TIPOLOGÍA Y CICLO DE VIDA DE LOS DATOS

Busquets Aran, Nil

05/01/2021

PRAC 2

Contenido

Práctica 2

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 datos Práctica 2 pág. 2 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.

Desarrollo

Descripción del dataset.

¿Por qué es importante y qué pregunta/problema pretende responder?

Este conjunto de datos se puede ver como tareas de clasificación o regresión. Las clases son ordenadas y no equilibradas (por ejemplo, hay vinos más normales que excelentes o pobres). Los algoritmos de detección atípicos podrían usarse para detectar los pocos vinos excelentes o pobres. Además, no estamos seguros de si todas las variables de entrada son relevantes. Por lo tanto, podría ser interesante probar los métodos de selección de características.

fixed acidity: La mayoría de los ácidos involucrados con el vino o fijos o no volátiles (no se evaporan fácilmente).

volatile acidity: La cantidad de ácido acético en el vino, que a niveles demasiado altos puede conducir a un sabor desagradable, vinagre.

citric acid: En pequeñas cantidades, el ácido cítrico puede añadir 'frescura' y sabor a los vinos.

residual sugar: La cantidad de azúcar restante después de las paradas de fermentación, es raro encontrar vinos con menos de 1 gramo/litro y los vinos con más de 45 gramos/litro se consideran dulces.

chlorides: La cantidad de sal en el vino.

free sulfur dioxide: La forma libre de SO2 existe en equilibrio entre sosmo molecular (como gas disuelto) e ion bisulfito; previene el crecimiento microbiano y la oxidación del vino.

Total sulfur dioxide: cantidad de formas libres y enlazadas de S02; en bajas concentraciones, so2 es en su mayoría indetectable en el vino, pero a concentraciones libres de SO2 superiores a 50 ppm, SO2 se hace evidente en la nariz y el sabor del vino.

density: La densidad de agua es cercana a la del agua dependiendo del porcentaje de contenido de alcohol y azúcar.

pH: describe cuán ácido o básico es un vino en una escala de 0 (muy ácido) a 14 (muy básico); la mayoría de los vinos están entre 3-4 en la escala de pH.

sulphates: Un aditivo vitivinícola que puede contribuir a los niveles de gas de dióxido de azufre (S02), que actúa como antimicrobiano y antioxidante.

alcohol: El porcentaje de alcoholemia del vino.

quality (score between 0 and 10): Variable de salida (basada en datos sensoriales, puntuación entre 0 y 10)

Integración datos

Integración y selección de los datos de interés a analizar.

A partir de este conjunto de datos se plantea la problemática de determinar qué variables influyen más sobre la calidad del vino. Además, se podrá crear modelos de regresión que permitan predecir la calidad del vino en función de sus características y contrastes de hipótesis que ayuden a identificar propiedades interesantes en las muestras.

Con este modelo de datos sería interesante ver que correlación y/o importancia tienen las diferentes características (fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sulphates, alcohol. quality) en la calidad del vino.

Limpieza de los datos.

La limpieza de datos es el proceso de corregir o eliminar datos incorrectos, dañados, con formato incorrecto, duplicados o incompletos dentro de un conjunto de datos. Si los datos son incorrectos, los resultados y los algoritmos no son fiables, aunque puedan parecer correctos. No hay una forma absoluta de prescribir los pasos exactos en el proceso de limpieza de datos porque los procesos variarán de un conjunto de datos a otro.

Antes de comenzar con la limpieza de los datos, procedemos a realizar la lectura del fichero en formato CSV en el que se encuentran. El resultado devuelto por la llamada a la función read_csv() será un objeto data.frame:

```
# Librerias que vamos a usar
import pandas as pd
import sns as sns

df = pd.read_csv("winequality-red.csv")

# Mostramos el numero de filas y columnas que tiene el dataset para tener una
idea clara
print("Rows, columns: " + str(df.shape))
```

```
Rows, columns: (1599, 12)
```

```
# Imprimimos la diez primeras lineas para ver si se ha importado correctamente
print(df.head(10))
```

Imprimimos el resultado de head para ver que los datos se han importado correctamente. El resultado de la importación es el siguiente:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
0	7.4	0.70	0.00	1.9	0.076	
1	7.8	0.88	0.00	2.6	0.098	
2	7.8	0.76	0.04	2.3	0.092	
3	11.2	0.28	0.56	1.9	0.075	
4	7.4	0.70	0.00	1.9	0.076	
5	7.4	0.66	0.00	1.8	0.075	
6	7.9	0.60	0.06	1.6	0.069	
7	7.3	0.65	0.00	1.2	0.065	
8	7.8	0.58	0.02	2.0	0.073	
9	7.5	0.50	0.36	6.1	0.071	

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	\
0	11.0	34.0	0.9978	3.51	0.56	
1	25.0	67.0	0.9968	3.20	0.68	
2	15.0	54.0	0.9970	3.26	0.65	
3	17.0	60.0	0.9980	3.16	0.58	
4	11.0	34.0	0.9978	3.51	0.56	
5	13.0	40.0	0.9978	3.51	0.56	
6	15.0	59.0	0.9964	3.30	0.46	
7	15.0	21.0	0.9946	3.39	0.47	
8	9.0	18.0	0.9968	3.36	0.57	
9	17.0	102.0	0.9978	3.35	0.80	

	alcohol	quality
0	9.4	5
1	9.8	5
2	9.8	5
3	9.8	6
4	9.4	5
5	9.4	5
6	9.4	5
7	10.0	7
8	9.5	7
9	10.5	5

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

```
# comprovamos si hay valores nulos, conocidos como NA, en alguna de las columnas
del dataframe
print(df.isna().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
dtype: int64
```

Obtenemos el siguiente resultado, con lo que observamos que no existe ningún valor nulo.

Observamos los distintos datos que contiene cada columna, todo y haber comprobado anteriormente que no existen valores nulos, pueden existir valores como 'unknown', '-', etc… que significan nulo igual, pero no son percibidos por la anterior función.

Substituimos los espacios por '_', para así poder ejecutar las funciones que requieren del nombre de la columna.

```
# Substituimos los espacios por _, para no tener problemas con futuras funciones
df.columns = df.columns.str.replace(' ', '_')
print(df.fixed_acidity.unique())

# Listamos los valores unicos para cada columna

print(df.fixed_acidity.unique())
print(df.volatile_acidity.unique())
print(df.citric_acid.unique())
print(df.residual_sugar.unique())
print(df.chlorides.unique())
print(df.free_sulfur_dioxide.unique())
print(df.total_sulfur_dioxide.unique())
print(df.density.unique())
print(df.pH.unique())
print(df.sulphates.unique())
print(df.alcohol.unique())
print(df.quality.unique())
```

Un ejemplo de resultado es:

```
[ 7.4  7.8 11.2  7.9  7.3  7.5  6.7  5.6  8.9  8.5  8.1  7.6  6.9  6.3
  7.1  8.3  5.2  5.7  8.8  6.8  4.6  7.7  8.7  6.4  6.6  8.6 10.2  7.
  7.2  9.3  8.   9.7  6.2  5.   4.7  8.4 10.1  9.4  9.   8.2  6.1  5.8
  9.2 11.5  5.4  9.6 12.8 11.  11.6 12.  15.  10.8 11.1 10.  12.5 11.8
 10.9 10.3 11.4  9.9 10.4 13.3 10.6  9.8 13.4 10.7 11.9 12.4 12.2 13.8
  9.1 13.5 10.5 12.6 14.  13.7  9.5 12.7 12.3 15.6  5.3 11.3 13.   6.5
 12.9 14.3 15.5 11.7 13.2 15.9 12.1  5.1  4.9  5.9  6.   5.5]
```

Observamos que no parece ningún valor diferente de lo normal, o de tipo string que pueda evidenciarse como valor nulo.

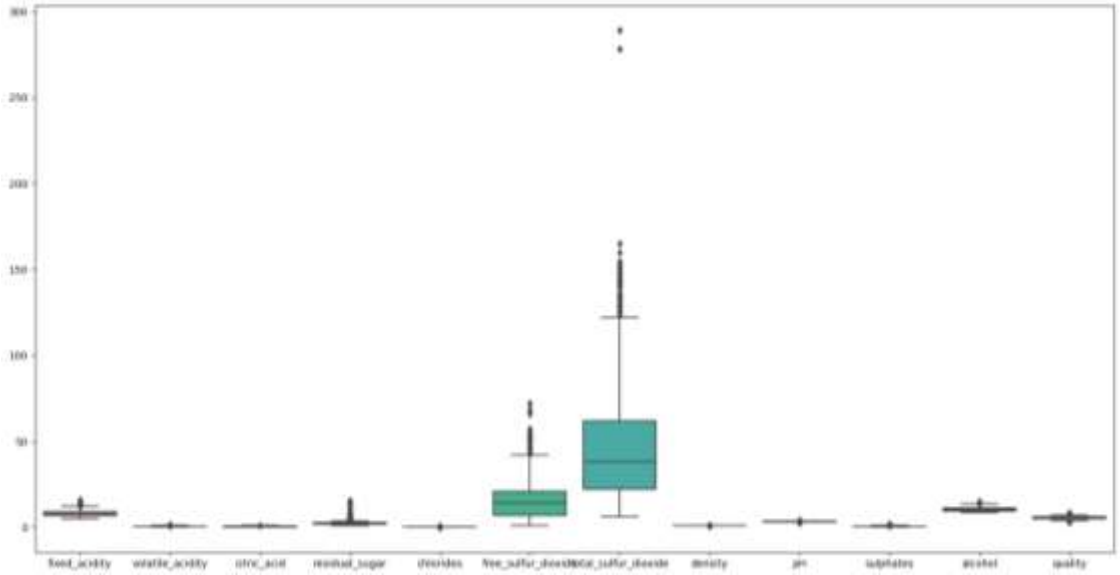
2. Identificación y tratamiento de valores extremos.

En el análisis clásico de cualquier muestra de datos se suele tratar los valores extremos debido a su gran impacto en el modelo de datos, los valores extremos son aquellos que parecen no ser congruentes sin los comparamos con el resto de los datos.

Formalmente, la Teoría de Valores Extremos (EVT de sus siglas en inglés) es la rama de la estadística que centra su estudio en los eventos asociados a las colas de la distribución (valores más altos o bajos de la variable sometida a estudio).

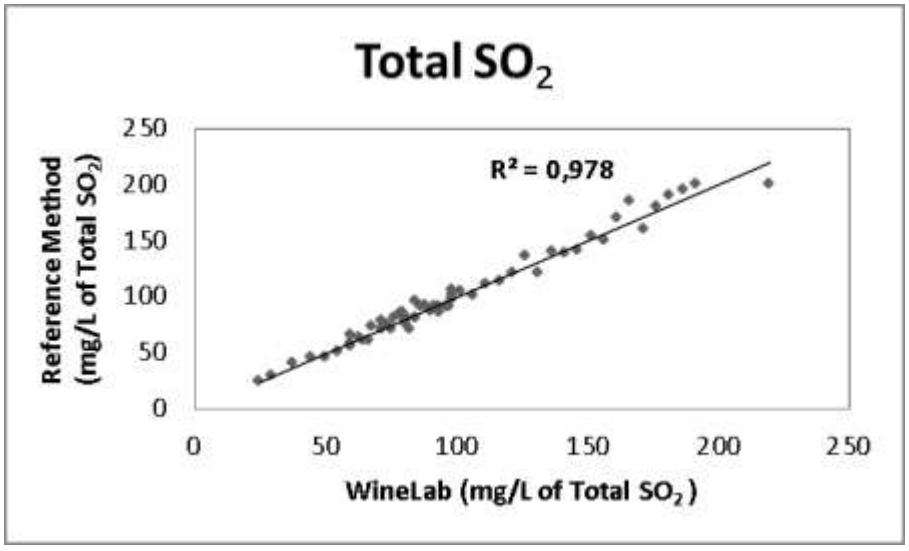
Creamos un boxplot de todas las columnas para ver si algún valor difiere mucho de la media.

```
# Create a boxplot
sns.boxplot(data=df, orient="v")
plt.show()
```



A primera vista parecería que todos los valores permanecen en un rango aceptable excepto 2 valores de total de dióxido de sulfuro.

Investigando sobre los valores del dióxido de sulfuro vemos que la mayoría se concentran entre 50 y 100 como también se representa en nuestro gráfico, pero cabe la posibilidad que haya valores altos cercanos a 250, con lo que los tomamos como válidos.



Análisis de los datos.

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

   Nuestro grupo que queremos analizar / comparar es la calidad del vino, para poder ver que dependencias tiene sobre los otros parámetros.
   Con lo que vamos a generar tres agrupaciones dependiendo de la calidad del vino, los nuevos grupos generados van a dividirse en:
   - Baja calidad de vino → [3, 4]
   - Media calidad de vino → [5, 6]
   - Alta calidad de vino → [7, 8]

```
# Create the new dataframes grouping by quality
low_quality_df = df[df.quality.isin([3, 4])]
medium_quality_df = df[df.quality.isin([5, 6])]
high_quality_df = df[df.quality.isin([7, 8])]

# Validate that the dataframes are correctly generated
print(low_quality_df.quality.unique())
print(medium_quality_df.quality.unique())
print(high_quality_df.quality.unique())
```

```
[4 3]
[5 6]
[7 8]
```

2. **Comprobación de la normalidad y homogeneidad de la varianza.**

   Normalidad

   Para la comprobación de que los valores que toman nuestras variables cuantitativas provienen de una población distribuida normalmente, utilizaremos la prueba de normalidad de AndersonDarling. Así, se comprueba que para que cada prueba se obtiene un p-valor superior al nivel de significación prefijado $\alpha = 0,05$. Si esto se cumple, entonces se considera que variable en cuestión sigue una distribución normal.

   Aplicamos la función de Anderson a cada columna del dataframe ya que todas las variables de dicho dataframe son numéricas.

```
# For each column of data we apply the Anderson Darling theorem
for column in df:
    print(column)
    print(anderson(df[column], dist="norm"))
```

   Obtenemos el siguiente resultado:

fixed_acidity

```
AndersonResult(statistic=28.14295750940846, critical_values=array([0.5
75, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

volatile_acidity

```
AndersonResult(statistic=5.6830748971976845, critical_values=array([0.
575, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

citric_acid

```
AndersonResult(statistic=17.54208740702461, critical_values=array([0.5
75, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

residual_sugar

```
AndersonResult(statistic=188.06444866412699, critical_values=array([0.
575, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

chlorides

```
AndersonResult(statistic=210.4491870499487, critical_values=array([0.5
75, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

free_sulfur_dioxide

```
AndersonResult(statistic=38.60990999200476, critical_values=array([0.5
75, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

total_sulfur_dioxide

```
AndersonResult(statistic=52.48865143001012, critical_values=array([0.5
75, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

density

```
AndersonResult(statistic=3.8675951923328284, critical_values=array([0.
575, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

pH

```
AndersonResult(statistic=1.8641116106432492, critical_values=array([0.
575, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

sulphates

```
AndersonResult(statistic=46.93219549223704, critical_values=array([0.5
75, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

alcohol

```
AndersonResult(statistic=34.91706402625596, critical_values=array([0.5
75, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

quality

```
AndersonResult(statistic=110.63276775926897, critical_values=array([0.
575, 0.654, 0.785, 0.916, 1.089]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

Con lo que se da a entender que ninguna característica sigue una distribución normal debido a que el estadístico obtenido en cada caso es siempre superior a los valores críticos de cada nivel significativo.
Probamos la prueba de Anderson a nivel de los nuevos datos agrupados por nivel de calidad del vino:

```
# Fo# For each column of data we apply the Anderson Darling theorem
for column in low_quality_df:
    print(column)
    print(anderson(low_quality_df[column], dist="norm"))

# For each column of data we apply the Anderson Darling theorem
for column in medium_quality_df:
    print(column)
    print(anderson(medium_quality_df[column], dist="norm"))

# For each column of data we apply the Anderson Darling theorem
for column in high_quality_df:
    print(column)
    print(anderson(high_quality_df[column], dist="norm"))
```

Con dicha prueba se observan diferencias comparado con el modelo de datos completo, ya que hay características en que se observa una distribución normal:

- volatile_acidity → statistic = 0.3908
- density → statistic = 0.2762
- pH → statistic = 0.8087
- alchol → statistic = 09295

Homogeneidad de la varianza

Seguidamente, pasamos a estudiar la homogeneidad de varianzas mediante la aplicación de un test de Fligner-Killeen. En este caso, estudiaremos esta homogeneidad en cuanto a los grupos conformados por los vinos de baja, media y alta calidad. En el siguiente prueba, la hipótesis nula consiste en que ambas varianzas son iguales.

```
# Apply the Flinger theory
for column in high_quality_df:
    print(column)
    print(fligner(low_quality_df[column], medium_quality_df[column],
high_quality_df[column]))
```

fixed_acidity
FlignerResult(statistic=16.951431578455242, pvalue=0.00020846992135310367)

volatile_acidity
FlignerResult(statistic=33.075008418852875, pvalue=6.574355395338488e-08)

citric_acid
FlignerResult(statistic=0.09478507093834285, pvalue=0.953712957941357)

residual_sugar
FlignerResult(statistic=4.92514259073604, pvalue=0.08521555466086454)

chlorides
FlignerResult(statistic=4.612191780452948, pvalue=0.09964953584103305)

free_sulfur_dioxide
FlignerResult(statistic=8.422558169104486, pvalue=0.014827390717266868)

total_sulfur_dioxide
FlignerResult(statistic=37.333787848640945, pvalue=7.817555522240422e-09)

density
FlignerResult(statistic=17.209075276200103, pvalue=0.00018327228072401752)

pH
FlignerResult(statistic=0.08450282148453186, pvalue=0.9586287407104894)

sulphates
FlignerResult(statistic=3.229175443528085, pvalue=0.19897268564095671)

alcohol
FlignerResult(statistic=3.109266097660288, pvalue=0.21126689305239074)

quality
FlignerResult(statistic=140.98508160870261, pvalue=2.4292852504386e-31)

Para los casos en los que obtenemos una p-value superior a 0,05 podemos aceptar la hipótesis de que las varianzas de ambas muestras son homogéneas.

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.

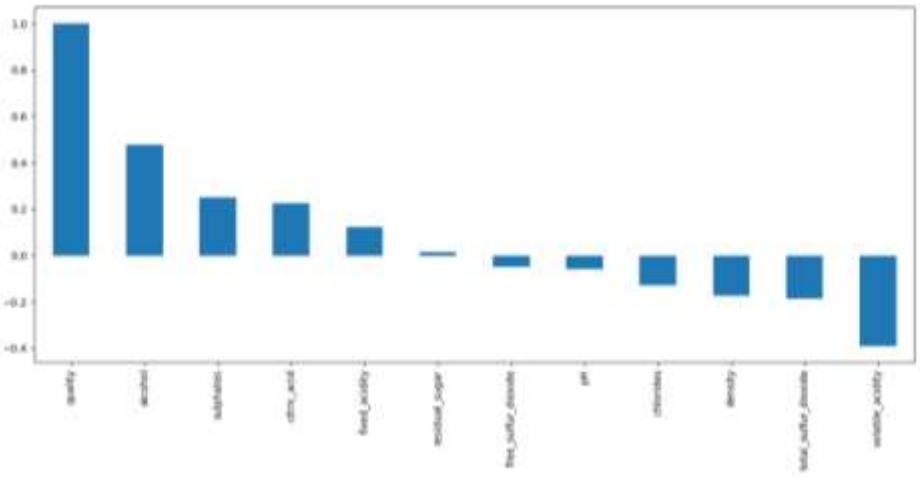
Correlación

Creamos una matriz de correlación para ver las influencias entre todas las variables para tener una imagen general del comportamiento del modelo.

```
# Correlations between variables
sns.heatmap(df.corr(), annot=True, fmt='.0%')
plt.show()
```

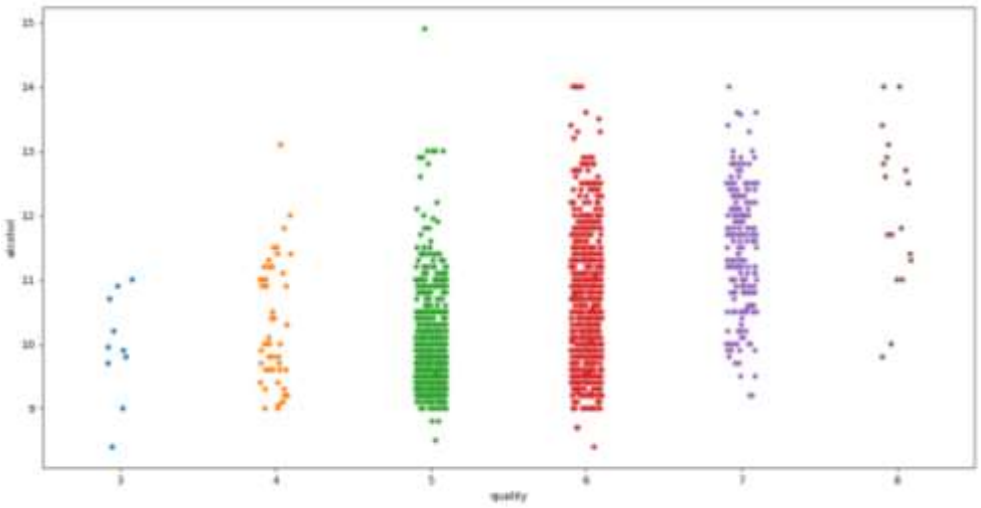
Como hemos focalizado desde un principio hemos querido ver que correlación tienen las variables frente la calidad del vino, vamos a mostrar los mismos datos, pero solo para el vino en forma de grafico para tener una imagen más nítida de lo que necesitamos.

```
# Calculate and order correlations
plt.figure(figsize=(10, 6)).subplots_adjust(bottom=0.25)
df.corr()['quality'].sort_values(ascending=False).plot(kind='bar')
plt.show()
```



Como vemos que la correlación más alta es el nivel de alcohol vamos a focalizar en esta para ver a profundidad el comportamiento.

```
# Matrix correlation between all variables
plt.figure(figsize=(10, 6))
sns.stripplot(data=df, x="quality", y="alcohol", jitter=True)
plt.show()
```



Podemos observar un comportamiento directo, ya que a más cantidad de alcohol la calidad del vino es superior.

Regresión

Tal y como se planteó en los objetivos de la actividad, resultará de mucho interés poder realizar predicciones sobre la calidad del vino dadas sus características más importantes.

Así, se calculará un modelo de regresión lineal utilizando regresores cuantitativos con el que poder realizar las predicciones de la calidad. Para obtener un modelo de regresión lineal considerablemente eficiente, lo que haremos será obtener varios modelos de regresión utilizando las variables que estén más correladas con respecto a la calidad, según el gráfico de barras obtenido anteriormente. Así, de entre todos los modelos que tengamos, escogeremos el mejor utilizando como criterio aquel que presente un mayor coeficiente de determinación (R2).

Creamos 4 modelos diferentes con las características 5 características más influentes sobre la calidad y reducimos una característica por modelo.

```
# We separate our features from our target feature (quality) and we split data
intro training and test
X1 = df.loc[:, ['alcohol', 'sulphates', 'citric_acid', 'volatile_acidity',
'total_sulfur_dioxide']]
X2 = df.loc[:, ['alcohol', 'sulphates', 'citric_acid', 'volatile_acidity']]
X3 = df.loc[:, ['alcohol', 'sulphates', 'citric_acid']]
X4 = df.loc[:, ['alcohol', 'sulphates']]
Y = df.iloc[:, 11]

# Create the test and training samples
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, Y, test_size=0.4,
random_state=42)
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, Y, test_size=0.4,
random_state=42)
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, Y, test_size=0.4,
random_state=42)
X4_train, X4_test, y4_train, y4_test = train_test_split(X4, Y, test_size=0.4,
random_state=42)

# Fit the model and make prediction
reg1 = LinearRegression()
reg2 = LinearRegression()
reg3 = LinearRegression()
reg4 = LinearRegression()
reg1.fit(X1_train, y1_train)
reg2.fit(X2_train, y2_train)
reg3.fit(X3_train, y3_train)
reg4.fit(X4_train, y4_train)
y1_prediction_lr = reg1.predict(X1_test)
y2_prediction_lr = reg2.predict(X2_test)
y3_prediction_lr = reg3.predict(X3_test)
y4_prediction_lr = reg4.predict(X4_test)
```

Seguimos con la evaluación del modelo mediante el coeficiente de determinación

```
# Evaluate our models
print(sqrt(mean_squared_error(y1_test, y1_prediction_lr)))
print(sqrt(mean_squared_error(y2_test, y2_prediction_lr)))
print(sqrt(mean_squared_error(y3_test, y3_prediction_lr)))
print(sqrt(mean_squared_error(y4_test, y4_prediction_lr)))
```

El resultado obtenido es el siguiente:

R^2(M1) = 0.6660516631947361

R^2(M2) = 0.6698700649462583

R^2(M3) = 0.6894807520440475

R^2(M4) = 0.6998235210158352

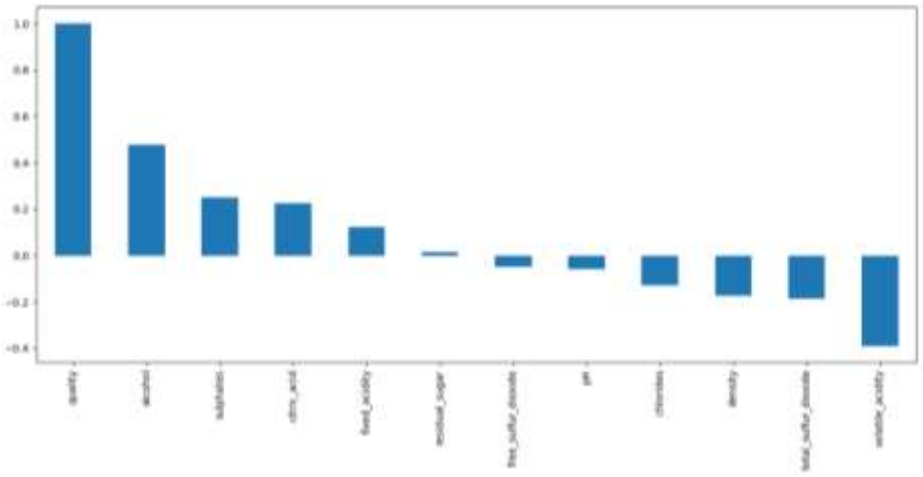
Como conclusión Podemos ver que el modelo más representativo es el que solo tiene 2 características, pero son las más influyentes.

Representación datos

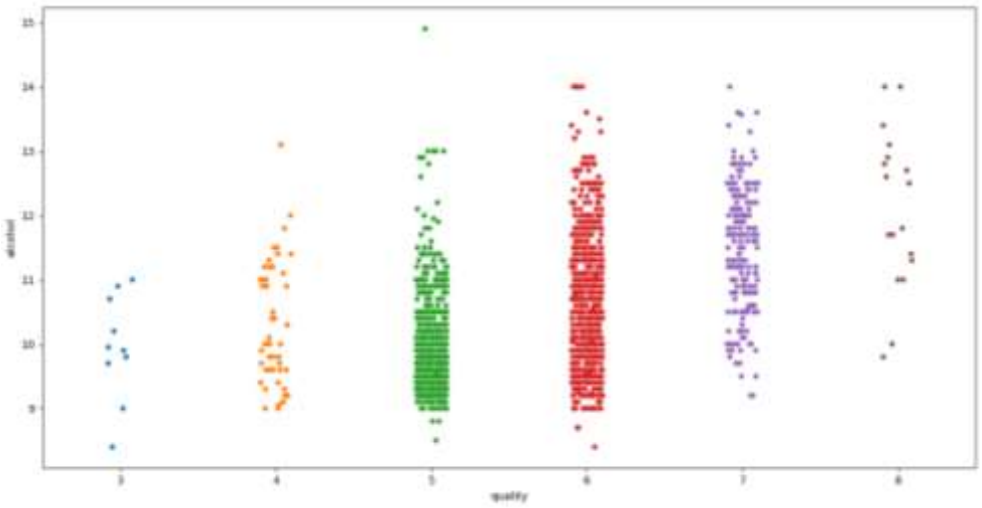
Representación de los resultados a partir de tablas y gráficas.

Imágenes I explicaciones representadas en el punto 4

1 Matriz de correlación



2 Correlación calidad



3 Correlación calidad - alcohol

Resolución problema

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

Podemos concluir que las características más importantes e influentes en la calidad del vino son la graduación de alcohol y sulphates.

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.

```
# library we are using
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import anderson
from scipy.stats import fligner
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from math import sqrt


# Create the dataframe coming from the csv by using pandas.read_csv() function
df = pd.read_csv("wine_quality-red.csv")

# print the head of the dataframe
print(df.head(10))
# show the number of rows and columns of the dataframe
print("Rows, columns: " + str(df.shape))

# Check null values
print(df.isna().sum())

# Replace spaces with _ for each column to avoid future problems
df.columns = df.columns.str.replace(' ', '_')

# Print all column names
for col_name in df.columns:
    print(col_name)

# List the unique values for each column
print(df.fixed_acidity.unique())
print(df.volatile_acidity.unique())
print(df.citric_acid.unique())
print(df.residual_sugar.unique())
print(df.chlorides.unique())
print(df.free_sulfur_dioxide.unique())
print(df.total_sulfur_dioxide.unique())
print(df.density.unique())
print(df.pH.unique())
print(df.sulphates.unique())
print(df.alcohol.unique())
print(df.quality.unique())
```

```
# Create a boxplot
sns.boxplot(data=df, orient="v")
plt.show()

# Create the new dataframes grouping by quality
low_quality_df = df[df.quality.isin([3, 4])]
medium_quality_df = df[df.quality.isin([5, 6])]
high_quality_df = df[df.quality.isin([7, 8])]

# Validate that the dataframes are correctly generated
print(low_quality_df.quality.unique())
print(medium_quality_df.quality.unique())
print(high_quality_df.quality.unique())

# For each column of data we apply the Anderson Darling theorem
for column in df:
    print(column)
    print(anderson(df[column], dist="norm"))

# For each column of data we apply the Anderson Darling theorem
for column in low_quality_df:
    print(column)
    print(anderson(low_quality_df[column], dist="norm"))

# For each column of data we apply the Anderson Darling theorem
for column in medium_quality_df:
    print(column)
    print(anderson(medium_quality_df[column], dist="norm"))

# For each column of data we apply the Anderson Darling theorem
for column in high_quality_df:
    print(column)
    print(anderson(high_quality_df[column], dist="norm"))

# Apply the Flinger theory
for column in high_quality_df:
    print(column)
    print(fligner(low_quality_df[column], medium_quality_df[column],
high_quality_df[column]))

# CORRELATION
# Correlations between variables
plt.figure(figsize=(10, 6)).subplots_adjust(bottom=0.25)
sns.heatmap(df.corr(), annot=True, fmt='.0%')
plt.show()

# Calculate and order correlations
plt.figure(figsize=(10, 6)).subplots_adjust(bottom=0.25)
df.corr()['quality'].sort_values(ascending=False).plot(kind='bar')
plt.show()

# Matrix correlation between all variables
plt.figure(figsize=(10, 6))
sns.stripplot(data=df, x="quality", y="alcohol", jitter=True)
plt.show()
```

```
# REGRESSION
# We separate our features from our target feature (quality) and we split data
intro training and test
X1 = df.loc[:, ['alcohol', 'sulphates', 'citric_acid', 'volatile_acidity',
'total_sulfur_dioxide']]
X2 = df.loc[:, ['alcohol', 'sulphates', 'citric_acid', 'volatile_acidity']]
X3 = df.loc[:, ['alcohol', 'sulphates', 'citric_acid']]
X4 = df.loc[:, ['alcohol', 'sulphates']]
Y = df.iloc[:, 11]

# Create the test and training samples
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, Y, test_size=0.4,
random_state=42)
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, Y, test_size=0.4,
random_state=42)
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, Y, test_size=0.4,
random_state=42)
X4_train, X4_test, y4_train, y4_test = train_test_split(X4, Y, test_size=0.4,
random_state=42)

# Fit the model and make prediction
reg1 = LinearRegression()
reg2 = LinearRegression()
reg3 = LinearRegression()
reg4 = LinearRegression()
reg1.fit(X1_train, y1_train)
reg2.fit(X2_train, y2_train)
reg3.fit(X3_train, y3_train)
reg4.fit(X4_train, y4_train)
y1_prediction_lr = reg1.predict(X1_test)
y2_prediction_lr = reg2.predict(X2_test)
y3_prediction_lr = reg3.predict(X3_test)
y4_prediction_lr = reg4.predict(X4_test)

# Evaluate our models
print(sqrt(mean_squared_error(y1_test, y1_prediction_lr)))
print(sqrt(mean_squared_error(y2_test, y2_prediction_lr)))
print(sqrt(mean_squared_error(y3_test, y3_prediction_lr)))
print(sqrt(mean_squared_error(y4_test, y4_prediction_lr)))
```

Bibliografía

Kaggle.com. 2021. *Kaggle: Your Machine Learning And Data Science Community*. [online] Available at: <https://www.kaggle.com/> [Accessed 1 January 2021].

Kaggle.com. 2021. *Covid-19 Case Surveillance Public Use Dataset*. [online] Available at: <https://www.kaggle.com/arashnic/covid19-case-surveillance-public-use-dataset> [Accessed 1 January 2021].

Data.cdc.gov. 2021. [online] Available at: <https://data.cdc.gov/Case-Surveillance/COVID-19-Case-Surveillance-Public-Use-Data/vbim-akqf> [Accessed 1 January 2021].

Cdrfoodlab.com. 2021. *Total Sulfur Dioxide - Wine Test*. [online] Available at: <https://www.cdrfoodlab.com/foods-beverages-analysis/total-sulfur-dioxide-wine/> [Accessed 4 January 2021].

2021. [online] Available at: <https://aprendeia.com/agrupando-los-datos-con-python/> [Accessed 4 January 2021].

Docs.scipy.org. 2021. *Scipy.Stats.Fligner — Scipy V1.6.0 Reference Guide*. [online] Available at: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.fligner.html> [Accessed 4 January 2021].

Medium. 2021. *Red Wine Quality Prediction Using Regression Modeling And Machine Learning*. [online] Available at: <https://towardsdatascience.com/red-wine-quality-prediction-using-regression-modeling-and-machine-learning-7a3e2c3e1f46> [Accessed 5 January 2021].

Briega, R., 2021. *Probabilidad Y Estadística Con Python*. [online] Relopezbriega.github.io. Available at: <https://relopezbriega.github.io/blog/2015/06/27/probabilidad-y-estadistica-con-python/> [Accessed 5 January 2021].

Contribución	Firma
Investigación previa	Nil Busquets Aran
Redacción de las respuestas	Nil Busquets Aran
Desarrollo código	Nil Busquets Aran