project_notebook

March 29, 2024

1 Super Wiwino project

Hello and welcom on my little analysis of the Wiwino database !

In this project I will first present the questions we have been asked by our beloved coaches, and then allow you to play a little bit with a tool I created to check the flavor profile by country.

[ ]:
```
import sqlite3
import matplotlib.pyplot as plt
import numpy as np
import ipywidgets as widgets
import pandas as pd

conn = sqlite3.connect("db/vivino.db")
cur = conn.cursor()
```

1.1 Highlighted wines

In order to increase the sales, the sales department asked us to find ten wines to highlight. Highlight are important to give the clients an idea of how wide our wine database is, and it is definitely very wide !

We went a little bit creative and here is our selection.

[ ]:
```
highlighted = pd.DataFrame(columns = ['char', 'name'])
```

[ ]:
```
query_fizz = """
            SELECT wines.name,
            vintages.year,
            wines.fizziness,
            vintages.price_euros

                FROM wines

            INNER JOIN vintages
                ON wines.id = vintages.wine_id

            GROUP BY wines.id

            ORDER BY fizziness DESC

            LIMIT 1
            """
fizziest = pd.read_sql_query(query_fizz, conn)
fizziest['char'] = 'Fizziest'
highlighted = pd.concat([highlighted, fizziest[['char', 'name']]])
```

[ ]:
```
query_sweet = """
            SELECT wines.name,
            vintages.year,
            wines.sweetness,
            vintages.price_euros

                FROM wines

            INNER JOIN vintages
                ON wines.id = vintages.wine_id

            GROUP BY wines.id

            ORDER BY sweetness DESC

            LIMIT 1
            """
sweetest = pd.read_sql_query(query_sweet, conn)
sweetest['char'] = 'Sweetest'
highlighted = pd.concat([highlighted, sweetest[['char', 'name']]])
```

[ ]:
```
query_intense = """
            SELECT wines.name,
            vintages.year,
            wines.intensity,
            vintages.price_euros

                FROM wines

            INNER JOIN vintages
                ON wines.id = vintages.wine_id

            GROUP BY wines.id

            ORDER BY intensity DESC

            LIMIT 1
            """
intensest = pd.read_sql_query(query_intense, conn)
```

```
intensest['char'] = 'Most intense'
highlighted = pd.concat([highlighted, intensest[['char', 'name']]])
```

```
[ ]: query_complex = """
                     SELECT wines.name,
                         vintages.year,
                         SUM(keywords_wine.count) AS complexity,
                         vintages.price_euros

                         FROM wines

                     INNER JOIN vintages
                         ON wines.id = vintages.wine_id

                     INNER JOIN keywords_wine
                         ON wines.id = keywords_wine.wine_id

                     GROUP BY wines.id

                     ORDER BY complexity DESC

                     LIMIT 1
                     """
     complexest = pd.read_sql_query(query_complex, conn)
     complexest['char'] = 'Most complex'
     highlighted = pd.concat([highlighted, complexest[['char', 'name']]])
```

```
[ ]: query_yeasty = """
                     SELECT wines.name,
                         vintages.year,
                         SUM(keywords_wine.count) AS yeastiness,
                         vintages.price_euros

                         FROM wines

                     INNER JOIN vintages
                         ON wines.id = vintages.wine_id

                     INNER JOIN keywords_wine
                         ON wines.id = keywords_wine.wine_id

                     INNER JOIN keywords
                         ON keywords_wine.keyword_id = keywords.id

                     WHERE keywords.name = "yeast"

                     GROUP BY wines.id
```

```
                ORDER BY yeastiness DESC

                LIMIT 1
                """
yeastiest = pd.read_sql_query(query_yeasty, conn)
yeastiest['char'] = 'Yeastiest'
highlighted = pd.concat([highlighted, yeastiest[['char', 'name']]])
```

[ ]:
```
query_tropical = """
                SELECT wines.name,
                    vintages.year,
                    SUM(keywords_wine.count) AS tropicalness,
                    vintages.price_euros

                    FROM wines

                INNER JOIN vintages
                    ON wines.id = vintages.wine_id

                INNER JOIN keywords_wine
                    ON wines.id = keywords_wine.wine_id

                WHERE keywords_wine.group_name = "tropical_fruit"

                GROUP BY wines.id

                ORDER BY tropicalness DESC

                LIMIT 1
                """
tropicalest = pd.read_sql_query(query_tropical, conn)
tropicalest['char'] = 'Most tropical'
highlighted = pd.concat([highlighted, tropicalest[['char', 'name']]])
```

[ ]:
```
query_fruity = """
                SELECT wines.name,
                    vintages.year,
                    SUM(keywords_wine.count) AS fruitiness,
                    vintages.price_euros

                    FROM wines

                INNER JOIN vintages
                    ON wines.id = vintages.wine_id

                INNER JOIN keywords_wine
```

```
                    ON wines.id = keywords_wine.wine_id

                WHERE keywords_wine.group_name IN ("tropical_fruit",␣
↪"citrus_fruit","tree_fruit", "black_fruit", "red_fruit", "dried_fruit")

                GROUP BY wines.id

                ORDER BY fruitiness DESC

                LIMIT 1
                """
fruitiest = pd.read_sql_query(query_fruity, conn)
fruitiest['char'] = 'Fruitiest'
highlighted = pd.concat([highlighted, fruitiest[['char', 'name']]])
```

[ ]:
```
query_old = """
                SELECT wines.name,
                    vintages.year,
                    vintages.price_euros

                    FROM wines

                INNER JOIN vintages
                    ON wines.id = vintages.wine_id

                INNER JOIN keywords_wine
                    ON wines.id = keywords_wine.wine_id

                GROUP BY wines.id

                ORDER BY vintages.year

                LIMIT 1
                """
oldest = pd.read_sql_query(query_old, conn)
oldest['char'] = 'Oldest'
oldest
highlighted = pd.concat([highlighted, oldest[['char', 'name']]])
```

[ ]:
```
query_big = """
                SELECT wines.name,
                    vintages.year,
                    vintages.price_euros,
                    vintages.bottle_volume_ml

                    FROM wines
                INNER JOIN vintages
                    ON wines.id = vintages.wine_id

                INNER JOIN keywords_wine
                    ON wines.id = keywords_wine.wine_id

                GROUP BY wines.id

                ORDER BY vintages.bottle_volume_ml

                LIMIT 1
                """
biggest = pd.read_sql_query(query_big, conn)
biggest['char'] = 'Biggest bottle'
highlighted = pd.concat([highlighted, biggest[['char', 'name']]])
```

```
[ ]: query_best = """
                SELECT wines.name,
                vintages.year,
                wines.ratings_average AS rating,
                vintages.price_euros

                    FROM wines

                INNER JOIN vintages
                    ON wines.id = vintages.wine_id

                GROUP BY wines.id

                ORDER by rating DESC

                LIMIT 1
                """
     best = pd.read_sql_query(query_best, conn)
     best['char'] = 'Best rated'
     highlighted = pd.concat([highlighted, best[['char', 'name']]])
```

```
[ ]: query_vp = """
                SELECT wines.name,
                vintages.year,
                wines.ratings_average,
                AVG(vintages.price_euros) as price,
                wines.ratings_average/vintages.price_euros as value_price
                    FROM wines
                INNER JOIN vintages
                    ON wines.id = vintages.wine_id
                ORDER BY value_price DESC
```

```
            LIMIT 1
            """
best_vp = pd.read_sql_query(query_vp, conn)
best_vp['char'] = 'Best value/price'
highlighted = pd.concat([highlighted, best_vp[['char', 'name']]])
```

[ ]:
```
highlighted.reset_index(drop = True)
```

[ ]:
```
                char                                        name
0           Fizziest  Les Riceys Cuvée Cyriès Brut Millesimé Champagne
1           Sweetest                       Aszú 6 Puttonyos Tokaj
2       Most intense                                   Red Blend
3       Most complex                                  Tignanello
4          Yeastiest                             Brut Champagne
5      Most tropical                                  Sauternes
6          Fruitiest                                  Sassicaia
7             Oldest                Ginés Liébana Pedro Ximénez
8     Biggest bottle                 Vin Santo di Montepulciano
9         Best rated                Batard-Montrachet Grand Cru
10  Best value/price                                      Siepi
```

1.2 On which country should the sales focus ?

In order to increase the sales, the wiwino company will focus on one particular country more than the other in the next year. We came up with a metric we called *thirstiness* (thanks to my collegue Brian Daza), which is how many users each country has, over how any wines are produced in that country. That is, how thirsty these people are for wines !

[ ]:
```
query_most_wines = """
                    SELECT name, SUM(users_count)/SUM(wines_count) as␣
 ↪thirstiness
                        FROM countries
                    GROUP BY name
                    ORDER BY 2 DESC
                    LIMIT 5
                    """

top_countries_wines = pd.read_sql_query(query_most_wines, conn)
top_countries_wines
```

[ ]:
```
         name  thirstiness
0  États-Unis           60
1      Suisse           47
2    Roumanie           33
3    Portugal           28
4      Israël           27
```

Our number one target should be the United States. This country is known for having a huge love of wine, particularly French wine because of its long history and poweful notion of “terroir”. Though they have big wine producing regions such as California, Oregon, Washington and New York, none of these have the precious “terroir” that USA drinkers are looking after so much.

Switzerland is also a notable target. Being sandwitched between France and Italy, the wine culture is very strong there too. But it is a primarily mountainous region mostly not suitable for wine production which requires low-ish altitudes and a hilly landscape.

1.3 Lemon meringue pie wines !

A subset of clients who like the following tastes have been identified : coffee, toast, green apple, cream, and citrus. As wine drinkers ourselves, this set of tastes distinctly reminds us of a *lemon meringue pie*. This often happens in creamy, slighlty tart, sometimes fizzy, white wines and this happens to be a type of wine I’m also in love with.

```
[ ]: query_lemonpie = """
                     SELECT wines.name,
                         vintages.year,
                         vintages.price_euros--,
                         --GROUP_CONCAT(DISTINCT keywords.name) as tastes
                     FROM wines

                     INNER JOIN vintages
                         ON wines.id = vintages.wine_id

                     INNER JOIN keywords_wine
                         ON wines.id = keywords_wine.wine_id

                     INNER JOIN keywords
                         ON keywords_wine.keyword_id = keywords.id

                     WHERE keywords.name IN ("toast", "coffee", "green apple",␣
       ↪"cream", "citrus")
                         AND keywords_wine.count >= 10

                     GROUP BY wines.id

                     HAVING COUNT(DISTINCT keywords.name) = 5
                     """
     lemonpie = pd.read_sql_query(query_lemonpie, conn)
     lemonpie
```

```
[ ]:                                        name  year  price_euros
     0              La Grande Année Brut Champagne  2012       293.75
     1          Cristal Brut Champagne (Millésimé)  1999      1900.00
     2                 Belle Epoque Brut Champagne  2013       247.50
     3                                     Vintage  1996      1473.75
     4                La Grande Dame Brut Champagne  2008       832.50
```

```
5                                          Brut Champagne  2000       638.83
6                                     Trebbiano d'Abruzzo  2009       420.00
7   Le Mesnil Blanc de Blancs (Cuvée S) Brut Champ…  1996      2882.50
8                     Sauternes (Premier Grand Cru Classé)  2009        92.95
9                      Comtes de Champagne Blanc de Blancs  2011       811.25
10                                              Sauternes  1962      1025.00
11       R.D Extra Brut Champagne (Récemment Dégorgé)  2004       533.75
12                                                     MV  2016       120.00
13            Dom Ruinart Blanc de Blancs Brut Champagne  2010       287.00
14                                  Blanc des Millénaires  2007       260.00
15                  Sir Winston Churchill Brut Champagne  2008       683.75
16                           P2 Plénitude Brut Champagne  2003       893.75
17              Cuvée des Enchanteleurs Brut Champagne  1988      1387.50
18                                          Grande Cuvée  N.V.       245.00
```

1.4 3 most common grapes !

```
[ ]: query_most_commom_grapes = """
                            SELECT grapes.name, COUNT(grapes.name) grape_name
                            FROM grapes
                            INNER JOIN most_used_grapes_per_country as mugpc
                                ON grapes.id = mugpc.grape_id
                            INNER JOIN countries
                                ON mugpc.country_code = countries.code
                            GROUP BY grapes.name
                            ORDER BY 2 DESC
                            """

     most_common_grapes = pd.read_sql_query(query_most_commom_grapes, conn)
     most_common_grapes = most_common_grapes.name.to_list()
```

```
[ ]: grapes_choose = widgets.Dropdown(
         options=most_common_grapes,
         description='Grape :',
         disabled=False,
     )


     classify_by = widgets.RadioButtons(
         options=['Value/price (ascending)', 'Price (ascending)', 'Price␣
      ↪(descending)', 'Rating'],
         description='Classify by :',
         disabled=False
     )

     display(grapes_choose)
display(classify_by)
```

```
Dropdown(description='Grape :', options=('Cabernet Sauvignon', 'Merlot',␣
↪'Chardonnay', 'Shiraz/Syrah', 'Pinot …

RadioButtons(description='Classify by :', options=('Value/price (ascending)',␣
↪'Price (ascending)', 'Price (des…
```

[ ]:
```
if classify_by.value == 'Value/price (ascending)' :
    classify = 'wines.ratings_average/vintages.price_euros DESC'
elif classify_by.value == 'Price (ascending)' :
    classify = 'vintages.price_euros ASC'
elif classify_by.value == 'Price (descending)' :
    classify = 'vintages.price_euros DESC'
elif classify_by.value == 'Rating' :
    classify = 'wines.ratings_average DESC'

query_grape = f"""
            SELECT grapes.name AS grape,
            wines.name AS wine,
            vintages.year,
            vintages.price_euros,
            wines.ratings_average AS rating,
            wines.ratings_average/vintages.price_euros *100 as value_over_price

            FROM wines

            INNER JOIN vintages
                ON wines.id = vintages.wine_id

            INNER JOIN regions
                ON wines.region_id = regions.id

            INNER JOIN countries
                ON regions.country_code = countries.code

            INNER JOIN most_used_grapes_per_country as mugpc
                ON countries.code = mugpc.country_code

            INNER JOIN grapes
                ON mugpc.grape_id = grapes.id

            WHERE grapes.name = '{grapes_choose.value}'

            GROUP BY wines.id

            ORDER BY {classify}
            LIMIT 5
            """
grape = pd.read_sql_query(query_grape, conn)
grape
```

```
[ ]:     grape                                        wine  year  price_euros  \
     0  Merlot  Amarone della Valpolicella Classico Riserva  2011      1046.25
     1  Merlot                  Fratini Bolgheri Superiore  2018       262.60
     2  Merlot                     Cristal Rosé Vinothèque  2000      1600.00
     3  Merlot                 Batard-Montrachet Grand Cru  2020      1149.50
     4  Merlot                                    Eszencia  2011       480.37

        rating  value_over_price
     0     4.8          0.458781
     1     4.8          1.827875
     2     4.8          0.300000
     3     4.8          0.417573
     4     4.7          0.978412
```

1.5 Country Leaderboard

Which country produces the bes wines in the dataset ?

Here is the leaderboard, by average rating of the wines in each country. We selected the countries that have a minimum of 20 wines encoded, to make sure countries that have one wine do not dominate the leaderboard because their only wine has a good rating.

```
[ ]: query_country_leaderboard = """
                                 SELECT countries.name,
                                 AVG(wines.ratings_average) as av_rating,
                                 COUNT(wines.id) AS nb_wines

                                 FROM countries

                                 INNER JOIN regions
                                     ON countries.code = regions.country_code

                                 INNER JOIN wines
                                     ON regions.id = wines.region_id

                                 GROUP BY countries.name

                                 HAVING nb_wines > 20

                                 ORDER BY av_rating DESC
                                 """
     country_leaderboard = pd.read_sql_query(query_country_leaderboard, conn)
     country_leaderboard
```

```
[ ]:              name  av_rating  nb_wines
     0      États-Unis   4.490541        74
     1  Afrique du Sud   4.459091        22
     2          France   4.447130       331
     3         Espagne   4.443617        94
     4        Portugal   4.435714        28
     5          Italie   4.430026       383
     6       Argentine   4.417391        23
```

```
[ ]: # query_like_cabernet = """
     #                     SELECT wines.name,
     #                     wines.ratings_average/vintages.price_euros as value_price,
     #                     vintages.year,
     #                     wines.ratings_average,
     #                     vintages.price_euros
     #                     FROM wines

     #                     INNER JOIN regions
     #                         ON wines.region_id = regions.id
     #                     INNER JOIN countries
     #                         ON regions.country_code = countries.code
     #                     INNER JOIN most_used_grapes_per_country as mugpc
     #                         ON countries.code = mugpc.country_code
     #                     INNER JOIN grapes
     #                         ON mugpc.grape_id = grapes.id
     #                     INNER JOIN vintages
     #                         ON wines.id = vintages.wine_id

     #                     WHERE grapes.name = "Cabernet Sauvignon"
     #                     ORDER BY value_price DESC
     #                     LIMIT 5
     #                     """
     # like_cabernet = pd.read_sql_query(query_like_cabernet, conn)
     # like_cabernet
```

```
[ ]: query_countries = """
                     SELECT name
                     FROM countries
                     """
     countries = pd.read_sql_query(query_countries, conn).values.tolist()
     countries = [item for row in countries for item in row]
```

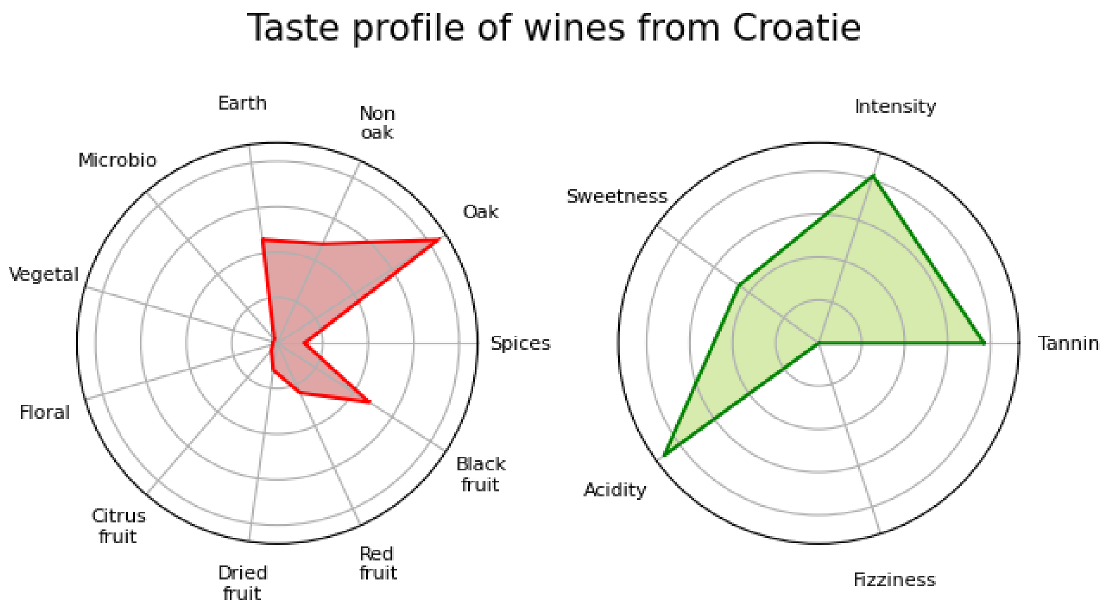
```
[ ]: country_choose = widgets.Dropdown(
         options=countries,
         value='France',
         description='Country:',
         disabled=False,
     )
     display(country_choose)
```

```
Dropdown(description='Country:', index=1, options=('Italie', 'France',␣
↪'États-Unis', 'Espagne', 'Portugal', 'A…
```

[ ]:
```
query_wine = f"""
            SELECT DISTINCT keywords_wine.group_name, SUM(keywords_wine.count)␣
↪as count
            FROM wines
            INNER JOIN keywords_wine
            ON wines.id = keywords_wine.wine_id
            INNER JOIN regions
            ON wines.region_id = regions.id
            INNER JOIN countries
            ON regions.country_code = countries.code
            WHERE countries.name == '{country_choose.value}'
            GROUP BY 1
            ORDER BY 2 DESC
            """

wine = pd.read_sql_query(query_wine, conn)

flavors = [ "spices", "oak", "non_oak", "earth", "microbio", "vegetal",␣
↪"floral", "tropical_fruit", "citrus_fruit", "tree_fruit", "dried_fruit"␣
↪,"red_fruit", "black_fruit"]
wine.group_name = wine.group_name.astype("category")
wine.group_name = wine.group_name.cat.set_categories(flavors)
wine = wine.sort_values(["group_name"])  ## 'sort' changed to 'sort_values'



fig = plt.figure(layout="constrained")
ax1 = fig.add_subplot(121, projection="polar")
# ax1 = fig.add_subplot(122, projection="polar")

# theta has 5 different angles, and the first one repeated
theta1 = np.arange(len(wine) + 1) / float(len(wine)) * 2 * np.pi
# values has the 5 values from 'Col B', with the first element repeated
values1 = wine['count'].values
values1 = np.append(values1, values1[0])

# draw the polygon and the mark the points for each angle/value combination
l1, = ax1.plot(theta1, values1, color="red", marker=",", label="count")
theta_ticks1 = wine.group_name.str.capitalize().replace('_', '\n', regex=True)
plt.xticks(theta1[:-1], theta_ticks1, color='black', size=8)
ax1.tick_params(pad=7) # to increase the distance of the labels to the plot
```

```
# fill the area of the polygon with green and some transparency
ax1.fill(theta1, values1, 'firebrick', alpha=0.4)
ax1.set_yticklabels([])




query_tastes = f"""
            SELECT
            SUM(tannin) as tannin,
            SUM(intensity) as intensity,
            SUM(sweetness) as sweetness,
            SUM(acidity) as acidity,
            SUM(fizziness) as fizziness
            FROM wines
            INNER JOIN regions
                ON wines.region_id = regions.id
            INNER JOIN countries
                ON regions.country_code = countries.code
            WHERE countries.name = '{country_choose.value}'
            GROUP BY countries.code
            """
tastes = pd.read_sql_query(query_tastes, conn)
tastes = tastes.transpose().reset_index().rename(columns = {'index' : 'tastes',␣
 ↪0 : 'count'}).fillna(0)

ax2 = fig.add_subplot(122, projection="polar")

# theta has 5 different angles, and the first one repeated
theta2 = np.arange(len(tastes) + 1) / float(len(tastes)) * 2 * np.pi
# values has the 5 values from 'Col B', with the first element repeated
values2 = tastes['count'].values
values2 = np.append(values2, values2[0])

# draw the polygon and the mark the points for each angle/value combination
l1, = ax2.plot(theta2, values2, color="green", marker=",", label="count")
theta_ticks2 = tastes.tastes.str.capitalize().replace('_', '\n', regex=True)
plt.xticks(theta2[:-1], theta_ticks2, color='black', size=8)
ax2.tick_params(pad=10) # to increase the distance of the labels to the plot
# fill the area of the polygon with green and some transparency
ax2.fill(theta2, values2, 'yellowgreen', alpha=0.4)
ax2.set_yticklabels([])

# plt.legend() # shows the legend, using the label of the line plot (useful␣
 ↪when there is more than 1 polygon)
fig.suptitle(f"Taste profile of wines from {country_choose.value}", fontsize =␣
 ↪15, x=0.5, y=0.9)
```

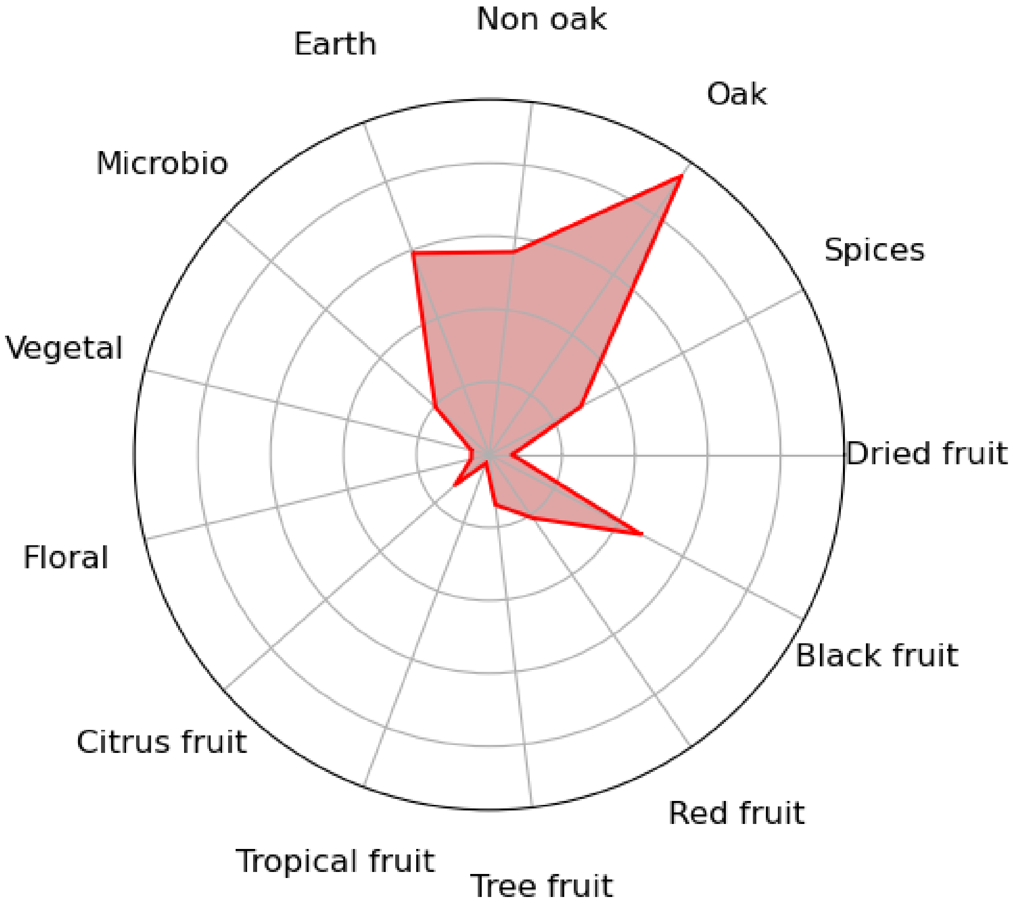
```
plt.show()
```



[ ]:
```
query_nbwines = """
                SELECT COUNT(*)
                FROM wines
                """
nb_wines = pd.read_sql_query(query_nbwines, conn)
nb_wines = nb_wines.values[0][0]

query_wine = f"""
            SELECT keywords_wine.group_name , SUM(keywords_wine.count) as count
            FROM wines
            JOIN keywords_wine
            ON wines.id = keywords_wine.wine_id
            WHERE wines.id IN (
                        SELECT id
                        FROM wines
                        ORDER BY wines.ratings_average DESC
                        LIMIT {round(nb_wines/10)}
                        )
            GROUP BY keywords_wine.group_name
            """

wine = pd.read_sql_query(query_wine, conn)

wine.group_name = wine.group_name.astype("category")
wine.group_name = wine.group_name.cat.set_categories(flavors)
```

```
wine = wine.sort_values(["group_name"])  ## 'sort' changed to 'sort_values'
wine




fig = plt.figure()
ax = fig.add_subplot(111, projection="polar")

# theta has 5 different angles, and the first one repeated
theta = np.arange(len(wine) + 1) / float(len(wine)) * 2 * np.pi
# values has the 5 values from 'Col B', with the first element repeated
values = wine['count'].values
values = np.append(values, values[0])

# draw the polygon and the mark the points for each angle/value combination
l1, = ax.plot(theta, values, color="red", marker=",", label="count")
theta_ticks = wine.group_name.str.capitalize().replace('_', ' ', regex=True)
plt.xticks(theta[:-1], theta_ticks, color='black', size=12)
ax.tick_params(pad=20) # to increase the distance of the labels to the plot
# fill the area of the polygon with green and some transparency
ax.fill(theta, values, 'firebrick', alpha=0.4)
ax.set_yticklabels([])

# plt.legend() # shows the legend, using the label of the line plot (useful␣
 ↪when there is more than 1 polygon)
plt.title(f"Flavor profile for the top wines")
plt.show()
```

Flavor profile for the top wines


Non oak
Oak
Spices
Dried fruit
Black fruit
Red fruit
Tree fruit
Tropical fruit
Citrus fruit
Floral
Vegetal
Microbio
Earth