variable]
    print(variable+' : Statistics=%.2f, p-value=%.2f' % (statistic,pvalue))
    varianza_igual.append(pvalue>=0.05)
#    [print(np.var(x, ddof=1)) for x in [X.loc[y==3,variable], X.loc[y==4,variable], X.loc[y==5,variable
#                                X.loc[y==7,variable], X.loc[y==8,variable]]]
```

```
fixed acidity : Statistics=13.12, p-value=0.00
volatile acidity : Statistics=12.95, p-value=0.00
citric acid : Statistics=1.06, p-value=0.30
residual sugar : Statistics=2.29, p-value=0.13
chlorides : Statistics=1.81, p-value=0.18
free sulfur dioxide : Statistics=1.75, p-value=0.19
total sulfur dioxide : Statistics=15.20, p-value=0.00
density : Statistics=17.03, p-value=0.00
pH : Statistics=0.06, p-value=0.81
sulphates : Statistics=0.71, p-value=0.40
alcohol : Statistics=1.45, p-value=0.23
```

En función de los test realizados determinamos que las únicas variables que tienen homogeneidad de la varianza en sus datos con relación a la calidad del vino son: `citric acid`, `residual sugar`, `chlorides`, `free sulfur dioxide`, `pH`, `sulphates` y `alcohol` por cuanto su estadístico *p-valor* es **superior** al nivel de significancia **α = 0.05**.

4.3 Pruebas Estadisticas

En función del objetivo del estudio procederemos a realizar pruebas estadísticas para comparar los grupos de datos definidos.

4.3.1 Test de Igualdad de Medias

In [563]:

```
for variable,var in zip(list(X),varianza_igual):
    statistic,pvalue = stats.ttest_ind(X.loc[alta_calidad.index,variable],X.loc[baja_calidad.index,variab
    print(variable+' : Statistics=%.2f, p-value=%.2f' % (statistic,pvalue))
```

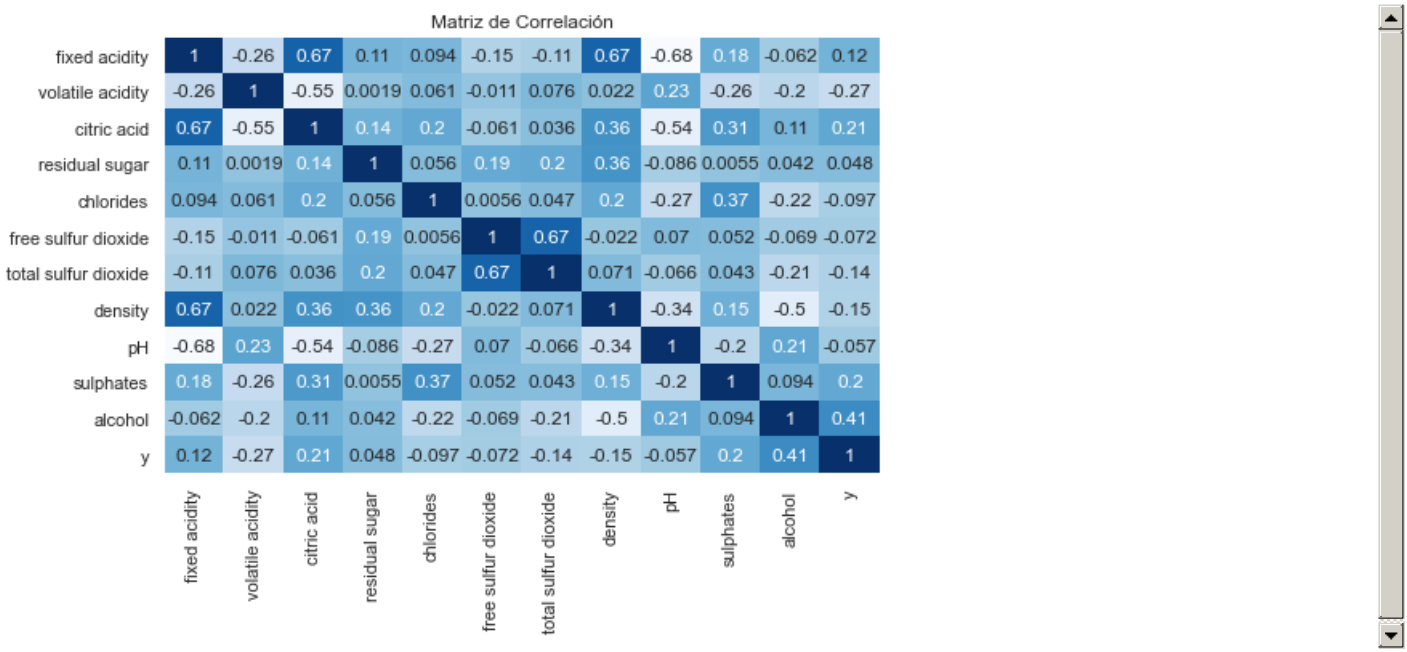
```
fixed acidity : Statistics=4.26, p-value=0.00
volatile acidity : Statistics=-12.95, p-value=0.00
citric acid : Statistics=8.79, p-value=0.00
residual sugar : Statistics=1.91, p-value=0.06
chlorides : Statistics=-3.91, p-value=0.00
free sulfur dioxide : Statistics=-2.87, p-value=0.00
total sulfur dioxide : Statistics=-5.63, p-value=0.00
density : Statistics=-5.27, p-value=0.00
pH : Statistics=-2.29, p-value=0.02
sulphates : Statistics=8.14, p-value=0.00
alcohol : Statistics=17.82, p-value=0.00
```

En función de los test realizados determinamos que las única variable que tiene medias iguales entre los dos grupos de análisis (alta_calidad, baja_calidad) es el atributo `residual sugar` por cuanto su estadístico *p-valor* es **superior** al nivel de significancia **α = 0.05**, el resto de variables tienen medias signitivamente diferentes y por tanto estas pueden ser útiles para clasificar la calidad del vino.

4.3.2 Correlación de Variables

In [564]:

```
fig = plt.figure(figsize=(8,5))
sns.heatmap(RedWine_data.drop(columns=['quality']).corr(),cmap='Blues',annot=True,cbar=False)
plt.title('Matriz de Correlación')
plt.show()
```



En la matriz de correlación se puede observar que el `alcohol` tiene una correlación media **0.41** con respecto a nuestra variable objetivo, siendo esta la más importante con relación al resto de variables.

4.3.3 Modelos de Clasificación del Vino

Definición de Funciones

In [565]:

```
def getMatrizConfusion(y_train,y_train_pred,y_test,y_test_pred):
    accuracy_train=accuracy_score(y_train, y_train_pred)*100
    #print("Accuracy Train: {0:.2f}%".format(accuracy_train))

    accuracy_test=accuracy_score(y_test, y_test_pred)*100
    print("Accuracy Test : {0:.2f}%".format(accuracy_test))

    matriz_confusion = confusion_matrix(y_test, y_test_pred)
    print('Matriz de Confusión:')
    sns.heatmap(matriz_confusion,annot=True,fmt="d",cbar=False,cmap="Blues")
    plt.xlabel('Observado')
    plt.ylabel('Estimado')
    plt.show()

def getParametrosOptimos(modelo,param_grid,X_train,y_train):

    crossValidation = GridSearchCV(modelo, param_grid, cv=4)
    crossValidation.fit(X_train, y_train)
    print('Valor Óptimo del Hiperparámetro:',crossValidation.best_params_)
    print('Accuracy Valor Óptimo: {0:.2f}%'.format(crossValidation.best_score_*100))

    filas=list(param_grid.values())[0]
    parametroFilas=list(param_grid.keys())[0]
    columnas=['score']
    parametroColumnas=[' ']
    if(len(param_grid)==2):
        columnas=list(param_grid.values())[1]
        parametroColumnas=list(param_grid.keys())[1]

    means_score=crossValidation.cv_results_['mean_test_score']*100
    mean_accuracy=pd.DataFrame(means_score.reshape(len(filas),len(columnas)),columns=columnas,index=filas

    fig, axes = plt.subplots(1, 2,figsize=(12,4))

    sns.heatmap(ax=axes[0],data=mean_accuracy, annot=True, fmt=".2f", cmap="RdBu",cbar=False)
    axes[0].set_title('Mean Test Score')
    axes[0].set_ylabel(parametroFilas)
    axes[0].set_xlabel(parametroColumnas)
    #plt.show()

    stds_score=crossValidation.cv_results_['std_test_score']*100
    std_accuracy=pd.DataFrame(stds_score.reshape(len(filas),len(columnas)),columns=columnas,index=filas)

    sns.heatmap(ax=axes[1],data=std_accuracy, annot=True, fmt=".2f", cmap="RdBu_r",cbar=False)
    axes[1].set_title('Std Test Score')
    axes[1].set_ylabel(parametroFilas)
    axes[1].set_xlabel(parametroColumnas)
    plt.show()
```

4.3.3.1 Conjuntos de datos de Entrenamiento, Test y Validación

Dividimos el *dataset* en dos subconjuntos, ***train*** (70% de los datos) y ***test*** (20% de los datos). Se nombra los conjuntos como: X_train, X_test, y_train, y_test, utilizando la opción `random_state = 24` y la implementación `train_test_split` de `sklearn`.

In [566]:

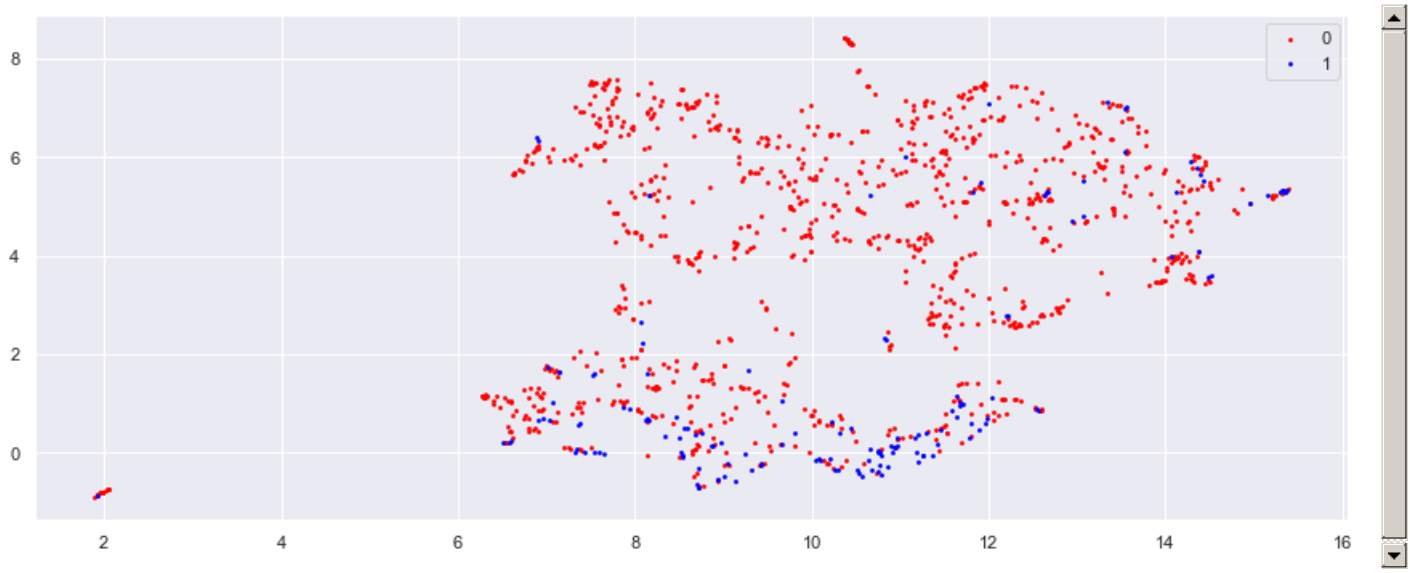
```
#X_balanced=X.loc[y==0,:].sample(n=200).append(X.loc[y==1,:].sample(n=200))
#y_balanced=RedWine_data.loc[X_balanced.index,'y']

#X_validation=X[~X.index.isin(X_balanced.index)]
#y_validation=RedWine_data.loc[~X.index.isin(X_balanced.index),'y']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,stratify=y,random_state=24)

scaler = StandardScaler()
scaler.fit(X_train)

X_train_scaled=pd.DataFrame(scaler.transform(X_train),columns = X.columns)
X_test_scaled=pd.DataFrame(scaler.transform(X_test),columns = X.columns)
#X_validation_scaled=pd.DataFrame(scaler.transform(X_validation),columns = X.columns)
```

4.3.3.2 Reducción de Dimensionalidad

Se realiza una proyección a 2 dimensiones de las muestras de X_train con UMAP y se proyecta el conjunto X_test a dos dimensiones luego lo visualizamos en un scatter plot, esto nos permitirá determinar si las clases van clasificarse de manera eficiente.

In [567]:

```
model = umap.UMAP(n_components=2, random_state=42)
model.fit(X_train_scaled)
X_train_projection = model.transform(X_train_scaled)
X_test_projection = model.transform(X_test_scaled)
X_validation_projection = model.transform(X_validation_scaled)

fig, ax = plt.subplots(1, 1, figsize=(12, 5))
for i in range(0,2):
    ax.scatter(X_train_projection[y_train == i,0], X_train_projection[y_train == i,1], s=3, label=str(i))
plt.legend()
plt.tight_layout()
```



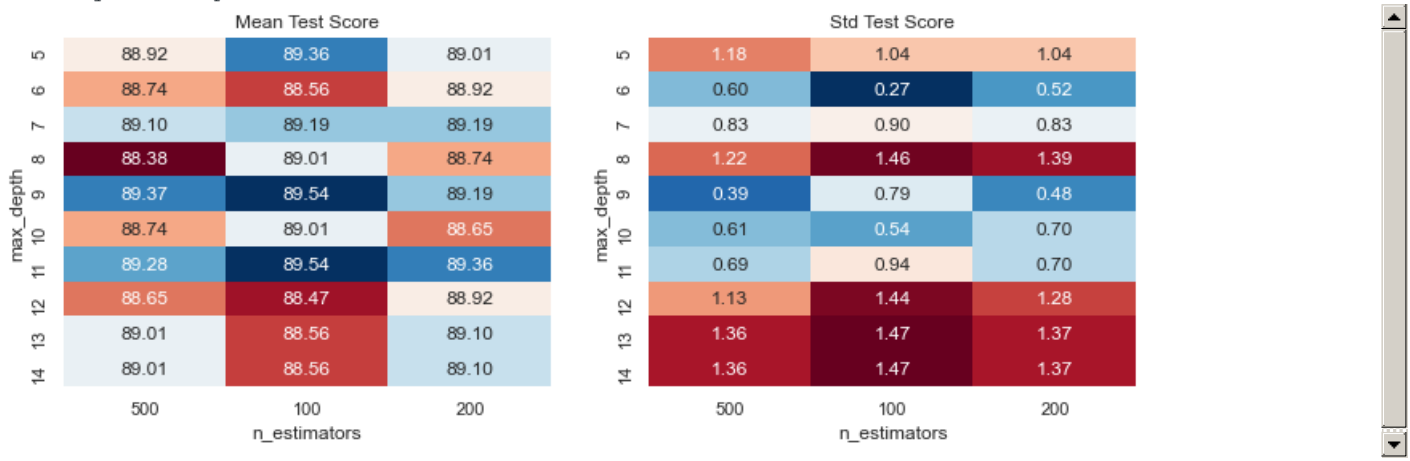
4.3.3.3 Entrenamiento Modelo Xtreme Gradient Boosting Classifier

Para seleccionar el mejor modelo primero calculamos el valor óptimo de los hiperparámetros `n_estimators` y `max_depth` . utilizando para esto una búsqueda Grid Search con validación cruzada para encontrar los valores óptimos, para cada combinación de valores, calculamos su promedio y la desviación estándar y finalmente en un *heatmap* para visualizamos la precisión según los diferentes valores de los hiperparámetros.

In [568]:

```
from xgboost import XGBClassifier

param_grid = {'max_depth': np.arange(5,15) ,
              'n_estimators': [500,100,200]  }
xtremeGradientboosting = XGBClassifier(use_label_encoder=False,eval_metric='logloss')
getParametrosOptimos(xtremeGradientboosting,param_grid,X_train_scaled, y_train)
```

```
Valor Óptimo del Hiperparámetro: {'max_depth': 9, 'n_estimators': 100}
Accuracy Valor Óptimo: 89.54%
```



Como resultado de la busqueda de parámetros óptimos se obtuvo los hiperparámetros `n_estimators=100` y `max_depth=9` con una accuracy en el conjunto de entrenamiento de **89.54%**

5. Resultados

Con los parámteros óptimos entrenamos el modelo y determinamos:

- El accuracy en el conjunto de test.
- La matriz de confusión de los resultados del modelo.

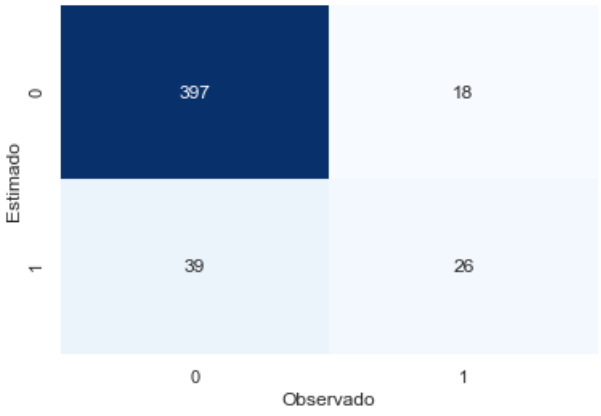
In [569]:

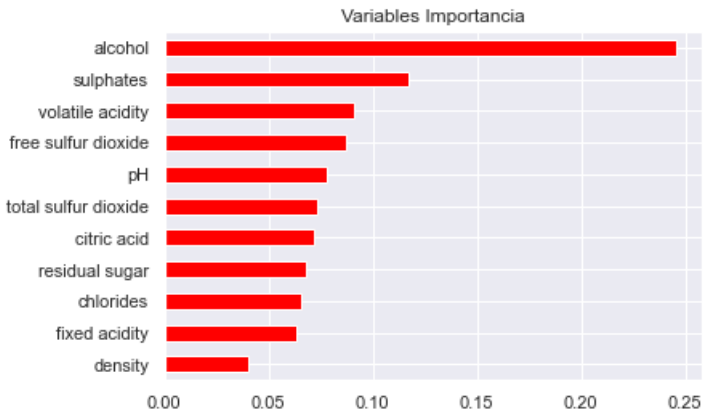
```
xtremeGradientBoosting = XGBClassifier(max_depth=9,n_estimators=100,use_label_encoder=False,eval_metric='
xtremeGradientBoosting.fit(X_train_scaled, y_train)

y_train_xgb = xtremeGradientBoosting.predict(X_train_scaled)
y_test_xgb = xtremeGradientBoosting.predict(X_test_scaled)
```

In [571]:

```
getMatrizConfusion(y_train,y_train_pred,y_test,y_test_xgb)

importances = pd.Series(xtremeGradientBoosting.feature_importances_, index = X_train.columns)
sorted_importances = importances.sort_values()
sorted_importances.plot(kind='barh', color='red')
plt.title('Variables Importancia')
plt.show()
```

```
Accuracy Test : 88.12%
Matriz de Confusión:
```





6. Conclusiones

En función de los resultdos, podemos concluir que se obtiene un modelo relativamente bueno para clasificar la calidad de los vinos, siendo más eficiente al clasificar a los de baja calidad, la precisión global del modelo es del **88.12%**.

Además hemos determinado que los atributos más importantes a la hora de clasificar a los vinos en Baja Calidad y Alta Calidad son: `alcohol` , `sulfatos` y `volatile acidity` , esto también se vio reflejado en el análisis de correlación realizado, puesto que estos atributos son los de indice de correlación más alto con relación a nuestra variable objetivo.