TommasoCiampoliniDataVis

October 6, 2023

1 Data Manipulation and Visualization - Lifestyle

progetto di: Tommaso Ciampolini

Questo notebook è stato sviluppato identificando come ipotetico cliente il **Governo Italiano**.

L'obiettivo di questa analisi è quello di mostrare l'indice di benessere dell'Italia rispetto agli altri Paesi, osservando il suo andamento negli ultimi anni e analizzando possibili fattori ad esso correlati con lo scopo di aiutare il Governo a prendere decisioni riguardanti le questioni pubbliche.

2 Discovery

La seguente analisi indaga **la condizione di benessere dei cittadini italiani** in relazione alla situazione globale e mostra i diversi **parametri correlati** al benessere della persona. L'obiettivo consiste nel dare un'indicazione su quali aspetti focalizzare l'attenzione nell'ottica di **incrementare lo stato di benessere dei cittadini.**

L'analisi presenta la seguente struttura logica:

- Selezione e descrizione dei dati
- Pulizia ed esplorazione dei dati scelti
- Preparazione dei dati per la rappresentazione
- Panoramica generale sullo stato di benessere globale, negli Stati dell'UE ed in Italia
- Andamento negli anni dell'indice di felicità nel mondo, in UE ed in Italia
- Analisi dei parametri correlati all'indice di felicità in UE ed in Italia
- Confronto della situazione italiana con la media UE
- Conclusioni

3 Data Selections

3.0.1 Indice di felicità

fonte: World Happiness Report ——- Download qui

The World Happiness Report è un'affermata ricerca a livello internazionale sullo **stato di benessere globale**.

I punteggi e le classifiche di felicità (Happiness Score) utilizzano i dati del sondaggio mondiale Gallup. I punteggi si basano sulle risposte alla principale domanda di valutazione della vita posta

nel sondaggio.

Questa domanda chiede agli intervistati di valutare la propria vita facendo riferimento alla scala di Cantril: 0 corrisponde alla peggiore vita possibile e 10 alla condizione migliore.

Le colonne che seguono il punteggio di felicità (HS) stimano la misura in cui HS è influenzato da ciascuno di questi sei fattori: - produzione economica, - supporto sociale, - aspettativa di vita, - libertà, - assenza di corruzione, - generosità.

3.0.2 Indice di libertà

fonte: The Human Freedom Index ——- Download qui

The Human Freedom Index riporta la misura globale delle **libertà personali, civili ed economiche** e viene rilasciato annualmente da *Cato Institute* e da *Fraser Institute*.

L'obiettivo di questo dataset è quello di tracciare un quadro ampio ma ragionevolmente accurato del **grado di libertà generale nel mondo** intesa come assenza di vincoli coercitivi. Utilizza 79 indicatori distinti di libertà personale ed economica, comprende 165 Paesi ed i suoi dati riguardano le analisi dal 2008 al 2021.

3.0.3 Carico di malattia

fonte: Global Burden of Disease Collaborative Network. Global Burden of Disease Study 2019 (GBD 2019) Results. Seattle, United States: Institute for Health Metrics and Evaluation (IHME), 2020. rielaborati da Our World in data. ——- Download qui

Viene preso come indicatore del carico di malattia della popolazione il DALY (disability-adjusted life year) su 100.000 individui per ogni patologia. Questo valore indica la misura della **gravità globale di una patologia** espressa come il numero di anni persi a causa della malattia per disabilità o per morte prematura.

3.0.4 Disoccupazione

fonte: World Development Indicators - World Bank (2022.05.26) rielaborati da Our World in data. ——- Download qui

Questo dataset si riferisce alla quota della forza lavoro che non ha un impiego ma è disponibile e in cerca di un'occupazione espresso in percentuale sulla popolazione nazionale.

3.0.5 Anni di scolarizzazione previsti

fonte: UNDP, Human Development Report (2021-22) rielaborati da Our World in data. ——- Download qui

Gli anni di scolarizzazione previsti sono il numero di anni che un bambino in età scolare dovrebbe trascorrere a scuola e all'università, compresi gli anni di ripetizione. È la somma dei rapporti di iscrizione specifici per età per l'istruzione primaria, secondaria, post-secondaria non terziaria e terziaria.

[ ]:

4 Preparation

[ ]:
```
# importo librerie necessarie

import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import pandas as pd
import os, shutil
import datetime
from functools import reduce
from math import ceil
import folium
import geojson
import difflib
```

[ ]:
```
# definisco una funzione per estrapolare colori da colormap

def get_colors_from_cmap(cmap, n_colors):
    color_map = cm.get_cmap(cmap, 256)
    colors = color_map(np.linspace(0, 1, n_colors))
    return colors
```

[ ]:
```
# imposto stile e interfaccia grafica

%matplotlib inline
plt.style.use('seaborn-darkgrid')
sns.set_style("darkgrid")
sns.set_palette("viridis")
sns.set_theme("notebook")
```

```
/var/folders/9t/l4905t7105997k8tkd9wwxt00000gn/T/ipykernel_14038/1127396030.py:4
: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are
deprecated since 3.6, as they no longer correspond to the styles shipped by
seaborn. However, they will remain available as 'seaborn-v0_8-<style>'.
Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn-darkgrid')
```

[ ]:
```
# importo i dataset dalla cartella Data
for f in os.listdir("Data"):
    file, ext = os.path.splitext(f)
    if ext == ".csv":
        globals()[f"{file}_df"] = pd.read_csv(f"Data/{f}")
        print(f"ho creato il dataframe: {file}_df")
    else:
```

```
        print(f"non riconosco l'estensione del file {f}, inserire manualmente␣
↪se necessario")
```

```
non riconosco l'estensione del file countries_mod.geojson, inserire manualmente
se necessario
non riconosco l'estensione del file countries.geojson, inserire manualmente se
necessario
non riconosco l'estensione del file .DS_Store, inserire manualmente se
necessario
ho creato il dataframe: WHR_2015_df
ho creato il dataframe: WHR_2016_df
ho creato il dataframe: WHR_2017_df
ho creato il dataframe: school_years_df
ho creato il dataframe: WHR_2019_df
ho creato il dataframe: freedom_df
ho creato il dataframe: WHR_2018_df
ho creato il dataframe: unemployment_df
ho creato il dataframe: burden_disease_df
```

[ ]:
```
# creo cartella immagini se non esiste già

# verifico se la cartella esiste
if not os.path.exists("immagini"):
    os.mkdir("immagini") # creo cartella se non esiste
    print(f"La cartella 'immagini è stata creata.")
else:
    print(f"La cartella immagini esiste già.")
```

```
La cartella immagini esiste già.
```

5 Data Cleaning & Data Exploration

5.1 Indice Felicità

[ ]:
```
WHR_2015_df.sample()
```

[ ]:
```
      Country             Region  Happiness Rank  Happiness Score  \
144  Cambodia  Southeastern Asia             145            3.819

     Standard Error  Economy (GDP per Capita)   Family  \
144         0.05069                   0.46038  0.62736

     Health (Life Expectancy)  Freedom  Trust (Government Corruption)  \
144                   0.61114  0.66246                        0.07247

     Generosity  Dystopia Residual
144     0.40359            0.98195
```

[ ]:
```
WHR_2016_df.sample()
```

[ ]:
```
    Country                      Region  Happiness Rank  Happiness Score  \
51  Belize  Latin America and Caribbean              52            5.956

    Lower Confidence Interval  Upper Confidence Interval  \
51                       5.71                      6.202

    Economy (GDP per Capita)   Family  Health (Life Expectancy)  Freedom  \
51                   0.87616  0.68655                   0.45569  0.51231

    Trust (Government Corruption)  Generosity  Dystopia Residual
51                        0.10771     0.23684            3.08039
```

[ ]:
```
WHR_2017_df.sample()
```

[ ]:
```
       Country  Happiness.Rank  Happiness.Score  Whisker.high  Whisker.low  \
5  Netherlands               6            7.377      7.427426     7.326574

   Economy..GDP.per.Capita.    Family  Health..Life.Expectancy.   Freedom  \
5                  1.503945  1.428939                  0.810696  0.585384

   Generosity  Trust..Government.Corruption.  Dystopia.Residual
5     0.47049                       0.282662           2.294804
```

[ ]:
```
WHR_2018_df.sample()
```

[ ]:
```
    Overall rank Country or region  Score  GDP per capita  Social support  \
77            78            Serbia  5.398           0.975           1.369

    Healthy life expectancy  Freedom to make life choices  Generosity  \
77                    0.685                         0.288       0.134

    Perceptions of corruption
77                      0.043
```

[ ]:
```
WHR_2019_df.sample()
```

[ ]:
```
     Overall rank Country or region  Score  GDP per capita  Social support  \
150           151             Yemen   3.38           0.287           1.163

     Healthy life expectancy  Freedom to make life choices  Generosity  \
150                    0.463                         0.143       0.108

     Perceptions of corruption
150                      0.077
```

5.1.1 Considerazioni

I dataset del World Happiness Report sono simili ma non identici. I nomi delle colonne variano leggermente ed anche il numero di colonne. E' necessario conformare i nomi delle colonne e scegliere quali utilizzare.

Mantengo le colonne che sono presenti in tutti gli anni e che hanno dati rilevanti per questa analisi:

- Country
- Happiness Score
- Economy (GDP per Capita)
- Social Support
- Health (Life Expectancy)
- Freedom to make life choices
- Trust (Government Corruption)
- Generosity
- year

Per rendere più agevole l'analisi unirò i dataset con i dati dei diversi anni in un unico dataset.

Fasi:

1. Aggiungo una colonna "year" ad ogni dataframe in cui viene riportato l'anno a cui si riferiscono i dati.
2. Rinomino le colonne di interesse ed elimino quelle non rilevanti
3. Verifico che i nomi degli Stati combacino
4. Unisco i dataset
5. Gestisco i valori mancanti, nulli o 0

-1 aggiungo la colonna year

[ ]:
```
# 1 aggiungo colonna "year" ai dataframe con anno corrispettivo
for anno in range(2015,2020):
    globals()[f'WHR_{anno}_df']['year'] = anno
```

-2 sistemo le colonne

[ ]:
```
# 2 rinomino colonne ed elimino quelle che non servono: 2015
WHR_2015_df.rename(columns={"Family":"Social Support",
                            "Freedom":"Freedom to make life choices"},
                   inplace=True)

WHR_2015_df = WHR_2015_df[["year",
                           "Country",
                           "Happiness Score",
                           "Economy (GDP per Capita)",
                           "Social Support",
                           "Health (Life Expectancy)",
                           "Freedom to make life choices",
                           "Generosity",
                           "Trust (Government Corruption)"]]
```

[ ]:
```
# 2 rinomino colonne ed elimino quelle che non servono: 2016
WHR_2016_df.rename(columns={"Family":"Social Support",
                            "Freedom":"Freedom to make life choices"},
                   inplace=True)

WHR_2016_df = WHR_2016_df[["year",
                           "Country",
                           "Happiness Score",
                           "Economy (GDP per Capita)",
                           "Social Support",
                           "Health (Life Expectancy)",
                           "Freedom to make life choices",
                           "Generosity",
                           "Trust (Government Corruption)"]]
```

[ ]:
```
# 2 rinomino colonne ed elimino quelle che non servono: 2017
WHR_2017_df.rename(columns={"Happiness.Score":"Happiness Score",
                            "Economy..GDP.per.Capita.":"Economy (GDP per␣
 ↪Capita)",
                            "Health..Life.Expectancy.":"Health (Life␣
 ↪Expectancy)",
                            "Family":"Social Support",
                            "Freedom":"Freedom to make life choices",
                            "Trust..Government.Corruption.":"Trust (Government␣
 ↪Corruption)"},
                   inplace=True)

WHR_2017_df = WHR_2017_df[["year",
                           "Country",
                           "Happiness Score",
                           "Economy (GDP per Capita)",
                           "Social Support",
                           "Health (Life Expectancy)",
                           "Freedom to make life choices",
                           "Generosity",
                           "Trust (Government Corruption)"]]
```

[ ]:
```
# 2 rinomino colonne ed elimino quelle che non servono: 2018
WHR_2018_df.rename(columns={"Country or region":"Country",
                            "Score":"Happiness Score",
                            "Social support": "Social Support",
                            "GDP per capita":"Economy (GDP per Capita)",
                            "Healthy life expectancy":"Health (Life␣
 ↪Expectancy)",
                            "Perceptions of corruption":"Trust (Government␣
 ↪Corruption)"},
                   inplace=True)
```

```
WHR_2018_df = WHR_2018_df[["year",
                           "Country",
                           "Happiness Score",
                           "Economy (GDP per Capita)",
                           "Social Support",
                           "Health (Life Expectancy)",
                           "Freedom to make life choices",
                           "Generosity",
                           "Trust (Government Corruption)"]]
```

[ ]:
```
# 2 rinomino colonne ed elimino quelle che non servono: 2019
WHR_2019_df.rename(columns={"Country or region":"Country",
                            "Score":"Happiness Score",
                            "Social support": "Social Support",
                            "GDP per capita":"Economy (GDP per Capita)",
                            "Healthy life expectancy":"Health (Life␣
 ↪Expectancy)",
                            "Perceptions of corruption":"Trust (Government␣
 ↪Corruption)"},
                   inplace=True)

WHR_2019_df = WHR_2019_df[["year",
                           "Country",
                           "Happiness Score",
                           "Economy (GDP per Capita)",
                           "Social Support",
                           "Health (Life Expectancy)",
                           "Freedom to make life choices",
                           "Generosity",
                           "Trust (Government Corruption)"]]
```

-3 verifico nomi Stati Prima di poter unire i dati in unico dataset bisogna verificare che i nomi degli Stati siano scritti allo stesso modo per ogni anno:

- creo una lista dei nomi che non combaciano tra i vari anni
- correggo i nomi dove possibile
- elimino eventuali Stati non presenti tutti gli anni

[ ]:
```
# 3 creo array con nomi degli stati per ogni anno

country_2015 = WHR_2015_df["Country"]
country_2016 = WHR_2016_df["Country"]
country_2017 = WHR_2017_df["Country"]
country_2018 = WHR_2018_df["Country"]
country_2019 = WHR_2019_df["Country"]
```

```
# paesi che sono presenti tutti gli anni
evry_year_country = (reduce(np.intersect1d,(country_2015,
                            country_2016,
                            country_2017,
                            country_2018,
                            country_2019)))

print("i Paesi presenti tutti gli anni sono: ", evry_year_country.shape[0])
print("\n")
print("I Paesi che NON sono presenti tutti gli anni sono:")
for anno in range(2015,2020):
    print(anno, "\n", np.setdiff1d(globals()[f"country_{anno}"],␣
 ↪evry_year_country))
```

```
i Paesi presenti tutti gli anni sono:  141


I Paesi che NON sono presenti tutti gli anni sono:
2015
 ['Angola' 'Central African Republic' 'Comoros' 'Djibouti' 'Hong Kong'
 'Laos' 'Lesotho' 'Macedonia' 'Mozambique' 'North Cyprus' 'Oman'
 'Somaliland region' 'Sudan' 'Suriname' 'Swaziland' 'Taiwan'
 'Trinidad and Tobago']
2016
 ['Angola' 'Belize' 'Comoros' 'Hong Kong' 'Laos' 'Macedonia' 'Namibia'
 'North Cyprus' 'Puerto Rico' 'Somalia' 'Somaliland Region' 'South Sudan'
 'Sudan' 'Suriname' 'Taiwan' 'Trinidad and Tobago']
2017
 ['Angola' 'Belize' 'Central African Republic' 'Hong Kong S.A.R., China'
 'Lesotho' 'Macedonia' 'Mozambique' 'Namibia' 'North Cyprus' 'Somalia'
 'South Sudan' 'Sudan' 'Taiwan Province of China' 'Trinidad and Tobago']
2018
 ['Angola' 'Belize' 'Central African Republic' 'Hong Kong' 'Laos' 'Lesotho'
 'Macedonia' 'Mozambique' 'Namibia' 'Northern Cyprus' 'Somalia'
 'South Sudan' 'Sudan' 'Taiwan' 'Trinidad & Tobago']
2019
 ['Central African Republic' 'Comoros' 'Gambia' 'Hong Kong' 'Laos'
 'Lesotho' 'Mozambique' 'Namibia' 'North Macedonia' 'Northern Cyprus'
 'Somalia' 'South Sudan' 'Swaziland' 'Taiwan' 'Trinidad & Tobago']
```

correzioni Si possono notare alcune differenze facilmente correggibili: - Hong Kong nel 2017 ha anche la dicitura S.A.R., China - North Cyprus diventa Northen Cyprus nel 2018 - Taiwan nel 2017 ha anche la dicitura Province of China - Trinidad and Tobago dal 2018 è Trinidad & Tobago

[ ]:
```
# correggo i nomi degli stati
```

```
WHR_2017_df.loc[WHR_2017_df["Country"]== "Hong Kong S.A.R., China", "Country"]␣
 ↪= "Hong Kong"
WHR_2018_df.loc[WHR_2018_df["Country"]== "Northern Cyprus", "Country"] = "North␣
 ↪Cyprus"
WHR_2019_df.loc[WHR_2019_df["Country"]== "Northern Cyprus", "Country"] = "North␣
 ↪Cyprus"
WHR_2017_df.loc[WHR_2017_df["Country"]== "Taiwan Province of China", "Country"]␣
 ↪= "Taiwan"
WHR_2018_df.loc[WHR_2018_df["Country"]== "Trinidad & Tobago", "Country"] =␣
 ↪"Trinidad and Tobago"
WHR_2019_df.loc[WHR_2019_df["Country"]== "Trinidad & Tobago", "Country"] =␣
 ↪"Trinidad and Tobago"
```

[ ]:
```
# verifico di aver fatto correttamente le modifiche
evry_year_country = (reduce(np.intersect1d,(country_2015,
                            country_2016,
                            country_2017,
                            country_2018,
                            country_2019)))

print("i Paesi presenti tutti gli anni sono: ", evry_year_country.shape[0])
print("\n")
```

```
i Paesi presenti tutti gli anni sono:  145
```

rispetto ai 141 Paesi precedenti, si sono aggiunti i 4 Paesi che ho modificato.

-4 unisco i dataframe Unisco in un unico dataset e seleziono i 145 Paesi

[ ]:
```
# 3 unisco i dataframe
WHR_bozza_df = pd.merge(WHR_2015_df, WHR_2016_df, how='outer')
WHR_bozza_df = pd.merge(WHR_bozza_df, WHR_2017_df, how='outer')
WHR_bozza_df = pd.merge(WHR_bozza_df, WHR_2018_df, how='outer')
WHR_bozza_df = pd.merge(WHR_bozza_df, WHR_2019_df, how='outer')


WHR_bozza_df.sample(5)
```

[ ]:

	year	Country	Happiness Score	Economy (GDP per Capita)	\
770	2019	Burundi	3.775	0.04600	
501	2018	Qatar	6.374	1.64900	
301	2016	Chad	3.763	0.42214	
19	2015	United Arab Emirates	6.901	1.42727	
556	2018	Azerbaijan	5.201	1.02400	

```
     Social Support  Health (Life Expectancy)  Freedom to make life choices  \
770         0.44700                   0.38000                       0.22000
501         1.30300                   0.74800                       0.65400
301         0.63178                   0.03824                       0.12807
19          1.12575                   0.80925                       0.64157
556         1.16100                   0.60300                       0.43000

     Generosity  Trust (Government Corruption)
770     0.17600                        0.18000
501     0.25600                        0.17100
301     0.18667                        0.04952
19      0.26428                        0.38583
556     0.03100                        0.17600
```

[ ]:
```
# seleziono i 145 Paesi che sono presenti in tutti gli anni
WHR_df = WHR_bozza_df.loc[WHR_bozza_df["Country"] == "PaeseInesistente"] #␣
 ↪dataframe vuoto ma con colonne giuste
for country in evry_year_country:
    WHR_df = pd.merge(WHR_df, WHR_bozza_df.loc[WHR_bozza_df["Country"] ==␣
 ↪country], how="outer") # aggiungo gli Stati presenti ogni anno

WHR_df
```

[ ]:
```
     year      Country  Happiness Score  Economy (GDP per Capita)  \
0    2015  Afghanistan            3.575                  0.319820
1    2016  Afghanistan            3.360                  0.382270
2    2017  Afghanistan            3.794                  0.401477
3    2018  Afghanistan            3.632                  0.332000
4    2019  Afghanistan            3.203                  0.350000
..    …           …                …                         …
720  2015     Zimbabwe            4.610                  0.271000
721  2016     Zimbabwe            4.193                  0.350410
722  2017     Zimbabwe            3.875                  0.375847
723  2018     Zimbabwe            3.692                  0.357000
724  2019     Zimbabwe            3.663                  0.366000

     Social Support  Health (Life Expectancy)  Freedom to make life choices  \
0          0.302850                  0.303350                      0.234140
1          0.110370                  0.173440                      0.164300
2          0.581543                  0.180747                      0.106180
3          0.537000                  0.255000                      0.085000
4          0.517000                  0.361000                      0.000000
..              …                         …                             …
720        1.032760                  0.334750                      0.258610
721        0.714780                  0.159500                      0.254290
722        1.083096                  0.196764                      0.336384
```

```
723        1.094000                  0.248000                       0.406000
724        1.114000                  0.433000                       0.361000

     Generosity  Trust (Government Corruption)
0      0.365100                       0.097190
1      0.312680                       0.071120
2      0.311871                       0.061158
3      0.191000                       0.036000
4      0.158000                       0.025000
..          …                             …
720    0.189870                       0.080790
721    0.185030                       0.085820
722    0.189143                       0.095375
723    0.132000                       0.099000
724    0.151000                       0.089000

[725 rows x 9 columns]
```

il DataFrame ha 725 righe (145 Paesi x 5 anni) quindi l'operazione di merge è andata come previsto

-5 valori mancanti

[ ]:
```
# verifico dati mancanti
WHR_df.isna().sum()
```

[ ]:
```
year                            0
Country                         0
Happiness Score                 0
Economy (GDP per Capita)        0
Social Support                  0
Health (Life Expectancy)        0
Freedom to make life choices    0
Generosity                      0
Trust (Government Corruption)   1
dtype: int64
```

manca solo il valore di una cella nella colonna Trust

[ ]:
```
# verifico presenza di zeri
zero = WHR_df.eq(0).sum()
zero
```

[ ]:
```
year                            0
Country                         0
Happiness Score                 0
Economy (GDP per Capita)        1
Social Support                  1
Health (Life Expectancy)        3
```

```
Freedom to make life choices     2
Generosity                       5
Trust (Government Corruption)    6
dtype: int64
```

considerando la natura dei dati, il valore 0 non ha senso come dato,

è più appropriato considerarli dati mancanti paragonabili a NaN.

Sostituisco gli 0 con np.NaN

[ ]:
```
# sostituisco gli zeri con np.NaN
WHR_df.replace(0, np.NaN, inplace=True)
```

[ ]:
```
# trovo percentuale di missing values nel dataset
print(f"""
PERCENTUALE DATI MANCANTI:

{(WHR_df.isnull().mean() *100).sort_values(ascending=False)}
""")
```

```
PERCENTUALE DATI MANCANTI:

Trust (Government Corruption)    0.965517
Generosity                       0.689655
Health (Life Expectancy)         0.413793
Freedom to make life choices     0.275862
Economy (GDP per Capita)         0.137931
Social Support                   0.137931
year                             0.000000
Country                          0.000000
Happiness Score                  0.000000
dtype: float64
```

i dati mancanti sono molto pochi, sempre sotto al 1%

5.2 Indice Libertà

[ ]:
```
freedom_df.sample(5)
```

[ ]:

	year	countries	ISO	region	hf_score	hf_rank	\
354	2017	Burundi	BDI	Sub-Saharan Africa	4.96	155.0	
671	2015	Barbados	BRB	Latin America & the Caribbean	7.73	55.0	
618	2016	Portugal	PRT	Western Europe	8.67	20.0	
59	2019	Greece	GRC	Eastern Europe	7.86	56.0	
562	2016	Hungary	HUN	Eastern Europe	8.01	44.0	

```
     hf_quartile  pf_rol_procedural  pf_rol_civil  pf_rol_criminal  …  \
354          4.0                NaN           NaN              NaN  …
671          2.0               7.67          6.52             5.91  …
618          1.0               8.17          6.92             6.41  …
59           2.0               6.60          5.86             4.99  …
562          2.0               6.56          5.01             4.75  …

     ef_regulation_business_adm  ef_regulation_business_bureaucracy  \
354                        3.43                                2.44
671                        4.18                                6.67
618                        3.29                                8.67
59                         2.62                                5.56
562                        3.21                                7.33

     ef_regulation_business_start  ef_regulation_business_bribes  \
354                          9.76                           1.48
671                          9.44                           7.66
618                          9.82                           7.84
59                           9.86                           7.05
562                          9.58                           7.58

     ef_regulation_business_licensing  ef_regulation_business_compliance  \
354                              9.76                               2.11
671                              5.00                               7.34
618                              9.07                               7.28
59                               7.98                               7.84
562                              7.57                               6.90

     ef_regulation_business  ef_regulation  ef_score  ef_rank
354                    4.83           5.88      5.77    144.0
671                    6.71           6.71      6.65     99.0
618                    7.66           6.98      7.65     40.0
59                     6.82           7.18      7.15     78.0
562                    7.03           7.64      7.60     43.0

[5 rows x 125 columns]
```

[ ]:
```
freedom_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1980 entries, 0 to 1979
Columns: 125 entries, year to ef_rank
dtypes: float64(119), int64(3), object(3)
memory usage: 1.9+ MB
```

5.2.1 Considerazioni

Il dataset è formato da innumerevoli indici che mettono in relazione diversi aspetti della vita con le libertà personali. Per questa analisi ci interessano gli indici di libertà personale (pf_score), libertà economica (ef_score), libertà umana (hf_score), l'anno ed il nome dello Stato.

Mantengo le seguenti colonne:

- year
- countries
- hf_score
- pf_score
- ef_score

fasi: 1. Seleziono le colonne utili 2. Rinomino le colonne 3. Gestisco i nomi degli Stati 4. Gestisco gli anni 5. Gestisco i valori mancanti, nulli o 0

-1 seleziono le colonne

[ ]:
```
# verifico i dati delle colonne utili
freedom_df[["year", "countries", "hf_score", "pf_score", "ef_score"]].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1980 entries, 0 to 1979
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   year        1980 non-null   int64
 1   countries   1980 non-null   object
 2   hf_score    1867 non-null   float64
 3   pf_score    1978 non-null   float64
 4   ef_score    1867 non-null   float64
dtypes: float64(3), int64(1), object(1)
memory usage: 77.5+ KB
```

i dati sembrano consistenti e del tipo corretto

[ ]:
```
# prendo solo le colonne con gli indici hf (human freedom), pf (personal␣
↪freedom) ed ef (economic freedom)
freedom_df = freedom_df[["year", "countries", "hf_score", "pf_score",␣
↪"ef_score"]]
```

-2 rinomino le colonne

[ ]:
```
# cambio il nome alle colonne per standardizzare tutti i dataframe
freedom_df.rename(columns={"countries": "Country",
                           "hf_score": "Human freedom score",
                           "pf_score": "Personal freedom score",
                           "ef_score": "Economic freedom score"}, inplace=True)
freedom_df
```

[ ]:
```
      year        Country  Human freedom score  Personal freedom score  \
0     2019        Albania                 8.14                    8.38
1     2019        Algeria                 5.26                    5.51
2     2019         Angola                 6.09                    6.50
3     2019      Argentina                 7.38                    8.73
4     2019        Armenia                 8.20                    8.32
…     …             …                     …                       …
1975  2008  Venezuela, RB                 5.44                    6.26
1976  2008        Vietnam                 5.97                    6.05
1977  2008    Yemen, Rep.                  NaN                    4.89
1978  2008         Zambia                 7.32                    7.40
1979  2008       Zimbabwe                 5.20                    5.76

      Economic freedom score
0                       7.81
1                       4.90
2                       5.50
3                       5.50
4                       8.03
…                       …
1975                    4.30
1976                    5.84
1977                     NaN
1978                    7.22
1979                    4.42

[1980 rows x 5 columns]
```

-3 gestisco gli anni

[ ]:
```
freedom_df["year"].min()
```

[ ]:
```
2008
```

[ ]:
```
freedom_df["year"].max()
```

[ ]:
```
2019
```

I dati di questo dataset partono dal 2008, visto che i dati riguardanti la felicità partono dal 2015, prendo solo i dati dal 2015 in poi anche per questo dataframe.

[ ]:
```
# filtro solo gli anni dopo il 2015
freedom_df = freedom_df.loc[freedom_df["year"] >= 2015,:]
freedom_df
```

[ ]:
```
   year       Country  Human freedom score  Personal freedom score  \
0  2019       Albania                 8.14                    8.38
1  2019       Algeria                 5.26                    5.51
```

```
2    2019         Angola                 6.09                    6.50
3    2019      Argentina                 7.38                    8.73
4    2019        Armenia                 8.20                    8.32
..    …             …                    …                       …
820  2015  Venezuela, RB                 4.52                    5.77
821  2015        Vietnam                 5.83                    5.67
822  2015    Yemen, Rep.                 4.53                    3.30
823  2015         Zambia                 7.07                    7.09
824  2015       Zimbabwe                 6.25                    6.18

     Economic freedom score
0                      7.81
1                      4.90
2                      5.50
3                      5.50
4                      8.03
..                      …
820                    2.77
821                    6.04
822                    6.25
823                    7.04
824                    6.34

[825 rows x 5 columns]
```

-4 gestisco gli Stati verifico se i nomi degli Stati combacia con il dataset WHR_df

[ ]:
```
# verifico se i nomi degli Stati combaciano all'interno del dataframe nei␣
 ↪diversi anni

print("numero di Stati:")
for y in range(2015,2020):
    print(f"{y}:", freedom_df.loc[freedom_df["year"]==y,"Country"].count()) #␣
 ↪numero Stati per anno


# confronto i nomi degli Stati nei diversi anni
count = 0
for y in range(2015,2020):
    for y_1 in range(2015,2020):
        if not np.array_equal(freedom_df.loc[freedom_df["year"]==y,"Country"],
                              freedom_df.
 ↪loc[freedom_df["year"]==y_1,"Country"]):
            print(f"{y} e {y_1} sono diversi!")
            count += 1
if count == 0:
```

```
    print("Sono tutti uguali!")
```

```
numero di Stati:
2015: 165
2016: 165
2017: 165
2018: 165
2019: 165
Sono tutti uguali!
```

[ ]:
```
# confronto i nomi degli Stati di freedom_df con quelli di WHR_df
C_non_inc = np.setdiff1d(freedom_df.loc[freedom_df["year"]==2015,"Country"],␣
 ↪evry_year_country)

print(f"nomi in freedom_df che non sono presenti in WHR_df sono␣
 ↪{len(C_non_inc)}")
```

```
nomi in freedom_df che non sono presenti in WHR_df sono 39
```

[ ]:
```
# trovo gli Stati spaiati in WHR_df
spaiati = np.setdiff1d(evry_year_country, freedom_df.
 ↪loc[freedom_df["year"]==2015,"Country"])

print(f"nomi spaiati in WHR_df sono {len(spaiati)}")
```

```
nomi spaiati in WHR_df sono 19
```

[ ]:
```
for c in spaiati:
    print(c)
```

```
Afghanistan
Congo (Brazzaville)
Congo (Kinshasa)
Egypt
Hong Kong
Iran
Ivory Coast
Kosovo
Kyrgyzstan
North Cyprus
Palestinian Territories
Russia
Slovakia
South Korea
Syria
Turkmenistan
Uzbekistan
Venezuela
```

Yemen

[ ]:
```
#cerco somiglianze tra i Paesi che non combaciano
for C in C_non_inc:
    #cerca una corrispondenza simile
    match = difflib.get_close_matches(C, spaiati, n=5, cutoff=0.6)
    if match:
        print(f"Possibile somiglianza tra: {C} e {match}")

    # cerco i Paesi che hanno le prime 5 lettere uguali
    for c in spaiati:
        if C[0:5] == c[0:5]: # se hanno le prime 5 lettere uguali
            print(f"Iniziano con le stesse lettere: {C} e {c}")
```

```
Iniziano con le stesse lettere: Congo, Dem. Rep. e Congo (Brazzaville)
Iniziano con le stesse lettere: Congo, Dem. Rep. e Congo (Kinshasa)
Iniziano con le stesse lettere: Congo, Rep. e Congo (Brazzaville)
Iniziano con le stesse lettere: Congo, Rep. e Congo (Kinshasa)
Iniziano con le stesse lettere: Egypt, Arab Rep. e Egypt
Possibile somiglianza tra: Hong Kong SAR, China e ['Hong Kong']
Iniziano con le stesse lettere: Hong Kong SAR, China e Hong Kong
Iniziano con le stesse lettere: Kyrgyz Republic e Kyrgyzstan
Iniziano con le stesse lettere: North Macedonia e North Cyprus
Iniziano con le stesse lettere: Russian Federation e Russia
Possibile somiglianza tra: Slovak Republic e ['Slovakia']
Iniziano con le stesse lettere: Slovak Republic e Slovakia
Possibile somiglianza tra: Suriname e ['Syria']
Iniziano con le stesse lettere: Syrian Arab Republic e Syria
Possibile somiglianza tra: Venezuela, RB e ['Venezuela']
Iniziano con le stesse lettere: Venezuela, RB e Venezuela
Possibile somiglianza tra: Yemen, Rep. e ['Yemen']
Iniziano con le stesse lettere: Yemen, Rep. e Yemen
```

[ ]:
```
# correggo i nomi
freedom_df.loc[freedom_df["Country"]=="Congo, Dem. Rep.", "Country"] = "Congo␣
 ↪(Kinshasa)"
freedom_df.loc[freedom_df["Country"]=="Congo, Rep.", "Country"] = "Congo␣
 ↪(Brazzaville)"
freedom_df.loc[freedom_df["Country"]=="Egypt, Arab Rep.", "Country"] = "Egypt"
freedom_df.loc[freedom_df["Country"]=="Guinea-Bissau", "Country"] = "Guinea"
freedom_df.loc[freedom_df["Country"]=="Hong Kong SAR, China", "Country"] =␣
 ↪"Hong Kong"
freedom_df.loc[freedom_df["Country"]=="Kyrgyz Republic", "Country"] =␣
 ↪"Kyrgyzstan"
freedom_df.loc[freedom_df["Country"]=="Russian Federation", "Country"] =␣
 ↪"Russia"
freedom_df.loc[freedom_df["Country"]=="Slovak Republic", "Country"] = "Slovakia"
```

```
freedom_df.loc[freedom_df["Country"]=="Syrian Arab Republic", "Country"] =␣
 ↪"Syria"
freedom_df.loc[freedom_df["Country"]=="Venezuela, RB", "Country"] = "Venezuela"
freedom_df.loc[freedom_df["Country"]=="Yemen, Rep.", "Country"] = "Yemen"
```

[ ]:
```
# verifico la correzione, ho fatto 11 correzioni. dovrei trovarmi 39-11= 28␣
 ↪nomi nella lista dei nomi
# che sono in freedom_df ma non in WHR_df
print("ora i nomi che non combaciano sono:")
len(np.setdiff1d(freedom_df.loc[freedom_df["year"]==2015,"Country"],␣
 ↪evry_year_country))
```

```
ora i nomi che non combaciano sono:
```

[ ]:
```
28
```

[ ]:
```
# mantengo i nomi degli Stati che combaciano

# creo nuovo dataframe vuoto, con solo le colonne
new_freedom_df = freedom_df.loc[freedom_df["Country"]== "PaeseInesistente"]
for country in evry_year_country:
    new_freedom_df = pd.merge(new_freedom_df, freedom_df.
 ↪loc[freedom_df["Country"]== country],
                              how="outer")

new_freedom_df
```

[ ]:
```
     year   Country  Human freedom score  Personal freedom score  \
0    2019   Albania                 8.14                    8.38
1    2018   Albania                 8.16                    8.43
2    2017   Albania                 8.16                    8.46
3    2016   Albania                 8.10                    8.39
4    2015   Albania                 8.10                    8.38
..    …       …                     …                       …
680  2019  Zimbabwe                 5.60                    6.07
681  2018  Zimbabwe                 5.82                    6.18
682  2017  Zimbabwe                 5.77                    6.19
683  2016  Zimbabwe                 5.95                    6.04
684  2015  Zimbabwe                 6.25                    6.18

     Economic freedom score
0                      7.81
1                      7.78
2                      7.73
3                      7.71
4                      7.71
..                      …
```

```
680                    4.94
681                    5.31
682                    5.19
683                    5.82
684                    6.34

[685 rows x 5 columns]
```

-5 valori mancanti

[ ]:
```
# percentuali dati mancanti
nan_perc = (new_freedom_df.isnull().mean() *100).sort_values(ascending=False)

# rappresento
print(f"""
PERCENTUALI DATI MANCANTI:

{nan_perc}
""")
```

```
PERCENTUALI DATI MANCANTI:

Human freedom score       0.291971
Economic freedom score    0.291971
year                      0.000000
Country                   0.000000
Personal freedom score    0.000000
dtype: float64
```

[ ]:
```
# verifico presenza di zeri
zero = new_freedom_df.eq(0).sum()

# rappresento
print(f"""
ZERI NEI DATI:

{zero}
""")
```

```
ZERI NEI DATI:

year                      0
Country                   0
Human freedom score       0
Personal freedom score    0
```

```
Economic freedom score    0
dtype: int64
```

5.3 Carico Malattia

[ ]:
```
burden_disease_df
```

[ ]:
```
          Country Code  Year     DALYs
0     Afghanistan  AFG  1990  86375.17
1     Afghanistan  AFG  1991  83381.07
2     Afghanistan  AFG  1992  79890.55
3     Afghanistan  AFG  1993  80292.52
4     Afghanistan  AFG  1994  83334.93
…           …  …   …       …
6835     Zimbabwe  ZWE  2015  66169.40
6836     Zimbabwe  ZWE  2016  64174.96
6837     Zimbabwe  ZWE  2017  62297.15
6838     Zimbabwe  ZWE  2018  60084.52
6839     Zimbabwe  ZWE  2019  58969.11

[6840 rows x 4 columns]
```

[ ]:
```
burden_disease_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6840 entries, 0 to 6839
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Country  6840 non-null   object
 1   Code     6150 non-null   object
 2   Year     6840 non-null   int64
 3   DALYs    6840 non-null   float64
dtypes: float64(1), int64(1), object(2)
memory usage: 213.9+ KB
```

i dati sono tutti non-nulli e del tipo corretto a parte per la colonna Code.

5.3.1 Considerazioni

Il dataset contiene i dati dal 1990 al 2019

4 colonne: Country, Code, Year, DALYs

FASI:

1. Elimino la colonna Code
2. Rinomino le colonne Year e DALYs
3. Elimino gli anni prima del 2015

4. Gestisco i nomi degli Stati
5. Gestisco i dati mancanti

-1 elimino la colonna Code

[ ]:
```
# elimino la colonna Code
burden_disease_df.drop(columns=("Code"), inplace=True)
```

[ ]:
```
burden_disease_df.sample(5)
```

[ ]:

	Country	Year	DALYs
4620	Peru	1990	42906.30
5389	Solomon Islands	2009	71717.31
2897	Japan	2007	17572.75
5515	South Korea	2015	17635.83
306	Australia	1996	24087.24

-2 rinomino le colonne

[ ]:
```
#rinomino colonne
burden_disease_df.rename(columns={"DALYs (Disability-Adjusted Life Years) - All␣
↪causes - Sex: Both - Age: Age-standardized (Rate)": "Burden disease (DALYs)",
                                  "Year": "year",
                                 "Entity": "Country"},
                         inplace=True)
```

[ ]:
```
burden_disease_df.sample(5)
```

[ ]:

	Country	year	DALYs
6413	Uruguay	2013	25547.46
5418	Somalia	2008	80198.59
551	Belgium	2001	23654.83
1013	Canada	2013	19536.04
824	Brunei	2004	30573.97

-3 gestisco gli anni

[ ]:
```
# elimino gli anni precedenti al 2015
burden_disease_df = burden_disease_df.loc[burden_disease_df["year"]>2014]
```

[ ]:
```
burden_disease_df.head(5)
```

[ ]:

	Country	year	DALYs
25	Afghanistan	2015	59166.96
26	Afghanistan	2016	58734.65
27	Afghanistan	2017	56378.69
28	Afghanistan	2018	57072.05
29	Afghanistan	2019	55424.65

-4 gestisco gli Stati

```
[ ]: # numero degli Stati nel dataset
disease_country = burden_disease_df["Country"].unique()
len(disease_country)
```

```
[ ]: 228
```

Il numero è così alto perchè oltre agli Stati sono presenti anche diverse regioni territoriali che comprendono diversi Stati, ad esempio: Africa Region, European Region ….

```
[ ]: # verifico gli Stati di disease_country che sono presenti anche in␣
  ↪evrevry_year_country
len(np.intersect1d(disease_country, evry_year_country))
```

```
[ ]: 137
```

137 Paesi sui 145 di WHR_df combaciano

verifico, come per i dataset precedenti, possibili somiglianze di nomi

```
[ ]: # cerco gli 8 Paesi in WHR_df che non hanno un corrispettivo in␣
  ↪burden_disease_df
spaiati = np.setdiff1d(evry_year_country, disease_country)
spaiati
```

```
[ ]: array(['Congo (Brazzaville)', 'Congo (Kinshasa)', 'Czech Republic',
       'Hong Kong', 'Ivory Coast', 'Kosovo', 'North Cyprus',
       'Palestinian Territories'], dtype=object)
```

```
[ ]: # trovo gli Stati di burden_disease_df che non sono in WHR_df
C_non_inc = np.setdiff1d(disease_country, evry_year_country)
print(f"gli Stati non inclusi sono {len(C_non_inc)}")
```

```
gli Stati non inclusi sono 91
```

```
[ ]: #cerco somiglianze tra i Paesi che non combaciano
for C in C_non_inc:
    #cerca una corrispondenza simile
    match = difflib.get_close_matches(C, spaiati, n=5, cutoff=0.6)
    if match:
        print(f"Possibile somiglianza tra: {C} e {match}")

    # cerco i Paesi che hanno le prime 5 lettere uguali
    for c in spaiati:
        if C[0:5] == c[0:5]: # se hanno le prime 5 lettere uguali
            print(f"Iniziano con le stesse lettere: {C} e {c}")
```

```
Possibile somiglianza tra: Central African Republic e ['Czech Republic']
Iniziano con le stesse lettere: Congo e Congo (Brazzaville)
```

```
Iniziano con le stesse lettere: Congo e Congo (Kinshasa)
Iniziano con le stesse lettere: Czechia e Czech Republic
Iniziano con le stesse lettere: North America (WB) e North Cyprus
Possibile somiglianza tra: North Korea e ['North Cyprus']
Iniziano con le stesse lettere: North Korea e North Cyprus
Iniziano con le stesse lettere: North Macedonia e North Cyprus
Iniziano con le stesse lettere: Northern Ireland e North Cyprus
Iniziano con le stesse lettere: Northern Mariana Islands e North Cyprus
Iniziano con le stesse lettere: Palestine e Palestinian Territories
```

[ ]:
```
# correggo i nomi degli Stati
burden_disease_df.loc[burden_disease_df["Country"]=="Czechia", "Country"] =␣
 ↪"Czech Republic"
burden_disease_df.loc[burden_disease_df["Country"]=="Congo", "Country"] =␣
 ↪"Congo (Kinshasa)"
burden_disease_df.loc[burden_disease_df["Country"]=="Palestine", "Country"] =␣
 ↪"Palestinian Territories"
```

[ ]:
```
# verifico correzioni
disease_country = burden_disease_df["Country"].unique()

new_spaiati = np.setdiff1d(evry_year_country, disease_country)
print(f"ora i Paesi spaiati sono {len(new_spaiati)}")
```

```
ora i Paesi spaiati sono 5
```

[ ]:
```
# mantengo i nomi degli Stati che combaciano

# creo nuovo dataframe vuoto con solo le colonne
new_burden_disease_df = burden_disease_df.
 ↪loc[burden_disease_df["Country"]=="PaeseInesistente"]

for country in evry_year_country:
    new_burden_disease_df = pd.merge(new_burden_disease_df,
                                     burden_disease_df.
 ↪loc[burden_disease_df["Country"]== country],
                                     how="outer")
```

[ ]:
```
new_burden_disease_df
```

[ ]:

	Country	year	DALYs
0	Afghanistan	2015	59166.96
1	Afghanistan	2016	58734.65
2	Afghanistan	2017	56378.69
3	Afghanistan	2018	57072.05
4	Afghanistan	2019	55424.65
..	…	…	…

```
695     Zimbabwe  2015  66169.40
696     Zimbabwe  2016  64174.96
697     Zimbabwe  2017  62297.15
698     Zimbabwe  2018  60084.52
699     Zimbabwe  2019  58969.11

[700 rows x 3 columns]
```

-5 valori mancanti

[ ]:
```python
# percentuali dati mancanti
nan_perc = (new_burden_disease_df.isnull().mean() *100).
  ↪sort_values(ascending=False)

# rappresento
print(f"""
PERCENTUALI DATI MANCANTI:

{nan_perc}
""")
```

```
PERCENTUALI DATI MANCANTI:

Country    0.0
year       0.0
DALYs      0.0
dtype: float64
```

[ ]:
```python
# verifico presenza di zeri
zero = new_burden_disease_df.eq(0).sum()

# rappresento
print(f"""
ZERI NEI DATI:

{zero}
""")
```

```
ZERI NEI DATI:

Country    0
year       0
DALYs      0
dtype: int64
```

5.4 Disoccupazione

[ ]:
```
unemployment_df
```

[ ]:
```
           Entity Code  Year  \
0     Afghanistan  AFG  1991
1     Afghanistan  AFG  1992
2     Afghanistan  AFG  1993
3     Afghanistan  AFG  1994
4     Afghanistan  AFG  1995
…             …  …   …
6257     Zimbabwe  ZWE  2017
6258     Zimbabwe  ZWE  2018
6259     Zimbabwe  ZWE  2019
6260     Zimbabwe  ZWE  2020
6261     Zimbabwe  ZWE  2021

      Unemployment, total (% of total labor force) (modeled ILO estimate)
0                                                10.649
1                                                10.821
2                                                10.723
3                                                10.726
4                                                11.179
…                                                 …
6257                                              4.785
6258                                              4.796
6259                                              4.833
6260                                              5.351
6261                                              5.174

[6262 rows x 4 columns]
```

[ ]:
```
unemployment_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6262 entries, 0 to 6261
Data columns (total 4 columns):
 #   Column                                                                Non-
Null Count  Dtype
---  ------
--------------  -----
 0   Entity                                                                6262
non-null   object
 1   Code                                                                  5828
non-null   object
 2   Year                                                                  6262
non-null   int64
 3   Unemployment, total (% of total labor force) (modeled ILO estimate)  6262
```

```
non-null   float64
dtypes: float64(1), int64(1), object(2)
memory usage: 195.8+ KB
```

Ad eccezione della colonna Code, le altre colonne sono consistenti e del tipo giusto

5.4.1 Considerazioni

i dati vanno dal 1991 al 2021

il dataset è composto da 4 features: Entity, Code, Unemployment, total (% of total labor force) (modeled ILO estimate)

Fasi:

1. Elimino la colonna Code
2. Rinomino le colonne
3. Gestisco gli anni
4. Gestisco i nomi degli Stati
5. Gestisco i dati mancanti

-1 elimino la colonna Code

[ ]:
```
# elimino colonne
unemployment_df = unemployment_df.drop(columns=["Code"])
```

[ ]:
```
unemployment_df.sample(5)
```

[ ]:
```
           Entity  Year  \
3124        Libya  2015
4653       Rwanda  1994
3978  New Zealand  2001
3572       Mexico  1998
4519     Portugal  2015

      Unemployment, total (% of total labor force) (modeled ILO estimate)
3124                                             19.531
4653                                              0.465
3978                                              5.430
3572                                              3.730
4519                                             12.440
```

-2 rinomino le colonne

[ ]:
```
# rinomino colonne
unemployment_df = unemployment_df.rename(columns={"Entity": "Country",
                                                  "Year": "year",
                                                  "Unemployment, total (% of␣
 ↪total labor force) (modeled ILO estimate)": "Unemployment %"})
```

[ ]:
```
unemployment_df.sample(5)
```

[ ]:

	Country	year	Unemployment %
683	Brunei	1992	4.861
6208	Zambia	1999	12.441
4321	Panama	2003	3.791
5552	Timor	1994	3.540
286	Bahamas	1998	7.650

-3 gestisco gli anni

[ ]:
```
# mantengo solo gli anni tra il 2015 e il 2019
unemployment_df = unemployment_df.loc[(unemployment_df["year"] > 2014) & ␣
↪(unemployment_df["year"] < 2020)]
```

[ ]:
```
unemployment_df.head(6)
```

[ ]:

	Country	year	Unemployment %
24	Afghanistan	2015	11.127000
25	Afghanistan	2016	11.158000
26	Afghanistan	2017	11.180000
27	Afghanistan	2018	11.152000
28	Afghanistan	2019	11.217000
55	Albania	2015	17.190001

-4 gestisco gli Stati

[ ]:
```
# numero degli Stati nel dataset
unemployment_country = unemployment_df["Country"].unique()
len(unemployment_country)
```

[ ]: 202

[ ]:
```
# verifico gli Stati di unemployment_country che sono presenti anche in␣
↪evrevry_year_country
len(np.intersect1d(unemployment_country, evry_year_country))
```

[ ]: 137

137 Paesi sui 145 di WHR_df combaciano

verifico, come per i dataset precedenti, possibili somiglianze di nomi

[ ]:
```
# cerco i 42 Paesi in TOT_df che non hanno un corrispettivo in unemployment_df
spaiati = np.setdiff1d(evry_year_country, unemployment_country)
print(f"i Paesi spaiati sono {len(spaiati)}")
spaiati
```

```
i Paesi spaiati sono 8
```

[ ]:
```
array(['Congo (Brazzaville)', 'Congo (Kinshasa)', 'Czech Republic',
       'Ivory Coast', 'Kosovo', 'North Cyprus', 'Palestinian Territories',
       'Taiwan'], dtype=object)
```

[ ]:
```
# trovo gli Stati di unemployment_df che non sono in TOT_df
C_non_inc = np.setdiff1d(unemployment_country, evry_year_country)
print(f"gli Stati non inclusi sono {len(C_non_inc)}")
```

```
gli Stati non inclusi sono 65
```

[ ]:
```
#cerco somiglianze tra i Paesi che non combaciano
for C in C_non_inc:
    #cerca una corrispondenza simile
    match = difflib.get_close_matches(C, spaiati, n=5, cutoff=0.6)
    if match:
        print(f"Possibile somiglianza tra: {C} e {match}")

    # cerco i Paesi che hanno le prime 5 lettere uguali
    for c in spaiati:
        if C[0:5] == c[0:5]: # se hanno le prime 5 lettere uguali
            print(f"Iniziano con le stesse lettere: {C} e {c}")
```

```
Possibile somiglianza tra: Central African Republic e ['Czech Republic']
Iniziano con le stesse lettere: Congo e Congo (Brazzaville)
Iniziano con le stesse lettere: Congo e Congo (Kinshasa)
Iniziano con le stesse lettere: Czechia e Czech Republic
Iniziano con le stesse lettere: North America e North Cyprus
Possibile somiglianza tra: North Korea e ['North Cyprus']
Iniziano con le stesse lettere: North Korea e North Cyprus
Iniziano con le stesse lettere: North Macedonia e North Cyprus
Iniziano con le stesse lettere: Palestine e Palestinian Territories
```

[ ]:
```
# correggo i nomi degli Stati

unemployment_df.loc[unemployment_df["Country"]=="Democratic Republic of Congo",␣
 ↪"Country"] = "Congo (Kinshasa)"
unemployment_df.loc[unemployment_df["Country"]=="Congo", "Country"] = "Congo␣
 ↪(Brazzaville)"
unemployment_df.loc[unemployment_df["Country"]=="Palestine", "Country"] =␣
 ↪"Palestinian Territories"
```

[ ]:
```
# verifico correzioni
unemployment_country = unemployment_df["Country"].unique()

new_spaiati = np.setdiff1d(evry_year_country, unemployment_country)
print(f"ora i Paesi spaiati sono {len(new_spaiati)}")
```

```
ora i Paesi spaiati sono 5
```

[ ]:
```
# mantengo i nomi degli Stati che combaciano

# creo nuovo dataframe vuoto, con solo le colonne
new_unemployment_df = unemployment_df.
 ↪loc[unemployment_df["Country"]=="PaeseInesistente"]

for country in evry_year_country:
    new_unemployment_df = pd.merge(new_unemployment_df, unemployment_df.
 ↪loc[unemployment_df["Country"]== country], how="outer")
```

[ ]:
```
new_unemployment_df
```

[ ]:

	Country	year	Unemployment %
0	Afghanistan	2015	11.127
1	Afghanistan	2016	11.158
2	Afghanistan	2017	11.180
3	Afghanistan	2018	11.152
4	Afghanistan	2019	11.217
..	…	…	…
695	Zimbabwe	2015	4.778
696	Zimbabwe	2016	4.788
697	Zimbabwe	2017	4.785
698	Zimbabwe	2018	4.796
699	Zimbabwe	2019	4.833

[700 rows x 3 columns]

-5 valori mancanti

[ ]:
```
# percentuali dati mancanti
nan_perc = (new_unemployment_df.isnull().mean() *100).
 ↪sort_values(ascending=False)

# rappresento
print(f"""
PERCENTUALI DATI MANCANTI:

{nan_perc}
""")
```

```
PERCENTUALI DATI MANCANTI:

Country           0.0
year              0.0
Unemployment %    0.0
dtype: float64
```

[ ]:
```
# verifico presenza di zeri
zero = new_unemployment_df.eq(0).sum()

# rappresento
print(f"""
ZERI NEI DATI:

{zero}
""")
```

```
ZERI NEI DATI:

Country           0
year              0
Unemployment %    0
dtype: int64
```

5.5 Anni di scolarizzazione previsti

[ ]:
```
school_years_df
```

[ ]:

	Entity	Code	Year	Expected Years of Schooling
0	Afghanistan	AFG	1990	2.504050
1	Afghanistan	AFG	1991	2.806550
2	Afghanistan	AFG	1992	3.109050
3	Afghanistan	AFG	1993	3.411550
4	Afghanistan	AFG	1994	3.714050
…	…	…	…	…
6266	Zimbabwe	ZWE	2017	11.853943
6267	Zimbabwe	ZWE	2018	11.981767
6268	Zimbabwe	ZWE	2019	12.110969
6269	Zimbabwe	ZWE	2020	12.110969
6270	Zimbabwe	ZWE	2021	12.110969

[6271 rows x 4 columns]

[ ]:
```
school_years_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6271 entries, 0 to 6270
Data columns (total 4 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Entity                      6271 non-null   object
 1   Code                        5951 non-null   object
 2   Year                        6271 non-null   int64
```

```
 3   Expected Years of Schooling  6271 non-null   float64
dtypes: float64(1), int64(1), object(2)
memory usage: 196.1+ KB
```

Ad eccezione della colonna Code, le altre colonne sono consistenti e del tipo giusto

5.5.1 Considerazioni

i dati vanno dal 1990 al 2021

il dataset è composto da 4 features: Entity, Code, Expected Years of Schooling

Fasi:

1. Elimino la colonna Code
2. Rinomino le colonne
3. Gestisco gli anni
4. Gestisco i nomi degli Stati
5. Gestisco i dati mancanti

-1 elimino la colonna Code

[ ]:
```
# elimino colonne
school_years_df = school_years_df.drop(columns=["Code"])
```

[ ]:
```
school_years_df.sample(5)
```

[ ]:

	Entity	Year	Expected Years of Schooling
3306	Luxembourg	1990	10.527326
4084	Niger	2015	5.918090
1603	East Asia and the Pacific (UNDP)	2013	12.816409
4352	Papua New Guinea	2003	6.541913
4843	Saudi Arabia	2010	13.996519

-2 rinomino le colonne

[ ]:
```
# rinomino colonne
school_years_df = school_years_df.rename(columns={"Entity": "Country",
                                                  "Year": "year"})
```

[ ]:
```
school_years_df.sample(5)
```

[ ]:

	Country	year	Expected Years of Schooling
5784	Tuvalu	2006	10.950456
5653	Tonga	2020	16.048570
2783	Japan	1992	13.065940
957	Cameroon	2019	13.108600
3182	Liberia	2016	10.125308

-3 gestisco gli anni

```
[ ]: # mantengo solo gli anni tra il 2015 e il 2019
school_years_df = school_years_df.loc[(school_years_df["year"] > 2014) & ␣
 ↪(school_years_df["year"] < 2020)]
```

```
[ ]: school_years_df.head(6)
```

```
[ ]:         Country  year  Expected Years of Schooling
     25  Afghanistan  2015                    10.180150
     26  Afghanistan  2016                    10.185949
     27  Afghanistan  2017                    10.191750
     28  Afghanistan  2018                    10.197550
     29  Afghanistan  2019                    10.263844
     57      Albania  2015                    15.076300
```

-4 gestisco gli Stati

```
[ ]: # numero degli Stati nel dataset
school_years_country = school_years_df["Country"].unique()
len(school_years_country)
```

```
[ ]: 204
```

```
[ ]: # verifico gli Stati di school_years_country che sono presenti anche in␣
 ↪evrevry_year_country
len(np.intersect1d(school_years_country, evry_year_country))
```

```
[ ]: 137
```

137 Paesi sui 145 di WHR_df combaciano

verifico, come per i dataset precedenti, possibili somiglianze di nomi

```
[ ]: # cerco i Paesi in TOT_df che non hanno un corrispettivo in school_years_df
spaiati = np.setdiff1d(evry_year_country, school_years_country)
print(f"i Paesi spaiati sono {len(spaiati)}")
spaiati
```

```
i Paesi spaiati sono 8
```

```
[ ]: array(['Congo (Brazzaville)', 'Congo (Kinshasa)', 'Czech Republic',
       'Ivory Coast', 'Kosovo', 'North Cyprus', 'Palestinian Territories',
       'Taiwan'], dtype=object)
```

```
[ ]: # trovo gli Stati di school_years_df che non sono in TOT_df
C_non_inc = np.setdiff1d(school_years_country, evry_year_country)
print(f"gli Stati non inclusi sono {len(C_non_inc)}")
```

```
gli Stati non inclusi sono 67
```

[ ]:
```
#cerco somiglianze tra i Paesi che non combaciano
for C in C_non_inc:
    #cerca una corrispondenza simile
    match = difflib.get_close_matches(C, spaiati, n=5, cutoff=0.6)
    if match:
        print(f"Possibile somiglianza tra: {C} e {match}")

    # cerco i Paesi che hanno le prime 5 lettere uguali
    for c in spaiati:
        if C[0:5] == c[0:5]: # se hanno le prime 5 lettere uguali
            print(f"Iniziano con le stesse lettere: {C} e {c}")
```

```
Possibile somiglianza tra: Central African Republic e ['Czech Republic']
Iniziano con le stesse lettere: Congo e Congo (Brazzaville)
Iniziano con le stesse lettere: Congo e Congo (Kinshasa)
Iniziano con le stesse lettere: Czechia e Czech Republic
Possibile somiglianza tra: North Korea e ['North Cyprus']
Iniziano con le stesse lettere: North Korea e North Cyprus
Iniziano con le stesse lettere: North Macedonia e North Cyprus
Iniziano con le stesse lettere: Palestine e Palestinian Territories
```

[ ]:
```
# correggo i nomi degli Stati

school_years_df.loc[school_years_df["Country"]=="Democratic Republic of Congo",␣
 ↪"Country"] = "Congo (Kinshasa)"
school_years_df.loc[school_years_df["Country"]=="Congo", "Country"] = "Congo␣
 ↪(Brazzaville)"
school_years_df.loc[school_years_df["Country"]=="Czechia", "Country"] = "Czech␣
 ↪Republic"
school_years_df.loc[school_years_df["Country"]=="Palestine", "Country"] =␣
 ↪"Palestinian Territories"
```

[ ]:
```
# verifico correzioni
school_years_country = school_years_df["Country"].unique()

new_spaiati = np.setdiff1d(evry_year_country, school_years_country)
print(f"ora i Paesi spaiati sono {len(new_spaiati)}")
```

```
ora i Paesi spaiati sono 4
```

[ ]:
```
# mantengo i nomi degli Stati che combaciano

# creo nuovo dataframe vuoto, con solo le colonne
new_school_years_df = school_years_df.
 ↪loc[school_years_df["Country"]=="PaeseInesistente"]

for country in evry_year_country:
```

```
    new_school_years_df = pd.merge(new_school_years_df, school_years_df.
 ↪loc[school_years_df["Country"]== country], how="outer")
```

[ ]:
```
new_school_years_df
```

[ ]:
```
          Country  year  Expected Years of Schooling
0     Afghanistan  2015                    10.180150
1     Afghanistan  2016                    10.185949
2     Afghanistan  2017                    10.191750
3     Afghanistan  2018                    10.197550
4     Afghanistan  2019                    10.263844
..            …   …                         …
700      Zimbabwe  2015                    11.602372
701      Zimbabwe  2016                    11.727483
702      Zimbabwe  2017                    11.853943
703      Zimbabwe  2018                    11.981767
704      Zimbabwe  2019                    12.110969

[705 rows x 3 columns]
```

-5 valori mancanti

[ ]:
```
# percentuali dati mancanti
nan_perc = (new_school_years_df.isnull().mean() *100).
 ↪sort_values(ascending=False)

# rappresento
print(f"""
PERCENTUALI DATI MANCANTI:

{nan_perc}
""")
```

```
PERCENTUALI DATI MANCANTI:

Country                         0.0
year                            0.0
Expected Years of Schooling     0.0
dtype: float64
```

[ ]:
```
# verifico presenza di zeri
zero = new_school_years_df.eq(0).sum()

# rappresento
print(f"""
```

```
ZERI NEI DATI:

{zero}
""")
```

```
ZERI NEI DATI:

Country                         0
year                            0
Expected Years of Schooling     0
dtype: int64
```

6 Data Transformation

6.1 Mondo

6.1.1 Unisco i DataFrame

[ ]:
```
# unisco i dataframe

lista_DF = [WHR_df, new_freedom_df, new_burden_disease_df, new_unemployment_df,␣
 ↪new_school_years_df] # lista di DF da unire
TOT_df = lista_DF[0]
for df in lista_DF[1:]:
    TOT_df = pd.merge(TOT_df, df, on=["year", "Country"], how='outer')
```

[ ]:
```
TOT_df.sample(5)
```

[ ]:
```
     year     Country  Happiness Score  Economy (GDP per Capita)  \
273  2018     Hungary            5.620                   1.17100
321  2016     Jamaica            5.510                   0.89333
348  2018      Kosovo            5.662                   0.85500
339  2019  Kazakhstan            5.809                   1.17300
456  2016       Nepal            4.793                   0.44626

     Social Support  Health (Life Expectancy)  Freedom to make life choices  \
273         1.40100                   0.73200                       0.25900
321         0.96372                   0.59469                       0.43597
348         1.23000                   0.57800                       0.44800
339         1.50800                   0.72900                       0.41000
456         0.69699                   0.50073                       0.37012

     Generosity  Trust (Government Corruption)  Human freedom score  \
273     0.06100                        0.02200                 7.73
321     0.22245                        0.04294                 7.87
```

```
348     0.27400                         0.02300                   NaN
339     0.14600                         0.09600                  6.77
456     0.38160                         0.07008                  7.16

     Personal freedom score  Economic freedom score     DALYs  Unemployment %  \
273                    7.90                    7.49  24785.17           3.710
321                    8.07                    7.60  26891.58          13.190
348                     NaN                     NaN       NaN             NaN
339                    6.24                    7.52  32474.94           4.800
456                    7.58                    6.57  37590.09           2.868

     Expected Years of Schooling
273                    15.129460
321                    13.188587
348                          NaN
339                    15.612930
456                    12.619200
```

6.1.2 Dati mancanti

Analizzo il nuovo dataframe comprendente tutti i dati di interesse, ripuliti ed ordinati

[ ]:
```
TOT_df.info()
```

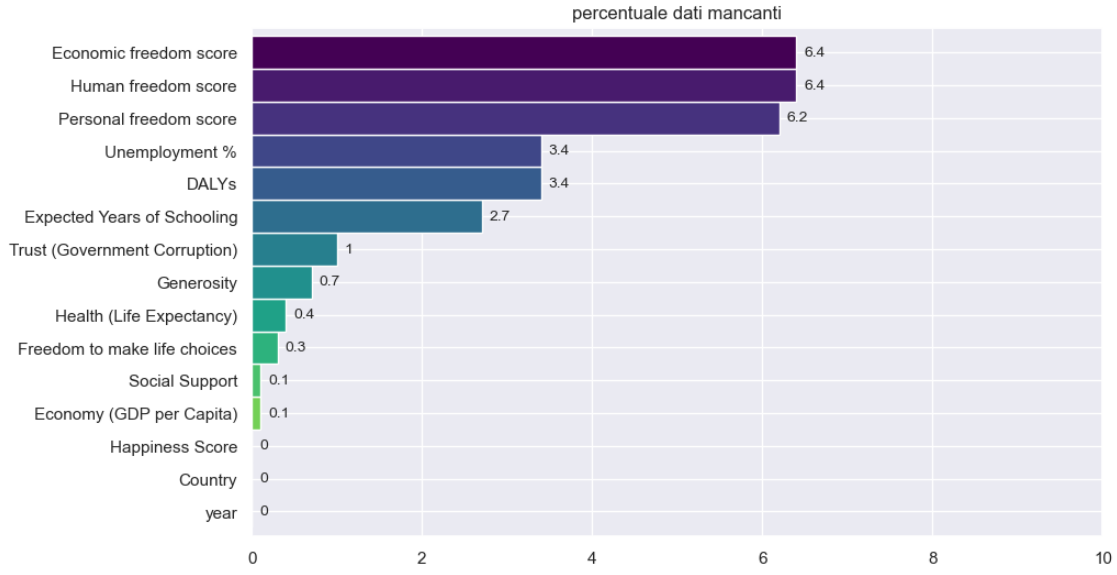
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 730 entries, 0 to 729
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   year                            730 non-null    int64
 1   Country                         730 non-null    object
 2   Happiness Score                 730 non-null    float64
 3   Economy (GDP per Capita)        729 non-null    float64
 4   Social Support                  729 non-null    float64
 5   Health (Life Expectancy)        727 non-null    float64
 6   Freedom to make life choices    728 non-null    float64
 7   Generosity                      725 non-null    float64
 8   Trust (Government Corruption)   723 non-null    float64
 9   Human freedom score             683 non-null    float64
 10  Personal freedom score          685 non-null    float64
 11  Economic freedom score          683 non-null    float64
 12  DALYs                           705 non-null    float64
 13  Unemployment %                  705 non-null    float64
 14  Expected Years of Schooling     710 non-null    float64
dtypes: float64(13), int64(1), object(1)
memory usage: 91.2+ KB
```

[ ]:
```
# percentuali dati mancanti
nan_perc = (TOT_df.isnull().mean() *100).map(lambda x: round(x,1)).
 ↪sort_values(ascending=False)

print("Percentuale di dati mancanti nel DataFrame TOT_df:")
nan_perc
```

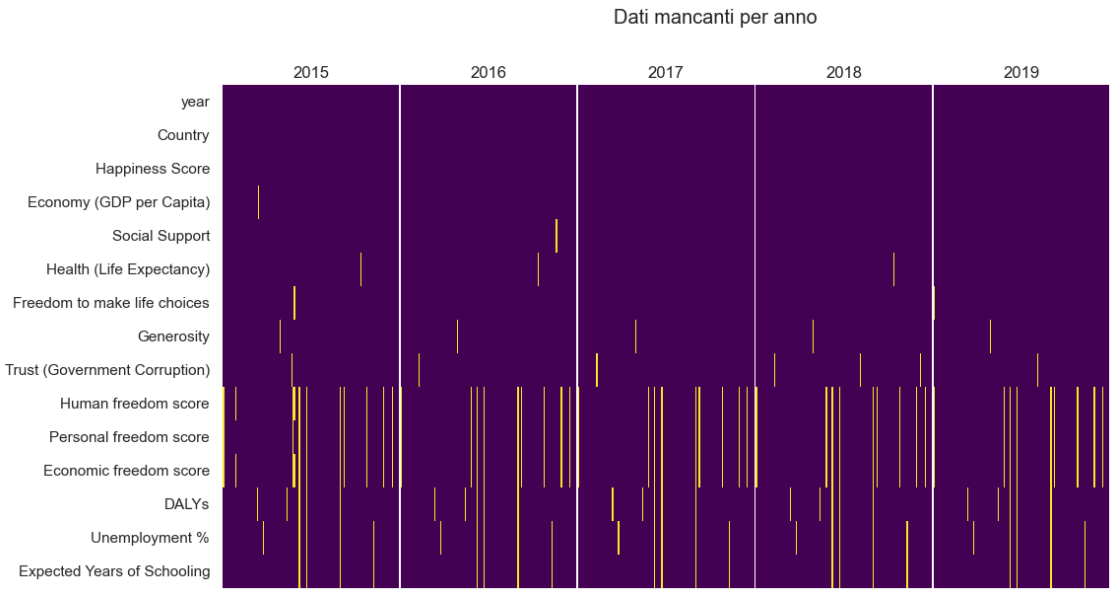
```
Percentuale di dati mancanti nel DataFrame TOT_df:
```

[ ]:
```
Human freedom score              6.4
Economic freedom score           6.4
Personal freedom score           6.2
DALYs                            3.4
Unemployment %                   3.4
Expected Years of Schooling      2.7
Trust (Government Corruption)    1.0
Generosity                       0.7
Health (Life Expectancy)         0.4
Freedom to make life choices     0.3
Economy (GDP per Capita)         0.1
Social Support                   0.1
year                             0.0
Country                          0.0
Happiness Score                  0.0
dtype: float64
```

[ ]:
```
# rappresento

# genero colori per il grafico
colors= get_colors_from_cmap("viridis_r",len(nan_perc))

# genero il grafico
plt.figure(figsize=(10,6))
ax = nan_perc.sort_values().plot(kind="barh", color=colors, width=1)
ax.set_xlim(0,10)
plt.title("percentuale dati mancanti")
ax.bar_label(ax.containers[0], padding=5, fontsize=10)

#salvo immagine
plt.savefig("immagini/percentuale_dati_mancanti.png", bbox_inches="tight")

plt.show()
```



[ ]:
```
# rappresento dati mancanti
fig, ax = plt.subplots(figsize=(10,6), ncols=5, nrows=1, sharey=True)
fig.subplots_adjust(left=0, right=0.9, bottom=0, top=0.85, wspace=0.01)
plt.suptitle("Dati mancanti per anno")

for i in range(0,5):
    sns.heatmap(TOT_df.loc[TOT_df["year"]==(2015+i)].isnull().transpose(),␣
 ↪cbar=False, cmap='viridis', xticklabels=False, ax=ax[i])
    ax[i].set_title(2015+i)

#salvo immagine
plt.savefig("immagini/dati_mancanti_per_anno.png", bbox_inches="tight")

plt.show()
```



Considerazioni Le prime 3 features: **year, Country e Happines Score** non hanno dati mancanti,

i Dtype delle colonne vanno bene, sono adatti al tipo di dato che rappresentano.

Le features con più dati mancanti sono Economic Freedom score e Human Freedom score con il 6,4% di dati mancanti. La percentuale di dati mancanti comunque non è alta e dalla figura "missing data per year" i dati mancanti sembrano essere disposti in modo casuale e senza correlazioni significative.

6.1.3 Descrizione dati

[ ]:
```
TOT_df.sample(5)
```

[ ]:
```
     year               Country  Happiness Score  Economy (GDP per Capita)  \
680  2015  United Arab Emirates            6.901                  1.427270
643  2018              Thailand            6.072                  1.016000
633  2018            Tajikistan            5.199                  0.474000
142  2017   Congo (Brazzaville)            4.291                  0.808964
622  2017                 Syria            3.462                  0.777153

     Social Support  Health (Life Expectancy)  Freedom to make life choices  \
680        1.125750                  0.809250                      0.641570
643        1.417000                  0.707000                      0.637000
633        1.166000                  0.598000                      0.292000
142        0.832044                  0.289957                      0.435026
622        0.396103                  0.500533                      0.081539

     Generosity  Trust (Government Corruption)  Human freedom score  \
```

```
680    0.264280                      0.385830                6.00
643    0.364000                      0.029000                6.56
633    0.187000                      0.034000                5.43
142    0.120852                      0.079618                5.50
622    0.493664                      0.151347                3.54

     Personal freedom score  Economic freedom score     DALYs  Unemployment %  \
680                    5.28                    7.01  30014.94           1.768
643                    6.35                    6.86  24384.93           0.770
633                    4.90                    6.18  37949.08           7.012
142                    5.74                    5.18       NaN          20.559
622                    2.46                    5.02  43624.81           8.752

     Expected Years of Schooling
680                    13.302456
643                    15.729759
633                    11.609001
142                    11.851453
622                     9.164220
```

[ ]:
```
# identifico dimensione DataFrame
TOT_df.shape
```

[ ]:
```
(730, 15)
```

[ ]:
```
TOT_df.describe()
```

[ ]:
```
              year  Happiness Score  Economy (GDP per Capita)  Social Support  \
count   730.000000       730.000000                729.000000      729.000000
mean   2017.000000         5.431170                  0.940587        1.094569
std       1.415183         1.122781                  0.398529        0.321438
min    2015.000000         2.839000                  0.015300        0.104190
25%    2016.000000         4.534250                  0.648457        0.877580
50%    2017.000000         5.427000                  1.004000        1.139350
75%    2018.000000         6.261500                  1.251420        1.351000
max    2019.000000         7.769000                  2.096000        1.644000

       Health (Life Expectancy)  Freedom to make life choices  Generosity  \
count                727.000000                    728.000000  725.000000
mean                   0.635175                      0.416054    0.220820
std                    0.234011                      0.146374    0.122931
min                    0.005565                      0.013000    0.001990
25%                    0.497931                      0.312832    0.132000
50%                    0.669000                      0.432726    0.202900
75%                    0.814495                      0.531393    0.283100
max                    1.141000                      0.724000    0.838075
```

```
       Trust (Government Corruption)  Human freedom score  \
count                     723.000000           683.000000
mean                        0.125408             7.196208
std                         0.106602             1.240608
min                         0.001000             3.490000
25%                         0.053995             6.375000
50%                         0.089283             7.210000
75%                         0.154725             8.315000
max                         0.551910             9.150000

       Personal freedom score  Economic freedom score         DALYs  \
count              685.000000              683.000000    705.000000
mean                 7.347197                6.977789  32700.151688
std                  1.586668                0.976078  13245.198847
min                  2.450000                2.670000  15045.110000
25%                  6.110000                6.310000  22832.380000
50%                  7.580000                7.140000  28416.860000
75%                  8.740000                7.750000  40025.480000
max                  9.670000                9.030000  67191.880000

       Unemployment %  Expected Years of Schooling
count      705.000000                   710.000000
mean         7.100474                    13.807085
std          5.370717                     2.961005
min          0.100000                     5.918090
25%          3.670000                    11.774572
50%          5.270000                    14.083815
75%          9.140000                    15.918522
max         28.469999                    23.088920
```

[ ]:
```
# identifico il numero degli Stati presi in esame
TOT_df["Country"].nunique()
```

[ ]:
```
145
```

Il dataframe ottenuto dopo la pulizia ed il raggruppamento è formato da **730 righe (146 paesi per 5 anni) e da 15 colonne.**

Gli anni presi in considerazione vanno **dal 2015 al 2019.**

il dataframe comprende **145 Stati.**

Happiness score ha un valore medio di 5.4 tra tutti gli Stati, il valore minimo è 2.8 ed il massimo è 7.7

6.2 Unione Europea

[ ]:
```
# creo lista con nomi degli Stati dell'UE
UE_country_list = ["Austria", "Belgium", "Bulgaria", "Croatia", "Cyprus",␣
 ↪"Czech Republic", "Denmark", "Estonia", "Finland", "France", "Germany",␣
 ↪"Greece", "Hungary", "Ireland", "Italy", "Latvia", "Lithuania",␣
 ↪"Luxembourg", "Malta", "Netherlands", "Poland", "Portugal", "Romania",␣
 ↪"Slovakia", "Slovenia", "Spain", "Sweden"]
```

[ ]:
```
# creo dataframe con Paesi UE

# creo nuovo dataframe vuoto, con solo le colonne
UE_df = TOT_df.loc[TOT_df["Country"]=="PaeseInesistente"]

# creo il dataframe UE
for country in UE_country_list:
    UE_df = pd.merge(UE_df, TOT_df.loc[TOT_df["Country"]== country],␣
 ↪how="outer")

UE_df
```

[ ]:

	year	Country	Happiness Score	Economy (GDP per Capita)	Social Support	\
0	2015	Austria	7.200	1.337230	1.297040	
1	2016	Austria	7.119	1.450380	1.083830	
2	2017	Austria	7.006	1.487097	1.459945	
3	2018	Austria	7.139	1.341000	1.504000	
4	2019	Austria	7.246	1.376000	1.475000	
..	…	…	…	…	…	
130	2015	Sweden	7.364	1.331710	1.289070	
131	2016	Sweden	7.291	1.451810	1.087640	
132	2017	Sweden	7.284	1.494387	1.478162	
133	2018	Sweden	7.314	1.355000	1.501000	
134	2019	Sweden	7.343	1.387000	1.487000	

	Health (Life Expectancy)	Freedom to make life choices	Generosity	\
0	0.890420	0.624330	0.330880	
1	0.805650	0.543550	0.328650	
2	0.815328	0.567766	0.316472	
3	0.891000	0.617000	0.242000	
4	1.016000	0.532000	0.244000	
..	…	…	…	
130	0.910870	0.659800	0.362620	
131	0.831210	0.582180	0.382540	
132	0.830875	0.612924	0.385399	
133	0.913000	0.659000	0.285000	
134	1.009000	0.574000	0.267000	

```
     Trust (Government Corruption)  Human freedom score  \
0                         0.186760                 8.78
1                         0.213480                 8.78
2                         0.221060                 8.74
3                         0.224000                 8.68
4                         0.226000                 8.67
..                             …                    …
130                       0.438440                 8.92
131                       0.408670                 8.87
132                       0.384399                 8.85
133                       0.383000                 8.84
134                       0.373000                 8.83

     Personal freedom score  Economic freedom score     DALYs  Unemployment %  \
0                      9.43                    7.88  19699.89            5.72
1                      9.47                    7.83  19512.38            6.01
2                      9.38                    7.85  19323.09            5.50
3                      9.26                    7.86  19194.66            4.85
4                      9.25                    7.86  19104.09            4.49
..                      …                       …       …               …
130                    9.63                    7.94  18329.46            7.43
131                    9.64                    7.79  18243.34            6.99
132                    9.60                    7.80  18160.94            6.72
133                    9.63                    7.74  18112.81            6.36
134                    9.63                    7.72  18069.16            6.83

     Expected Years of Schooling
0                       15.95141
1                       16.08119
2                       16.09207
3                       16.06675
4                       16.00796
..                           …
130                     18.56860
131                     18.83243
132                     19.48234
133                     19.69102
134                     19.41853

[135 rows x 15 columns]
```

il nuovo DataFrame UE_df è formato da 135 colonne (tutti i 27 Paesi europei per 5 anni)

6.3 Italia

[ ]:
```
# creo DataFrame con dati dell'Italia
ITA_df = TOT_df.loc[TOT_df["Country"]=="Italy"]
ITA_df
```

[ ]:

	year	Country	Happiness Score	Economy (GDP per Capita)	Social Support	\
310	2015	Italy	5.948	1.251140	1.197770	
311	2016	Italy	5.977	1.354950	1.041670	
312	2017	Italy	5.964	1.395067	1.444923	
313	2018	Italy	6.000	1.264000	1.501000	
314	2019	Italy	6.223	1.294000	1.488000	

	Health (Life Expectancy)	Freedom to make life choices	Generosity	\
310	0.954460	0.262360	0.22823	
311	0.851020	0.188270	0.16684	
312	0.853144	0.256451	0.17279	
313	0.946000	0.281000	0.13700	
314	1.039000	0.231000	0.15800	

	Trust (Government Corruption)	Human freedom score	\
310	0.029010	8.50	
311	0.025560	8.49	
312	0.028028	8.49	
313	0.028000	8.49	
314	0.030000	8.49	

	Personal freedom score	Economic freedom score	DALYs	Unemployment %	\
310	9.08	7.70	18344.38	11.90	
311	9.09	7.64	18151.14	11.69	
312	9.12	7.61	18107.97	11.21	
313	9.12	7.61	18152.28	10.61	
314	9.12	7.61	18185.86	9.95	

	Expected Years of Schooling
310	16.03337
311	15.97876
312	16.08890
313	16.17504
314	16.22679

6.4 Andamento HS negli anni

[ ]:
```
# creo un DataFrame che rappresenti l'andamento di HS dell'Italia dell'UE e del␣
 ↪mondo negli anni.

# raggruppo i df per anno
```

```
TOT_y_mean = pd.Series(TOT_df.groupby("year")["Happiness Score"].mean(),␣
 ↪name="HS WORLD")
UE_y_mean = pd.Series(UE_df.groupby("year")["Happiness Score"].mean(), name="HS␣
 ↪UE")
ITA_y_mean = pd.Series(ITA_df.groupby("year")["Happiness Score"].mean(),␣
 ↪name="HS ITA")

# Calcolo la deviazione standard per ogni serie
UE_y_std = pd.Series(UE_df.groupby("year")["Happiness Score"].std(), name="STD␣
 ↪UE")
TOT_y_std = pd.Series(TOT_df.groupby("year")["Happiness Score"].std(),␣
 ↪name="STD WORLD")

# genero nuovo DF con dati che mi interessano
IUT_y_df = pd.concat([ITA_y_mean, UE_y_mean, UE_y_std, TOT_y_mean, TOT_y_std],␣
 ↪axis=1)
# mostro i dati
display(IUT_y_df)
```

year	HS ITA	HS UE	STD UE	HS WORLD	STD WORLD
2015	5.948	6.135185	0.917903	5.408479	1.162995
2016	5.977	6.182963	0.863192	5.400582	1.154321
2017	5.964	6.247259	0.786144	5.410603	1.119847
2018	6.000	6.341259	0.747588	5.444685	1.103059
2019	6.223	6.444037	0.730114	5.491500	1.084625

6.4.1 Classifiche di Felicità

[ ]:
```
# classifica mondiale
WORLD_classifica = TOT_df.groupby("Country")["Happiness Score"].mean().
 ↪sort_values(ascending=False).round(2).reset_index(drop=False)
WORLD_classifica['index'] = range(1, len(WORLD_classifica) + 1)
WORLD_classifica.set_index('index', inplace=True)

WORLD_classifica
```

[ ]:

index	Country	Happiness Score
1	Denmark	7.55
2	Norway	7.54
3	Finland	7.54
4	Switzerland	7.51
5	Iceland	7.51
…	…	…
141	Afghanistan	3.51
142	Tanzania	3.47

```
143         Rwanda             3.44
144          Syria             3.29
145        Burundi             3.08

[145 rows x 2 columns]
```

[ ]:
```
# classifica UE
UE_classifica = UE_df.groupby("Country")["Happiness Score"].mean().
 ↪sort_values(ascending=False).round(2).reset_index(drop=False)
UE_classifica['index'] = range(1, len(UE_classifica) + 1)
UE_classifica.set_index('index', inplace=True)

UE_classifica
```

[ ]:

index	Country	Happiness Score
1	Denmark	7.55
2	Finland	7.54
3	Netherlands	7.40
4	Sweden	7.32
5	Austria	7.14
6	Ireland	6.96
7	Luxembourg	6.94
8	Germany	6.93
9	Belgium	6.92
10	Czech Republic	6.65
11	Malta	6.53
12	France	6.52
13	Spain	6.35
14	Slovakia	6.11
15	Italy	6.02
16	Poland	5.98
17	Lithuania	5.93
18	Slovenia	5.89
19	Cyprus	5.73
20	Romania	5.70
21	Latvia	5.68
22	Estonia	5.64
23	Croatia	5.46
24	Hungary	5.33
25	Portugal	5.30
26	Greece	5.15
27	Bulgaria	4.62

6.4.2 Mappa della Felicità

- prendo le coordinate dei confini degli Stati per poter disegnare la mappa.

- aggiusto i nomi degli Stati.
- preparo il file per rappresentare la mappa.

[ ]:
```
#converto lista nomi degli Stati in set
country_set = set(evry_year_country)

#carico il file geojson con coordinate dei confini degli Stati
with open("Data/countries.geojson") as f:
    data_stati = geojson.load(f)

# identifico Stati che non combaciano
not_matching_names = set(map(lambda x: x['properties']['ADMIN'],␣
 ↪data_stati['features'])) - country_set

#cerco somiglianze tra i Paesi che non combaciano
for C in not_matching_names:
    #cerca una corrispondenza simile
    match = difflib.get_close_matches(C, country_set, n=4, cutoff=0.7)
    if match:
        print(f"Possibile somiglianza tra: {C} e {match}")

    # cerco i Paesi che hanno le prime 5 lettere uguali
    for c in country_set:
        if C[0:5] == c[0:5]: # se hanno le prime 5 lettere uguali
            print(f"Iniziano con le stesse lettere: {C} e {c}")
```

```
Possibile somiglianza tra: Swaziland e ['Thailand', 'Switzerland']
Iniziano con le stesse lettere: Dominica e Dominican Republic
Possibile somiglianza tra: Namibia e ['Zambia']
Possibile somiglianza tra: United States of America e ['United States']
Iniziano con le stesse lettere: United States of America e United States
Iniziano con le stesse lettere: United States of America e United Arab Emirates
Iniziano con le stesse lettere: United States of America e United Kingdom
Iniziano con le stesse lettere: Indian Ocean Territories e India
Possibile somiglianza tra: Northern Cyprus e ['North Cyprus']
Iniziano con le stesse lettere: Northern Cyprus e North Cyprus
Iniziano con le stesse lettere: United Republic of Tanzania e United States
Iniziano con le stesse lettere: United Republic of Tanzania e United Arab
Emirates
Iniziano con le stesse lettere: United Republic of Tanzania e United Kingdom
Iniziano con le stesse lettere: South Georgia and South Sandwich Islands e South
Korea
Iniziano con le stesse lettere: South Georgia and South Sandwich Islands e South
Africa
Possibile somiglianza tra: Aland e ['Poland']
Possibile somiglianza tra: North Korea e ['South Korea']
Iniziano con le stesse lettere: North Korea e North Cyprus
```

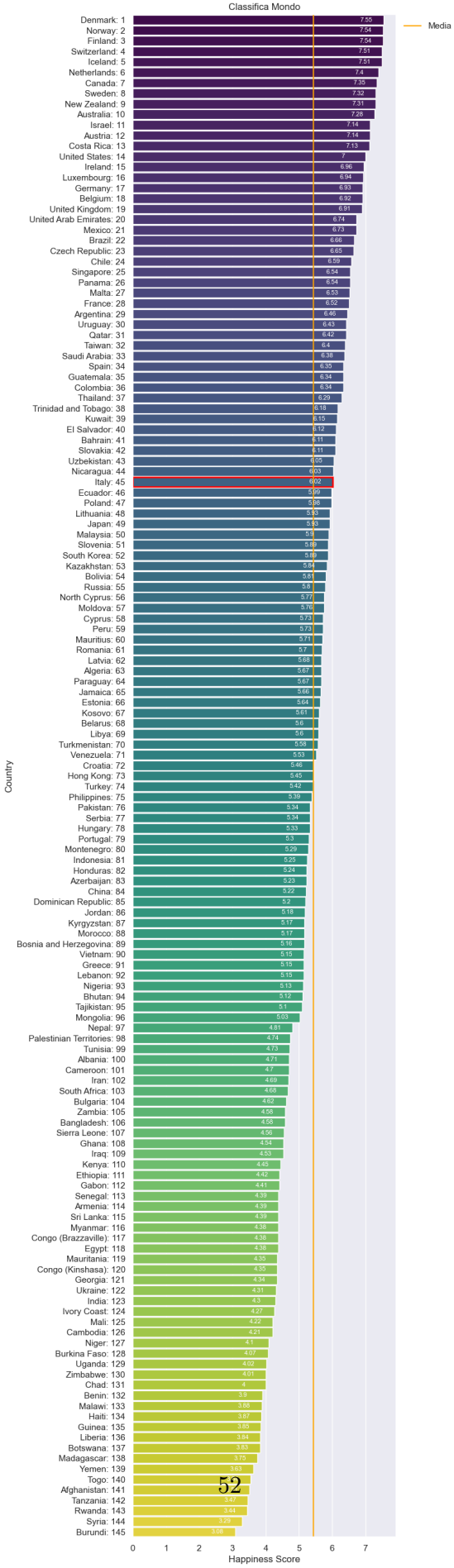
```
Possibile somiglianza tra: Guyana e ['Ghana']
Iniziano con le stesse lettere: South Sudan e South Korea
Iniziano con le stesse lettere: South Sudan e South Africa
Iniziano con le stesse lettere: Northern Mariana Islands e North Cyprus
Iniziano con le stesse lettere: United States Virgin Islands e United States
Iniziano con le stesse lettere: United States Virgin Islands e United Arab
Emirates
Iniziano con le stesse lettere: United States Virgin Islands e United Kingdom
Possibile somiglianza tra: Somalia e ['Romania']
Possibile somiglianza tra: Angola e ['Mongolia']
Iniziano con le stesse lettere: Guinea Bissau e Guinea
Iniziano con le stesse lettere: United States Minor Outlying Islands e United
States
Iniziano con le stesse lettere: United States Minor Outlying Islands e United
Arab Emirates
Iniziano con le stesse lettere: United States Minor Outlying Islands e United
Kingdom
Possibile somiglianza tra: Gambia e ['Zambia', 'Cambodia']
Possibile somiglianza tra: Brunei e ['Burundi']
Iniziano con le stesse lettere: Palestine e Palestinian Territories
Possibile somiglianza tra: Greenland e ['Ireland']
Possibile somiglianza tra: Hong Kong S.A.R. e ['Hong Kong']
Iniziano con le stesse lettere: Hong Kong S.A.R. e Hong Kong
Iniziano con le stesse lettere: Cyprus No Mans Area e Cyprus
```

[ ]:
```python
# Crea un dizionario per i nomi da sostituire
replace_dict = {"United States of America": "United States",
                "Hong Kong S.A.R.": "Hong Kong",
                "Northern Cyprus": "North Cyprus",
                "Dominica": "Dominican Republic",
                "Palestine": "Palestinian Territories"}

#itero sulle feature del file geojson
for feature in data_stati['features']:
    name = feature['properties']['ADMIN']
    new_name = replace_dict.get(name, name)
    feature['properties']['ADMIN'] = new_name

#salvo il file modificato
with open("Data/countries_mod.geojson", "w") as f:
    geojson.dump(data_stati, f)
```

7 Data Visualization

7.0.1 Classifica Mondo

[ ]:
```
#grafico della classifica di HS nel mondo

#imposto i colori
colors = get_colors_from_cmap("viridis", WORLD_classifica.shape[0])

# imposto grafico
fig, ax = plt.subplots(figsize=(6,35))
plt.title("Classifica Mondo")
plt.xlabel("HS")

# genero grafico e imposto etichette barre
sns.barplot(data=WORLD_classifica, orient="h", x="Happiness Score",␣
 ↪y="Country", width=0.9, palette=colors, ax=ax)
ax.bar_label(ax.containers[0], padding=-30, fontsize=8, color="w") # inserisco␣
 ↪valori HS sulle barre

# creo yticks con numero classifica e nome Stato
yticklabels = []
for i, country in enumerate(WORLD_classifica["Country"]):
    yticklabels.append(f"{country}: {i+1}")
ax.set_yticklabels(yticklabels)

# evidenzio l'Italia
italy_index = WORLD_classifica[WORLD_classifica['Country'] == 'Italy'].
 ↪index[0]-1
ax.containers[0][italy_index].set_edgecolor("red")
ax.containers[0][italy_index].set_linewidth(2)

# aggiungo linea della media
ax.axvline(WORLD_classifica["Happiness Score"].mean(), color="orange",␣
 ↪label="Media")

# gestisco legenda
plt.legend(bbox_to_anchor=(1,1))

#salvo immagine
plt.savefig("immagini/classifica_mondo_hs.png", bbox_inches="tight")

plt.show()
```


Classifica Mondo
Media
Denmark: 1 7.55
Norway: 2 7.54
Finland: 3 7.54
Switzerland: 4 7.51
Iceland: 5 7.51
Netherlands: 6 7.4
Canada: 7 7.35
Sweden: 8 7.32
New Zealand: 9 7.31
Australia: 10 7.28
Israel: 11 7.14
Austria: 12 7.14
Costa Rica: 13 7.13
United States: 14 7
Ireland: 15 6.96
Luxembourg: 16 6.94
Germany: 17 6.93
Belgium: 18 6.92
United Kingdom: 19 6.91
United Arab Emirates: 20 6.74
Mexico: 21 6.73
Brazil: 22 6.66
Czech Republic: 23 6.65
Chile: 24 6.59
Singapore: 25 6.54
Panama: 26 6.54
Malta: 27 6.53
France: 28 6.52
Argentina: 29 6.46
Uruguay: 30 6.43
Qatar: 31 6.42
Taiwan: 32 6.4
Saudi Arabia: 33 6.38
Spain: 34 6.35
Guatemala: 35 6.34
Colombia: 36 6.34
Thailand: 37 6.29
Trinidad and Tobago: 38 6.18
Kuwait: 39 6.15
El Salvador: 40 6.12
Bahrain: 41 6.11
Slovakia: 42 6.11
Uzbekistan: 43 6.05
Nicaragua: 44 6.03
Italy: 45 6.02
Ecuador: 46 5.99
Poland: 47 5.98
Lithuania: 48 5.93
Japan: 49 5.93
Malaysia: 50 5.9
Slovenia: 51 5.89
South Korea: 52 5.89
Kazakhstan: 53 5.84
Bolivia: 54 5.81
Russia: 55 5.8
North Cyprus: 56 5.77
Moldova: 57 5.76
Cyprus: 58 5.73
Peru: 59 5.73
Mauritius: 60 5.71
Romania: 61 5.7
Latvia: 62 5.68
Algeria: 63 5.67
Paraguay: 64 5.67
Jamaica: 65 5.66
Estonia: 66 5.64
Kosovo: 67 5.61
Belarus: 68 5.6
Libya: 69 5.6
Turkmenistan: 70 5.58
Venezuela: 71 5.53
Croatia: 72 5.46
Hong Kong: 73 5.45
Turkey: 74 5.42
Philippines: 75 5.39
Pakistan: 76 5.34
Serbia: 77 5.34
Hungary: 78 5.33
Portugal: 79 5.3
Montenegro: 80 5.29
Indonesia: 81 5.25
Honduras: 82 5.24
Azerbaijan: 83 5.23
China: 84 5.22
Dominican Republic: 85 5.2
Jordan: 86 5.18
Kyrgyzstan: 87 5.17
Morocco: 88 5.17
Bosnia and Herzegovina: 89 5.16
Vietnam: 90 5.15
Greece: 91 5.15
Lebanon: 92 5.15
Nigeria: 93 5.13
Bhutan: 94 5.12
Tajikistan: 95 5.1
Mongolia: 96 5.03
Nepal: 97 4.81
Palestinian Territories: 98 4.74
Tunisia: 99 4.73
Albania: 100 4.71
Cameroon: 101 4.7
Iran: 102 4.69
South Africa: 103 4.68
Bulgaria: 104 4.62
Zambia: 105 4.58
Bangladesh: 106 4.58
Sierra Leone: 107 4.56
Ghana: 108 4.54
Iraq: 109 4.53
Kenya: 110 4.45
Ethiopia: 111 4.42
Gabon: 112 4.41
Senegal: 113 4.39
Armenia: 114 4.39
Sri Lanka: 115 4.39
Myanmar: 116 4.38
Congo (Brazzaville): 117 4.38
Egypt: 118 4.38
Mauritania: 119 4.35
Congo (Kinshasa): 120 4.35
Georgia: 121 4.34
Ukraine: 122 4.31
India: 123 4.3
Ivory Coast: 124 4.27
Mali: 125 4.22
Cambodia: 126 4.21
Niger: 127 4.1
Burkina Faso: 128 4.07
Uganda: 129 4.02
Zimbabwe: 130 4.01
Chad: 131 4
Benin: 132 3.9
Malawi: 133 3.88
Haiti: 134 3.87
Guinea: 135 3.85
Liberia: 136 3.84
Botswana: 137 3.83
Madagascar: 138 3.75
Yemen: 139 3.63
Togo: 140 3.51
Afghanistan: 141 3.47
Tanzania: 142 3.47
Rwanda: 143 3.44
Syria: 144 3.29
Burundi: 145 3.08
Country
0 1 2 3 4 5 6 7
Happiness Score

Il grafico soprastante mostra la classifica degli Stati del mondo per punteggio Happiness Score.

I primi Paesi per felicità sono la Danimarca, Norvegia e la Finlandia ed i Paesi Bassi con rispettivamente un punteggio di 7.55, 7.54 e 7.54.

I Paesi più infelici del mondo sono il Burundi, la Syria ed il Rwanda con un punteggio rispettivamente di 3.08, 3,29 e 3.44.

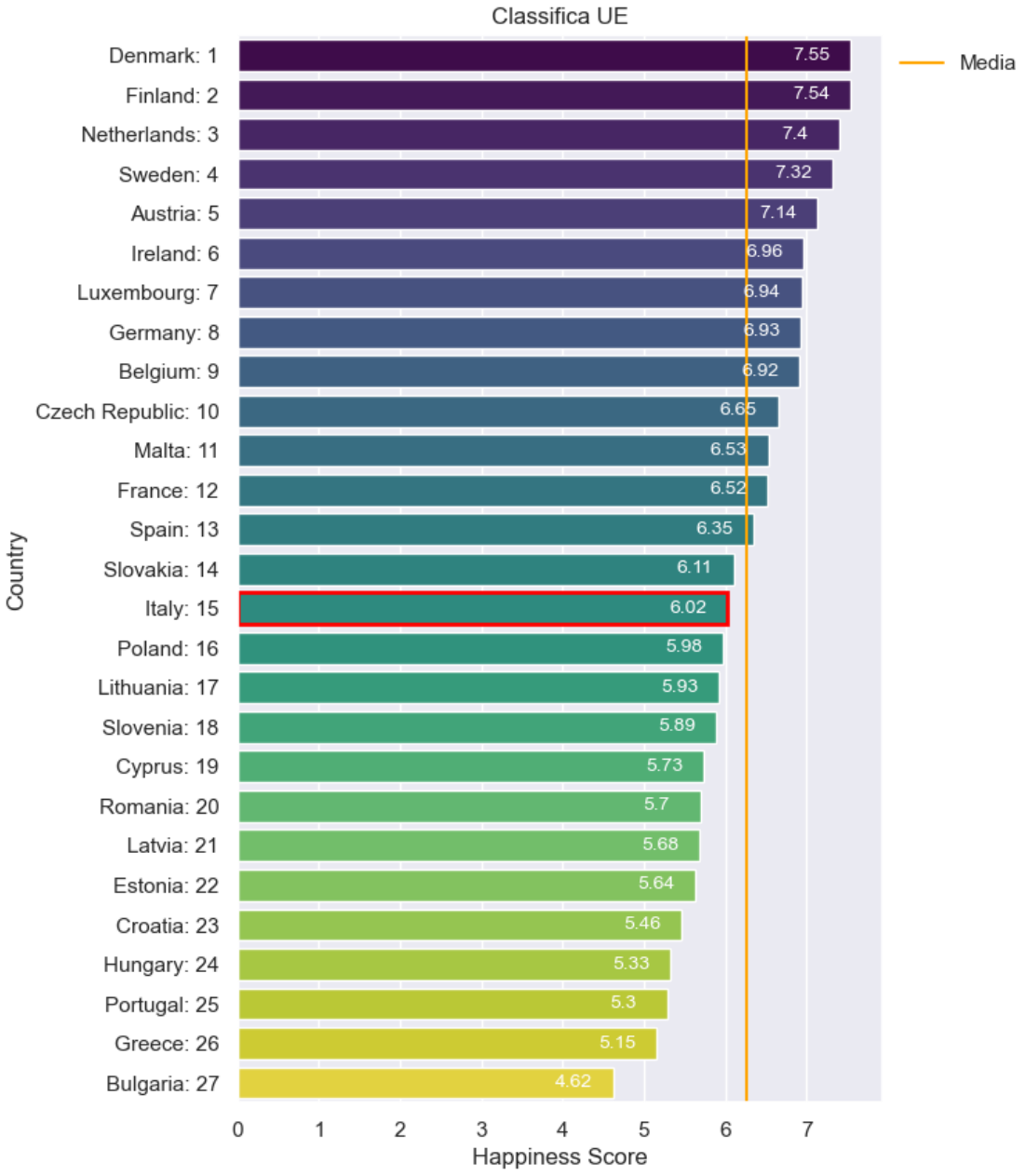
La media di Happiness Score è di 5.4.

L'Italia si trova in 45° posizione con una media di 6.02, sopra alla media globale.

7.0.2 Classifica UE

[ ]:
```
#grafico della classifica di HS in Unione Europea

#imposto i colori
colors = get_colors_from_cmap("viridis", UE_classifica.shape[0])

# imposto grafico
fig, ax = plt.subplots(figsize=(6,10))
plt.title("Classifica UE")
plt.xlabel("HS")

# creo grafico ed etichette
sns.barplot(data=UE_classifica, orient="h", x="Happiness Score", y="Country",␣
 ↪palette=colors, ax=ax)
ax.bar_label(ax.containers[0], padding=-30, fontsize=10, color="w") # inserisco␣
 ↪valori HS sulle barre

# creo yticks con numero classifica e nome stato
yticklabels = []
for i, country in enumerate(UE_classifica["Country"]):
    yticklabels.append(f"{country}: {i+1}")
ax.set_yticklabels(yticklabels)

# evidenzio l'Italia
italy_index = UE_classifica[UE_classifica['Country'] == 'Italy'].index[0]-1
ax.containers[0][italy_index].set_edgecolor("red")
ax.containers[0][italy_index].set_linewidth(2)
# aggiungo line della media
ax.axvline(UE_classifica["Happiness Score"].mean(), color="orange",␣
 ↪label="Media")

plt.legend(bbox_to_anchor=(1,1))

#salvo immagine
```

```
plt.savefig("immagini/classifica_UE_hs.png", bbox_inches="tight")

plt.show()
```



Il grafico soprastante mostra la classifica degli Stati UE per punteggio Happiness Score.

I primi Paesi per felicità sono la Danimarca, la Finlandia ed i Paesi Bassi con rispettivamente un punteggio di 7.55, 7.54 e 7.40.

I Paesi più infelici dell'Unione Europea sono la Bulgaria, la Grecia ed il Portogallo con

un punteggio rispettivamente di 4.62, 5.15 e 5.3.

La media di Happiness Score in Unione Europea è di 6.27.

L'Italia si trova in 15° posizione con una media di 6.02

7.0.3 Mappa della Felicità

Mondo

[ ]:
```
# genero la mappa Mondo
mappaMondo = folium.Map(location=[43,12],
                    zoom_start=1.5,
                    tiles="Stamen Watercolor",
                    width='100%',
                    height='80%',
                    no_touch=True,
                    zoom_control=False)

# aggiungo i confini
folium.GeoJson("Data/countries_mod.geojson").add_to(mappaMondo)

# coloro in base a Happiness Score
folium.Choropleth(
    geo_data=data_stati,
    data=WORLD_classifica,
    columns=("Country", "Happiness Score"),
    key_on="feature.properties.ADMIN",
    bins=20,
    fill_color="viridis",
    nan_fill_color='black',
    fill_opacity=0.8,
    line_color='black',
    line_weight=1,
    line_opacity=1,
    legend_name='Happiness Score').add_to(mappaMondo)

#salvo immagine
mappaMondo.save("immagini/mappa_mondo_hs.html")

mappaMondo
```

[ ]:
```
<folium.folium.Map at 0x7fcbdd989a90>
```

La mappa mostra il **valore di HS nei diversi Stati del mondo** (media dei 5 anni presi in considerazione).

Si può notare come **i Paesi del nord Europa e del nord America siano i più felici insieme all'Australia.**

I Paesi con indice di felicità più basso invece sono distribuiti principalmente nel con-

tinente africano e nel sud-ovest asiatico.

Unione Europea

[ ]:
```
# genero la mappa UE
mappaUE = folium.Map(location=[55,15],
                     zoom_start=3.4,
                     tiles="Stamen Watercolor",
                     width='100%',
                     height='80%',
                     no_touch=True,
                     zoom_control=False)

# aggiungo i confini
folium.GeoJson("Data/countries_mod.geojson").add_to(mappaUE)

# coloro in base a Happiness Score
folium.Choropleth(
    geo_data=data_stati,
    data=UE_classifica,
    columns=("Country", "Happiness Score"),
    key_on="feature.properties.ADMIN",
    bins=20,
    fill_color="viridis",
    nan_fill_color='black',
    fill_opacity=0.8,
    line_color='black',
    line_weight=1,
    line_opacity=1,
    legend_name='Happiness Score').add_to(mappaUE)

#salvo immagine
mappaUE.save("immagini/mappa_UE_hs.html")

mappaUE
```

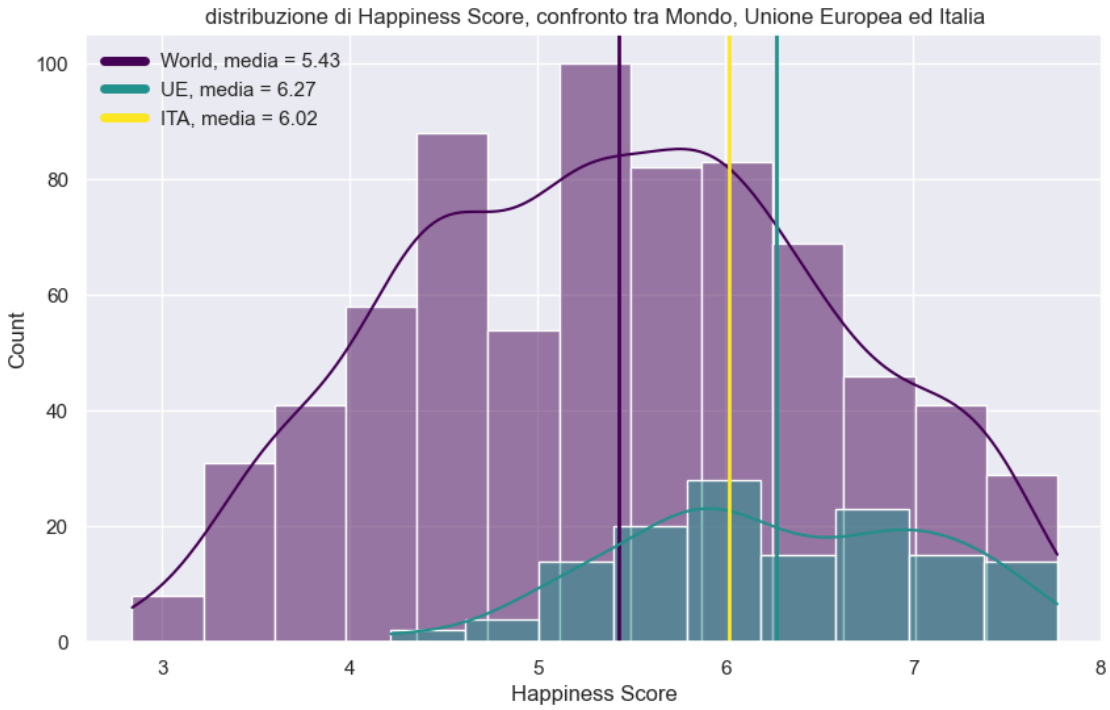
[ ]:
```
<folium.folium.Map at 0x7fcbf57a51c0>
```

Per quanto riguarda l'**Unione Europea** si può osservare che **i Paesi con HS più alto sono quelli del nord (Danimarca, Svezia, Finlandia...)** mentre **i più infelici quelli del sud-est (Grecia, Ungheria, Bulgaria) ed il Portogallo.**

7.0.4 Distribuzione HS

[ ]:
```
# metto a confronto le distribuzioni ed i valori di HS
plt.figure(figsize=(10,6))
plt.title("distribuzione di Happiness Score, confronto tra Mondo, Unione␣
 ↪Europea ed Italia")
```

```
colors = get_colors_from_cmap("viridis",3) #identifico i colori del grafico

# distribuzione e media globale
TOT_media = np.round(TOT_df["Happiness Score"].mean(),2)
sns.histplot(data=TOT_df["Happiness Score"], kde=True, color=colors[0])
plt.axvline(TOT_media, label=f'World, media = {TOT_media}', color=colors[0],␣
 ↪linewidth=2)

# distribuzione e media europea
UE_media = np.round(UE_df["Happiness Score"].mean(),2)
sns.histplot(data=UE_df["Happiness Score"], kde=True, color=colors[1])
plt.axvline(UE_media, label=f'UE, media = {UE_media}', color=colors[1],␣
 ↪linewidth=2)

# media italiana
ITA_media = np.round(ITA_df["Happiness Score"].mean(),2)
plt.axvline(ITA_media, label=f'ITA, media = {ITA_media}', color=colors[2],␣
 ↪linewidth=2)

# creo la legenda e modifico settaggi
legend = plt.legend(loc=2)
for line in legend.get_lines():
    line.set_linewidth(5)

#salvo immagine
plt.savefig("immagini/distribuzione_hs.png", bbox_inches="tight")

plt.show()
```



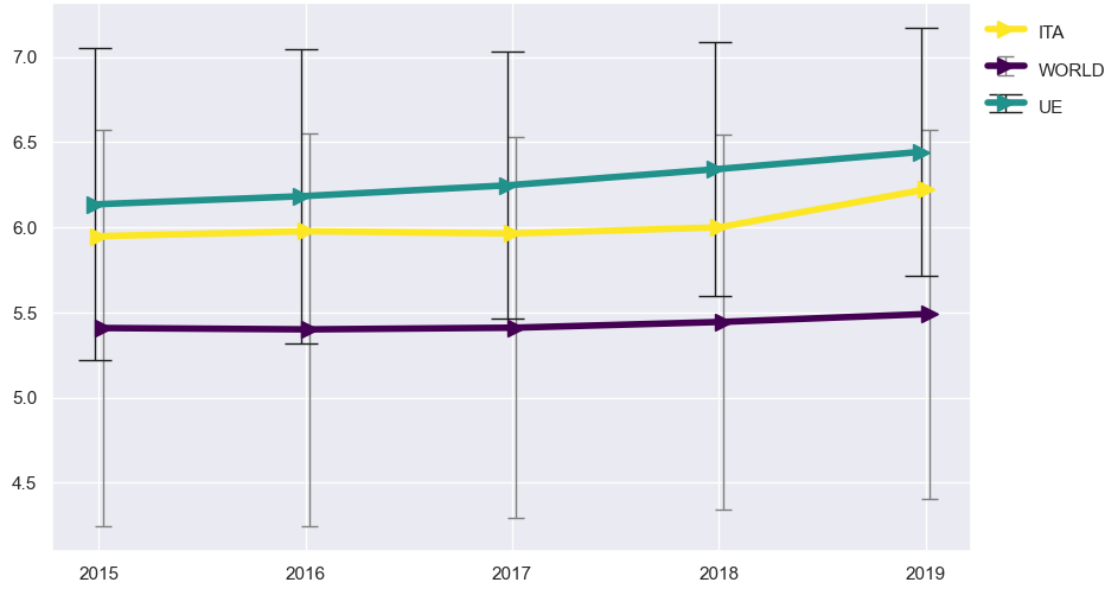
Dal grafico delle distribuzioni si nota come i **Paesi dell'UE abbiano mediamente un valore di HS più elevato rispetto al resto del mondo**. I punteggi di HS in UE infatti vanno da 4.5 a 7.5 circa con media 6.27 rispetto ai dati di HS globali che partano da un valore di circa 2.5 con una media di 5.4.

Si nota inoltre che il valore medio di **HS in Italia è di 6.02, maggiore della media globale ma minore della media europea.**

7.0.5 Andamento HS negli anni

[ ]:
```
#genero il grafico
colors = get_colors_from_cmap("viridis", 3) # imposto i colori
fig, ax = plt.subplots(figsize=(10,6))

# Disegnare andamento HS nel tempo e deviazione std
ax.errorbar(TOT_y_mean.index+0.02, TOT_y_mean.values, yerr=TOT_y_std.values,
            fmt='->', color=colors[0], markersize=10, linewidth=4,␣
 ↪ecolor='gray',
            elinewidth=1, capsize=5, label="WORLD") #mondo
ax.errorbar(UE_y_mean.index-0.02, UE_y_mean.values, yerr=UE_y_std.values,
            fmt='->', color=colors[1], markersize=10, linewidth=4, ecolor='k',
            elinewidth=1, capsize=10, label="UE") #UE
ax.plot(ITA_y_mean, color=colors[2], linewidth=4, marker=">", markersize=10,
        label="ITA") #ITA
ax.set_xticks([2015,2016,2017,2018,2019])
```

```
plt.legend(bbox_to_anchor=(1,1),handleheight=2)

#salvo immagine
plt.savefig("immagini/andamento_negli_anni_hs.png", bbox_inches="tight")

plt.show()
```



Si può osservare un **andamento crescente di tutte e tre le medie di HS**. La media globale incrementa di poco il proprio valore negli anni, solo 0.1 punti. L'Unione Europea e l'Italia incrementano leggermente di più, aumentando di 0.3 punti in 5 anni.

Si osserva inoltre che **la media di HS dell'UE rimane la più alta delle tre per tutti i 5 anni**, seguita dall'Italia ed infine dal mondo.

Si osserva che la media di **HS in Italia è sempre più bassa della media Europea ma più alta della media globale.

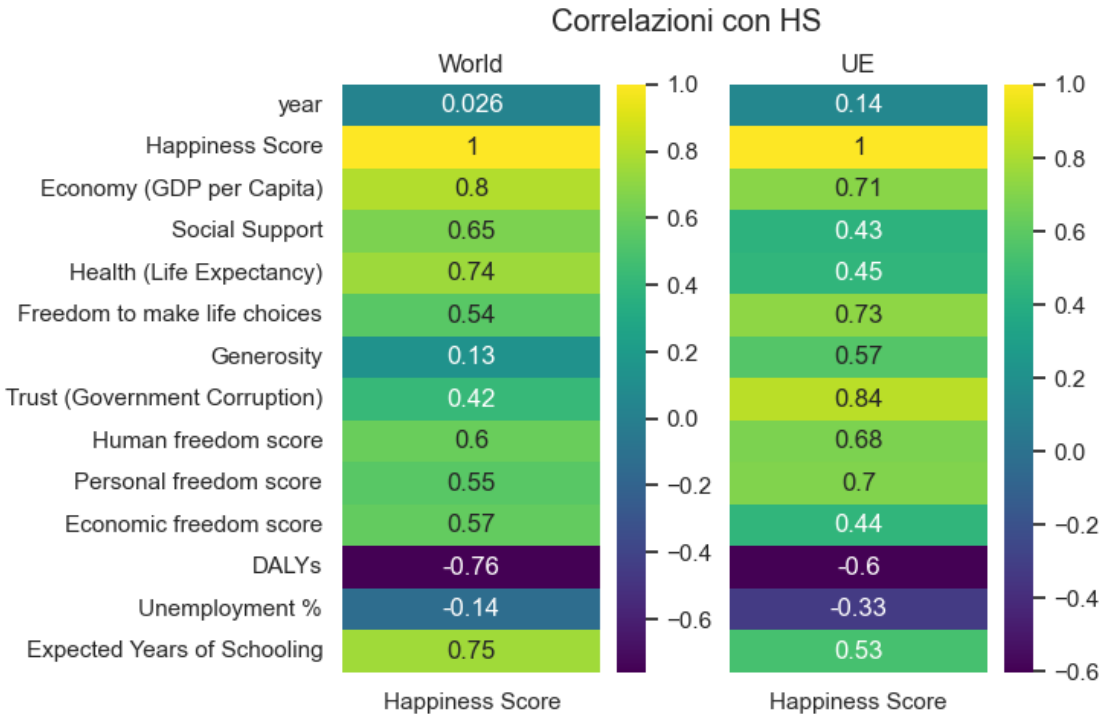
Rispetto al contesto socio-economico in cui il nostro Paese vive ed agisce (UE) l'Italia risulta avere un indice HS più basso.

Proviamo a capire se ci sono parametri correlati ad HS che l'Italia potrebbe migliorare.

7.0.6 Correlazioni con HS

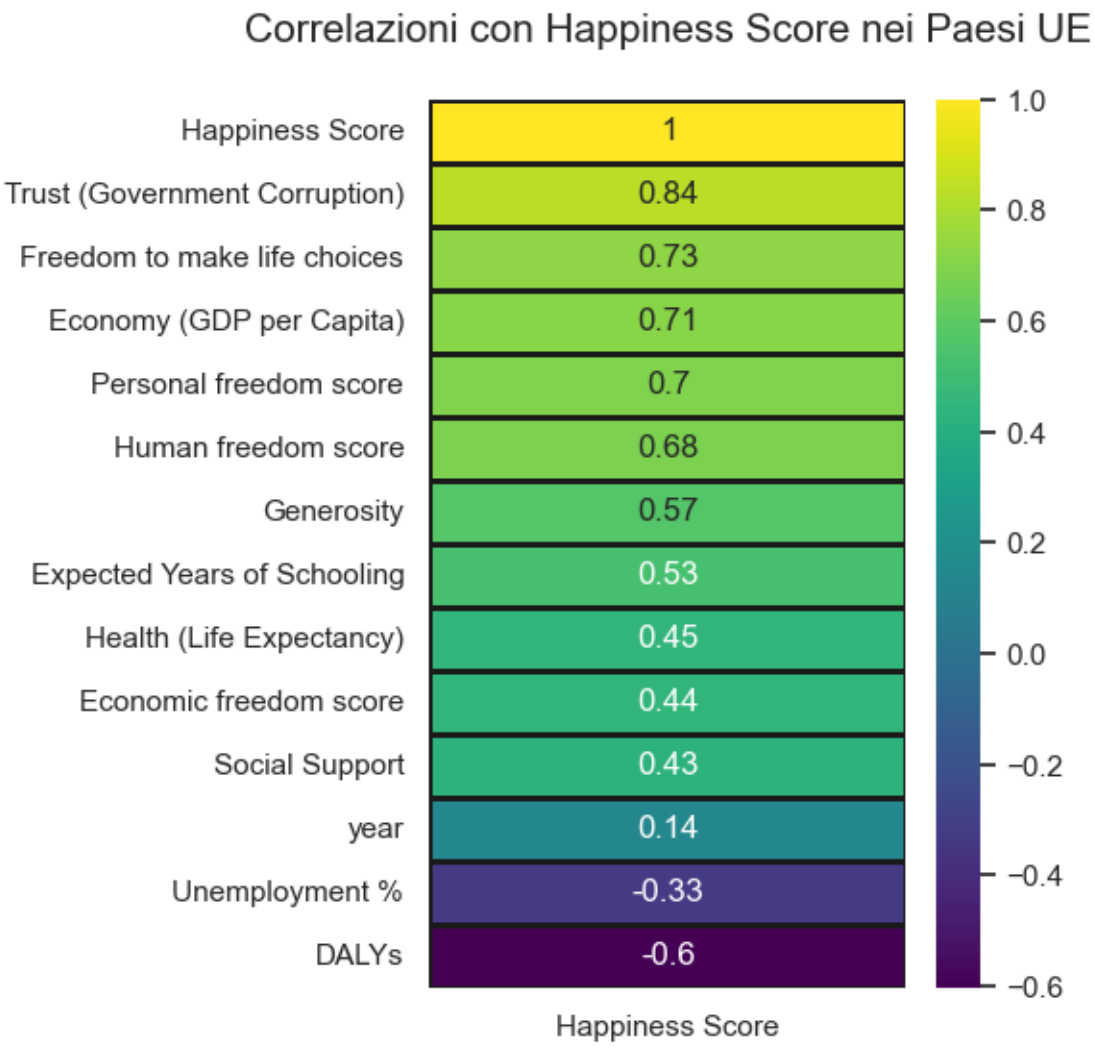
[ ]:
```
# cerco correlazioni tra le colonne e HS per identificare le features più␣
↪importanti su cui poter agire

fig, axs = plt.subplots(1, 2, figsize=(6, 5), sharey=True)
```

```
fig.suptitle("Correlazioni con HS")
axs[0].set_title("World")
axs[1].set_title("UE")

# Calcolo la matrice di correlazione per ogni DataFrame
TOT_corr = TOT_df.corr(numeric_only=True)
UE_corr = UE_df.corr(numeric_only=True)

# Seleziono la colonna 'Happiness Score'
score_corr_TOT = TOT_corr['Happiness Score']
score_corr_UE = UE_corr['Happiness Score']

# Disegno il grafico di heatmap per ogni DataFrame
sns.heatmap(score_corr_TOT.to_frame(), annot=True, cmap="viridis", ax=axs[0])
sns.heatmap(score_corr_UE.to_frame(), annot=True, cmap="viridis", ax=axs[1])

#salvo immagine
plt.savefig("immagini/correlazioni_mondo-UE_hs.png", bbox_inches="tight")

plt.show()
```



[ ]:
```
# grafico solo UE ordinato per correlazione
```

```
# ordino
sorted_score_corr_UE = score_corr_UE.sort_values(ascending=False)
# imposto grafico
fig, ax = plt.subplots(figsize=(4,6))
plt.title("Correlazioni con Happiness Score nei Paesi UE\n",␣
 ↪fontdict={"fontsize":15})
sns.heatmap(sorted_score_corr_UE.to_frame(), annot=True,␣
 ↪cmap="viridis",linewidths=1, linecolor='k', ax=ax)

#salvo immagine
plt.savefig("immagini/correlazioni_UE_sorted.png", bbox_inches="tight")

plt.show()
```



I valori ed i colori nella tabella soprastante rappresentano il **grado di correlazione tra le varie**

colonne del DataFrame e HS. Valori negativi indicano una **correlazione inversa** con grado che aumenta da 0 a -1. **Valori positivi** indicano invece una **correlazione di tipo diretto** con grado che aumenta da 0 a 1.

I grafici riguardanti il mondo, a sinistra, e gli Stati dell'Unione Europea, a destra, sono affiancati.

Le due heatmap sono simili ma non uguali, sono infatti presenti delle differenze importanti per questa analisi.

Queste differenze fanno capire che in base alla zona del mondo, alla situazione socio-economica e culturale, cambia il modo in cui un determinato parametro influenza il benessere dei cittadini.

Di conseguenza risulta fondamentale capire il contesto socio-politico-culturale del Paese per poter attuare azioni politiche mirate all'incremento della felicità dei cittadini.

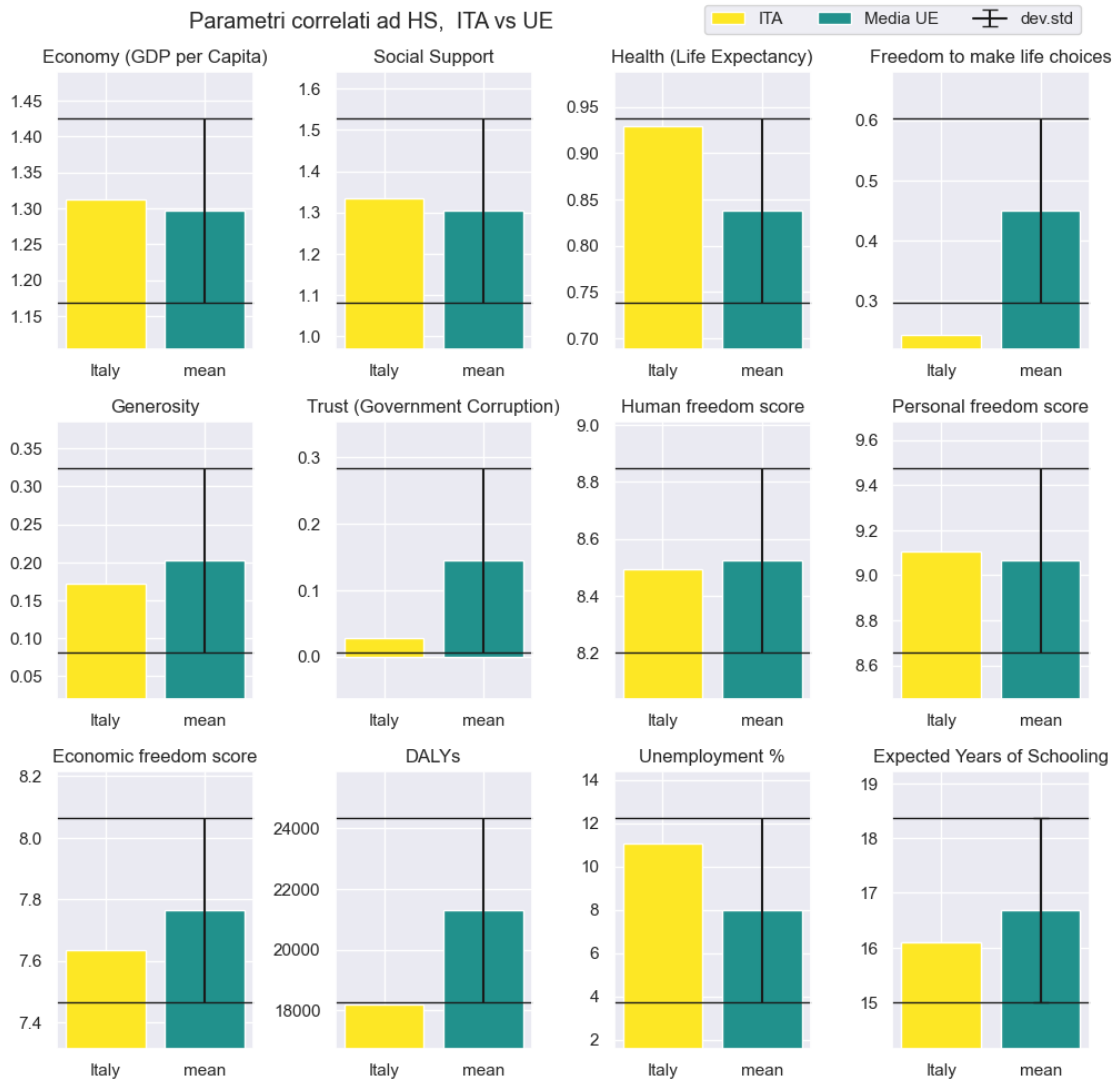
Per questo motivo confrontare i dati dell'Italia con quelli dell'Unione Europea, invece che con quelli mondiali, risulta più appropriato in quanto il contesto sociale, politico ed economico è più simile.

Analizzando il grafico riguardante l'Unione Europea si osserva che: - Le uniche colonne **correlate negativamente con HS sono Burden disease(DALYs) e Unemployment%, rispettivmente con valori di -0.6 e -0.33.** - la colonna con **indice di correlazione più alto è Trust (Government Corruption)** che indica il grado di fiducia nelle istituzioni da parte dei cittadini, con un valore di **0.84** - la colonna con la minore correlazione è **year**, che sta ad indicare un **lieve incremento di HS negli anni** ma non molto significativo - **tutte le altre colonne sono correlate positivamente con un indice tra lo 0.43 e lo 0.73**

7.0.7 Parametri correlati ad HS –> ITA vs UE

[ ]:
```
#imposto i colori
colors = get_colors_from_cmap("viridis", 3)

# Calcolo media e std
mean_df = UE_df.groupby('Country').mean(numeric_only=True) # media per ogni␣
 ↪stato
std_df = UE_df.groupby('Country').std(numeric_only=True) # std per ogni stato
mean_all = UE_df.mean(numeric_only=True) # media totale
std_all = UE_df.std(numeric_only=True) # std totale

# imposto il grafico
fig, axs = plt.subplots(nrows=3, ncols=4, figsize=(10, 10))
axs=axs.ravel()

#itero attraverso le colonne del DataFrame per creare i diversi assi
for idx, column in enumerate(UE_df.columns[3:]):
    axs[idx].bar(mean_df.loc['Italy'].name, mean_df.loc['Italy'][column],
                 color=colors[2], label="ITA") # barra ITA
    axs[idx].bar("mean", mean_all[column], color=colors[1], label="Media UE") #␣
 ↪barra UE
```

```
    axs[idx].errorbar("mean", mean_all[column], std_all[column],
                      fmt='', capsize=150, color='k', label="dev.std") # error␣
 ↪bar
    axs[idx].set_title(column)
    axs[idx].set_ylim(mean_all[column]-(1.5*std_all[column]),
                      mean_all[column]+(1.5*std_all[column])) # regolo ylim in␣
 ↪base a std

plt.suptitle("Parametri correlati ad HS,  ITA vs UE",␣
 ↪horizontalalignment="right")
plt.tight_layout()

# imposto la legenda
dev_std_handle = axs[idx].errorbar("mean", mean_all[column], std_all[column],
                      fmt='', capsize=5, color='k', label="dev.std") # handle␣
 ↪dev std
ITA_handle = axs[idx].bar(mean_df.loc['Italy'].name, mean_df.
 ↪loc['Italy'][column],
                color=colors[2], label="ITA") # handle ITA
UE_handle = axs[idx].bar("mean", mean_all[column], color=colors[1],␣
 ↪label="Media UE") # handle UE

plt.legend(handles=[ITA_handle, UE_handle, dev_std_handle], bbox_to_anchor=(1,␣
 ↪3.8),ncols=4, frameon=True)

#salvo immagine
plt.savefig("immagini/parametri_correlati_ITA_vs_UE.png", bbox_inches="tight")

plt.show()
```



I grafici soprastanti mettono in evidenza la differenza tra i valori correlati ad HS dell'Italia e della media in UE.

Le differenze più significative riguardano: la libertà di fare scelte di vita ed il carico di malattia, che hanno una differenza maggiore ad una deviazione standard.

La libertà di fare scelte di vita , che ha una correlazione con HS di **0.73, risulta minore in Italia rispetto alla media UE. Questo indica come incida fortemente sulla felicità della popolazione ed è un aspetto su cui focalizzare l'attenzione.**

Il carico di malattia, che ha un indice di correlazione negativo con HS, pari a **-0.6, è minore in Italia rispetto alla media UE, ad indicare la migliore situazione sanitaria italiana. Questo è confermato anche dal valore dell'aspettativa di vita che, anche se in minor misura, indica una situazione migliore in Italia.**

La fiducia nelle istituzioni (Trust), che ha l'indice di correlazione con HS più alto, pari a **0.84,**

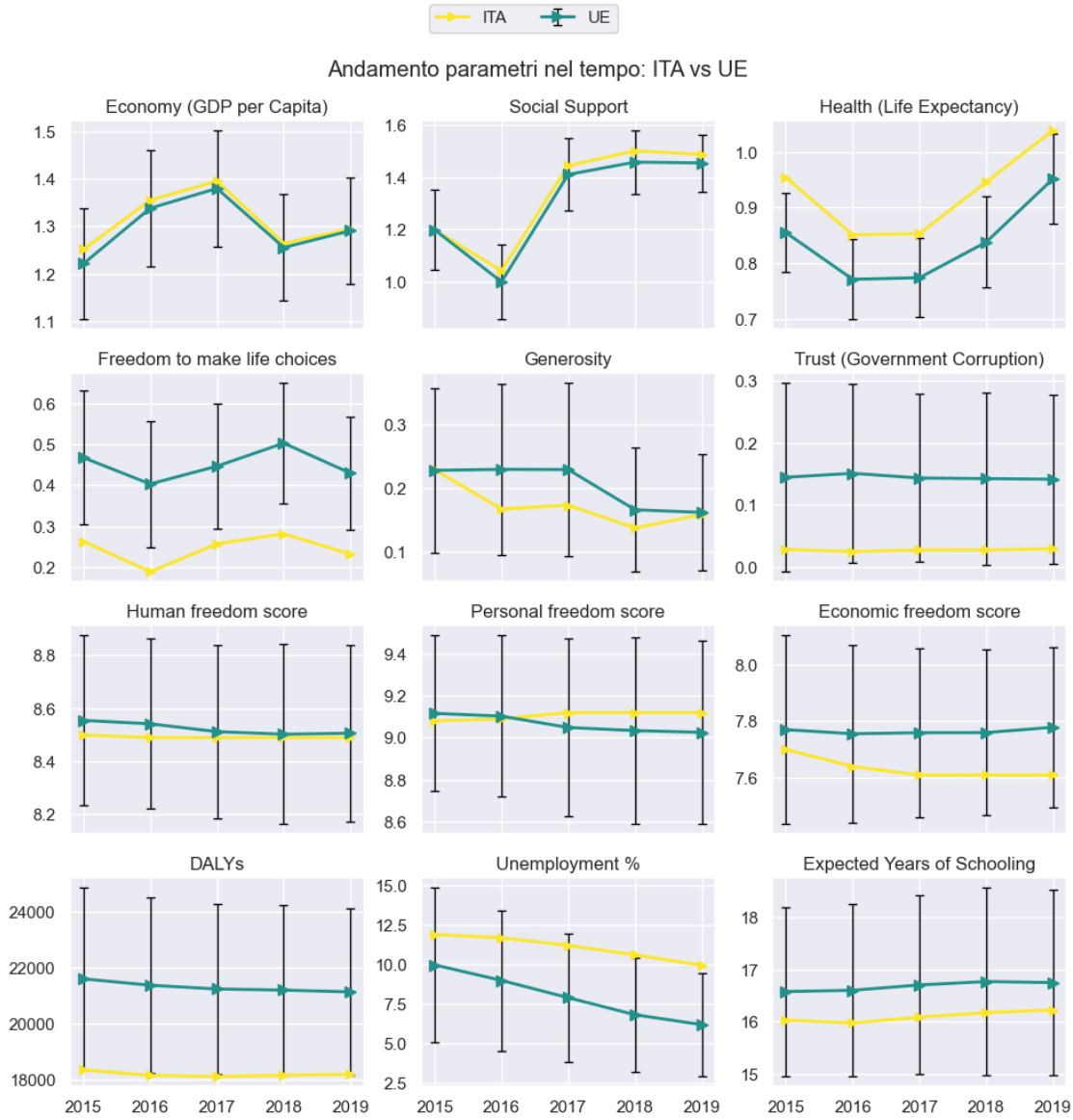
risulta minore in Italia quasi di una deviazione standard. Questo indica che la fiducia nelle istituzioni è un aspetto cruciale su cui l'Italia dovrebbe lavorare per aumentare la felicità dei cittadini.

La disoccupzione, con indice di correlazione **-0.33, risulta maggiore in Italia indicando una situazione peggiore nel nostro Paese rispetto alla media UE.** Tuttavia la differenza rispetto alla media UE non è elevata e l'indice di correlazione non sembra molto rilevante. Questo indica che la disoccupazione è un aspetto su cui l'Italia potrebbe lavorare per incrementare l'Happiness Score italiano ma non è un aspetto cruciale come i parametri visti precedentemente.

Per gli altri parametri la differenza tra Italia ed UE è irrisoria indicando che la situazione italiana in questi ambiti è in linea con quella europea.

7.0.8 Andamento parametri correlati ad HS

[ ]:
```
# seleziono colori grafico
colors=get_colors_from_cmap("viridis", 3)

fig, axs = plt.subplots(nrows=4, ncols=3, figsize=(10,10), sharex=True)
axs=axs.ravel()

for i, column in enumerate(UE_df.columns[3:]):

    UEf_grouped = UE_df.groupby("year")[column].mean() # raggruppo per anno
    UEf_std = UE_df.groupby("year")[column].std() # prendo dev std UE
    ITAf_grouped = ITA_df.groupby("year")[column].mean() # raggruppo per anno
    # grafico UE
    axs[i].errorbar(UEf_grouped.index, UEf_grouped.values,
                            yerr=UEf_std.values,  fmt='->', color=colors[1],
                            markersize=7, linewidth=2, ecolor='black',
                            elinewidth=1, capsize=3, label="UE")
    # grafico ITA
    axs[i].plot(ITAf_grouped, color=colors[2], linewidth=2, marker=">",␣
 ↪label="ITA")
    axs[i].set_title(column) # titoli assi
    axs[i].set_xticks=[UEf_grouped.index]


fig.suptitle("Andamento parametri nel tempo: ITA vs UE")
plt.tight_layout()
legend = plt.legend(ncol=2, bbox_to_anchor=(-0.5, 5.25), frameon=True)

#salvo immagine
plt.savefig("andamento_parametri_nel_tempo_ITA_vs_UE.png", bbox_inches="tight")

plt.show()
```



i grafici soprastanti mostrano l'andamento negli anni dei diversi parametri rappresentati dalle colonne del DataFrame dell'Italia e dell'Unione Europea.

si può osservare che: - Economy (GDP per capita), Social Support, Human Freedom Score, Personal Freedom Score e Generosity hanno valori simili ed un andamento negli anni comparabile.

- La colonna con la correlazione più alta con HS, **Trust (Government corruption) (correlazione con HS 0.84) ha un andamento simile tra Italia e UE ma con valori sempre inferiori in Italia. La differenza è di quasi una deviazione standard. Questi dati indicano che in Italia la fiducia nelle istituzioni è poca e questo incide molto sulla felicità della popolazione.**
- La seconda colonna per correlazione con HS è **Freedom to make life choices, con valore**

di 0.73. Nel grafico soprastante si può notare come l'**Italia abbia valori** di Freedom to make life choices **costantemente inferiori alla media UE di oltre una dev. std. Questo indica che la libertà di fare scelte di vita è un aspetto cruciale su cui focalizzare l'attenzione per poter aumentare il benessere dei cittadini.**

- **Dal punto di vista sanitario l'Italia sembra invece trovarsi in una situazione migliore rispetto alla media dell'UE. Sia i dati sull'aspettativa di vita che quelli sul carico di malattia sono migliori rispetto alla media europea.** L'aspettativa di vita infatti è sempre maggiore della media europea mentre il carico di malattia è inferiore alla media UE in tutti gli anni, entrambi con una differenza pari o poco superiore ad una dev.std.
- I dati riguardanti gli **anni previsti di scolarizzazione indicano un andamento similare tra Italia e UE ma con valori leggermente inferiori in Italia.** La differenza è minore di metà della dev. std.
- Per quanto riguarda la **libertà economica troviamo una situazione simile alla precedente ma con una leggera tendenza al peggioramento in Italia.**
- **La percentuale di disoccupazione infine indica un miglioramento graduale negli anni sia in Italia che in UE ma con valori di disoccupazione sempre minori in UE e un miglioramento meno deciso in Italia.** la differenza aumenta negli anni fino a superare una dev. std. nel 2019.

8 Conclusioni

!apt-get install texlive texlive-xetex texlive-latex-extra pandoc !pip install pypandoc

from google.colab import drive drive.mount('/content/drive')

!cp "drive/My Drive/Colab Notebooks/TommasoCiampoliniMLP.ipynb" ./ !jupyter nbconvert –to PDF "TommasoCiampoliniMLP.ipynb"L'analisi condotta permette di fare le seguenti considerazioni:

- **L'indice di felicità in Italia (6.0) risulta maggiore rispetto alla media globale (5.4) ma è minore rispetto alla media UE (6.3).**
- L'Italia si trova al **45°** posto nella classifica globale per HS e al **15°** in quella europea.
- **L'indice di felicità è cresciuto leggermente in tutto il mondo dal 2015 al 2019, indicando probabimente un complessivo miglioramento delle condizioni di vita**
- I parametri correlati maggiormente con HS in UE sono:
  - La fiducia nelle istituzioni,
  - La libertà di fare scelte di vita,
  - Il guadagno economico personale
- **Situazione italiana:**
  - **Punti di forza:**
    - **Situazione sanitaria:** l'Italia è abbondantemente sopra alla media UE per quanto riguarda il carico di malattia e leggermente sopra alla media per quanto riguarda l'aspettativa di vita. Questo indica una situazione positiva in ambito sanitario che influisce notevolmente sulla felicità degli italiani.

- **Punti deboli:**
  - **Libertà di fare scelte di vita:** i dati riguardanti questo aspetto mostrano una differenza consistente a sfavore dell'Italia. Questo parametro inoltre ha una forte correlazione con l'indice di felicità. Questo indica che misure volte a migliorare la libertà delle persone di poter fare scelte di vita sono cruciali per migliorare la felicità degli italiani.
  - **Fiducia nel governo:** i dati riguardanti questo aspetto mostrano una differenza significativa tra l'Italia e la media UE. Questo parametro è quello maggiormente correlato con l'indice di felicità e perciò migliorare la fiducia dei cittadini nelle istituzioni è vitale per poter aumentare l'HS italiano.
  - **Disoccupazione:** anche se in misura inferiore rispetto ai due parametri precedenti, la situazione italiana è peggiore rispetto alla media UE e questo incide sull'HS dell'Italia. Risulta quindi importante focalizzare l'attenzione anche su questo aspetto.

converto in pdf

[ ]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
```

```
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
pandoc is already the newest version (2.9.2.1-3ubuntu2).
pandoc set to manually installed.
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java libcommons-logging-java
  libcommons-parent-java libfontbox-java libfontenc1 libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
  libruby3.0 libsynctex2 libteckit0 libtexlua53 libtexluajit2 libwoff1
  libzzip-0-13 lmodern poppler-data preview-latex-style rake ruby
  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0
  rubygems-integration t1utils teckit tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-recommended texlive-pictures texlive-plain-generic tipa
  xfonts-encodings xfonts-utils
Suggested packages:
  fonts-noto fonts-freefont-otf | fonts-freefont-ttf libavalon-framework-java
  libcommons-logging-java-doc libexcalibur-logkit-java liblog4j1.2-java
  poppler-utils ghostscript fonts-japanese-mincho | fonts-ipafont-mincho
  fonts-japanese-gothic | fonts-ipafont-gothic fonts-arphic-ukai
  fonts-arphic-uming fonts-nanum ri ruby-dev bundler debhelper gv
  | postscript-viewer perl-tk xpdf | pdf-viewer xzdec
  texlive-fonts-recommended-doc texlive-latex-base-doc python3-pygments
  icc-profiles libfile-which-perl libspreadsheet-parseexcel-perl
  texlive-latex-extra-doc texlive-latex-recommended-doc texlive-luatex
  texlive-pstricks dot2tex prerex texlive-pictures-doc vprerex
```

```
  default-jre-headless tipa-doc
The following NEW packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java libcommons-logging-java
  libcommons-parent-java libfontbox-java libfontenc1 libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
  libruby3.0 libsynctex2 libteckit0 libtexlua53 libtexluajit2 libwoff1
  libzzip-0-13 lmodern poppler-data preview-latex-style rake ruby
  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0
  rubygems-integration t1utils teckit tex-common tex-gyre texlive texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures
  texlive-plain-generic texlive-xetex tipa xfonts-encodings xfonts-utils
0 upgraded, 55 newly installed, 0 to remove and 18 not upgraded.
Need to get 182 MB of archives.
After this operation, 572 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-droid-fallback all
1:6.0.1r16-1.1build1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1
[2,696 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 poppler-data all
0.4.11-1 [2,171 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-common all 6.17
[33.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-urw-base35 all
20200910-1 [6,367 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9-common
all 9.55.0~dfsg1-0ubuntu5.4 [752 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libidn12 amd64
1.38-4ubuntu1 [60.0 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/main amd64 libijs-0.35 amd64
0.35-15build2 [16.5 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 libjbig2dec0 amd64
0.19-3build2 [64.7 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9 amd64
9.55.0~dfsg1-0ubuntu5.4 [5,032 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libkpathsea6
amd64 2021.20210626.59705-1ubuntu0.1 [60.3 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/main amd64 libwoff1 amd64
1.0.2-1build4 [45.2 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy/universe amd64 dvisvgm amd64
2.13.1-1 [1,221 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-lmodern all
2.004.5-6.1 [4,532 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-noto-mono all
20201225-1build1 [397 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-texgyre all
20180621-3.1 [10.2 MB]
```

```
Get:17 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libapache-pom-java
all 18-1 [4,720 B]
Get:18 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-parent-
java all 43-1 [10.8 kB]
Get:19 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-logging-
java all 1.2-2 [60.3 kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfontenc1 amd64
1:1.1.4-1build3 [14.7 kB]
Get:21 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libptexenc1
amd64 2021.20210626.59705-1ubuntu0.1 [39.1 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy/main amd64 rubygems-integration
all 1.18 [5,336 B]
Get:23 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby3.0 amd64
3.0.2-7ubuntu2.4 [50.1 kB]
Get:24 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-rubygems all
3.3.5-2 [228 kB]
Get:25 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby amd64 1:3.0~exp1
[5,100 B]
Get:26 http://archive.ubuntu.com/ubuntu jammy/main amd64 rake all 13.0.6-2 [61.7
kB]
Get:27 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:28 http://archive.ubuntu.com/ubuntu jammy/universe amd64 ruby-webrick all
1.7.0-3 [51.8 kB]
Get:29 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-xmlrpc all
0.3.2-1ubuntu0.1 [24.9 kB]
Get:30 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libruby3.0
amd64 3.0.2-7ubuntu2.4 [5,113 kB]
Get:31 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsynctex2
amd64 2021.20210626.59705-1ubuntu0.1 [55.5 kB]
Get:32 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libteckit0 amd64
2.5.11+ds1-1 [421 kB]
Get:33 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexlua53
amd64 2021.20210626.59705-1ubuntu0.1 [120 kB]
Get:34 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexluajit2
amd64 2021.20210626.59705-1ubuntu0.1 [267 kB]
Get:35 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libzzip-0-13 amd64
0.13.72+dfsg.1-1.1 [27.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-encodings all
1:1.0.5-0ubuntu2 [578 kB]
Get:37 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-utils amd64
1:7.7+6build2 [94.6 kB]
Get:38 http://archive.ubuntu.com/ubuntu jammy/universe amd64 lmodern all
2.004.5-6.1 [9,471 kB]
Get:39 http://archive.ubuntu.com/ubuntu jammy/universe amd64 preview-latex-style
all 12.2-1ubuntu1 [185 kB]
Get:40 http://archive.ubuntu.com/ubuntu jammy/main amd64 t1utils amd64
1.41-4build2 [61.3 kB]
```

```
Get:41 http://archive.ubuntu.com/ubuntu jammy/universe amd64 teckit amd64
2.5.11+ds1-1 [699 kB]
Get:42 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-gyre all
20180621-3.1 [6,209 kB]
Get:43 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 texlive-
binaries amd64 2021.20210626.59705-1ubuntu0.1 [9,848 kB]
Get:44 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-base all
2021.20220204-1 [21.0 MB]
Get:45 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-fonts-
recommended all 2021.20220204-1 [4,972 kB]
Get:46 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-base
all 2021.20220204-1 [1,128 kB]
Get:47 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-
recommended all 2021.20220204-1 [14.4 MB]
Get:48 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive all
2021.20220204-1 [14.3 kB]
Get:49 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libfontbox-java all
1:1.8.16-2 [207 kB]
Get:50 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libpdfbox-java all
1:1.8.16-2 [5,199 kB]
Get:51 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-pictures
all 2021.20220204-1 [8,720 kB]
Get:52 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-extra
all 2021.20220204-1 [13.9 MB]
Get:53 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-plain-
generic all 2021.20220204-1 [27.5 MB]
Get:54 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tipa all 2:1.3-21
[2,967 kB]
Get:55 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-xetex all
2021.20220204-1 [12.4 MB]
Fetched 182 MB in 16s (11.6 MB/s)
Extracting templates from packages: 100%
Preconfiguring packages …
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 120879 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1build1_all.deb
…
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2.1_all.deb …
Unpacking fonts-lato (2.0-2.1) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/02-poppler-data_0.4.11-1_all.deb …
Unpacking poppler-data (0.4.11-1) …
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

[ ]:
```
!cp "drive/My Drive/Colab Notebooks/TommasoCiampoliniDataVis.ipynb" ./
!jupyter nbconvert --to PDF "TommasoCiampoliniDataVis.ipynb"
```

```
[NbConvertApp] Converting notebook Progetto Finale Data Science di Tommaso
Ciampolini.ipynb to PDF
[NbConvertApp] Support files will be in Progetto Finale Data Science di Tommaso
Ciampolini_files/
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
```

```
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Making directory ./Progetto Finale Data Science di Tommaso
Ciampolini_files
[NbConvertApp] Writing 203446 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 481725 bytes to Progetto Finale Data Science di Tommaso
```

Ciampolini.pdf