Rapport MAD
Projet d'Analyse de données

DOZ Louka et LAGUILLON Céline
Année universitaire 2021 - 2022

Sommaire

1 Pré-analyse

1.1 Source

Lien : https://archive.ics.uci.edu/ml/datasets/Wine+Quality.
Les données proviennent du site UC Irvine, proposant de nombreux jeux de données.

1.2 Importation des données

La première étape est d'importer les données comme ceci :

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

df_red = pd.read_csv("winequality-red.csv", sep=";")
df_white = pd.read_csv("winequality-white.csv", sep=";")
```

Deux jeux de données ont été importés: un sur les vins rouges, un sur les vins blancs. Nous obtenons alors deux tableaux comme ci-dessous.

[2]:
```
df_red
```

[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
0	7.4	0.700	0.00	1.9	0.076	
1	7.8	0.880	0.00	2.6	0.098	
2	7.8	0.760	0.04	2.3	0.092	
3	11.2	0.280	0.56	1.9	0.075	
4	7.4	0.700	0.00	1.9	0.076	
...	...	...	...	...	...	
1594	6.2	0.600	0.08	2.0	0.090	
1595	5.9	0.550	0.10	2.2	0.062	
1596	6.3	0.510	0.13	2.3	0.076	
1597	5.9	0.645	0.12	2.0	0.075	
1598	6.0	0.310	0.47	3.6	0.067	

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	\
0	11.0	34.0	0.99780	3.51	0.56	
1	25.0	67.0	0.99680	3.20	0.68	
2	15.0	54.0	0.99700	3.26	0.65	
3	17.0	60.0	0.99800	3.16	0.58	
4	11.0	34.0	0.99780	3.51	0.56	
...	...	...	...	...	...	
1594	32.0	44.0	0.99490	3.45	0.58	
1595	39.0	51.0	0.99512	3.52	0.76	
1596	29.0	40.0	0.99574	3.42	0.75	
1597	32.0	44.0	0.99547	3.57	0.71	

```
1598                18.0                  42.0  0.99549  3.39       0.66

      alcohol  quality
0         9.4        5
1         9.8        5
2         9.8        5
3         9.8        6
4         9.4        5
...       ...      ...
1594     10.5        5
1595     11.2        6
1596     11.0        6
1597     10.2        5
1598     11.0        6

[1599 rows x 12 columns]
```

[3]:
```
df_white
```

[3]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.0              0.27         0.36            20.7      0.045
1               6.3              0.30         0.34             1.6      0.049
2               8.1              0.28         0.40             6.9      0.050
3               7.2              0.23         0.32             8.5      0.058
4               7.2              0.23         0.32             8.5      0.058
...             ...               ...          ...             ...        ...
4893            6.2              0.21         0.29             1.6      0.039
4894            6.6              0.32         0.36             8.0      0.047
4895            6.5              0.24         0.19             1.2      0.041
4896            5.5              0.29         0.30             1.1      0.022
4897            6.0              0.21         0.38             0.8      0.020

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    45.0                 170.0  1.00100  3.00       0.45
1                    14.0                 132.0  0.99400  3.30       0.49
2                    30.0                  97.0  0.99510  3.26       0.44
3                    47.0                 186.0  0.99560  3.19       0.40
4                    47.0                 186.0  0.99560  3.19       0.40
...                   ...                   ...      ...   ...        ...
4893                 24.0                  92.0  0.99114  3.27       0.50
4894                 57.0                 168.0  0.99490  3.15       0.46
4895                 30.0                 111.0  0.99254  2.99       0.46
4896                 20.0                 110.0  0.98869  3.34       0.38
4897                 22.0                  98.0  0.98941  3.26       0.32

      alcohol  quality
0         8.8        6
```

```
1          9.5        6
2         10.1        6
3          9.9        6
4          9.9        6
...        ...      ...
4893      11.2        6
4894       9.6        5
4895       9.4        6
4896      12.8        7
4897      11.8        6

[4898 rows x 12 columns]
```

Il y a deux tableaux, l'un de 1599 lignes, l'autre de 4898 lignes et tous les deux de 13 colonnes. Ces colonnes sont les mêmes entre les deux tableaux, de même pour le type des variables. Les tableaux peuvent alors être fusionnés sans formatage.

[4]:
```
df = pd.concat([df_red, df_white], ignore_index=True)
df
```

[4]:
```
      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
...             ...               ...          ...             ...        ...
6492            6.2              0.21         0.29             1.6      0.039
6493            6.6              0.32         0.36             8.0      0.047
6494            6.5              0.24         0.19             1.2      0.041
6495            5.5              0.29         0.30             1.1      0.022
6496            6.0              0.21         0.38             0.8      0.020

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
...                   ...                   ...      ...   ...        ...
6492                 24.0                  92.0  0.99114  3.27       0.50
6493                 57.0                 168.0  0.99490  3.15       0.46
6494                 30.0                 111.0  0.99254  2.99       0.46
6495                 20.0                 110.0  0.98869  3.34       0.38
6496                 22.0                  98.0  0.98941  3.26       0.32

      alcohol  quality
```

```
0          9.4        5
1          9.8        5
2          9.8        5
3          9.8        6
4          9.4        5
...        ...      ...
6492      11.2        6
6493       9.6        5
6494       9.4        6
6495      12.8        7
6496      11.8        6

[6497 rows x 12 columns]
```

On obtient alors un tableaux de 14 variables et 6497 individus.

1.3 Description des variables

Colonne	Signification
fixed acidity	L'acidité du vin.
volatile acidity	La quantité d'acide acétique dans le vin.
citric acid	La quantité d'acide citrique dans le vin.
residual sugar	La quantité de sucre restant dans le vin après la fermentation.
chlorides	La quantité de sel dans le vin.
free sulfur dioxide	La forme libre du SO2 existant en équilibre dans le vin.
total sulfur dioxide	La quantité de formes libres et liées de S02.
density	La densité de l'eau dans le vin, qui dépend du pourcentage d'alcool et de la quantité de sucre.
pH	Le pH décrit le degré d'acidité ou de basicité d'un vin sur une échelle allant de 0 (très acide) à 14 (très basique).
sulphates	La quantité d'additifs dans le vin.
alcohol	Le pourcentage d'alcool dans le vin.
quality	La qualité du vin sur une échelle de 0 à 10.

1.4 Données manquantes

[5]:
```
is_NaN = df.isnull()
row_has_NaN = is_NaN.any(axis=1)
rows_with_NaN = df[row_has_NaN]
df = df.drop(rows_with_NaN.index)
display(rows_with_NaN)
rows_with_NaN.shape
```

```
Empty DataFrame
Columns: [fixed acidity, volatile acidity, citric acid, residual sugar, chlorides,␣
 ↪free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol,␣
 ↪quality]
```

```
Index: []
```

[5]:
```
(0, 12)
```

Ce code affiche toutes les lignes qui contiennent au moins une valeur nulle. Nous pouvons alors conclure qu'aucune valeur n'est nulle puisqu'aucune ligne n'est affichée.

1.5 Type des variables

Nous pouvons étudier les valeurs uniques des variables pour voir quelle forme elles prennent. **Nous n'affichons pas toutes les variables dans le rapport pour éviter de trop afficher.**

[6]:
```
for c in ['chlorides', 'total sulfur dioxide', 'quality']:
    print(c, " : ", np.sort(pd.unique(df[c])))
```

```
chlorides  :  [0.009 0.012 0.013 0.014 0.015 0.016 0.017 0.018 0.019 0.02  0.021
0.022
 0.023 0.024 0.025 0.026 0.027 0.028 0.029 0.03  0.031 0.032 0.033 0.034
 0.035 0.036 0.037 0.038 0.039 0.04  0.041 0.042 0.043 0.044 0.045 0.046
 0.047 0.048 0.049 0.05  0.051 0.052 0.053 0.054 0.055 0.056 0.057 0.058
 0.059 0.06  0.061 0.062 0.063 0.064 0.065 0.066 0.067 0.068 0.069 0.07
 0.071 0.072 0.073 0.074 0.075 0.076 0.077 0.078 0.079 0.08  0.081 0.082
 0.083 0.084 0.085 0.086 0.087 0.088 0.089 0.09  0.091 0.092 0.093 0.094
 0.095 0.096 0.097 0.098 0.099 0.1   0.101 0.102 0.103 0.104 0.105 0.106
 0.107 0.108 0.109 0.11  0.111 0.112 0.113 0.114 0.115 0.116 0.117 0.118
 0.119 0.12  0.121 0.122 0.123 0.124 0.125 0.126 0.127 0.128 0.13  0.132
 0.133 0.135 0.136 0.137 0.138 0.142 0.143 0.144 0.145 0.146 0.147 0.148
 0.149 0.15  0.152 0.153 0.154 0.156 0.157 0.158 0.159 0.16  0.161 0.165
 0.166 0.167 0.168 0.169 0.17  0.171 0.172 0.173 0.174 0.175 0.176 0.178
 0.179 0.18  0.184 0.185 0.186 0.19  0.194 0.197 0.2   0.201 0.204 0.205
 0.208 0.209 0.211 0.212 0.213 0.214 0.216 0.217 0.222 0.226 0.23  0.235
 0.236 0.239 0.24  0.241 0.243 0.244 0.25  0.255 0.263 0.267 0.27  0.271
 0.29  0.301 0.332 0.337 0.341 0.343 0.346 0.358 0.36  0.368 0.369 0.387
 0.401 0.403 0.413 0.414 0.415 0.422 0.464 0.467 0.61  0.611]
total sulfur dioxide  :  [  6.    7.    8.    9.   10.   11.   12.   13.   14.
15.   16.   17.
  18.   19.   20.   21.   22.   23.   24.   25.   26.   27.   28.   29.
  30.   31.   32.   33.   34.   35.   36.   37.   38.   39.   40.   41.
  42.   43.   44.   45.   46.   47.   48.   49.   50.   51.   52.   53.
  54.   55.   56.   57.   58.   59.   60.   61.   62.   63.   64.   65.
  66.   67.   68.   69.   70.   71.   72.   73.   74.   75.   76.   77.
  77.5  78.   79.   80.   81.   82.   83.   84.   85.   86.   87.   88.
  89.   90.   91.   92.   93.   94.   95.   96.   97.   98.   99.  100.
 101.  102.  103.  104.  105.  106.  107.  108.  109.  110.  111.  112.
 113.  114.  115.  115.5 116.  117.  118.  119.  120.  121.  122.  123.
 124.  125.  126.  127.  128.  129.  129.5 130.  131.  132.  133.  134.
 135.  136.  137.  138.  139.  140.  141.  142.  143.  144.  145.  146.
 147.  148.  149.  150.  151.  152.  153.  154.  155.  156.  157.  158.
```

```
 159.  160.  161.  162.  162.5 163.  164.  164.5 165.  166.  167.  168.
 169.  170.  171.  172.  173.  174.  175.  176.  176.5 177.  178.  179.
 180.  181.  182.  183.  184.  185.  186.  187.  188.  189.  189.5 190.
 191.  192.  193.  194.  195.  196.  197.  198.  199.  200.  201.  202.
 203.  204.  205.  206.  207.  208.  209.  210.  211.  212.  212.5 213.
 214.  215.  216.  216.5 217.  217.5 218.  218.5 219.  219.5 220.  221.
 222.  223.  224.  225.  226.  227.  228.  229.  230.  231.  232.  233.
 234.  234.5 235.  236.  237.  238.  238.5 240.  241.  242.  243.  244.
 245.  246.  247.  248.  249.  249.5 251.  252.  253.  255.  256.  259.
 260.  272.  278.  282.  289.  294.  303.  307.5 313.  344.  366.5 440. ]
quality  :  [3 4 5 6 7 8 9]
```

Nous pouvons conclure qu'aucune valeur aberrante n'a été trouvée et que quality est la seule variable qualitative. C'est une variable qualitative ordinale, puisqu'elle contient un ordre.
L'ensemble des autres variables sont quantitatives discrètes puisque leurs valeurs sont dénombrables.

1.6 Problématique

Nous allons étudier ce jeux de données pour tenter de répondre à la problématique suivante : Quelles sont les composantes principales du vin qui impactent sa qualité ?

Pour ce faire, nous allons étudier les variables de manière univariée et bivariée, et effectuer une Analyse en Composantes Principales, pour tenter de mettre en lumière les variables les plus importantes et leur corrélation avec la qualité du vin.

2 Analyse univariée

2.1 Quality

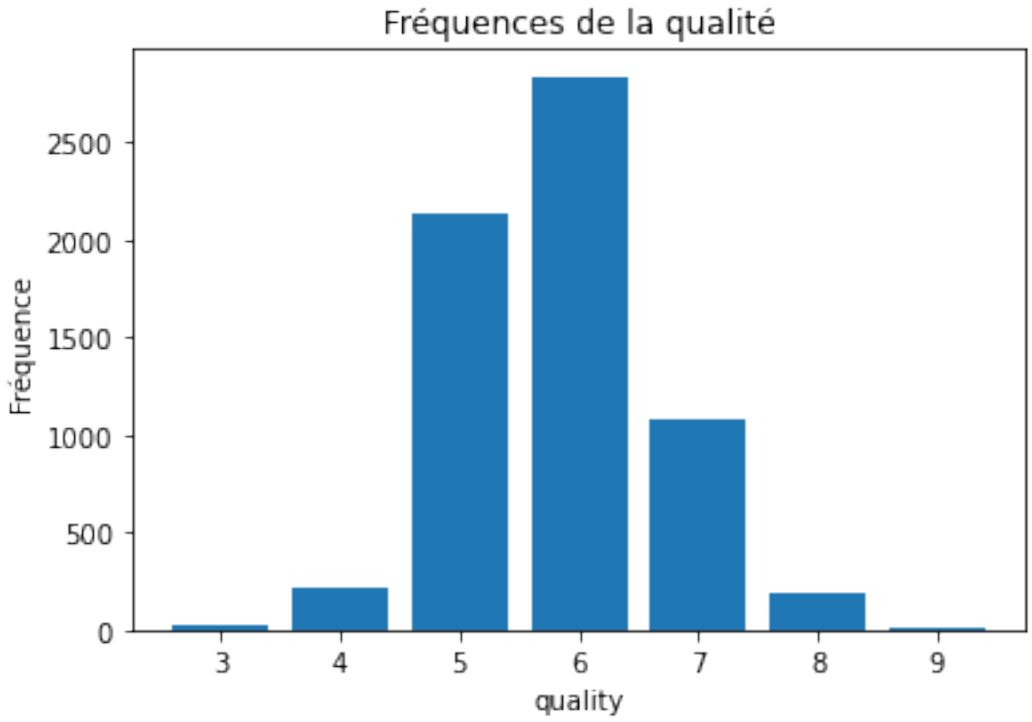
La qualité du vin est la variable importante de nos données, que nous cherchons par la suite à justifier.

[7]:
```
print(df['quality'].describe(include='all'))
```

```
count    6497.000000
mean        5.818378
std         0.873255
min         3.000000
25%         5.000000
50%         6.000000
75%         6.000000
max         9.000000
Name: quality, dtype: float64
```

Nous obtenons une qualité moyenne de 5.82 et un écart-type de 0.87, indiquant que les données sont plutôt homogènes.
La preuve en est que 25% des vins ont une note en dessous de 5 et que 75% ont en dessous de 6.
Enfin, la plus petite note est 3 et la plus grande est 9.

[8]:
```
x = np.sort(df['quality'].unique())

plt.bar(x, df['quality'].value_counts().sort_index(), align='center')
plt.xticks(x)
plt.xlabel('quality')
plt.ylabel('Fréquence')
plt.title('Fréquences de la qualité')
plt.show()
```

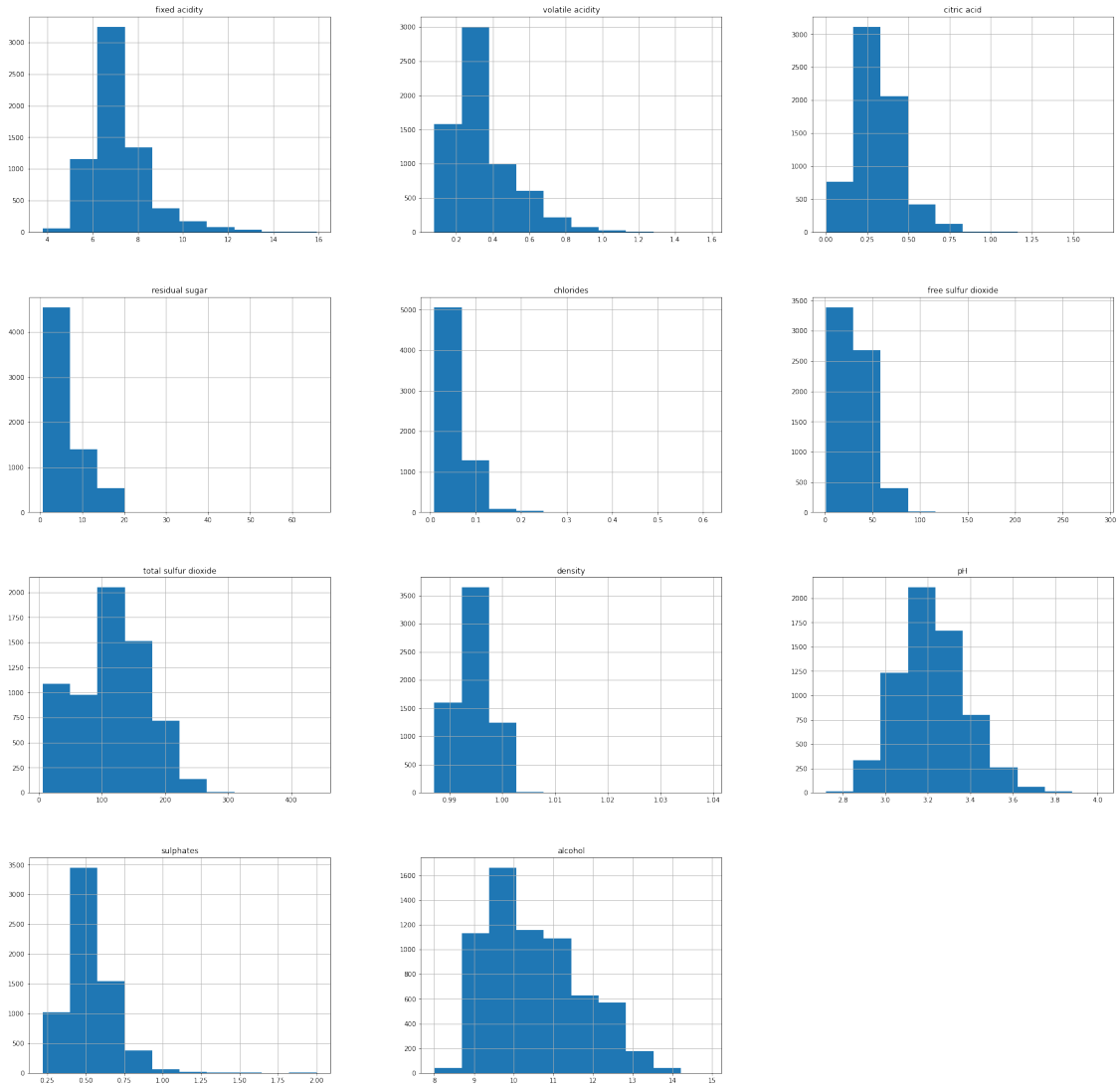


Le graphe ci-dessus confirme ce qui a été déduit, les données sont plutôt homogènes avec une forte concentration sur les valeur 5 et 6.

2.2 Les autres variables

Afin de mieux cerner les données, les autres données du jeu de données sont analysées.

[9]:
```
df.drop('quality', axis=1).hist(figsize=(30,30));
```



[10]:
```
df.drop('quality', axis=1).describe(include='all')
```

[10]:

	fixed acidity	volatile acidity	citric acid	residual sugar	\
count	6497.000000	6497.000000	6497.000000	6497.000000	
mean	7.215307	0.339666	0.318633	5.443235	
std	1.296434	0.164636	0.145318	4.757804	
min	3.800000	0.080000	0.000000	0.600000	
25%	6.400000	0.230000	0.250000	1.800000	
50%	7.000000	0.290000	0.310000	3.000000	
75%	7.700000	0.400000	0.390000	8.100000	
max	15.900000	1.580000	1.660000	65.800000	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	\
count	6497.000000	6497.000000	6497.000000	6497.000000	

```
mean      0.056034            30.525319           115.744574     0.994697
std       0.035034            17.749400            56.521855     0.002999
min       0.009000             1.000000             6.000000     0.987110
25%       0.038000            17.000000            77.000000     0.992340
50%       0.047000            29.000000           118.000000     0.994890
75%       0.065000            41.000000           156.000000     0.996990
max       0.611000           289.000000           440.000000     1.038980

                pH    sulphates      alcohol
count  6497.000000  6497.000000  6497.000000
mean      3.218501     0.531268    10.491801
std       0.160787     0.148806     1.192712
min       2.720000     0.220000     8.000000
25%       3.110000     0.430000     9.500000
50%       3.210000     0.510000    10.300000
75%       3.320000     0.600000    11.300000
max       4.010000     2.000000    14.900000
```

Dans l'ensemble, nous obtenons des répartitions avec un pic au centre, proche de là où se trouve la moyenne. Il semble y avoir des écarts-types plutôt petits et des valeurs maximales loin du troisième quartile.
Cela témoigne peut-être de la concordance entre les variables ou avec la qualité ? L'analyse bivariée, qui suit, va permettre d'approfondir ce point.

3 Analyse bivariée

Cette étape va permettre de voir s'il est possible d'établir une certaine relation entre la qualité et le reste des variables.
Commençons par étudier la matrice des corrélations.

3.1 Matrice des corrélations

Grâce à notre pré-analyse (cf. *1.5. Type des variables*), nous savons qu'il y a des valeurs très disparates. Les variables sont majoritairement comprises entre 0 et 1, comme les `chlorides`. D'autres variables atteignent 440; c'est le cas de `total sulfur dioxide`. Pour éviter qu'une variable gagne en poids car les valeurs qui la compose sont plus élevées, nous allons construire une matrice centrée réduite.

[11]:
```
from sklearn.preprocessing import StandardScaler

df_centered = pd.DataFrame(StandardScaler().fit_transform(df), columns=df.
 ↪columns, index=df.index)
df_centered
```

[11]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0          0.142473          2.188833    -2.192833       -0.744778   0.569958
1          0.451036          3.282235    -2.192833       -0.597640   1.197975
```

```
2          0.451036          2.553300    -1.917553       -0.660699   1.026697
3          3.073817         -0.362438     1.661085       -0.744778   0.541412
4          0.142473          2.188833    -2.192833       -0.744778   0.569958
...             ...               ...          ...             ...        ...
6492      -0.783214         -0.787650    -0.197054       -0.807837  -0.486252
6493      -0.474652         -0.119460     0.284686        0.537425  -0.257883
6494      -0.551792         -0.605417    -0.885253       -0.891916  -0.429160
6495      -1.323198         -0.301694    -0.128234       -0.912936  -0.971538
6496      -0.937495         -0.787650     0.422326       -0.975995  -1.028631

      free sulfur dioxide  total sulfur dioxide   density        pH  \
0               -1.100140             -1.446359  1.034993  1.813090
1               -0.311320             -0.862469  0.701486 -0.115073
2               -0.874763             -1.092486  0.768188  0.258120
3               -0.762074             -0.986324  1.101694 -0.363868
4               -1.100140             -1.446359  1.034993  1.813090
...                   ...                   ...       ...       ...
6492            -0.367664             -0.420128 -1.186161  0.320319
6493             1.491697              0.924588  0.067824 -0.426067
6494            -0.029599             -0.083949 -0.719251 -1.421248
6495            -0.593041             -0.101642 -2.003251  0.755710
6496            -0.480353             -0.313966 -1.763127  0.258120

      sulphates   alcohol   quality
0      0.193097 -0.915464 -0.937230
1      0.999579 -0.580068 -0.937230
2      0.797958 -0.580068 -0.937230
3      0.327510 -0.580068  0.207999
4      0.193097 -0.915464 -0.937230
...         ...       ...       ...
6492  -0.210144  0.593818  0.207999
6493  -0.478971 -0.747766 -0.937230
6494  -0.478971 -0.915464  0.207999
6495  -1.016626  1.935402  1.353228
6496  -1.419867  1.096912  0.207999

[6497 rows x 12 columns]
```

Maintenant, nous pouvons réaliser la matrice des corrélations. Pour une analyse plus facile, les valeurs absolues ont été utilisées pour faciliter l'interprétation, puisque plus la corrélation (en valeur absolue) est proche de 1, plus les variables sont corrélées.

[12]:
```
df_corr = df_centered.corr()
df_corr.abs().style.background_gradient(cmap='YlOrRd')
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	0.219008	0.324436	0.111981	0.298195	0.282735	0.329054	0.458910	0.252700	0.299568	0.095452	0.076743
volatile acidity	0.219008	1.000000	0.377981	0.196011	0.377124	0.352557	0.414476	0.271296	0.261454	0.225984	0.037640	0.265699
citric acid	0.324436	0.377981	1.000000	0.142451	0.038998	0.133126	0.195242	0.096154	0.329808	0.056197	0.010493	0.085532
residual sugar	0.111981	0.196011	0.142451	1.000000	0.128940	0.402871	0.495482	0.552517	0.267320	0.185927	0.359415	0.036980
chlorides	0.298195	0.377124	0.038998	0.128940	1.000000	0.195045	0.279630	0.362615	0.044708	0.395593	0.256916	0.200666
free sulfur dioxide	0.282735	0.352557	0.133126	0.402871	0.195045	1.000000	0.720934	0.025717	0.145854	0.188457	0.179838	0.055463
total sulfur dioxide	0.329054	0.414476	0.195242	0.495482	0.279630	0.720934	1.000000	0.032395	0.238413	0.275727	0.265740	0.041385
density	0.458910	0.271296	0.096154	0.552517	0.362615	0.025717	0.032395	1.000000	0.011686	0.259478	0.686745	0.305858
pH	0.252700	0.261454	0.329808	0.267320	0.044708	0.145854	0.238413	0.011686	1.000000	0.192123	0.121248	0.019506
sulphates	0.299568	0.225984	0.056197	0.185927	0.395593	0.188457	0.275727	0.259478	0.192123	1.000000	0.003029	0.038485
alcohol	0.095452	0.037640	0.010493	0.359415	0.256916	0.179838	0.265740	0.686745	0.121248	0.003029	1.000000	0.444319
quality	0.076743	0.265699	0.085532	0.036980	0.200666	0.055463	0.041385	0.305858	0.019506	0.038485	0.444319	1.000000

[12]: `<pandas.io.formats.style.Styler at 0x7faf62b46d60>`

Cette matrice des corrélations nous apprend que la variable qui est la plus corrélée avec la qualité du vin est: `alcohol` (0.444319). `density` (0.305858) semble aussi davantage corrélée que les autres variables.
Cependant, ces valeurs sont extrèmement faibles: elles n'atteignent même pas 0.5. Il est donc difficile d'annoncer que ces variables sont liées avec de telles valeurs.
Nous pouvons aussi déterminer que la qualité du vin sera légèrement corrélée aux variables `volatile acidity` et `chlorides`. En revanche, elle ne sera pas corrélée avec les variables `fixed acidity`, `citric acid`, `free sulfur dioxide`, `total sulfur dioxide`, `pH` et `sulphates`.
Nous supposons alors que seulement quelques variables affectent la qualité du vin.

Nous obtenons tout de même des corrélations plus fortes entre `density` et `residual sugar` (0.552517) avec une corrélation parmis les plus hautes de la matrice, ainsi qu'entre `density` et `alcohol` (0.686745).
Cette constatation est plutôt logique au vu des corrélations les plus fortes pour `quality`, que nous avons évoqué juste avant, mais elle est expliquable par le fait que `density` dépend notamment du pourcentage d'alcool et de la quantité de sucre.

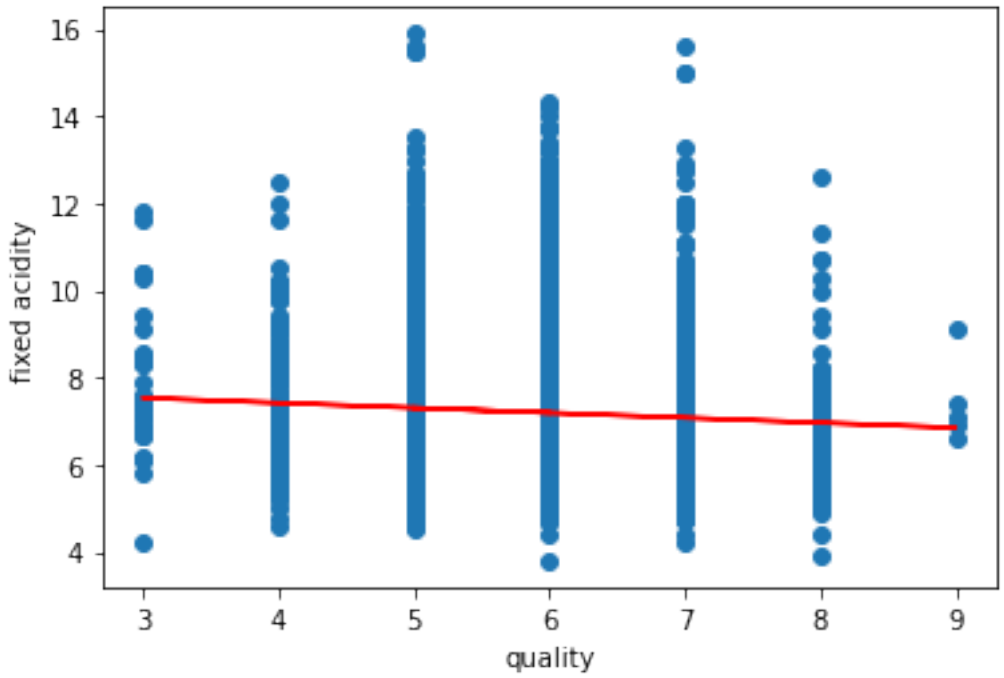
Mais surtout, les variables `total sulfur dioxide` et `free sulfur dioxide` ont la plus forte corrélation de la matrice (0.720934). Encore une fois, cela s'explique par le fait que `free sulfur dioxide` est inclus dans `total sulfur dioxide`.

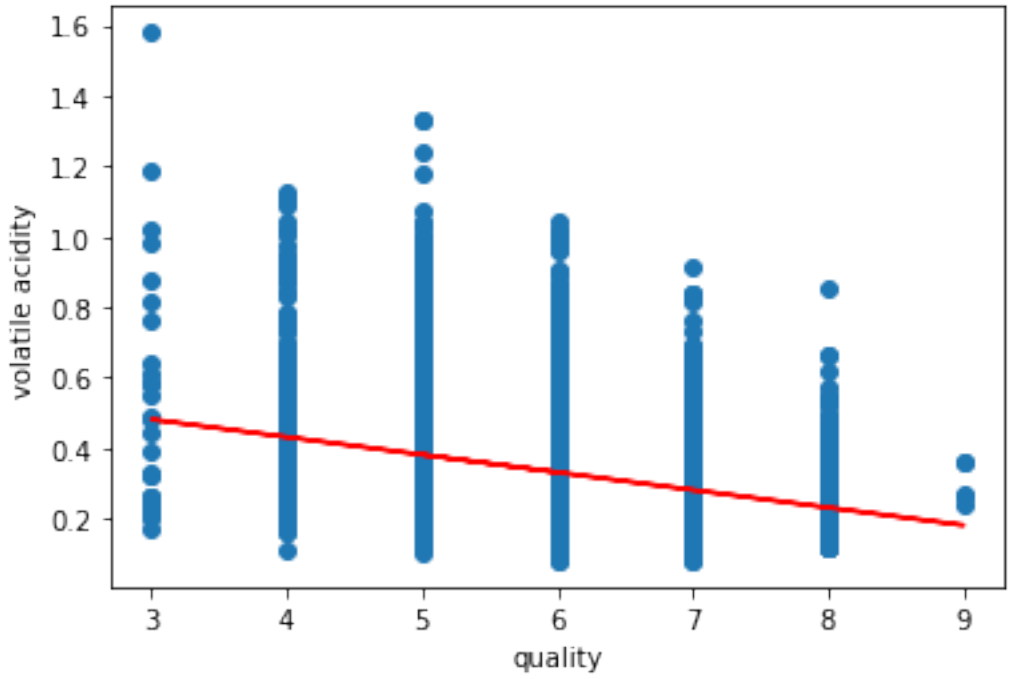
Il peut être surpenant de ne pas retrouver de plus grande corrélation entre `pH`, `fixed acidity`, `volatile acidity` et `citric acid`, puisque l'acidité du vin devrait être influencée par la quantité d'acide qui s'y trouve.
Tout cela n'est, malheureusement, pas encore suffisant pour répondre à la problématique.

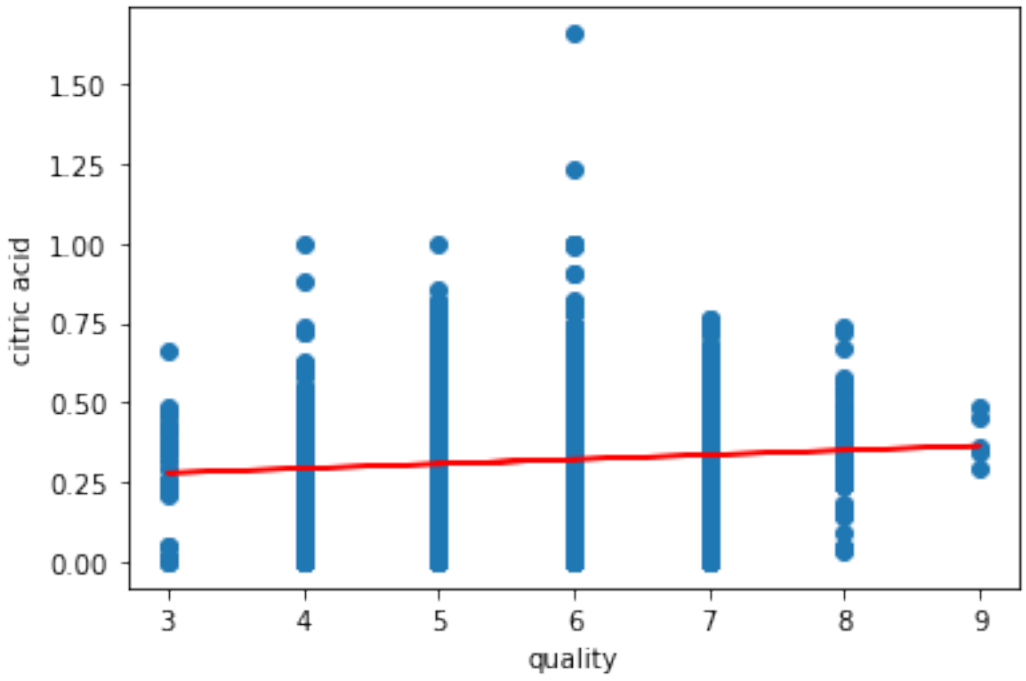
Tentons alors une approche à l'aide de régressions.

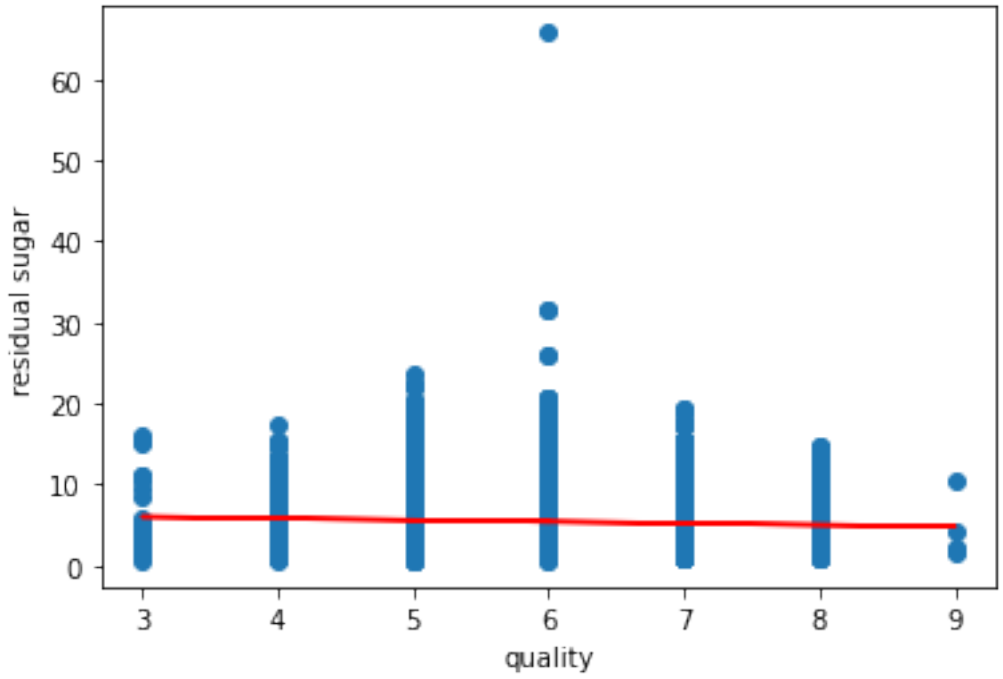
3.2 Régressions

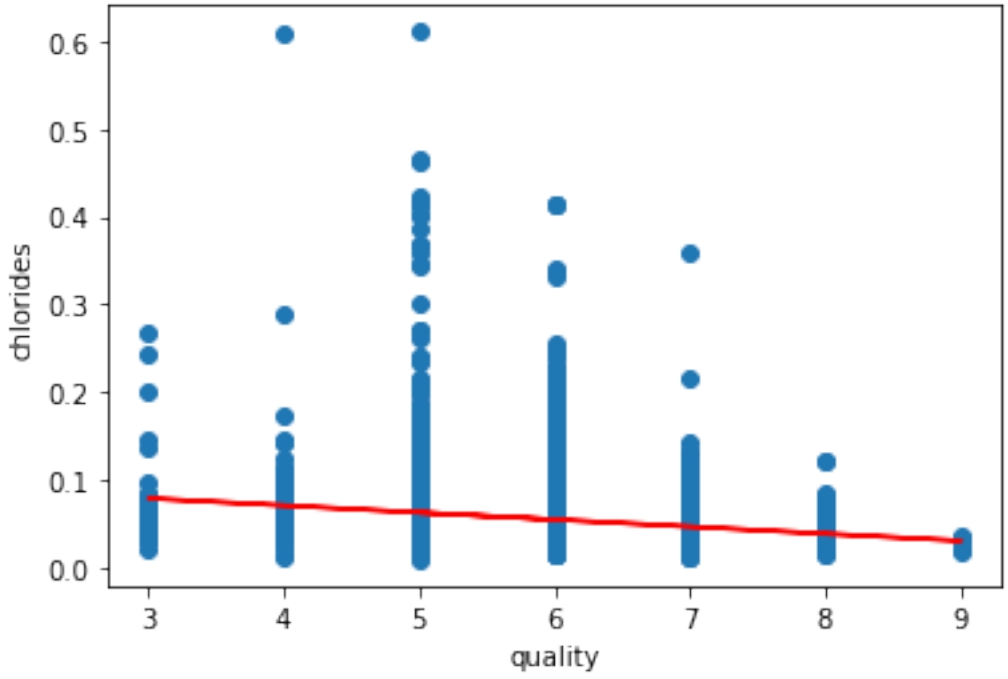
[13]:
```
for c in df.columns[:-1]:
    y = df[c]
    x = df['quality']
    plt.scatter(x, y)
    m, b = np.polyfit(x, y, 1)
    plt.plot(x, m * x + b, color = 'red')
    plt.ylabel(c)
    plt.xlabel('quality')
    plt.show()
```

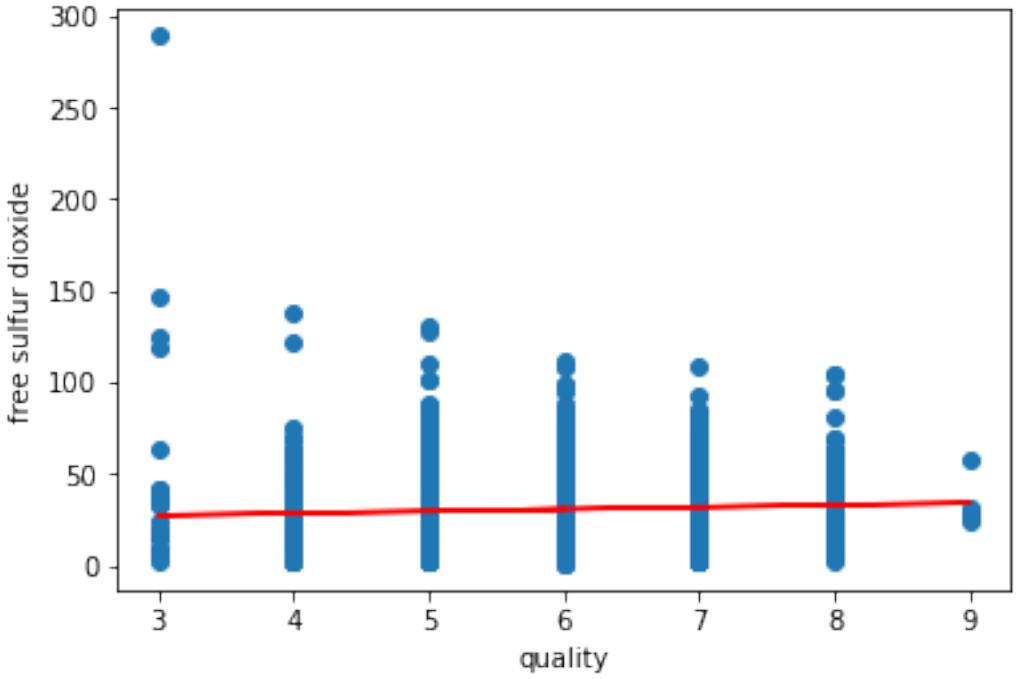


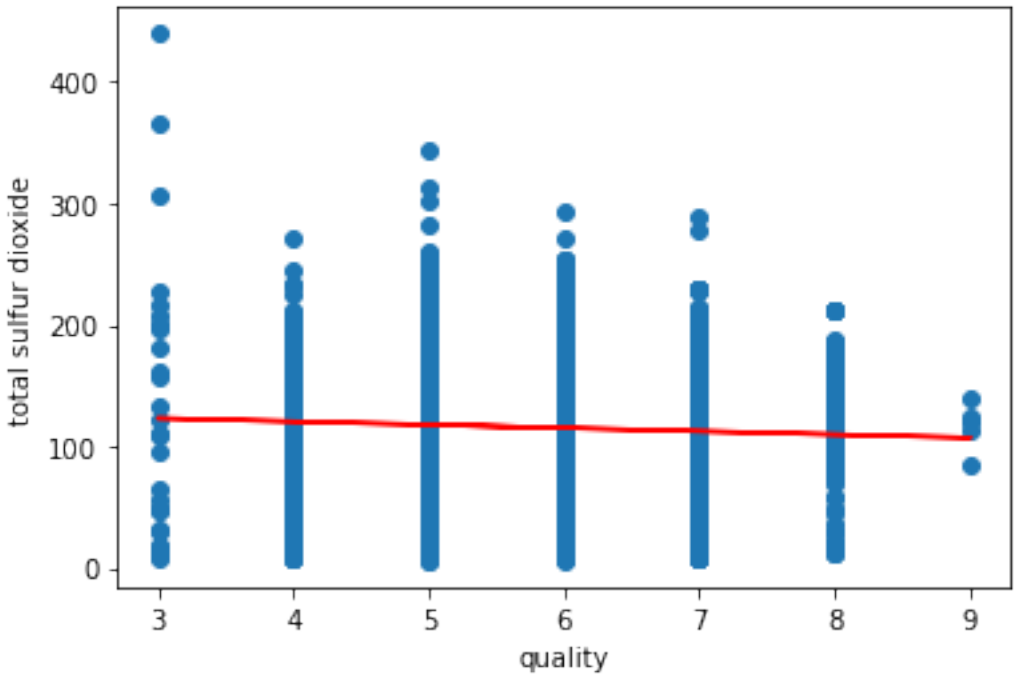

volatile acidity
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
3
4
5
6
7
8
9
quality

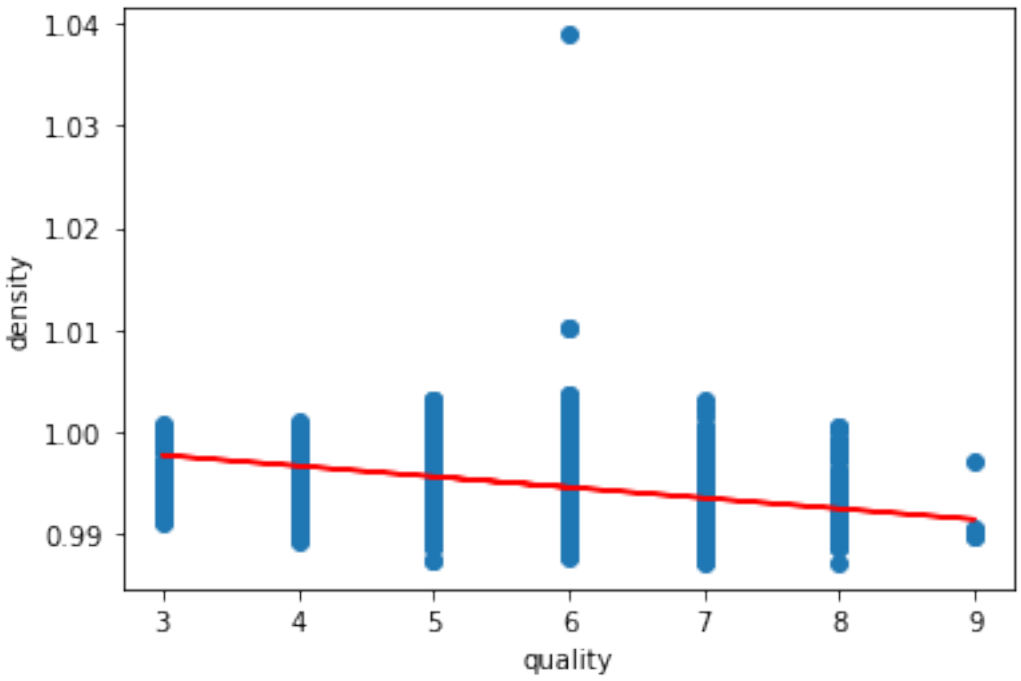

citric acid
1.50
1.25
1.00
0.75
0.50
0.25
0.00
3
4
5
6
7
8
9
quality

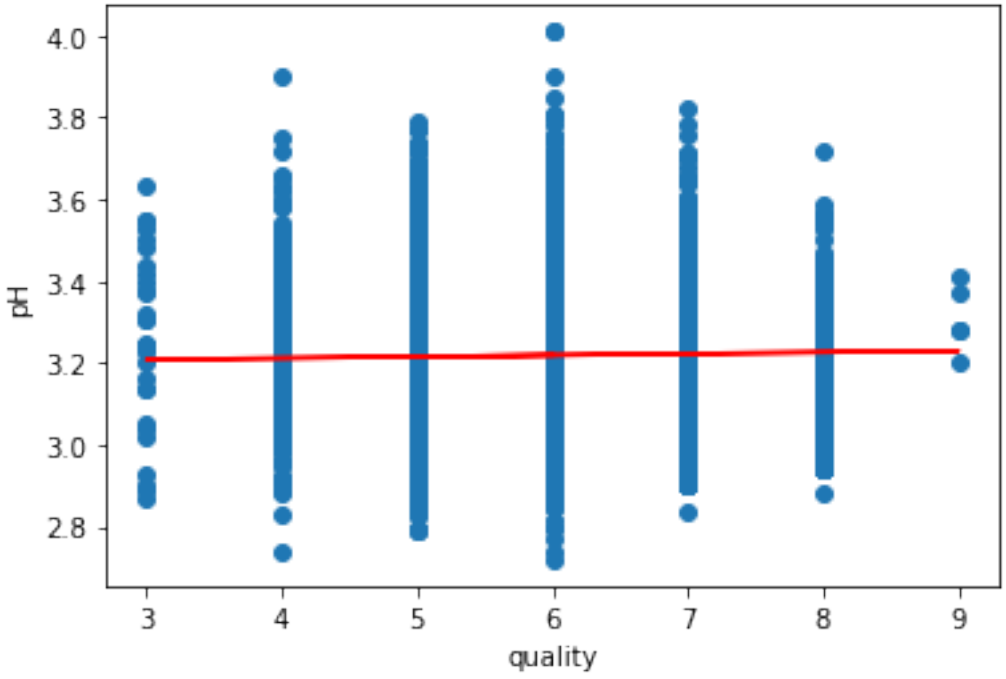

residual sugar
60
50
40
30
20
10
0
3
4
5
6
7
8
9
quality

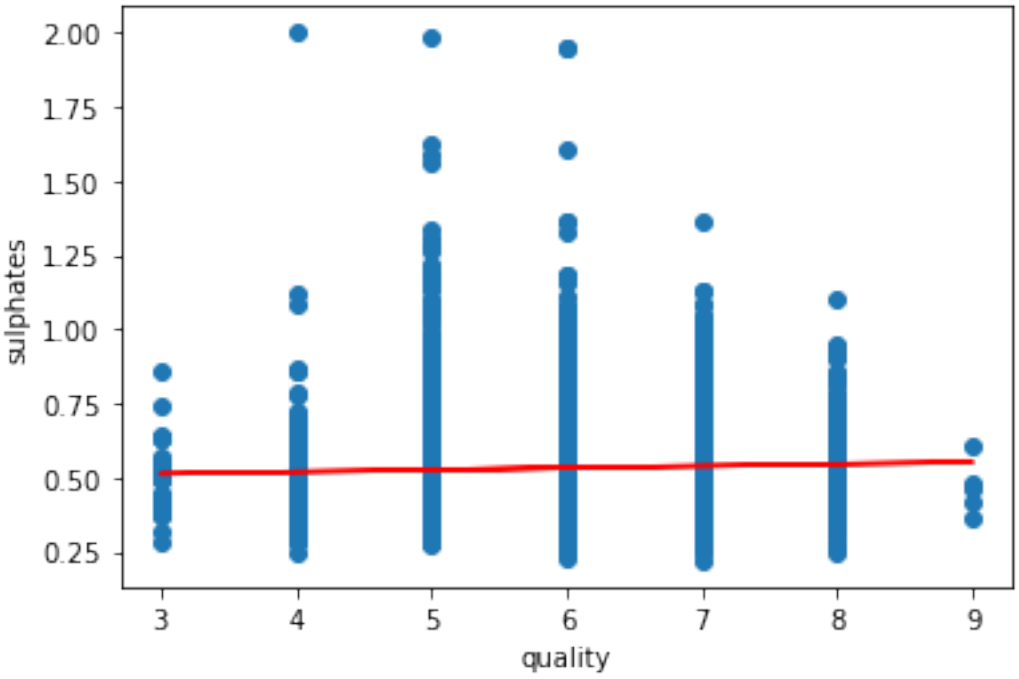

chlorides
0.6
0.5
0.4
0.3
0.2
0.1
0.0
3
4
5
6
7
8
9
quality

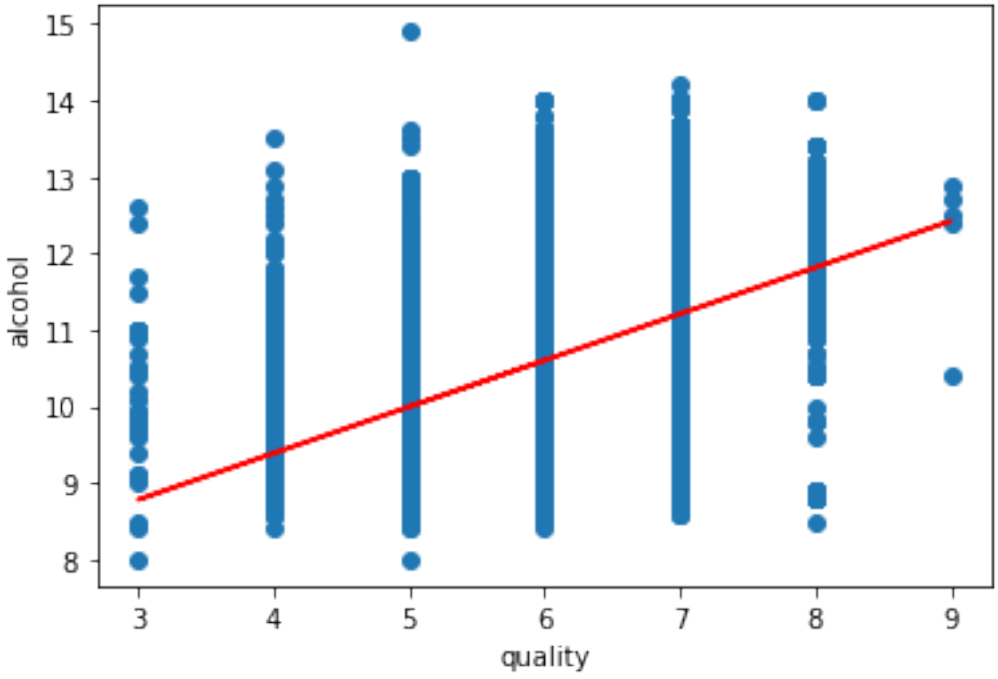

free sulfur dioxide
quality
300
250
200
150
100
50
0
3
4
5
6
7
8
9


total sulfur dioxide
quality
400
300
200
100
0
3
4
5
6
7
8
9


1.04
1.03
1.02
1.01
1.00
0.99
density
3
4
5
6
7
8
9
quality


4.0
3.8
3.6
3.4
3.2
3.0
2.8
pH
3
4
5
6
7
8
9
quality





Avec ces régressions, nous remarquons bien la corrélation entre `quality` et `alcohol`, où plus il y a d'alcool, plus le vin est de bonne qualité.
De même, la bonne visibilité de la corrélation avec `density`, avec l'idée que moins de densité,

semble rendre le vin meilleur.

Ces régressions montrent des relations qui auraient été sous-estimées avec la matrice des corrélations, notamment la variable `volatile acidity`, qui plus elle est élevée, moins le vin est bon. Ce n'est pas vraiment étonnant puisque sa corrélation avec `quality` est proche de celle de `density`.
La variable `chlorides` semble aussi montrer de légers traits de corrélation, où un taux plus élevé entraine un vin de moins bonne qualité.

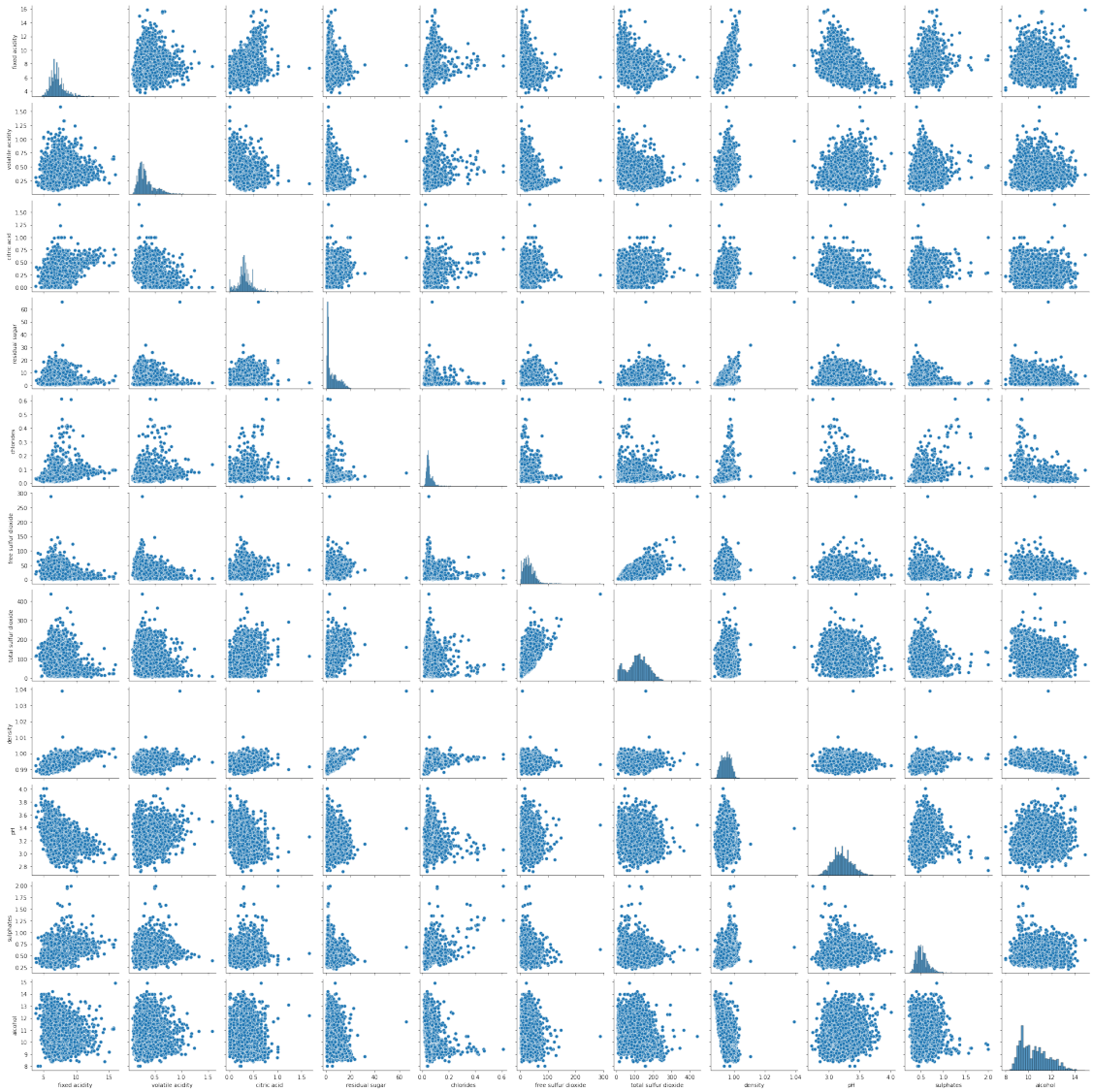
La matrice des corrélations ne fournit pas d'autres données exploitables.

Nous pouvons être critiques sur ces graphes, dans le sens où sans la droite, il ne serait visuellement pas toujours évident de déterminer une qualité de vin par rapport à la valeur d'une autre variable.

Une matrice de graphe est représentée pour donner une idée des relations entre les variables.

[14]:
```
sns.pairplot(df.drop('quality', axis=1))
```

[14]:
```
<seaborn.axisgrid.PairGrid at 0x7faf64209f70>
```



Il n'apparait aucune différence par rapport à la matrice des corrélations. Nous distinguons les variables les plus corrélées, de celles qui ne le sont pas comme, par exemple:
- `density` et `alcohol` dessinent une régression visible
- `alcohol` et `fixed acidity` ne dessinent rien de particulier

4 Analyse en Composantes Principales

Pour essayer d'appuyer ou améliorer les résultats précédents, nous voudrions simplifier nos données et réduire notre nombre de variables en ne conservant que les plus importantes, grâce à l'ACP.

[15]:
```
from sklearn.decomposition import PCA
```

```
pca = PCA()
pca_scores = pca.fit_transform(df_centered)
eigval = (pca.n_components_ - 1) / pca.n_components_ * pca.explained_variance_

index = ['Axe' + str(i) for i in range(1, df_corr.shape[0] + 1)]
df_pca = pd.DataFrame([eigval, (pca.explained_variance_ratio_ * 100), (np.
 ↪cumsum(pca.explained_variance_ratio_ * 100))], index=['Eigenvalue', 'Explained␣
 ↪variance percent', 'Explained variance percent cumulative'], columns=index).
 ↪transpose()
df_pca
```

[15]:

	Eigenvalue	Explained variance percent	\
Axe1	2.788514	25.346226	
Axe2	2.429407	22.082117	
Axe3	1.504946	13.679223	
Axe4	0.979724	8.905210	
Axe5	0.770577	7.004171	
Axe6	0.605459	5.503326	
Axe7	0.516920	4.698554	
Axe8	0.473057	4.299857	
Axe9	0.420239	3.819769	
Axe10	0.274137	2.491774	
Axe11	0.208654	1.896563	
Axe12	0.030058	0.273210	

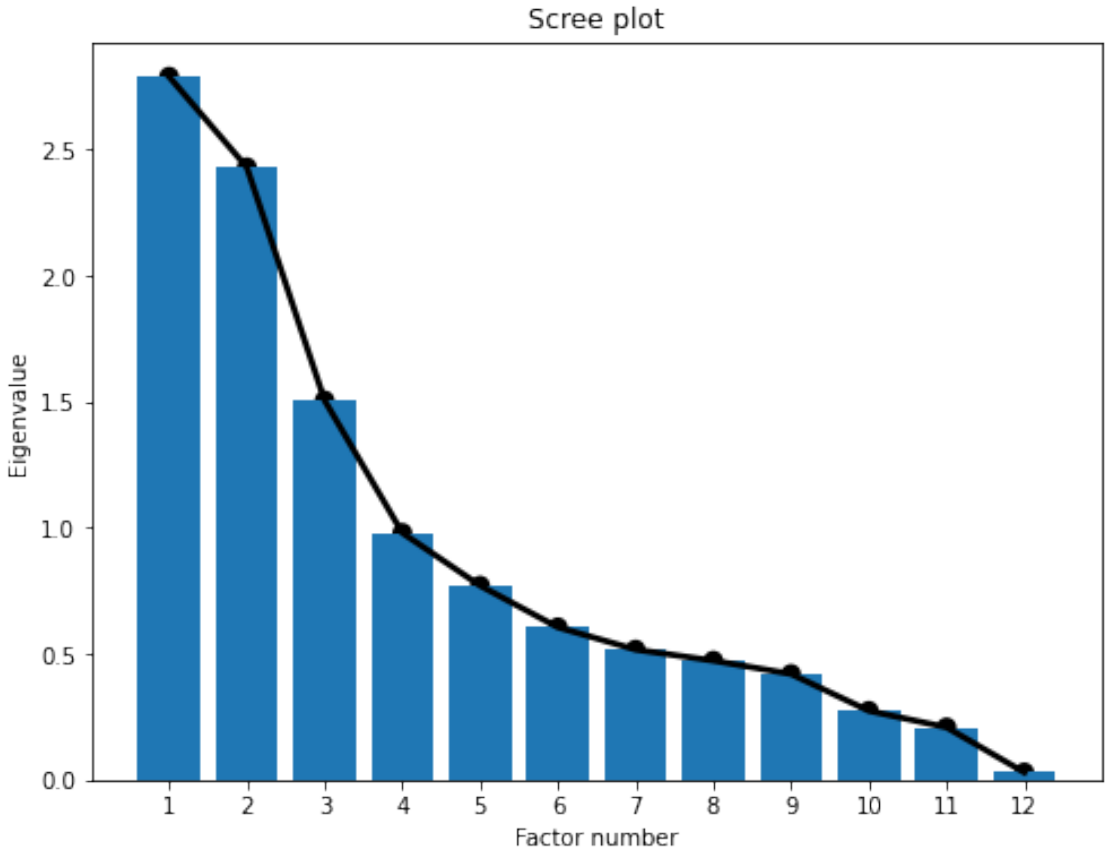
	Explained variance percent cumulative
Axe1	25.346226
Axe2	47.428343
Axe3	61.107566
Axe4	70.012777
Axe5	77.016947
Axe6	82.520274
Axe7	87.218827
Axe8	91.518684
Axe9	95.338453
Axe10	97.830228
Axe11	99.726790
Axe12	100.000000

4.1 Choix des composantes

Nous allons représenter les valeurs propres sur un graphe pour utiliser la métode des éboulis afin de choisir le nombre de composantes à conserver.

[16]:
```
factors = [x for x in range(df_pca.shape[0])]
egv = df_pca['Eigenvalue'].sort_values(ascending=False)
```

```
(fig, ax) = plt.subplots(figsize=(8, 6))
sns.pointplot(x=factors, y=egv, color = 'black')
ax.set_xticks(factors)
ax.set_xticklabels(range(1, df_pca.shape[0] + 1))
ax.set_xlabel('Factor number')
ax.set_ylabel('Eigenvalue')

plt.title("Scree plot")
plt.bar(factors, egv)
plt.show()
```

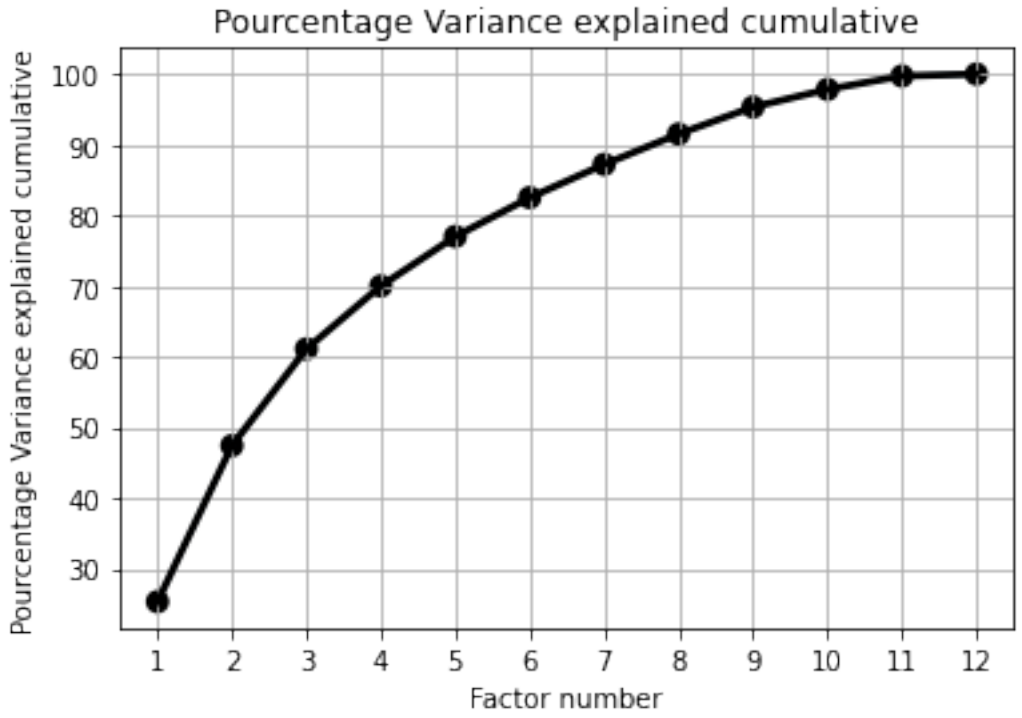


Nous remarquons 3 coudes en k = 4, k = 7 et k = 10. Retenir 4 composantes semble être la meilleure solution, mais il est peut-être possible de n'en retenir que 3.
Le pourcentage de variance cumulée va permettre de faire un choix.

[17]:
```
evp = df_pca['Explained variance percent cumulative']
factors2 = [x for x in range(df_pca.shape[0])]

plt.plot(evp)
ax = plt.axes()
```

```
sns.pointplot(x=factors2, y=evp, color = 'black')
ax.set_xticks(factors2)
ax.set_xticklabels(range(1, df_pca.shape[0] + 1))
ax.set_xlabel('Factor number')
ax.set_ylabel('Pourcentage Variance explained cumulative')
plt.grid(True)
plt.title("Pourcentage Variance explained cumulative")
plt.show()
```

```
<ipython-input-17-2fcae04abbb2>:5: MatplotlibDeprecationWarning: Adding an axes
using the same arguments as a previous axes currently reuses the earlier
instance.  In a future version, a new instance will always be created and
returned.  Meanwhile, this warning can be suppressed, and the future behavior
ensured, by passing a unique label to each axes instance.
  ax = plt.axes()
```



Nous constatons que les trois premières composantes accaparent 61% de l'information, notamment les deux premières qui en accaparent 50%. Les autres composantes sont beaucoup moins importantes.
Une quatrième composante pourrait permettre de couvrir 71% de l'information, mais sa valeur propre est inférieure à 1 et son pourcentage de variance expliquée est assez faible par rapport aux trois premières composantes.
De ce fait, nous ne la retenons pas et nous choisissont donc k = 3 composants.

4.2 Cercles des corrélations

Une matrice des corrélations peut alors être réalisée entre axes et facteurs, nous permettant de tracer le tout sous forme de cercles des corrélations.

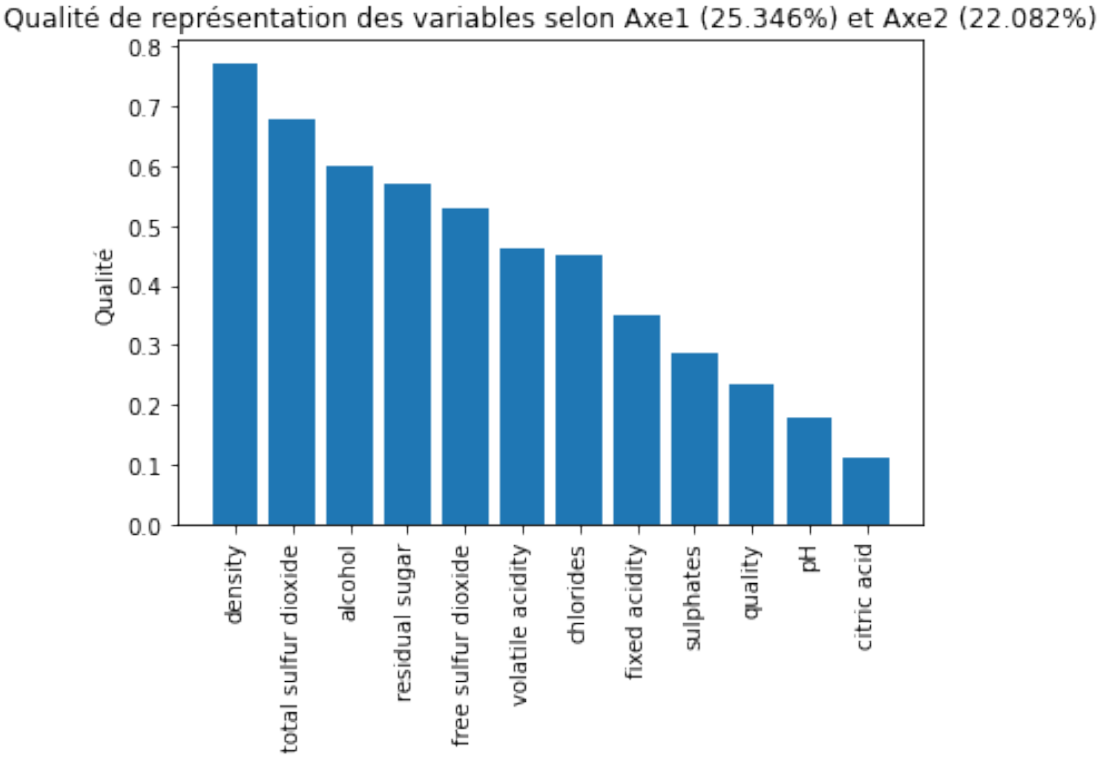
[18]:
```
biplot_index = [v + " (" + str(round(df_pca['Explained variance percent'][i],␣
 ↪3)) + "%)" for i, v in enumerate(df_pca.index)]
sqrt_eigval = np.sqrt(eigval)

#corrélation des variables avec les axes
corvar = np.zeros((df.shape[1],df.shape[1]))
for k in range(df.shape[1]):
    corvar[:,k] = pca.components_[k,:] * sqrt_eigval[k]

cos2var = corvar**2
df_cos2var = pd.DataFrame(cos2var, index=df.columns, columns=df_pca.index)

def show_quality_representation(x, y):
    tmp = pd.DataFrame(df_cos2var[df_cos2var.columns[x]] + df_cos2var[df_cos2var.
 ↪columns[y]], index=df.columns)
    Y = tmp.sort_values(tmp.columns[0], ascending=False)
    plt.bar(range(len(Y.index)), Y[tmp.columns[0]], align='center')
    ax = plt.axes()
    ax.set_xticks(range(len(Y.index)))
    ax.set_xticklabels(Y.index, rotation='vertical')
    plt.ylabel('Qualité')
    plt.title('Qualité de représentation des variables selon ' + biplot_index[x]␣
 ↪+ ' et ' + biplot_index[y])
    plt.show()

#cercle des corrélations
def show_biplot(x, y):
    fig, axes = plt.subplots(figsize=(8,8))
    axes.set_xlim(-1.2,1.2)
    axes.set_ylim(-1.2,1.2)

    for j in range(df.shape[1]):
        plt.arrow(0, 0, corvar[j,x], corvar[j,y],color = 'r',alpha = 1,␣
 ↪head_width=0.05, head_length=0.05)
        plt.annotate(df.columns[j],(corvar[j,x],corvar[j,y]))

    plt.plot([-1.2,1.2],[0,0],color='silver',linestyle='--',linewidth=1)
    plt.plot([0,0],[-1.2,1.2],color='silver',linestyle='--',linewidth=1)
    cercle = plt.Circle((0,0),1,color='blue',fill=False)
    axes.add_artist(cercle)
    plt.xlabel(biplot_index[x])
    plt.ylabel(biplot_index[y])
    plt.show()
```

4.2.1 Cercle des corrélations Axe1 et Axe2

Il s'agit d'abord de montrer la qualité de la représentations des variables (représentée par le $\cos^2$ de la corrélation variables/axes) sur les axes 1 et 2, afin de déterminer les variables que l'on peut se permettre d'interpréter.

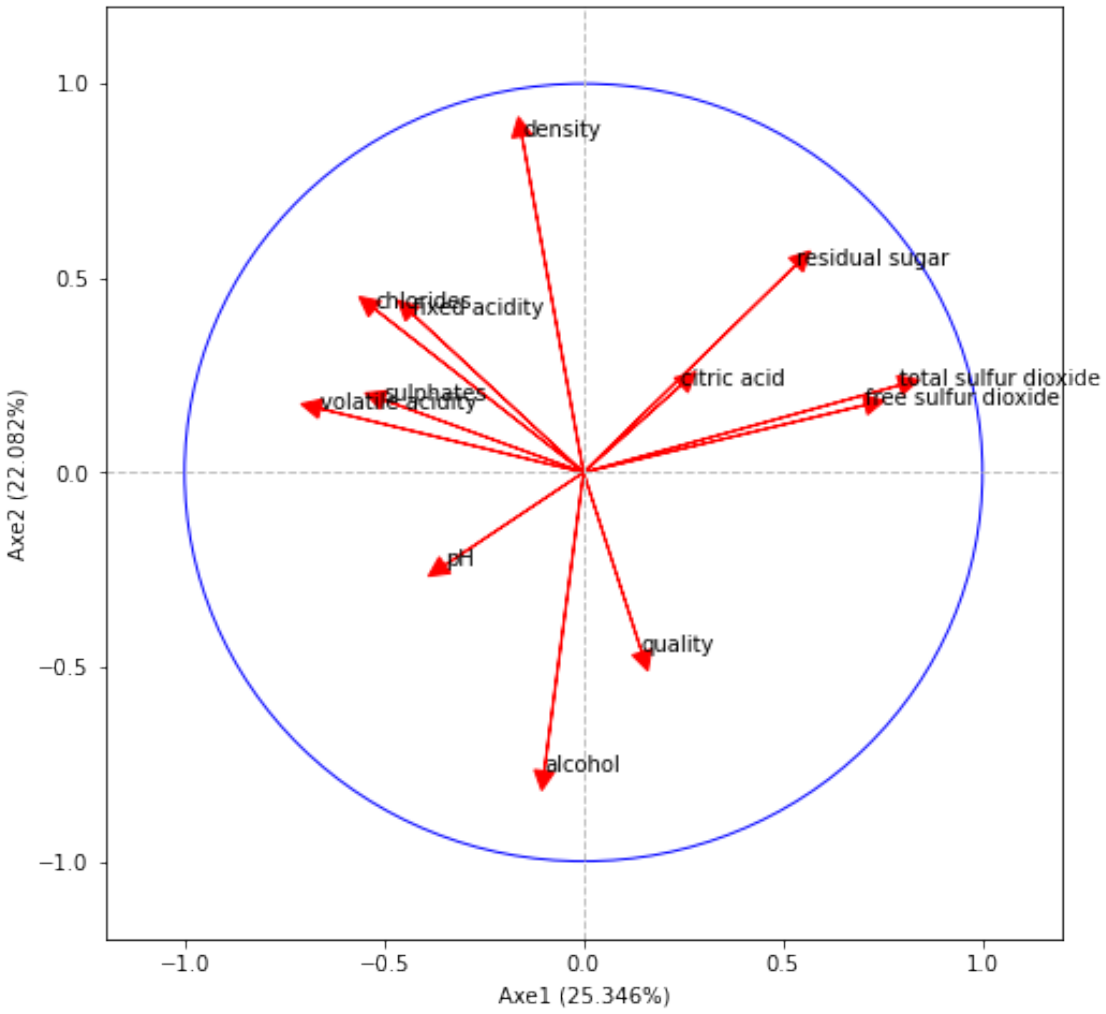
[19]:
```
show_quality_representation(0, 1)
```

```
<ipython-input-18-cef12f625d7e>:16: MatplotlibDeprecationWarning: Adding an axes
using the same arguments as a previous axes currently reuses the earlier
instance.  In a future version, a new instance will always be created and
returned.  Meanwhile, this warning can be suppressed, and the future behavior
ensured, by passing a unique label to each axes instance.
  ax = plt.axes()
```



Nous pouvons voir que l'Axe 1 et l'Axe 2 représentent assez bien la majorité des variables. Nous allons tout de même prendre en compte toutes les variables jusqu'à `quality`, car c'est cette variable qui nous intéresse. Nous pouvons alors interpréter: `density`, `total sulfur dioxide`, `alcohol`, `residual sugar`, `free sulfur dioxide`, `volatile acidity`, `chlorides`, `fixed acidity`, `sulphates` et `quality`.
Traçons maintenant le cercle des corélations associé.

[20]:
```
show_biplot(0, 1)
```



Sur ce cercle, l'effet de taille de l'Axe2 est visible. Il y apparait que les vins qui contiennent des valeurs élevées pour les variables `volatile acidity`, `chlorides`, `fixed acidity` et `sulphates`, sont aussi des vins qui contienent une valeur élevée `residual sugar`, ainsi que tous types de `sulfur dioxide`.
La relation globale entre ces variables est déterminée par `density`.

Nous remarquons tout de suite que `density` est corrélée négativement à `quality`, qui elle-même est corrélée positivement avec `alcohol`.

Enfin, `quality` est indépendante de `total sulfur dioxide` et `free sulfur dioxide`.
De même, `alcohol` est indépendante de `volatile acidity`.

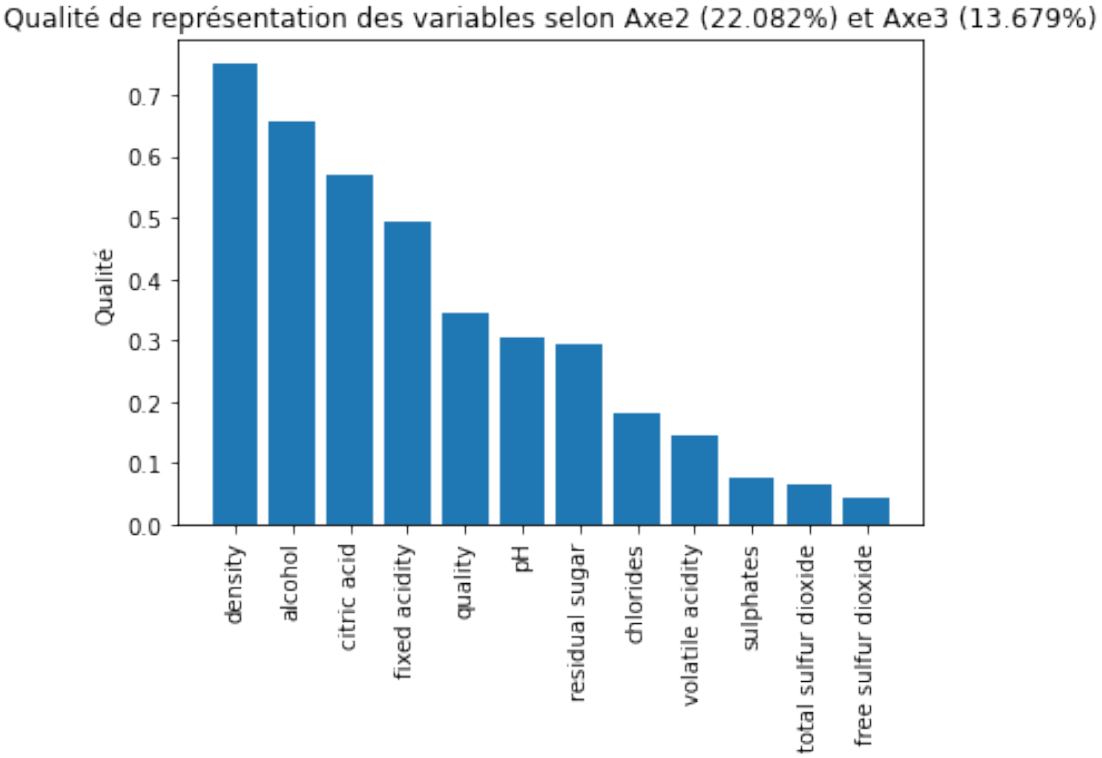
Jusque là, les résultats coïcident avec les graphes de l'analyse bivariée. (cf. *3. Analyse bivariée*).

4.2.2 Cercle des corrélations Axe2 et Axe3

Le processus est réitéré pour s'intéresser à la représentation de la qualité sur l'axe 2 et 3.

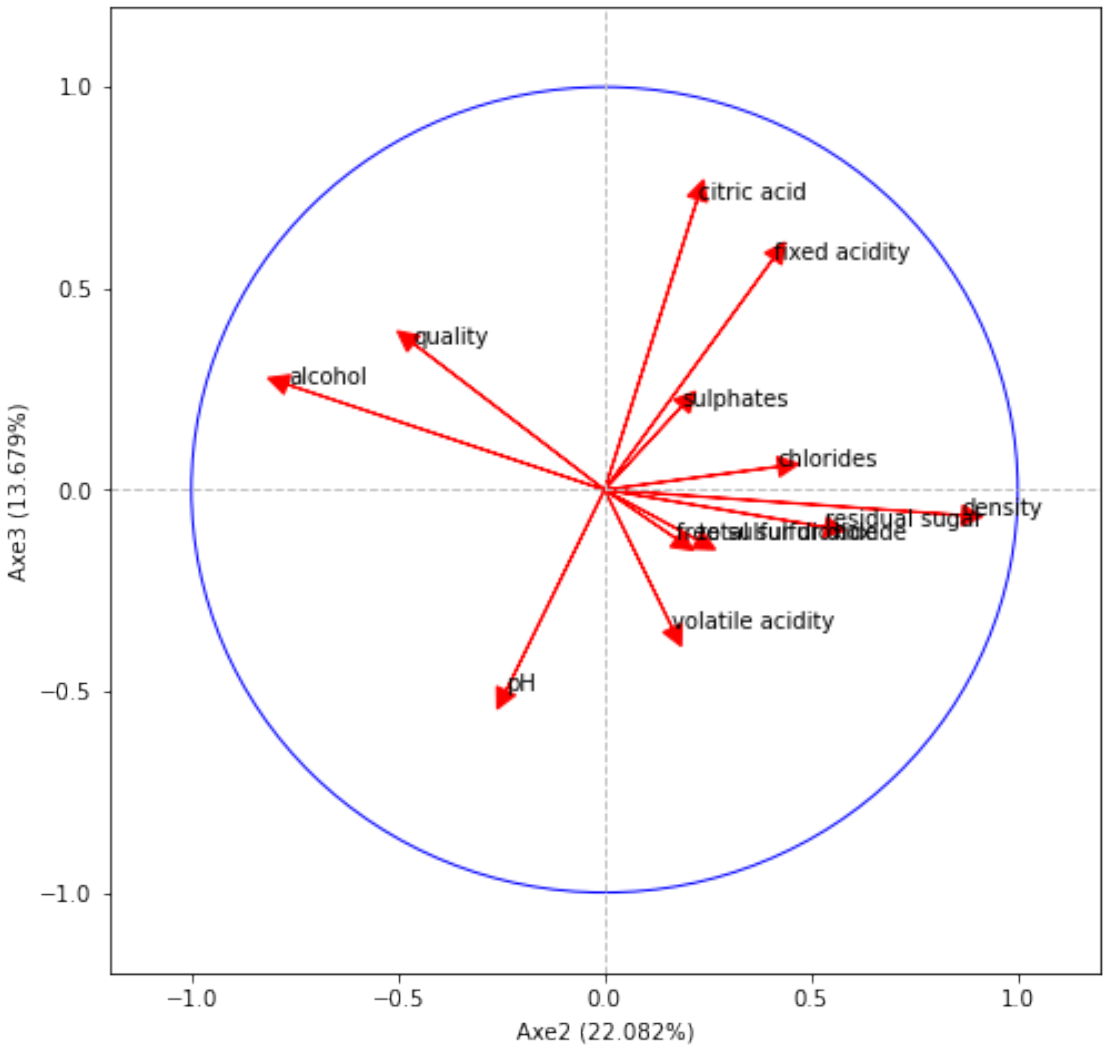
[21]:
```
show_quality_representation(1, 2)
```

```
<ipython-input-18-cef12f625d7e>:16: MatplotlibDeprecationWarning: Adding an axes
using the same arguments as a previous axes currently reuses the earlier
instance.  In a future version, a new instance will always be created and
returned.  Meanwhile, this warning can be suppressed, and the future behavior
ensured, by passing a unique label to each axes instance.
  ax = plt.axes()
```



Ce graphe va permettre de déterminer les variables que nous pourrons interpréter: `density`, `alcohol`, `citric acid`, `fixed acidity`, `quality`, `pH`, et `residual sugar`.
Passons enfin au cercle des corrélations.

[22]:
```
show_biplot(1, 2)
```



Sur ce cercle, nous ne percevons pas d'effet de taille entre l'Axe2 et l'Axe3.

Cependant, `quality` est toujours corrélée positivement avec `alcohol`.

Cette fois, `alcohol` est corrélée négativement avec `density`, mais aussi avec `residual sugar`. Cela s'explique par le fait que la densité est la densité d'eau dans le vin, et qu'elle dépend de l'alcool, ainsi que du sucre. Moins il y a d'alcool, plus la densité de l'eau augmente.
De plus, s'il y a moins d'alcool, il y a aussi plus de sucre. La corrélation négative est justifiée.

Enfin, on remarque que `alcohol` et `quality` sont indépendants de l'acidité du vin, définis ici par: `pH`, `citric acid` et `fixed acidity`. Cependant, ce n'est pas tout à fait vrai, l'analyse bivariée (cf. *3. Analyse bivariée*) et le cercle des corrélation Axe1 et Axe2 (cf. *4.2.1. Cercle des corrélations Axe1 et Axe2*), montrent qu'il existe tout de même une certaine corrélation entre `quality` et `volatile acidity`.

Encore une fois, nous obtenons des résultats qui conïcident avec les graphes de l'analyse bivariée.

5 Régressions (méthodes supervisées) et classifications

[23]:
```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

models = {
    "LinearRegression": LinearRegression(),
    "LogisticRegression": LogisticRegression(max_iter=500),
    "DecisionTreeClassifier": DecisionTreeClassifier(max_depth=5),
    "RandomForestClassifier": RandomForestClassifier(n_estimators=200,␣
 ↪max_depth=5),
    "KNeighborsClassifier": KNeighborsClassifier(n_neighbors = 3)
}

ycol = 'quality'
Y = df.get(ycol)
X = df.get([col for col in df.columns if col != ycol])
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)

for i in models:
    if(i == "LogisticRegression"):
        X_train = StandardScaler().fit_transform(X_train)

    model = models[i].fit(X_train, y_train)
    print("-------------- ", i, " ---------------")
    print("Train score : {}".format(model.score(X_train, y_train)))
    print("Test score : {}".format(model.score(X_test, y_test)))
    print("")
```

```
--------------  LinearRegression  ---------------
Train score : 0.2992523560486904
Test score : 0.25976731297901734

--------------  LogisticRegression  ---------------
Train score : 0.552241677891091
Test score : 0.03230769230769231

--------------  DecisionTreeClassifier  ---------------
Train score : 0.5753319222628439
```

```
Test score : 0.4576923076923077

--------------  RandomForestClassifier  ---------------
Train score : 0.5918799307292669
Test score : 0.4592307692307692

--------------  KNeighborsClassifier  ---------------
Train score : 0.7814123532807389
Test score : 0.3007692307692308
```

Le meilleur score qu'il est possible d'obtenir avec 20% des données pour l'entrainement, est 0.4592307692307692 avec un RandomForestClassifier, ce qui est faible.

6 Conclusion

D'après l'analyse univariée et bivariée de nos variables, nous avons estimé que la qualité d'un vin était déterminée par: - la variable `density`, par une corrélation négative; - la variable `alcohol`, par une corrélation positive; - la variable `volatile acidity`, par une corrélation négative.

L'Analyse en Composantes Principales a permis de mettre en lumière les variables vraiment importantes dans la qualité du vin. Il apparait que `density` et `alcohol` sont les variables qui ont un réel effet sur la qualité du vin.

Nous pouvons alors conclure qu'un vin de qualité, est un vin qui: - contient une faible densité d'eau, exprimée en fonction de l'alcool et du sucre; - un pourcentage d'alcool élevé.

On peut aussi conclure que la quantité de SO2, libre ou liée (`total sulfur dioxide` et `free sulfur dioxide`), ainsi que l'acidité (`pH`, `citric acid` et `fixed acidity`), n'ont aucun effet dans la qualité du vin.
`volatile acidity` ne semble pas être totalement détachée de `quality`, ce qui est assez étonnant puisque `fixed acidity` (l'acidité du vin), est indépendante de la qualité.
Cependant, l'analyse PCA a montré que ce n'était pas une variable importante dans la déterminiation de la qualité du vin.

Pour finir, nous pouvons nous demander si le jeu de données était suffisamment bien constitué. En effet, les faibles corrélations, les graphes nuages de points peu explicites, les trois axes principaux de l'ACP qui représente seulement 61% de l'information (4 axes pour 71%, 6 axes pour 82%, 8 axes pour 91%) et enfin, le meilleur score de prédiction (classifications et régressions confondues) ne dépassant pas 50%, laissent à penser que les données étaient difficilement exploitables.
Malgré de longues recherches, nous n'avons pas trouvé de jeu de données à la fois facilement exploitable et suffisamment documenté.