Análisis de Calidad de Vinos Tintos y Blancos

Descarga de Datos

```
1 import requests
2 import os
3 import pandas as pd
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import seaborn as sns
```

```
1
2 url_red = 'http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv'
3 url_white = 'http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv'
4
5 os.makedirs('../data/raw', exist_ok=True)
6 os.makedirs('../data/processed', exist_ok=True)
7
8 with open('../data/raw/winequality-red.csv', 'wb') as red:
9     red.write(requests.get(url_red).content)
10 with open('../data/raw/winequality-white.csv', 'wb') as white:
11     white.write(requests.get(url_white).content)
```

Combinar los Datos

```
1
2 red_df = pd.read_csv('../data/raw/winequality-red.csv', sep=';')
3 white_df = pd.read_csv('../data/raw/winequality-white.csv', sep=';')
4
5 red_df['type'] = 'red'
6 white_df['type'] = 'white'
7
8 wines_df = pd.concat([red_df, white_df], ignore_index=True)
9 wines_df.to_csv('../data/processed/wines.csv', index=False)
```

Conteo de Registros y Variables

```
1 print(wines_df.info())
2 print(wines_df.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         6497 non-null   float64
 1   volatile acidity      6497 non-null   float64
 2   citric acid           6497 non-null   float64
 3   residual sugar        6497 non-null   float64
 4   chlorides             6497 non-null   float64
 5   free sulfur dioxide   6497 non-null   float64
 6   total sulfur dioxide  6497 non-null   float64
 7   density               6497 non-null   float64
 8   pH                    6497 non-null   float64
 9   sulphates             6497 non-null   float64
 10  alcohol               6497 non-null   float64
 11  quality               6497 non-null   int64
 12  type                  6497 non-null   object
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
None
(6497, 13)
```

```
var_type = wines_df.dtypes.value_counts()
print(f'Observamos que tenemos {wines_df.shape[0]} registros, con {wines_df.isnull().sum().sum()} valores nu
print(f'El tipo de variables es: \n\n{var_type}')

```

```
Observamos que tenemos 6497 registros, con 0 valores nulos y 13 variables
El tipo de variables es:

float64    11
int64       1
object      1
Name: count, dtype: int64
```

Filtrar Atípicos y Manejar Datos Ausentes

A continuación, realizaremos un estudio de los datos con el objetivo de obtener conclusiones sobre las variables que influyen en la obtención de un vino de alta calidad, en contraposición a aquellas que resultan en vinos de calidad inferior.

```
wines_df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	6
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	

Filtrar Datos Ausentes

```
# Crear una copia para filtrar valores atípicos y nulos
df_wines_copy = wines_df.copy()
df_wines_copy.columns = df_wines_copy.columns.str.replace(' ', '_')
df_wines_copy.describe()
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxid
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.00000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.74457
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.52185
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.00000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.00000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.00000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.00000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.00000

```
df_wines_copy['type'].value_counts()
```

type	count
white	4898
red	1599

dtype: int64

```
print(df_wines_copy.isnull().sum())
```

```
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
type                    0
dtype: int64
```

```
# df_wines_copy.dropna(inplace=True)
```

Filtrar Valores Atípicos

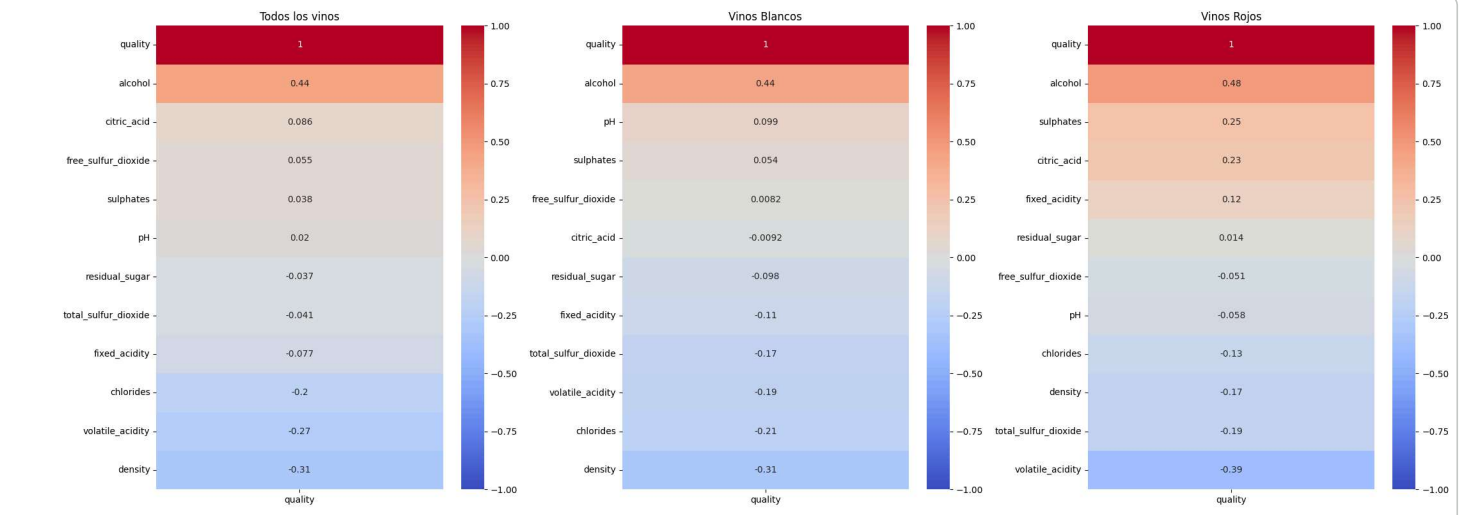
```
# Lista de columnas numéricas a analizar (en este caso todas las que aparecen en describe)
columnas_numericas = df_wines_copy.select_dtypes(include=['float64', 'int64']).columns
```

```
# Calcular la matriz de correlación para todos los tipos de vinos
corr_matrix = df_wines_copy[columnas_numericas].corr()
```

```
# Calcular la matriz de correlación para todos los vinos blancos

corr_matrix_white = df_wines_copy[df_wines_copy['type'] == 'white'][columnas_numericas].corr()
```

```
# Calcular la matriz de correlación para todos los vinos tintos

corr_matrix_red = df_wines_copy[df_wines_copy['type'] == 'red'][columnas_numericas].corr()
```

```
# Visualizar la correlación entre cada variable y la calidad del vino entre todos los vinos, rojos y blancos

fig, axes = plt.subplots(1, 3, figsize=(22, 8))

sns.heatmap(corr_matrix[['quality']].sort_values(by='quality', ascending=False),
            annot=True, cmap='coolwarm', vmin=-1, vmax=1, ax=axes[0])
axes[0].set_title('Todos los vinos')

sns.heatmap(corr_matrix_white[['quality']].sort_values(by='quality', ascending=False),
            annot=True, cmap='coolwarm', vmin=-1, vmax=1, ax=axes[1])
axes[1].set_title('Vinos Blancos')

sns.heatmap(corr_matrix_red[['quality']].sort_values(by='quality', ascending=False),
            annot=True, cmap='coolwarm', vmin=-1, vmax=1, ax=axes[2])
axes[2].set_title('Vinos Rojos')

plt.tight_layout()
plt.show()
```



Definir un umbral para las correlaciones negativas e Identificacion de las variables que afectan negativamente a la calidad del vino por separado.

```
# Definir un umbral para las correlaciones negativas
correlation_threshold = -0.1  # Por ejemplo, -0.1
```

```
# Identificar las variables que afectan negativamente a la calidad del vino tinto
negative_corr_vars_red = corr_matrix_red['quality'][corr_matrix_red['quality'] < correlation_threshold].inde
print("Variables que afectan negativamente a la calidad del vino tinto:", negative_corr_vars_red)

# Identificar las variables que afectan negativamente a la calidad del vino blanco
negative_corr_vars_white = corr_matrix_white['quality'][corr_matrix_white['quality'] < correlation_threshold
print("Variables que afectan negativamente a la calidad del vino blanco:", negative_corr_vars_white)
```

```
Variables que afectan negativamente a la calidad del vino tinto: ['volatile_acidity', 'chlorides', 'total_sulfur_dioxide'
Variables que afectan negativamente a la calidad del vino blanco: ['fixed_acidity', 'volatile_acidity', 'chlorides', 'tot
```

Funicon para filtrar los atipicos solo de las variables identificadas.

```
def remove_negative_outliers_iqr(df, negative_vars):
    df_filtered = df.copy()
    for column in negative_vars:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df_filtered = df_filtered[(df_filtered[column] >= lower_bound) & (df_filtered[column] <= upper_bound
    return df_filtered
```

```
# Separar los DataFrames por tipo de vino
df_red = df_wines_copy[df_wines_copy['type'] == 'red'].copy()
df_white = df_wines_copy[df_wines_copy['type'] == 'white'].copy()

# Aplicar la función de filtrado a los DataFrames de vinos tintos y blancos
df_red_filtered = remove_negative_outliers_iqr(df_red, negative_corr_vars_red)
df_white_filtered = remove_negative_outliers_iqr(df_white, negative_corr_vars_white)
```

```
# Realizamos un concatenado de los dos dataframes filtrados

df_wines_filtered = pd.concat([df_red_filtered, df_white_filtered], ignore_index=True)
```

```
print("Descripción del DataFrame original:")
display(df_wines_copy.describe())

print("\nDescripción del DataFrame filtrado:")
display(df_wines_filtered.describe())
```

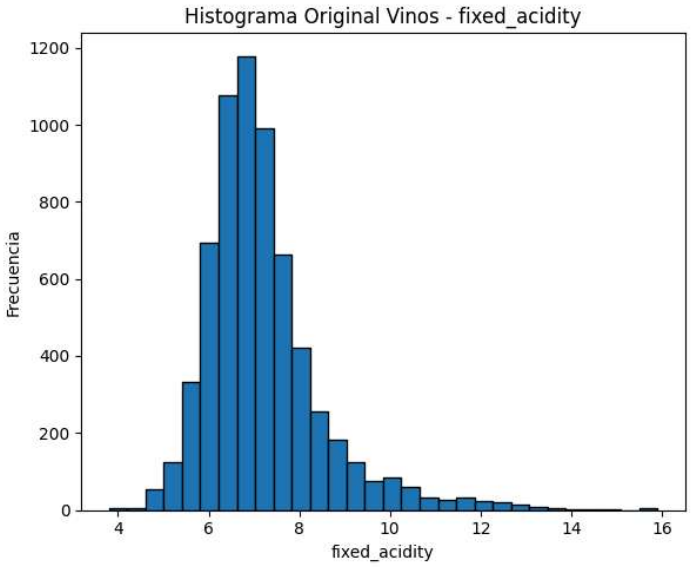
Descripción del DataFrame original:

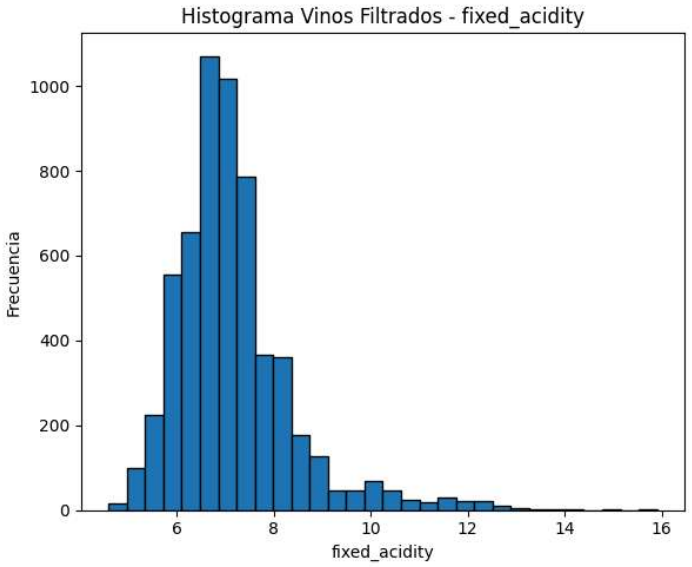
	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxid
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.00000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.74457
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.52185
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.00000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.00000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.00000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.00000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.00000

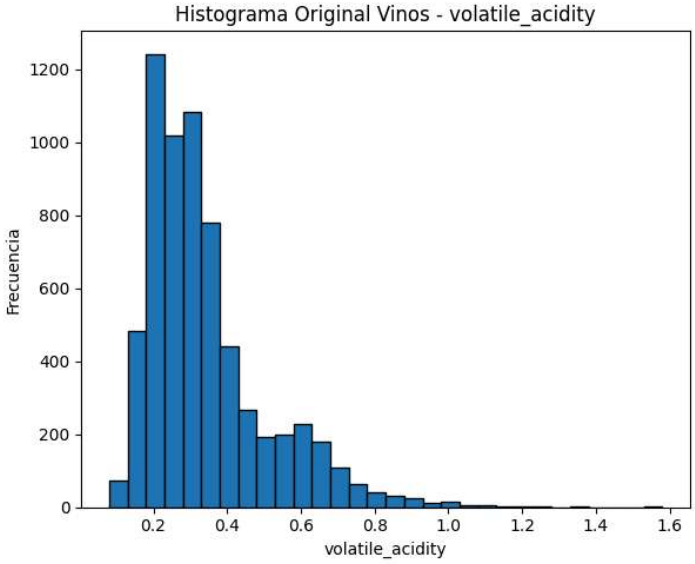
Descripción del DataFrame filtrado:

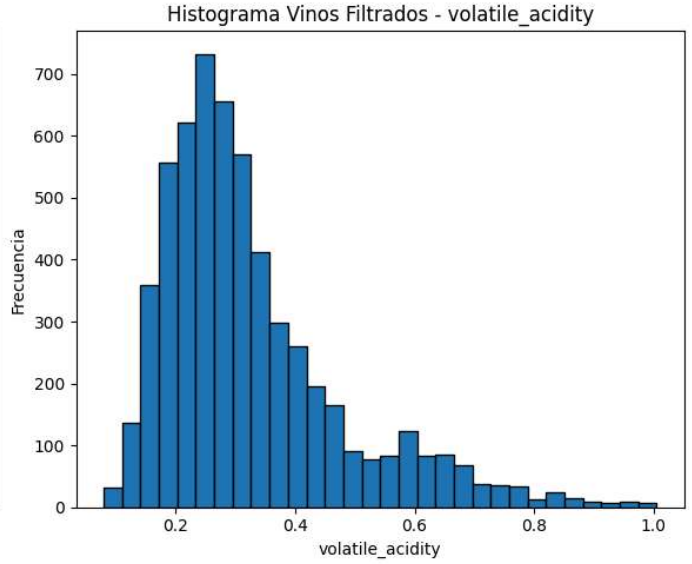
	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxid
count	5794.000000	5794.000000	5794.000000	5794.000000	5794.000000	5794.000000	5794.00000
mean	7.163971	0.325676	0.315906	5.482560	0.051123	30.687004	115.03702
std	1.218262	0.151971	0.137875	4.692735	0.019153	17.252842	55.70251
min	4.600000	0.080000	0.000000	0.600000	0.015000	1.000000	6.00000
25%	6.400000	0.220000	0.250000	1.800000	0.038000	17.000000	77.00000
50%	7.000000	0.280000	0.310000	3.000000	0.046000	29.000000	117.00000
75%	7.600000	0.380000	0.390000	8.200000	0.060000	42.000000	155.00000
max	15.900000	1.005000	1.660000	23.500000	0.119000	128.000000	255.00000

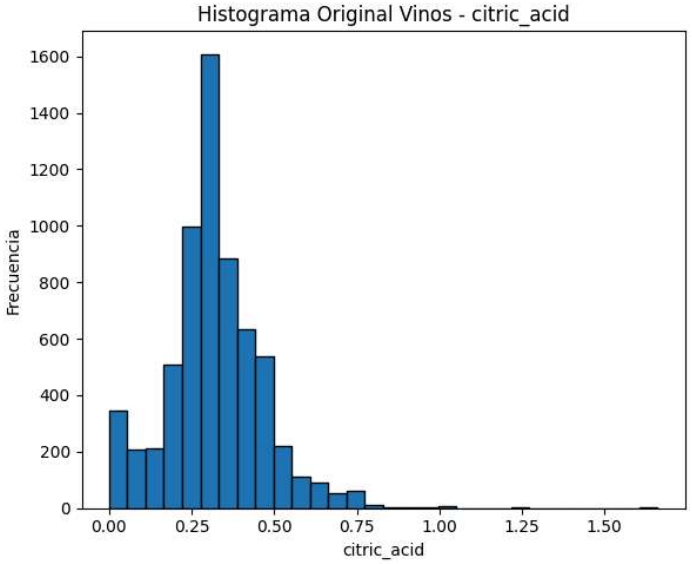
```
# Comparación de histogramas entre el DataFrame original y el filtrado

for col in columnas_numericas:
    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    plt.hist(df_wines_copy[col], bins=30, edgecolor='black')
    plt.title(f'Histograma Original Vinos - {col}')
    plt.xlabel(col)
    plt.ylabel('Frecuencia')

    plt.subplot(1, 2, 2)
    plt.hist(df_wines_filtered[col], bins=30, edgecolor='black')
    plt.title(f'Histograma Vinos Filtrados - {col}')
    plt.xlabel(col)
    plt.ylabel('Frecuencia')

    plt.tight_layout()
    plt.show()

```

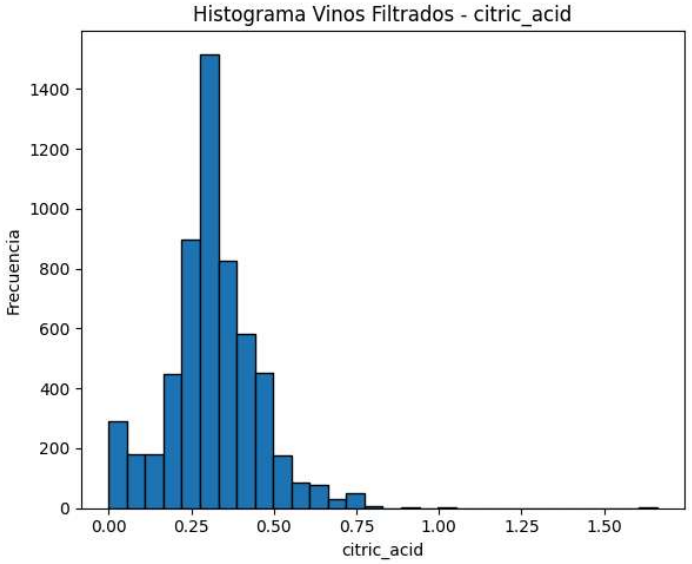

Histograma Original Vinos - fixed_acidity
Frecuencia
fixed_acidity

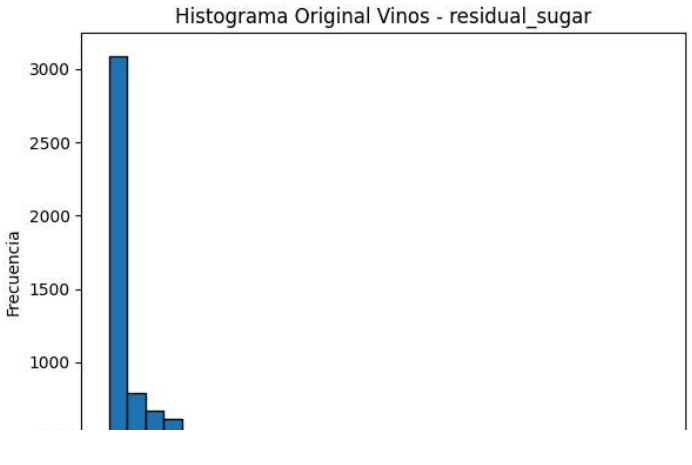

Histograma Vinos Filtrados - fixed_acidity
Frecuencia
fixed_acidity

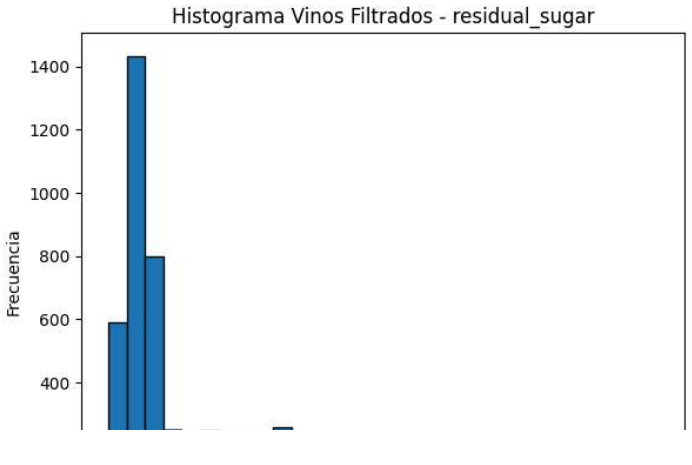

Histograma Original Vinos - volatile_acidity
Frecuencia
volatile_acidity


Histograma Vinos Filtrados - volatile_acidity
Frecuencia
volatile_acidity


Histograma Original Vinos - citric_acid
Frecuencia
citric_acid


Histograma Vinos Filtrados - citric_acid
Frecuencia
citric_acid


Histograma Original Vinos - residual_sugar
Frecuencia


Histograma Vinos Filtrados - residual_sugar
Frecuencia