TCC Expectativa de Vida

Livrarias importadas

Pip Install

In [132…

```
# !pip install inflection
# !pip install pandas
# !pip install sqlalchemy
# !pip install geopy
# !pip install pycountry-convert
# !pip install seaborn
# !pip install plotly
# !pip install pycountry-convert
# !pip install melt
# !pip install sklearn
# !pip install Image
# !pip install missingno
# !pip install boruta
# !pip install xgboost
# !pip install xlrd
# !pip install -U notebook-as-pdf
# !pyppeteer-install
```

Import

In [5]:

```
import pandas                                         as pd
import numpy                                          as np
from geopy.geocoders        import Nominatim
from pycountry_convert      import country_alpha2_to_continent_code,country_name_to_countr
import inflection
import sqlite3
from matplotlib             import pyplot             as plt
from IPython.core.display   import HTML
import seaborn                                        as sns
from IPython.display        import Image
from scipy.stats            import shapiro
from sqlalchemy             import create_engine
import missingno                                      as msno
import plotly.express                                 as px
from scipy                  import stats
from sklearn.impute         import KNNImputer
from sklearn.preprocessing  import MinMaxScaler
from sklearn.preprocessing  import LabelEncoder
from sklearn.preprocessing  import RobustScaler
from sklearn.metrics        import mean_absolute_error, mean_squared_error
from sklearn.linear_model   import LinearRegression,Lasso,Ridge
from sklearn.ensemble       import RandomForestRegressor
from boruta                 import BorutaPy
import xgboost                                        as xgb


import warnings
warnings.filterwarnings('ignore')
```

Funçoes auxiliares

In [6]:

```python
def cross_validation( x_training, kfold, model_name, model, verbose=False ): #funcao que i

    mae_list = []
    mape_list = []
    rmse_list = []
    for k in reversed( range( 1, kfold+1 ) ):
        if verbose:
            print( '\nKFold Number: {}'.format( k ) )
        # start and end date for validation
        validation_start_date = x_training['year'].max() - (k*3)
        validation_end_date = x_training['year'].max() -((k-1)*3)
        # filtering dataset
        training = x_training[x_training['year'] < validation_start_date]
        validation = x_training[(x_training['year'] >= validation_start_date) & (x_trainin

        # training and validation dataset
        # training
        xtraining = training.drop( ['year', 'life_expectancy'], axis=1 )
        ytraining = training['life_expectancy']

        # validation
        xvalidation = validation.drop( ['year', 'life_expectancy'], axis=1 )
        yvalidation = validation['life_expectancy']

        # model
        m = model.fit( xtraining, ytraining )

        # prediction
        yhat = m.predict( xvalidation )

        #performance
        m_result = ml_error( model_name, np.expm1( yvalidation ), np.expm1( yhat ) )

        # store performance of each kfold iteration
        mae_list.append(  m_result['MAE'] )
        mape_list.append( m_result['MAPE'] )
        rmse_list.append( m_result['RMSE'] )

    return pd.DataFrame( {'Model Name': model_name,
                          'MAE CV': np.round( np.mean( mae_list ), 2 ).astype( str ) + ' -
                          'MAPE CV': np.round( np.mean( mape_list ), 2 ).astype( str ) + '
                          'RMSE CV': np.round( np.mean( rmse_list ), 2 ).astype( str ) + '


def mean_percentage_error( y, yhat ):
    '''funçao que retorna o erro percentual medio'''
    return np.mean( ( y - yhat ) / y )


def mean_absolute_percentage_error( y, yhat ):
    '''funçao que retorna o percentual do erro medio absoluto'''
    return np.mean( np.abs( ( y - yhat ) / y ) )


def ml_error( model_name, y, yhat ):
    '''funçao que retorna um dataframe com os indicadores de performance'''
    mae = mean_absolute_error( y, yhat )
    mape = mean_absolute_percentage_error( y, yhat )
    rmse = np.sqrt( mean_squared_error( y, yhat ) )

    return pd.DataFrame( { 'Model Name': model_name,
                           'MAE': mae,
                           'MAPE': mape,
                           'RMSE': rmse }, index=[0] )

def cramer_v( x, y ):
```

```
    '''funçao que retorna a matrix de crammer'''
    cm = pd.crosstab( x, y ).to_numpy()
    n = cm.sum()
    r, k = cm.shape

    chi2 = stats.chi2_contingency( cm )[0]
    chi2corr = max( 0, chi2 - (k-1)*(r-1)/(n-1) )

    kcorr = k - (k-1)**2/(n-1)
    rcorr = r - (r-1)**2/(n-1)
    return np.sqrt( (chi2corr/n) / ( min( kcorr-1, rcorr-1 ) ) )




def jupyter_settings():
    '''funçao que define os parametros do %notebook'''
    %matplotlib inline
    %pylab inline

    plt.style.use( 'bmh' )
    plt.rcParams['figure.figsize'] = [25, 12]
    plt.rcParams['font.size'] = 24

    display( HTML( '<style>.container { width:75% !important; }</style>') )
    pd.options.display.max_columns = None
    pd.options.display.max_rows = None
    pd.set_option( 'display.expand_frame_repr', False )

    sns.set()

# Supress Scientific Notation
np.set_printoptions(suppress=True)
pd.set_option('display.float_format', '{:.2f}'.format)
```

Aquisiçao de dados

Dataset Expectativa de vida

In [7]:

```
# **1** Dataframe da OMS e das Nações Unidas com a variavel life-expectation
df_expec=pd.read_csv('../Datasets/Life_Expectancy_Data.csv',parse_dates=[1])
df_expec.head()
```

Out[7]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Poli
0	Afghanistan	2015-01-01	Developing	65.00	263.00	62	0.01	71.28	65.00	1154	...	6.0
1	Afghanistan	2014-01-01	Developing	59.90	271.00	64	0.01	73.52	62.00	492	...	58.0
2	Afghanistan	2013-01-01	Developing	59.90	268.00	66	0.01	73.22	64.00	430	...	62.0
3	Afghanistan	2012-01-01	Developing	59.50	272.00	69	0.01	78.18	67.00	2787	...	67.0
4	Afghanistan	2011-01-01	Developing	59.20	275.00	71	0.01	7.10	68.00	3013	...	68.0

5 rows × 22 columns

In [8]:
```
print('numero de linhas:{}'.format(df_expec.shape[0]))
print('numero de colunas:{}'.format(df_expec.shape[1]))
```

```
numero de linhas:2938
numero de colunas:22
```

Dataset Emission

In [9]:
```
#  **2** Dataframe com emissao de dados
df_emi=pd.read_csv('../Datasets/emission data.csv')
df_emi.head()
```

Out[9]:

	Country	1751	1752	1753	1754	1755	1756	1757	1758	1759	...	2008	2009	
0	Afghanistan	0	0	0	0	0	0	0	0	0	...	85152637.00	91912951.00	10(
1	Africa	0	0	0	0	0	0	0	0	0	...	31830766884.00	33019042498.00	3421:
2	Albania	0	0	0	0	0	0	0	0	0	...	228794816.00	233169632.00	237
3	Algeria	0	0	0	0	0	0	0	0	0	...	2894819537.00	3015005401.00	3132
4	Americas (other)	0	0	0	0	0	0	0	0	0	...	77460245651.00	79617865087.00	8187'

5 rows × 268 columns

Verificaçao do nomes em country da tabela emissao

In [10]:
```
countryISO=df_expec['Country'].unique()
emissioncountry=df_emi['Country'].unique()
(set(countryISO) ^ set(emissioncountry))-set(countryISO)
(set(countryISO) ^ set(emissioncountry))-set(emissioncountry)
```

Out[10]:
```
{'Bolivia (Plurinational State of)',
 'Brunei Darussalam',
 'Cabo Verde',
 'Czechia',
 "Côte d'Ivoire",
 "Democratic People's Republic of Korea",
 'Democratic Republic of the Congo',
 'Iran (Islamic Republic of)',
 "Lao People's Democratic Republic",
 'Micronesia (Federated States of)',
 'Monaco',
 'Republic of Korea',
 'Republic of Moldova',
 'Russian Federation',
 'San Marino',
 'Syrian Arab Republic',
 'The former Yugoslav republic of Macedonia',
 'Timor-Leste',
 'United Kingdom of Great Britain and Northern Ireland',
 'United Republic of Tanzania',
 'United States of America',
 'Venezuela (Bolivarian Republic of)',
 'Viet Nam'}
```

In [11]:
```
countryISO_unmatch=['Bolivia (Plurinational State of)',
 'Brunei Darussalam',
```

```
 'Cabo Verde',
 'Czechia',
 "Côte d'Ivoire",
 "Democratic People's Republic of Korea",
 'Democratic Republic of the Congo',
 'Iran (Islamic Republic of)',
 "Lao People's Democratic Republic",
 'Micronesia (Federated States of)',
 'Republic of Korea',
 'Republic of Moldova',
 'Russian Federation',
 'Syrian Arab Republic',
 'The former Yugoslav republic of Macedonia',
 'Timor-Leste',
 'United Kingdom of Great Britain and Northern Ireland',
 'United Republic of Tanzania',
 'United States of America',
 'Venezuela (Bolivarian Republic of)',
 'Viet Nam']

countryemission_unmatch=['Bolivia','Brunei',
 'Cape Verde', 'Czech Republic', "Cote d'Ivoire",'North Korea','Democratic Republic of Cor

dictemi = dict(zip(countryemission_unmatch,countryISO_unmatch))
df_emi['Country']=df_emi['Country'].replace(dictemi)
```

In [12]:

```
# Building DataFrame
df_emi=df_emi[['Country','2000','2001','2002','2003','2004','2005','2006','2007','2008',
               '2009','2010', '2011', '2012', '2013', '2014', '2015']]

df_emi=df_emi.melt(id_vars=['Country'])  # empilha os dados com granularidade em country
df_emi.columns=['Country','Year','Emission']

df_emi.head()
```

Out[12]:

	Country	Year	Emission
0	Afghanistan	2000	71717793.00
1	Africa	2000	23640083267.00
2	Albania	2000	196932672.00
3	Algeria	2000	2118624684.00
4	Americas (other)	2000	60974588046.00

Dataset Demographics

In [13]:

```
# **3** Dataframe com a populaçao separada em masculina feminina e demografia
df_demo=pd.read_csv('../Datasets/WPP2019_TotalPopulationBySex.csv')
df_demo.head()
```

Out[13]:

	LocID	Location	VarID	Variant	Time	MidPeriod	PopMale	PopFemale	PopTotal	PopDensity
0	4	Afghanistan	2	Medium	1950	1950.50	4099.24	3652.87	7752.12	11.87
1	4	Afghanistan	2	Medium	1951	1951.50	4134.76	3705.39	7840.15	12.01
2	4	Afghanistan	2	Medium	1952	1952.50	4174.45	3761.55	7936.00	12.16
3	4	Afghanistan	2	Medium	1953	1953.50	4218.34	3821.35	8039.68	12.31

	LocID	Location	VarID	Variant	Time	MidPeriod	PopMale	PopFemale	PopTotal	PopDensity
4	4	Afghanistan	2	Medium	1954	1954.50	4266.48	3884.83	8151.32	12.49

Verificaçao do nomes em country da tabela demo

In [14]:

```
countryISO=df_expec['Country'].unique()
democountry=df_demo['Location'].unique()
(set(countryISO) ^ set(democountry))-set(countryISO)
print((set(countryISO) ^ set(democountry))-set(democountry))
```

```
{'United Kingdom of Great Britain and Northern Ireland', 'Swaziland', 'Micronesia (Federat
ed States of)', "Democratic People's Republic of Korea", 'The former Yugoslav republic of
Macedonia'}
```

In [15]:

```
countryISO_unmatch=["Democratic People's Republic of Korea",'Micronesia (Federated States

countrydemo_unmatch=["Dem. People's Republic of Korea", 'Micronesia (Fed. States of)','Nor

dictdemo = dict(zip(countrydemo_unmatch,countryISO_unmatch))
df_demo['Location']=df_demo['Location'].replace(dictdemo)
```

In [16]:

```
## Seleçao das colunas
df_demo=df_demo.loc[(df_demo['Time']>=2000 )& (df_demo['Time']<=2015),['Location','Time','
df_demo.columns=['Country','Year','Pop male','Pop female','Pop total','Density']
```

In [17]:

```
df_demo.head()
```

Out[17]:

	Country	Year	Pop male	Pop female	Pop total	Density
50	Afghanistan	2000	10689.51	10090.45	20779.96	31.83
51	Afghanistan	2001	11117.75	10489.24	21606.99	33.10
52	Afghanistan	2002	11642.11	10958.67	22600.77	34.62
53	Afghanistan	2003	12214.63	11466.24	23680.87	36.27
54	Afghanistan	2004	12763.73	11962.96	24726.69	37.87

Merge Datasets

In [18]:

```
# Trasforming variable date
df_demo['Year']=df_demo['Year'].astype(str)
df_emi['Year'] = pd.to_datetime( df_emi['Year'] ).dt.strftime('%Y')
df_expec['Year'] = pd.to_datetime( df_expec['Year'] ).dt.strftime('%Y')
df_demo['Year'] = pd.to_datetime( df_demo['Year']).dt.strftime('%Y')
```

In [19]:

```
# Merge Dataframes

df=pd.merge(df_expec,df_emi,how='left',on=['Country','Year'])
df=pd.merge(df,df_demo,how='left',on=['Country','Year'])
df=df.drop(['Population'], axis=1)
df.head()
```

Out[19]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	GD
0	Afghanistan	2015	Developing	65.00	263.00	62	0.01	71.28	65.00	1154	...	584.2
1	Afghanistan	2014	Developing	59.90	271.00	64	0.01	73.52	62.00	492	...	612.7
2	Afghanistan	2013	Developing	59.90	268.00	66	0.01	73.22	64.00	430	...	631.7
3	Afghanistan	2012	Developing	59.50	272.00	69	0.01	78.18	67.00	2787	...	669.9
4	Afghanistan	2011	Developing	59.20	275.00	71	0.01	7.10	68.00	3013	...	63.5

5 rows × 26 columns

DF2 API geolocation

In [20]:

```
## Return latitude
def geolocate_lat(country):
    geolocator = Nominatim(user_agent='catuserbot')

    try:
        # Geolocate the center of the country
        loc = geolocator.geocode(country)
        # And return latitude
        return loc.latitude

    except:
        # Return missing value
        return np.nan

## Return longitude
def geolocate_long(country):
    geolocator = Nominatim(user_agent='catuserbot')

    try:
        # Geolocate the center of the country
        loc = geolocator.geocode(country)
        # And return longitude
        return loc.longitude

    except:
        # Return missing value
        return np.nan
```

In [21]:

```
# api that get lat and long for each country (takes 2 hours to run)
#df['lat'],df['long']=df['Country'].apply(lambda x: geolocate_lat(x)),df['Country'].apply
```

In [22]:

```
#df.to_csv('Datasets/DF_complete.csv',index=False) #create a file con lat and long becaus
```

In [23]:

```
df= pd.read_csv('../Datasets/DF_complete.csv')
```

Populating Country code and continent code from function

In [24]:

```
#function to convert to alpah2 country codes
```

```python
def get_code(col):
    try:
        cn_a2_code = country_name_to_country_alpha2(col)
    except:
        cn_a2_code = 'Unknown'
    try:
        cn_continent = country_alpha2_to_continent_code(cn_a2_code)
    except:
        cn_continent = 'Unknown'
    return (cn_a2_code)

#function to convert to alpah2 country continent

def get_continent(col):
    try:
        cn_a2_code = country_name_to_country_alpha2(col)
    except:
        cn_a2_code = 'Unknown'
    try:
        cn_continent = country_alpha2_to_continent_code(cn_a2_code)
    except:
        cn_continent = 'Unknown'
    return (cn_continent)
```

In [25]:

```python
df['continent'],df['code']=df['Country'].apply(lambda x: get_continent(x)),df['Country'].a
```

In [26]:

```python
aux = df.loc[df['continent']=='Unknown','Country'].unique()
aux
```

Out[26]:

```
array(['Bolivia (Plurinational State of)', 'Iran (Islamic Republic of)',
       'Micronesia (Federated States of)', 'Republic of Korea',
       'The former Yugoslav republic of Macedonia', 'Timor-Leste',
       'Venezuela (Bolivarian Republic of)'], dtype=object)
```

In [27]:

```python
cont=['SA','AS','OC','AS','EU','AS','SA']
```

In [28]:

```python
for i in range(len(aux)):
    for j in range(len(df)):
        if aux[i] == df.loc[j,'Country'] :
            df.loc[j,'continent']=cont[i]
```

Changing columns name

In [29]:

```python
df.columns
```

Out[29]:

```
Index(['Country', 'Year', 'Status', 'Life expectancy ', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',
       'Diphtheria ', ' HIV/AIDS', 'GDP', ' thinness  1-19 years',
       ' thinness 5-9 years', 'Income composition of resources', 'Schooling',
       'Emission', 'Pop male', 'Pop female', 'Pop total', 'Density', 'lat',
       'long', 'continent', 'code'],
      dtype='object')
```

In [30]:

```python
cols_original=['Country', 'Year', 'Status', 'Life expectancy ', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',
```

```
       'Diphtheria ', ' HIV/AIDS', 'GDP', ' thinness  1-19 years',
       ' thinness 5-9 years', 'Income composition of resources', 'Schooling',
       'Emission', 'Pop male', 'Pop female', 'Pop total', 'Density', 'lat',
       'long', 'continent', 'code']

# snakecase = lambda x: inflection.underscore(x)
# cols = list(map(snakecase,cols_original))
# df.columns=cols


cols=['country', 'year', 'status', 'life_expectancy', 'adult_mortality',
       'infant_deaths', 'alcohol', 'percentage_expenditure', 'hepatitis_b',
       'measles', 'bmi', 'under_five_deaths', 'polio', 'total_expenditure',
       'diphtheria', 'hiv_aids', 'gdp', 'thinness_1_19_years',
       'thinness_5_9_years', 'income_composition_of_resources', 'schooling',
       'emission', 'pop_male', 'pop_female', 'pop_total', 'density','lat', 'long',
       'continent', 'code']

df.columns=cols
```

In [31]:

```
df.head()
```

Out[31]:

	country	year	status	life_expectancy	adult_mortality	infant_deaths	alcohol	percentage_expenditure	h
0	Afghanistan	2015	Developing	65.00	263.00	62	0.01	71.28	
1	Afghanistan	2014	Developing	59.90	271.00	64	0.01	73.52	
2	Afghanistan	2013	Developing	59.90	268.00	66	0.01	73.22	
3	Afghanistan	2012	Developing	59.50	272.00	69	0.01	78.18	
4	Afghanistan	2011	Developing	59.20	275.00	71	0.01	7.10	

5 rows × 30 columns

Data Preparation

In [32]:

```
jupyter_settings()
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [33]:

```
df1=df.copy()
```

Creating Features

In [34]:

```
# female percent
df1['perc_female']=df1['pop_female']/df1['pop_total']

# emission per population tax
df1['emission_pop']=df1['emission']/df1['pop_total']
```

Data Info

In [35]:

```
print('numero de linhas:{}'.format(df.shape[0]))
print('numero de colunas:{}'.format(df.shape[1]))
```

```
numero de linhas:2938
numero de colunas:30
```

In [36]:

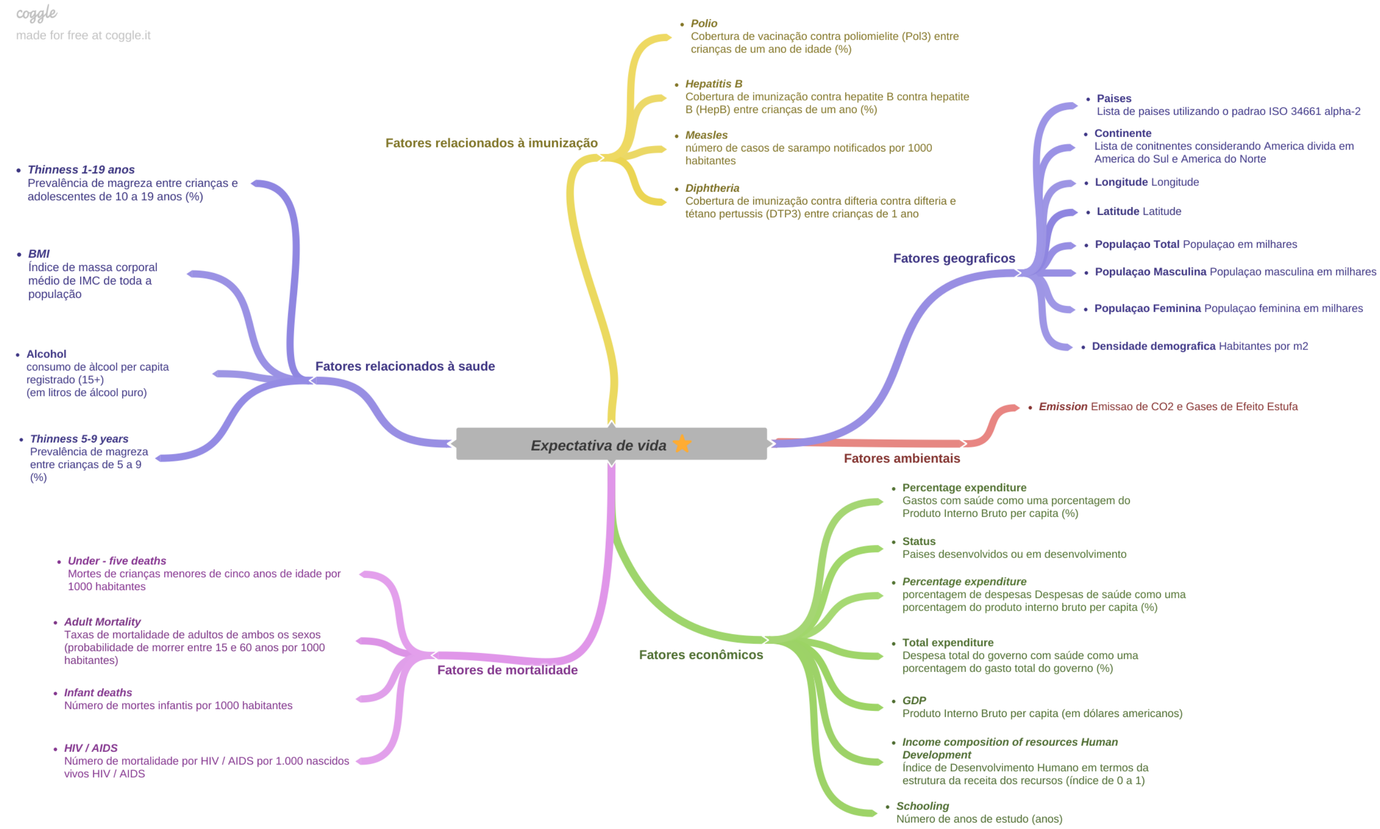
```python
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 32 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   country                          2938 non-null   object 
 1   year                             2938 non-null   int64  
 2   status                           2938 non-null   object 
 3   life_expectancy                  2928 non-null   float64
 4   adult_mortality                  2928 non-null   float64
 5   infant_deaths                    2938 non-null   int64  
 6   alcohol                          2744 non-null   float64
 7   percentage_expenditure           2938 non-null   float64
 8   hepatitis_b                      2385 non-null   float64
 9   measles                          2938 non-null   int64  
 10  bmi                              2904 non-null   float64
 11  under_five_deaths                2938 non-null   int64  
 12  polio                            2919 non-null   float64
 13  total_expenditure                2712 non-null   float64
 14  diphtheria                       2919 non-null   float64
 15  hiv_aids                         2938 non-null   float64
 16  gdp                              2490 non-null   float64
 17  thinness_1_19_years              2904 non-null   float64
 18  thinness_5_9_years               2904 non-null   float64
 19  income_composition_of_resources  2771 non-null   float64
 20  schooling                        2775 non-null   float64
 21  emission                         2936 non-null   float64
 22  pop_male                         2912 non-null   float64
 23  pop_female                       2912 non-null   float64
 24  pop_total                        2922 non-null   float64
 25  density                          2922 non-null   float64
 26  lat                              2922 non-null   float64
 27  long                             2920 non-null   float64
 28  continent                        2938 non-null   object 
 29  code                             2938 non-null   object 
 30  perc_female                      2912 non-null   float64
 31  emission_pop                     2920 non-null   float64
dtypes: float64(24), int64(4), object(4)
memory usage: 734.6+ KB
```

In [37]:

```python
Image('../img/Expectativa_de_vida_star.png')
```

Out[37]:



Descriptive Statistics

In [38]:
```
num_attributes = df1.select_dtypes( include=['int64', 'float64'] )
cat_attributes = df1.select_dtypes( exclude=['int64', 'float64', 'datetime64[ns]'] )
```

Numerical Atributes

In [39]:
```
# Central Tendency - mean, median
ct1 = pd.DataFrame( num_attributes.apply( np.mean ) ).T
ct2 = pd.DataFrame( num_attributes.apply( np.median ) ).T

# dispersion - std, min, max, range, skew, kurtosis
d1  = pd.DataFrame( num_attributes.apply( np.std ) ).T
d2  = pd.DataFrame( num_attributes.apply( min ) ).T
d3  = pd.DataFrame( num_attributes.apply( max ) ).T
d4  = pd.DataFrame( num_attributes.apply( lambda x: x.max() - x.min() ) ).T
d5  = pd.DataFrame( num_attributes.apply( lambda x: x.skew() ) ).T
d6  = pd.DataFrame( num_attributes.apply( lambda x: x.kurtosis() ) ).T

# concatenar
m = pd.concat( [d2, d3, d4, ct1, ct2, d1, d5, d6] ).T.reset_index()
m.columns = ['attributes', 'min', 'max', 'range', 'mean', 'median', 'std', 'skew', 'kurtos
m
```

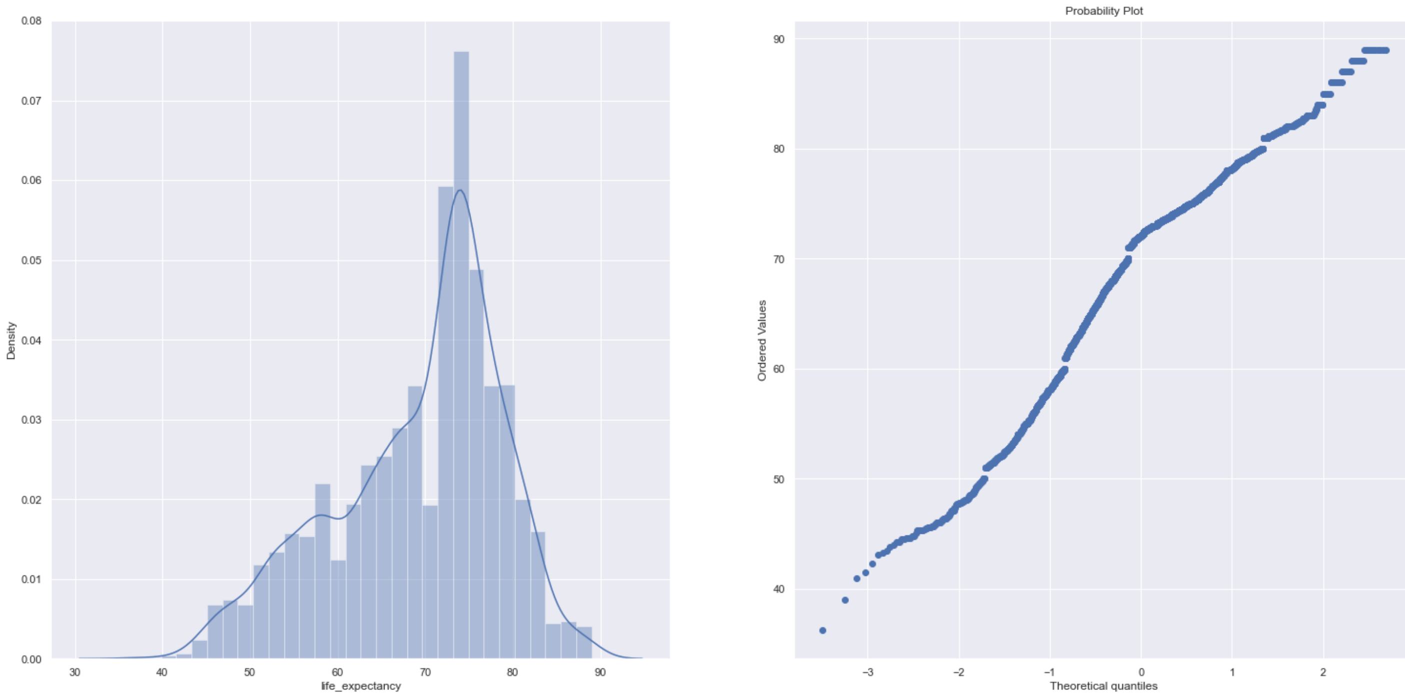
Out[39]:

	attributes	min	max	range	mean	median	
0	year	2000.00	2015.00	15.00	2007.52	2008.00	4
1	life_expectancy	36.30	89.00	52.70	69.22	NaN	9

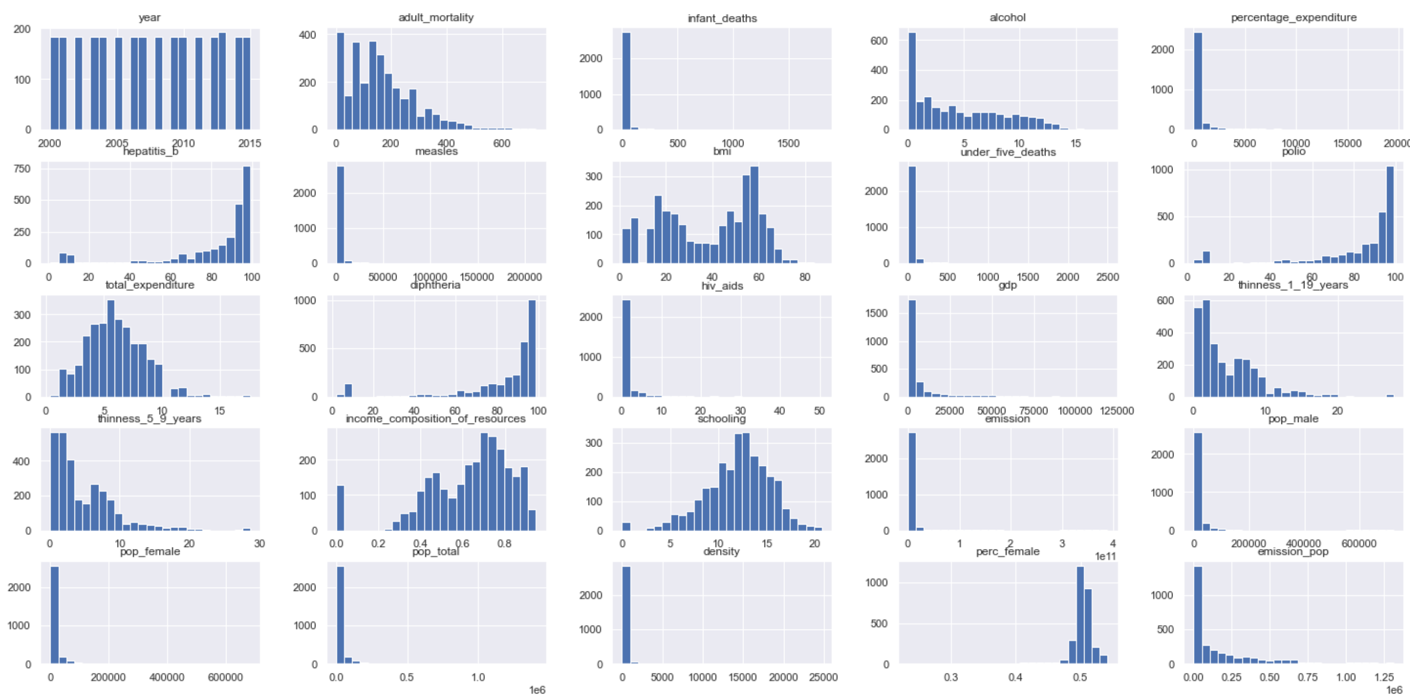
	attributes	min	max	range	mean	median	s
2	adult_mortality	1.00	723.00	722.00	164.80	NaN	124
3	infant_deaths	0.00	1800.00	1800.00	30.30	3.00	117
4	alcohol	0.01	17.87	17.86	4.60	NaN	4
5	percentage_expenditure	0.00	19479.91	19479.91	738.25	64.91	1987
6	hepatitis_b	1.00	99.00	98.00	80.94	NaN	25
7	measles	0.00	212183.00	212183.00	2419.59	17.00	11465
8	bmi	1.00	87.30	86.30	38.32	NaN	20
9	under_five_deaths	0.00	2500.00	2500.00	42.04	4.00	160
10	polio	3.00	99.00	96.00	82.55	NaN	23
11	total_expenditure	0.37	17.60	17.23	5.94	NaN	2
12	diphtheria	2.00	99.00	97.00	82.32	NaN	23
13	hiv_aids	0.10	50.60	50.50	1.74	0.10	5
14	gdp	1.68	119172.74	119171.06	7483.16	NaN	14267
15	thinness_1_19_years	0.10	27.70	27.60	4.84	NaN	4
16	thinness_5_9_years	0.10	28.60	28.50	4.87	NaN	4
17	income_composition_of_resources	0.00	0.95	0.95	0.63	NaN	0
18	schooling	0.00	20.70	20.70	11.99	NaN	3
19	emission	0.00	389000000000.00	389000000000.00	6553095034.87	NaN	29181208055
20	pop_male	35.71	722508.01	722472.30	18584.31	NaN	70463
21	pop_female	40.30	684339.86	684299.56	18285.66	NaN	66363
22	pop_total	1.61	1406847.87	1406846.26	36743.90	NaN	136593
23	density	1.54	24764.43	24762.89	177.21	NaN	701
24	lat	-41.50	64.98	106.48	18.42	NaN	24
25	long	-175.20	179.16	354.36	18.45	NaN	65
26	perc_female	0.23	0.54	0.31	0.50	NaN	0
27	emission_pop	0.00	1313605.50	1313605.50	183512.06	NaN	257204

In [40]:

```
# Response variable
plt.subplot( 1, 2, 1)
sns.distplot(df1['life_expectancy']);

plt.subplot( 1, 2, 2 )
stats.probplot(df1['life_expectancy'],dist='norm',plot=pylab)
pylab.show()
```



In [41]:
```python
cols_drop = ['lat', 'long','life_expectancy']
num_attributes1=num_attributes.drop(cols_drop, axis=1 )
num_attributes1.hist( bins=25 );
```



Categorical Atributes

In [42]:
```python
cat_attributes.apply( lambda x: x.unique().shape[0] )
```

Out[42]:
```
country      193
status         2
continent      6
code         188
dtype: int64
```

Missing values

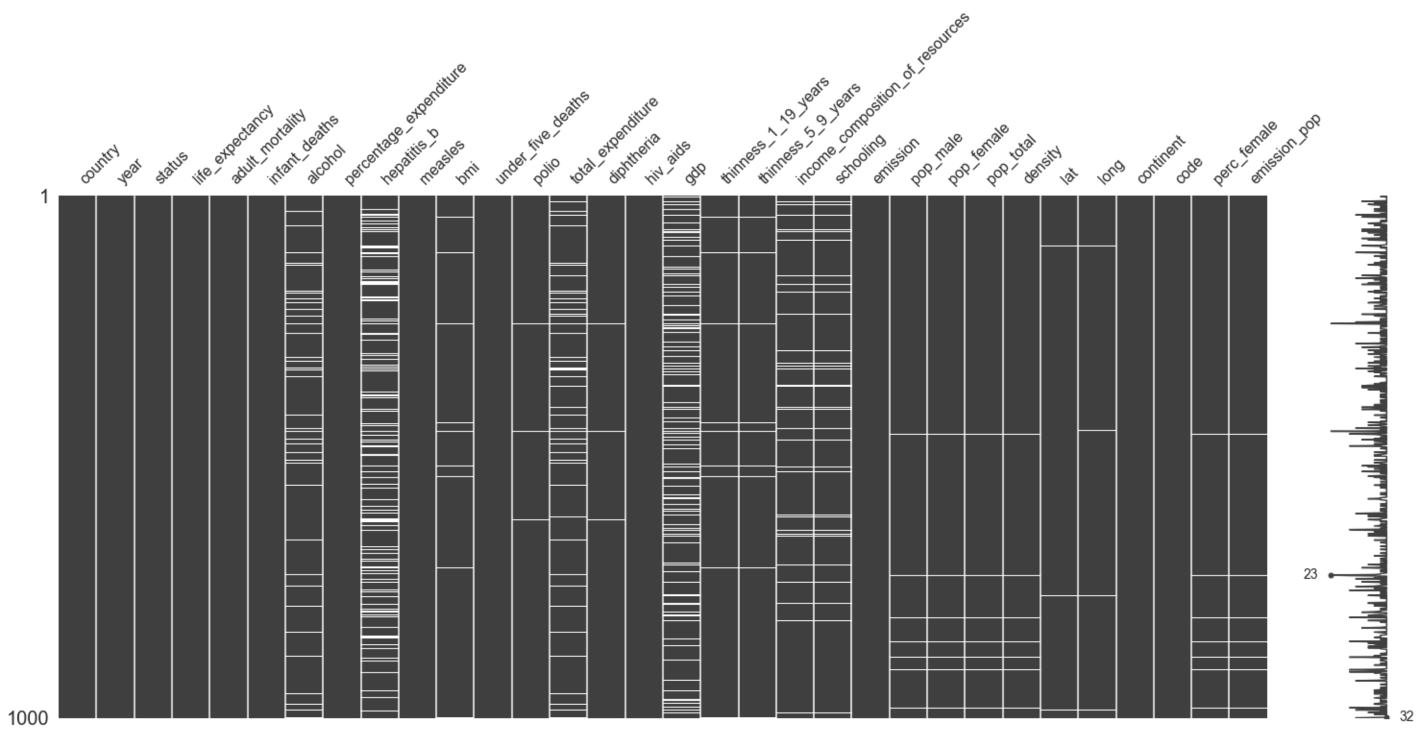
In [45]:
```python
# duplicate
```

```
df1.drop_duplicates()

print('not exist data duplicated' )

print('numero de linhas:{}'.format(df1.shape[0]))
print('numero de colunas:{}'.format(df1.shape[1]))
```

```
not exist data duplicated
numero de linhas:2938
numero de colunas:32
```

In [46]:

```
%matplotlib inline
msno.matrix(df1.sample(1000));
```



In [47]:

```
aux=df1.isna().sum().sort_values(ascending=False)
aux1=df1.isna().sum().sort_values(ascending=False)/df1.shape[0]*100

na=pd.concat([aux,aux1],axis=1)
na.columns=['NaN', 'NaN %']
na
```

Out[47]:

	NaN	NaN %
hepatitis_b	553	18.82
gdp	448	15.25
total_expenditure	226	7.69
alcohol	194	6.60
income_composition_of_resources	167	5.68
schooling	163	5.55
thinness_1_19_years	34	1.16
bmi	34	1.16
thinness_5_9_years	34	1.16

	NaN	NaN %
perc_female	26	0.88
pop_female	26	0.88
pop_male	26	0.88
polio	19	0.65
diphtheria	19	0.65
long	18	0.61
emission_pop	18	0.61
pop_total	16	0.54
density	16	0.54
lat	16	0.54
adult_mortality	10	0.34
life_expectancy	10	0.34
emission	2	0.07
year	0	0.00
hiv_aids	0	0.00
under_five_deaths	0	0.00
measles	0	0.00
percentage_expenditure	0	0.00
infant_deaths	0	0.00
continent	0	0.00
code	0	0.00
status	0	0.00
country	0	0.00

Bias gdp

In [48]:

```
df1=df1.drop(['gdp'], axis=1) # drop original collumn

df_gdp=pd.read_excel('../Datasets/GDPPC-USD-countries.xls',header=2)# New set Dataframe wi
df_gdp=df_gdp[['Country',2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010, 2011, 201
df_gdp=df_gdp.melt(id_vars=['Country'])  # stacks the data with granularity in the country
df_gdp.columns=['country','year','gdp']
```

In [49]:

```
# Checking the country names

countryISO=df['country'].unique()

gdpcountry=df_gdp['country'].unique()

(set(countryISO) ^ set(gdpcountry))-set(countryISO)

(set(countryISO) ^ set(gdpcountry))-set(gdpcountry)
```

Out[49]:
```
{"Democratic People's Republic of Korea",
 'Democratic Republic of the Congo',
 "Lao People's Democratic Republic",
 'Micronesia (Federated States of)',
 'Niue',
 'Saint Vincent and the Grenadines',
 'Swaziland',
 'The former Yugoslav republic of Macedonia',
 'United Kingdom of Great Britain and Northern Ireland',
 'United Republic of Tanzania',
 'United States of America'}
```

In [50]:
```
countryISO_unmatch=["Democratic People's Republic of Korea",
 'Democratic Republic of the Congo',
 "Lao People's Democratic Republic",
 'Micronesia (Federated States of)',
 'Saint Vincent and the Grenadines',
 'The former Yugoslav republic of Macedonia',
 'United Kingdom of Great Britain and Northern Ireland',
 'United Republic of Tanzania',
 'United States of America']

countrygdp_unmatch=[
 'D.P.R. of Korea',
 'D.R. of the Congo',
 "Lao People's DR",
 'Micronesia (FS of)',
 'St. Vincent and the Grenadines',
 'North Macedonia',
 'United Kingdom',
 'U.R. of Tanzania: Mainland',
 'United States',
]


dicgdp = dict(zip(countrygdp_unmatch,countryISO_unmatch))
df_gdp['country']=df_gdp['country'].replace(dicgdp)


df_gdp['year']=df_gdp['year'].astype(str)
df_gdp['year'] = pd.to_datetime( df_gdp['year'] ).dt.strftime('%Y')
df1['year']=df1['year'].astype(str)
df1['year'] = pd.to_datetime( df1['year'] ).dt.strftime('%Y')


df1=pd.merge(df1,df_gdp,how='left',on=['country','year'])

df1.head()
```

Out[50]:

	country	year	status	life_expectancy	adult_mortality	infant_deaths	alcohol	percentage_expenditure	h
0	Afghanistan	2015	Developing	65.00	263.00	62	0.01	71.28	
1	Afghanistan	2014	Developing	59.90	271.00	64	0.01	73.52	
2	Afghanistan	2013	Developing	59.90	268.00	66	0.01	73.22	
3	Afghanistan	2012	Developing	59.50	272.00	69	0.01	78.18	
4	Afghanistan	2011	Developing	59.20	275.00	71	0.01	7.10	

NaN com algoritmo KNN

In [51]:
```
knn_imputer = KNNImputer(n_neighbors=5, weights="uniform",metric='nan_euclidean')
```

```
df1['hepatitis_b'] = knn_imputer.fit_transform(df1[['hepatitis_b']])
df1['alcohol'] = knn_imputer.fit_transform(df1[['alcohol']])
df1['total_expenditure'] = knn_imputer.fit_transform(df1[['total_expenditure']])
df1['income_composition_of_resources'] = knn_imputer.fit_transform(df1[['income_compositio
df1['schooling'] = knn_imputer.fit_transform(df1[['schooling']])
```

NaN lat long

In [52]:

```
null_cols_code=df1.loc[df1['long'].isnull(),'country'].unique()
null_cols_code
```

Out[52]:

```
array(['Brunei Darussalam', 'Sierra Leone',
       'The former Yugoslav republic of Macedonia'], dtype=object)
```

In [53]:

```
lat=[-16.2837065,32.4207423,41.6137143]
lon=[-63.5493965,53.6830157,21.743258]

for i in range(len(null_cols_code)):
    for j in range(len(df1)):
        if null_cols_code[i] == df1.loc[j,'country'] :
            df1.loc[j,'lat']=lat[i]
            df1.loc[j,'long']=lon[i]
```

Eliminar NaN

In [54]:

```
df1.isna().sum().sort_values(ascending=False)
```

Out[54]:

```
thinness_1_19_years                34
bmi                                34
thinness_5_9_years                 34
gdp                                33
perc_female                        26
pop_female                         26
pop_male                           26
polio                              19
diphtheria                         19
emission_pop                       18
density                            16
pop_total                          16
adult_mortality                    10
life_expectancy                    10
emission                            2
code                                0
continent                           0
long                                0
lat                                 0
income_composition_of_resources     0
schooling                           0
infant_deaths                       0
status                              0
year                                0
hiv_aids                            0
total_expenditure                   0
under_five_deaths                   0
measles                             0
hepatitis_b                         0
percentage_expenditure              0
alcohol                             0
country                             0
dtype: int64
```

In [55]:

```
# drop NaN's
df1= df1.dropna(subset=['gdp','polio','bmi','life_expectancy'])
```

In [56]:
```
df1.isna().sum().sort_values(ascending=False)
```

Out[56]:
```
country                           0
year                              0
emission_pop                      0
perc_female                       0
code                              0
continent                         0
long                              0
lat                               0
density                           0
pop_total                         0
pop_female                        0
pop_male                          0
emission                          0
schooling                         0
income_composition_of_resources   0
thinness_5_9_years                0
thinness_1_19_years               0
hiv_aids                          0
diphtheria                        0
total_expenditure                 0
polio                             0
under_five_deaths                 0
bmi                               0
measles                           0
hepatitis_b                       0
percentage_expenditure            0
alcohol                           0
infant_deaths                     0
adult_mortality                   0
life_expectancy                   0
status                            0
gdp                               0
dtype: int64
```

In [57]:
```
print('novo numero de linhas:{}'.format(df1.shape[0]))
print('novo numero de colunas:{}'.format(df1.shape[1]))
```

```
novo numero de linhas:2872
novo numero de colunas:32
```

Exploratory Data Analysis

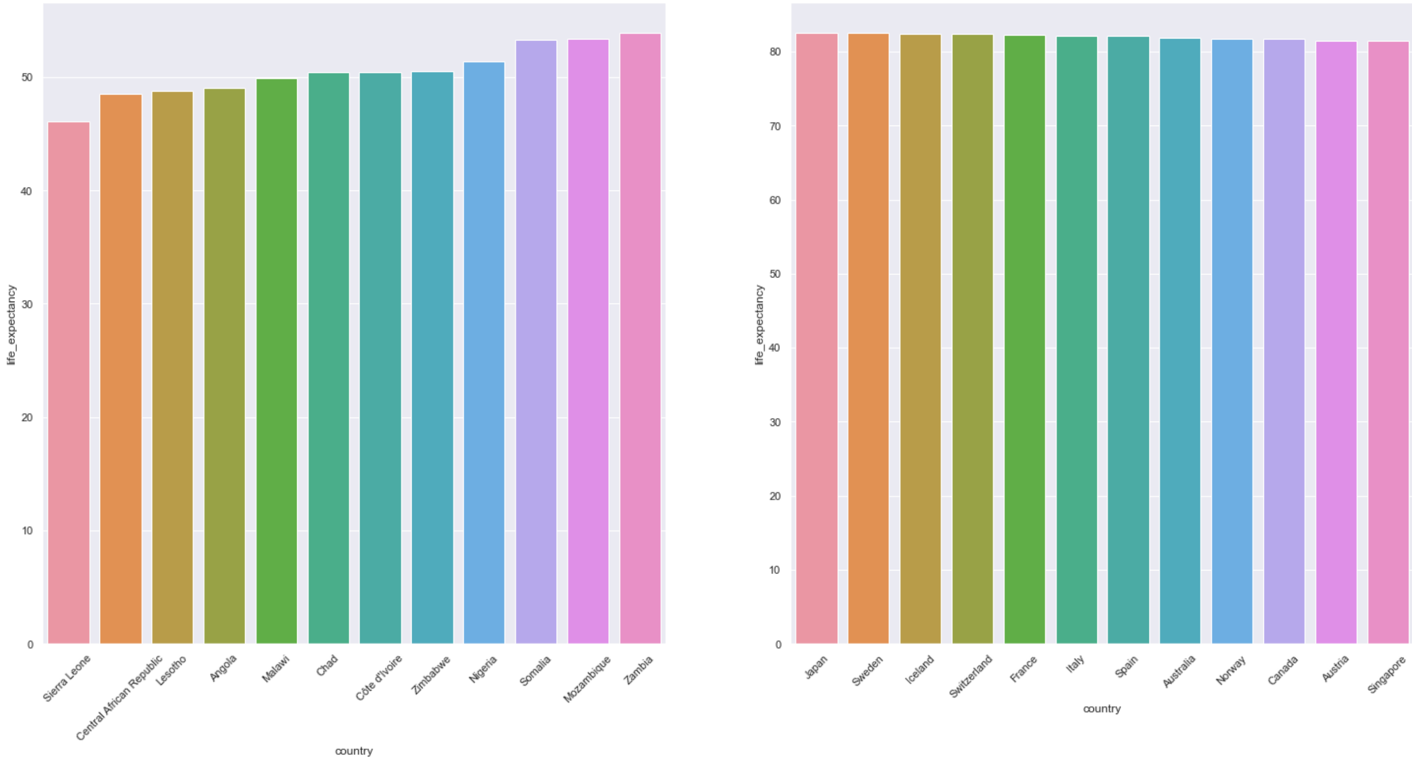
In [58]:
```
jupyter_settings()
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [59]:
```
df2=df1.copy()
```

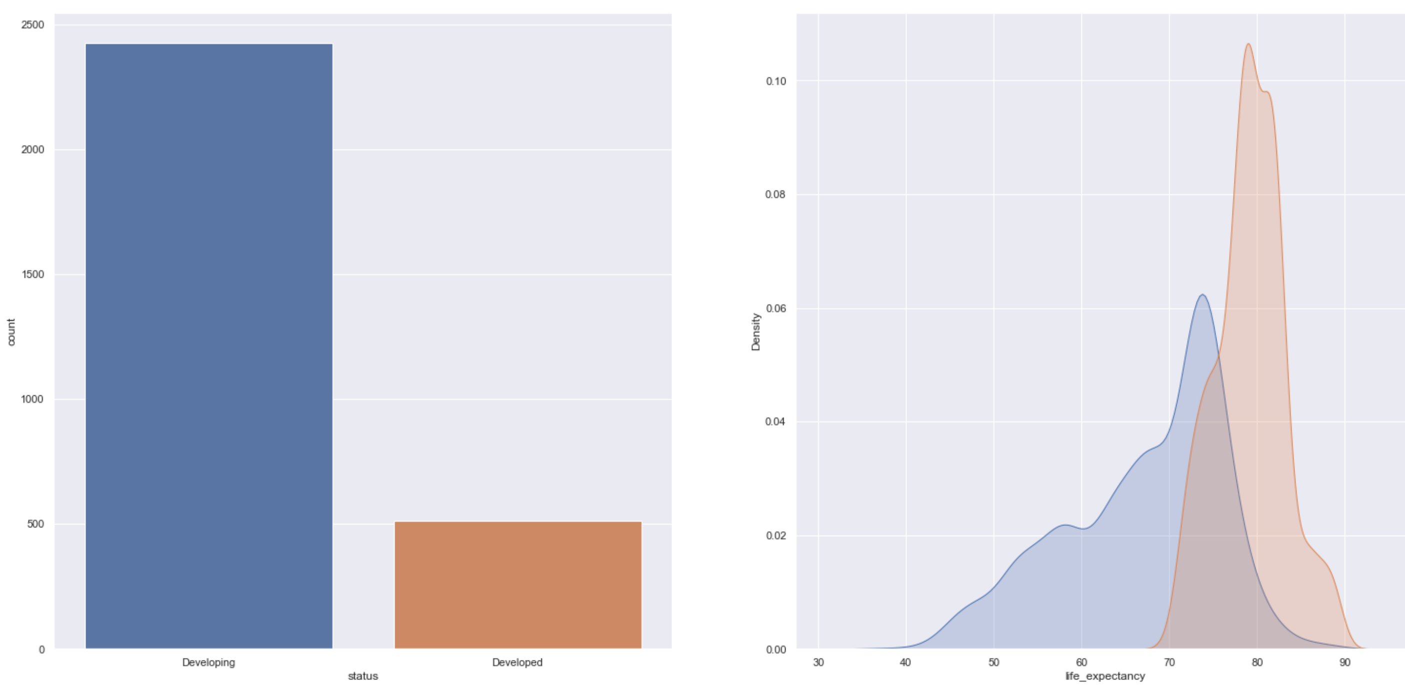
In [63]:
```
## continent
plt.subplot( 1, 2, 2)
a=df2[['country','life_expectancy']].groupby('country').mean().sort_values(ascending=False
a=a.head(12)
sns.barplot(x='country',y='life_expectancy',data=a);
plt.xticks(rotation=45);
```

```
plt.subplot( 1, 2, 1 )
b=df2[['country','life_expectancy']].groupby('country').mean().sort_values(ascending=True,
b=b.head(12)
sns.barplot(x='country',y='life_expectancy',data=b);
plt.xticks(rotation=45);
```



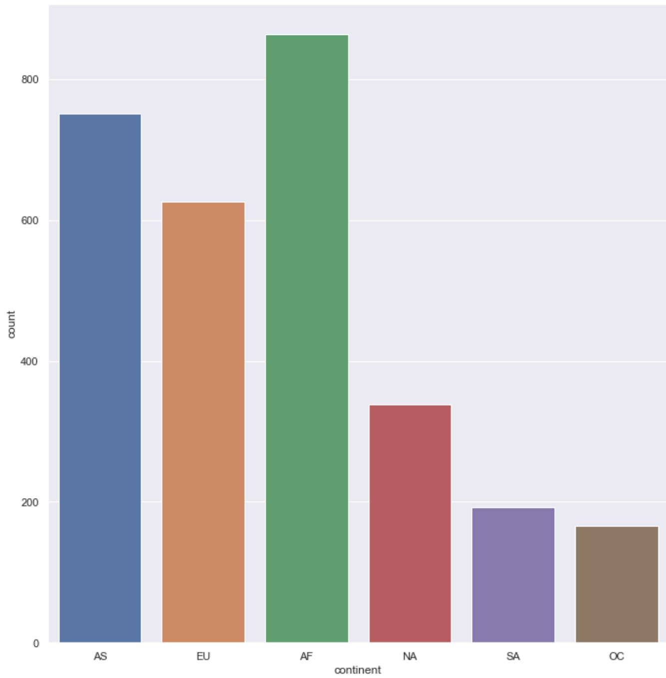
In [43]:

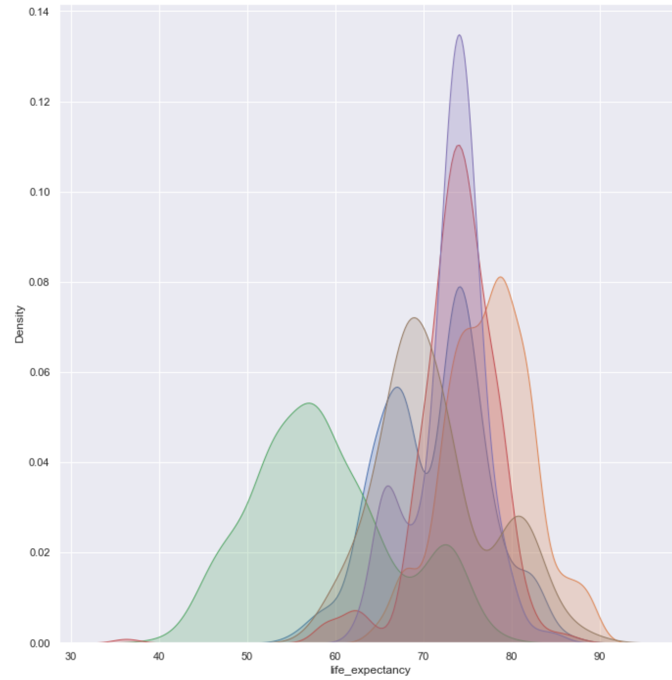
```
## Status
plt.subplot( 1, 2, 1 )

sns.countplot( df1['status'] )

plt.subplot( 1, 2, 2 )
sns.kdeplot( df1[df1['status'] =='Developing']['life_expectancy'], label='status', shade=T
sns.kdeplot( df1[df1['status'] =='Developed']['life_expectancy'], label='status', shade=Tr
```



In [44]:

```
## continent
plt.subplot( 1, 2, 1 )
sns.countplot( df1['continent'] )

plt.subplot( 1, 2, 2 )
sns.kdeplot( df1[df1['continent'] =='AS']['life_expectancy'], label='continent', shade=Tru
sns.kdeplot( df1[df1['continent'] =='EU']['life_expectancy'], label='continent', shade=Tru
sns.kdeplot( df1[df1['continent'] =='AF']['life_expectancy'], label='continent', shade=Tru
sns.kdeplot( df1[df1['continent'] =='NA']['life_expectancy'], label='continent', shade=Tru
sns.kdeplot( df1[df1['continent'] =='SA']['life_expectancy'], label='continent', shade=Tru
sns.kdeplot( df1[df1['continent'] =='OC']['life_expectancy'], label='continent', shade=Tru
```





H1 - Países densamente povoados ou altamaente populosos tendem a ter menor expectativa de vida?

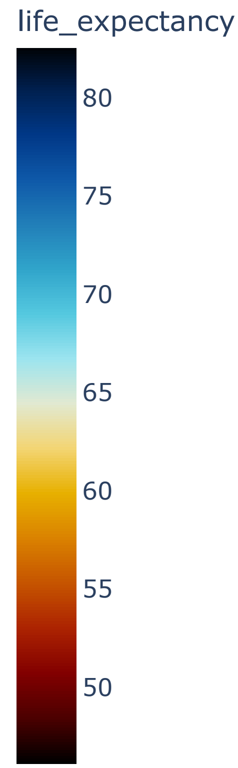
In [60]:
```
map=df2[['continent','status','country','lat','long','pop_total','density','life_expectanc
map.head()
```

Out[60]:

	continent	country	status	lat	long	pop_total	density	life_expectancy
0	AF	Algeria	Developing	28.00	3.00	34821.37	14.62	73.62
1	AF	Angola	Developing	-11.88	17.57	21622.86	17.34	49.02
2	AF	Benin	Developing	9.53	2.26	8628.81	76.52	57.57
3	AF	Botswana	Developing	-23.17	24.59	1888.51	3.33	56.05
4	AF	Burkina Faso	Developing	12.08	-1.69	14618.29	53.43	55.64

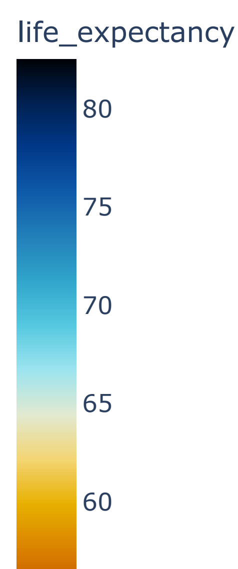
In [61]:
```
    fig=px.scatter_mapbox(map,
                          hover_name='country',
                          hover_data=["life_expectancy", "density",'status'],
                          lat='lat',
                          lon='long',
                          size='density',
                          color='life_expectancy',
                          color_continuous_scale=px.colors.cyclical.IceFire_r,
                          size_max=60,
                          zoom=1)
```

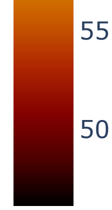
```
    fig.update_layout(mapbox_style='open-street-map')
    fig.update_layout(height=400, margin={'r':0,'l':0,'t':0,'b':0})
    fig.show()
```



In [62]:

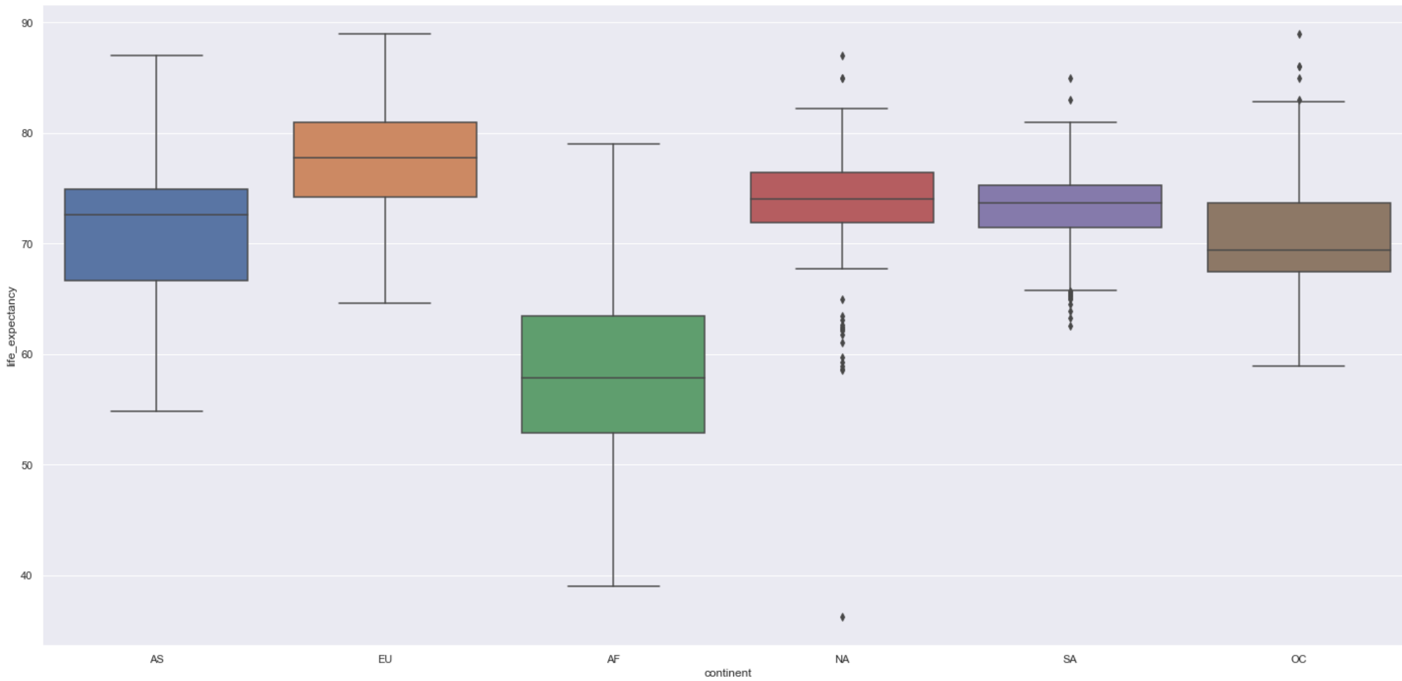
```
fig=px.scatter_mapbox(map,
                          hover_name='country',
                          hover_data=["life_expectancy", "density",'status'],
                          lat='lat',
                          lon='long',
                          size='pop_total',
                          color='life_expectancy',
                          color_continuous_scale=px.colors.cyclical.IceFire_r,
                          size_max=60,
                          zoom=1)

fig.update_layout(mapbox_style='open-street-map')
fig.update_layout(height=400, margin={'r':0,'l':0,'t':0,'b':0})
fig.show()
```



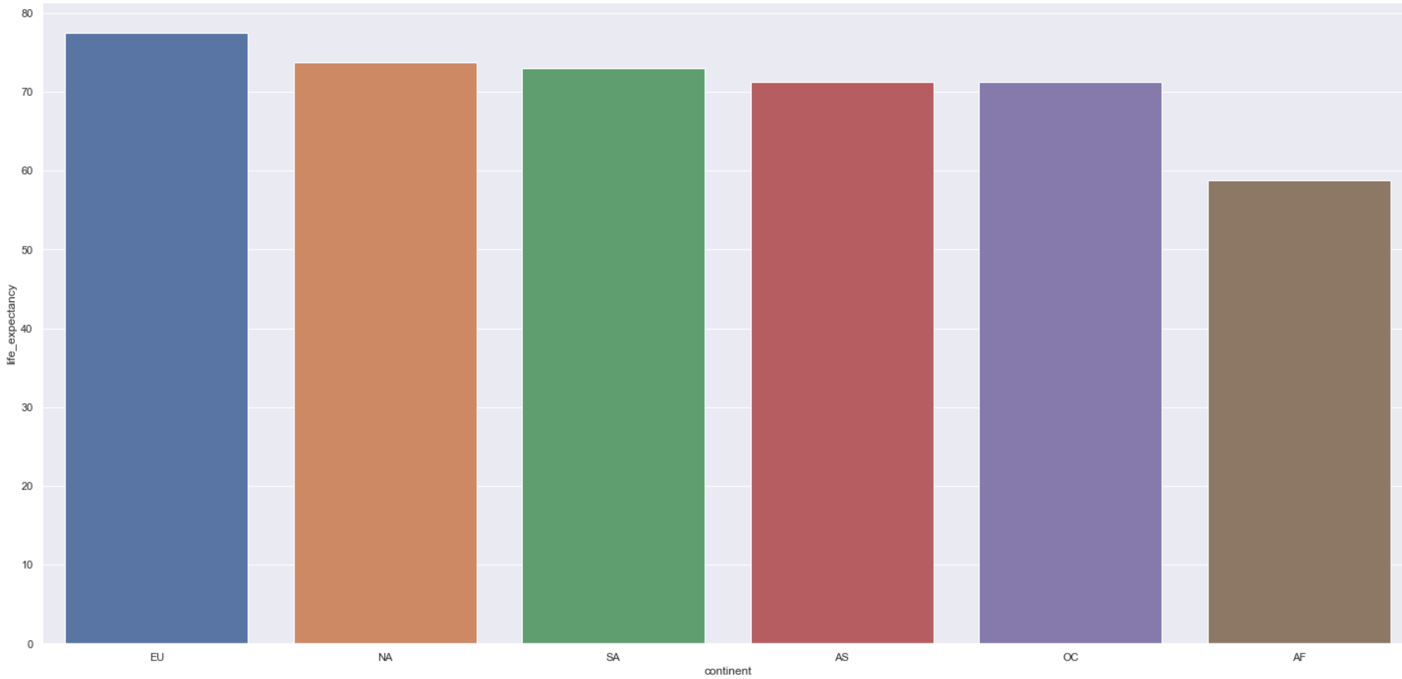


H2- Como è a variaçao da expectativa de vida dentro dos continentes

In [64]:
```
aux = df2[['continent','country','life_expectancy','year']]

#life_expectancy per continent
sns.boxplot( x='continent', y='life_expectancy',data=df2);
```

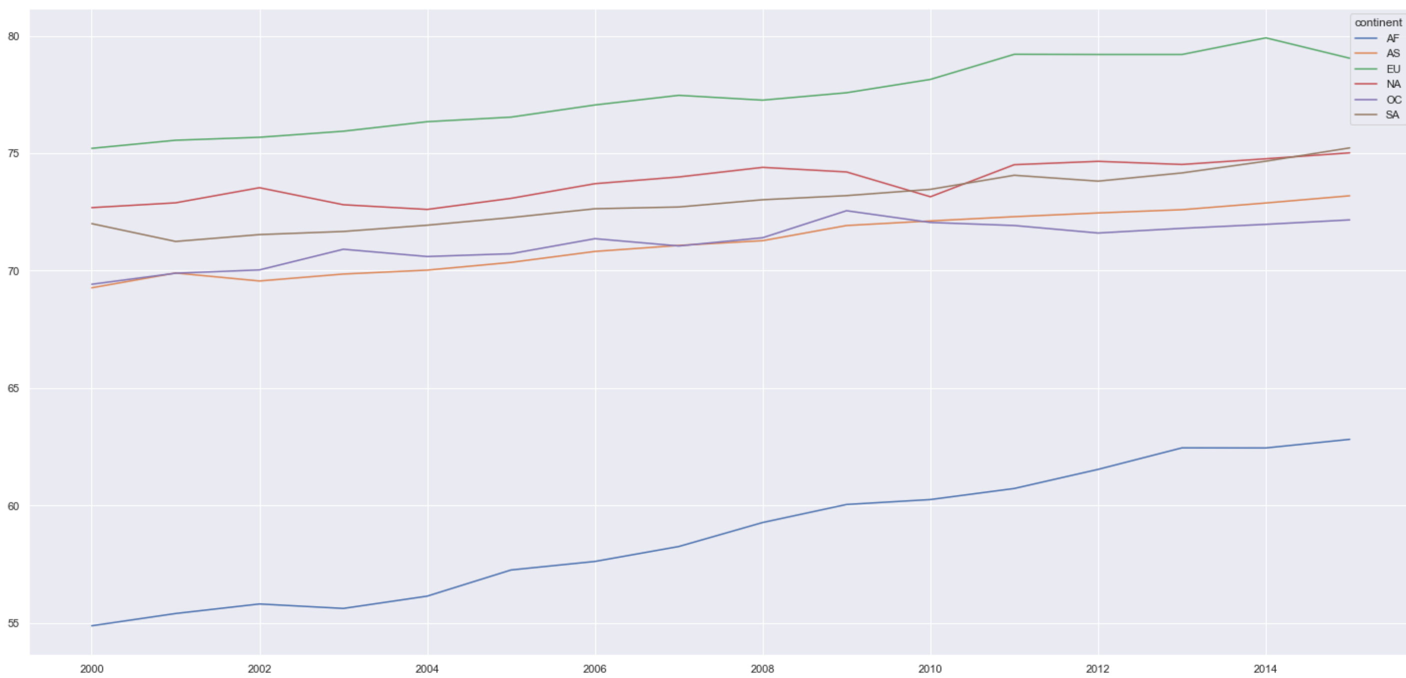


In [65]:
```
## continent

a=df2[['continent','life_expectancy']].groupby('continent').mean().sort_values(ascending=F
a=a.head(12)
sns.barplot(x='continent',y='life_expectancy',data=a);
```

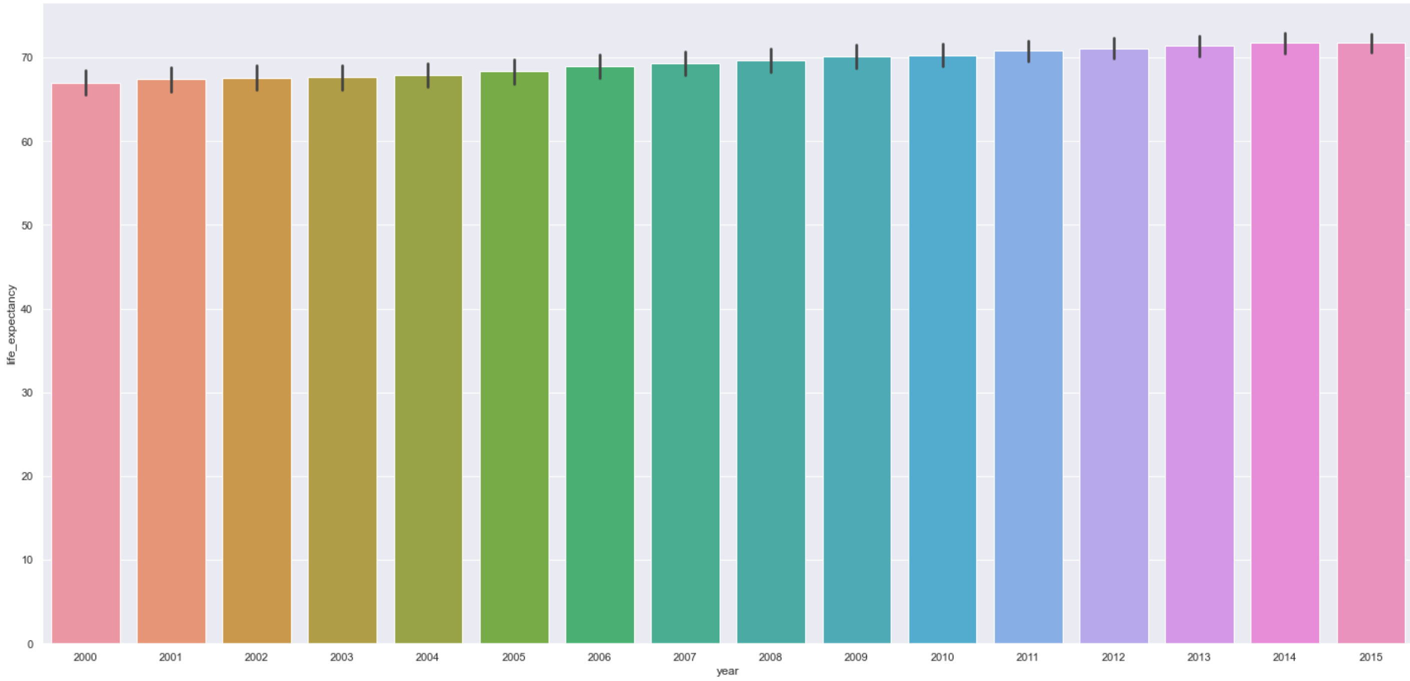


H3 - Como foi a evolucaçao da expectativa de vida ao longo dos anos

In [66]:
```
#life_expectancy mean per year
aux1 = df2[['year','continent','life_expectancy']].groupby(['continent','year']).mean().re
aux1.pivot(index='year',columns='continent',values='life_expectancy').plot();
```

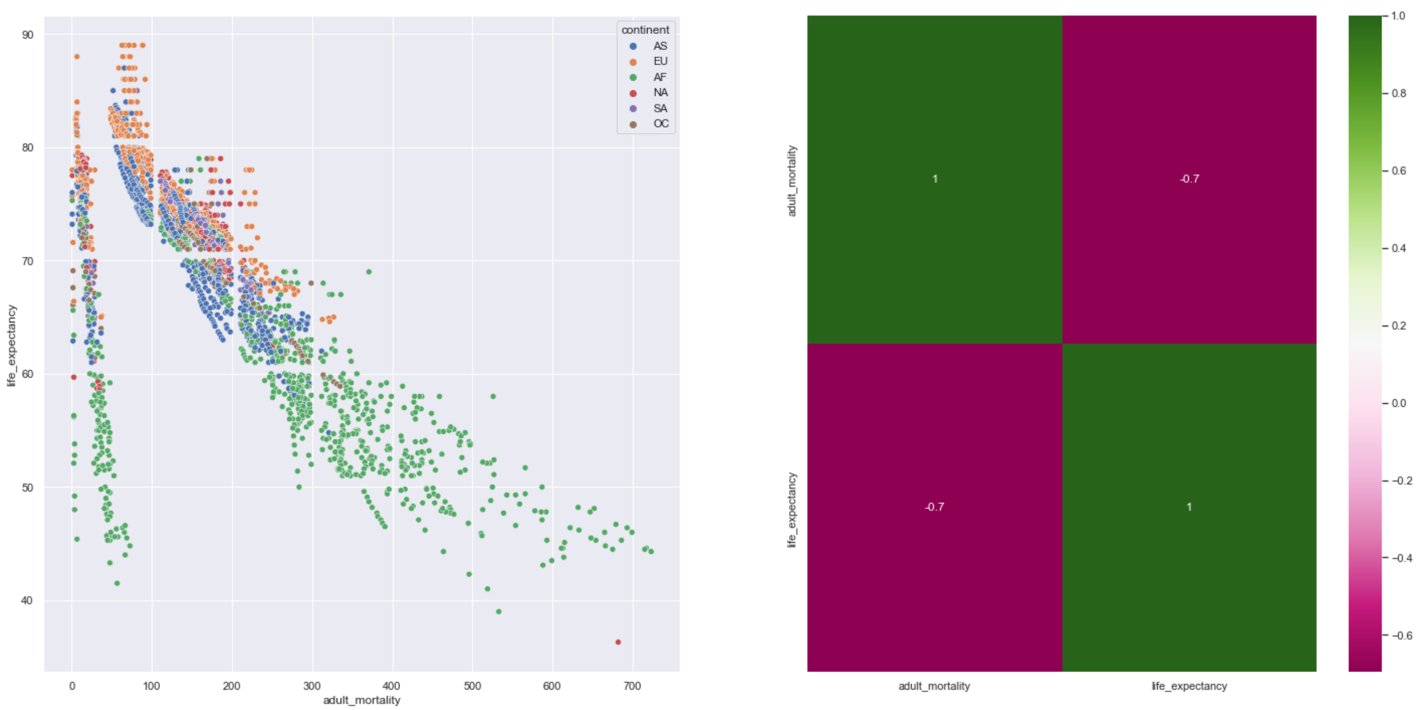


In [67]:
```
#life_expectancy mean per year
aux1 = df2[['year','country','life_expectancy']].groupby(['country','year']).mean().reset_
sns.barplot(x='year',y='life_expectancy',data=aux1);
```



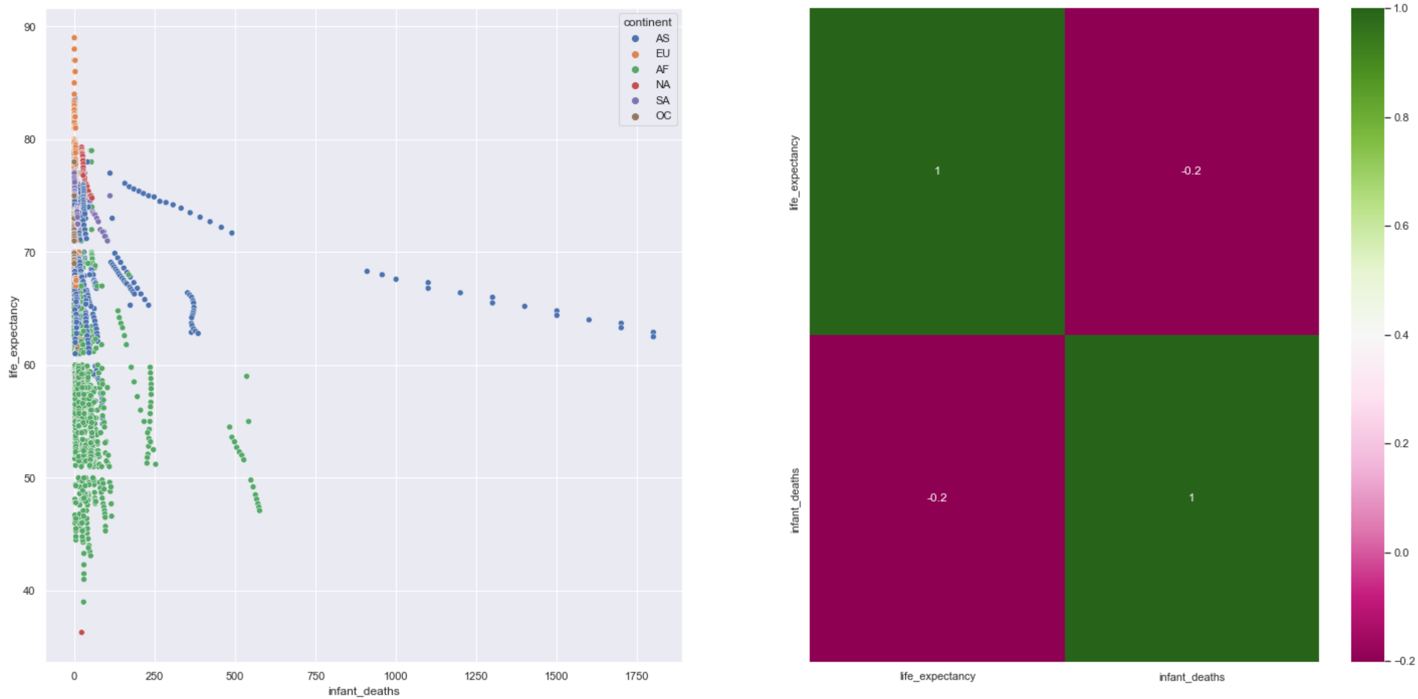
H4 - Como as taxas de mortalidade de bebês e adultos afetam a expectativa de vida?

In [68]:
```
aux2 = df2[['adult_mortality','life_expectancy','continent']]

plt.subplot( 1, 2, 1 )
sns.scatterplot(x='adult_mortality',y='life_expectancy',hue='continent',data=df2)

plt.subplot( 1, 2, 2 )
sns.heatmap(aux2.corr(method='pearson'),annot=True,cmap="PiYG");
```



In [69]:
```
aux3 = df2[['life_expectancy','infant_deaths','continent']]

plt.subplot( 1, 2, 1 )
sns.scatterplot(x='infant_deaths',y='life_expectancy',hue='continent', data=df2);
```

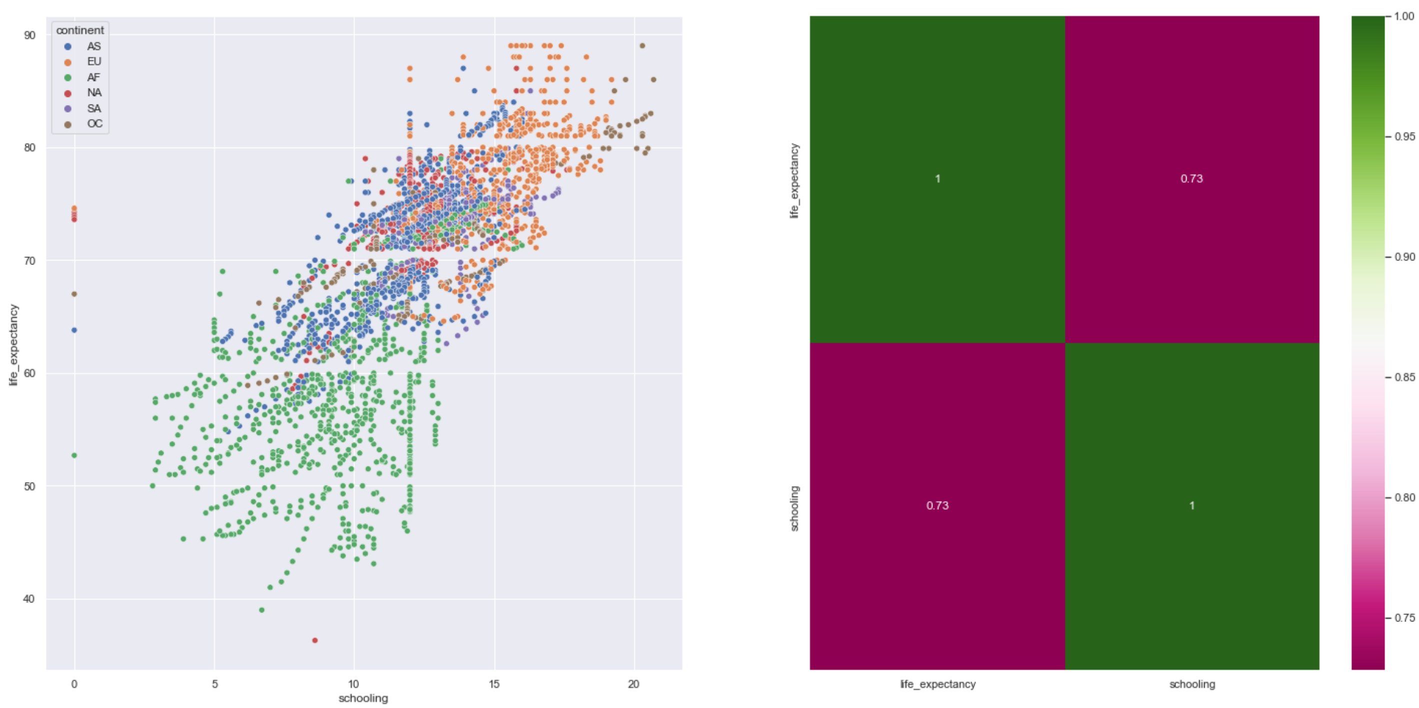
```
plt.subplot( 1, 2, 2 )
sns.heatmap(aux3.corr(method='pearson'),annot=True,cmap="PiYG");
```



H5 - Qual é o impacto da escolaridade na expectativa de vida?

In [183…

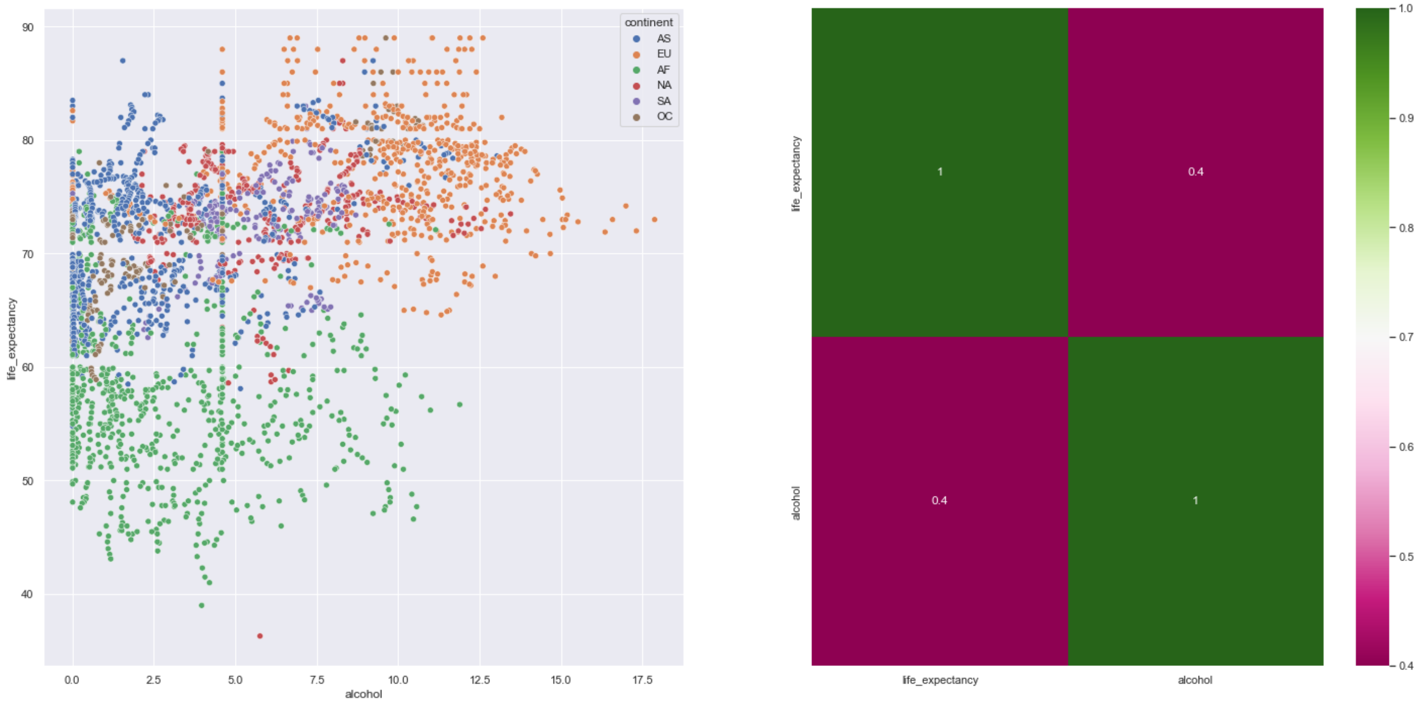
```
aux4 = df2[['life_expectancy','schooling','continent']]

plt.subplot( 1, 2, 1 )
sns.scatterplot(x='schooling',y='life_expectancy',hue='continent', data=df2);

plt.subplot( 1, 2, 2 )
sns.heatmap(aux4.corr(method='pearson'),annot=True,cmap="PiYG");
```



H6 - A expectativa de vida tem uma relação positiva ou negativa com o consumo de álcool?

In [184…

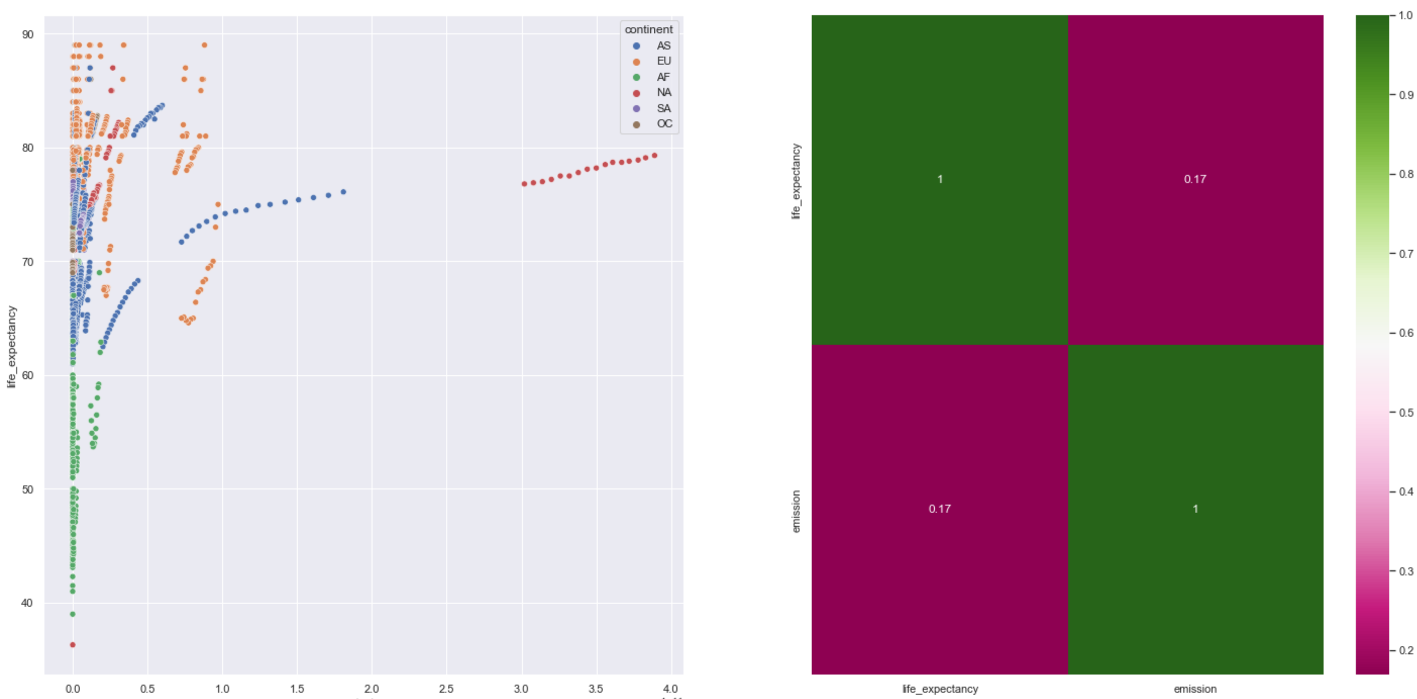
```
aux5 = df2[['life_expectancy','alcohol','continent']]

plt.subplot( 1, 2, 1 )
sns.scatterplot(x='alcohol',y='life_expectancy',hue='continent', data=df2);

plt.subplot( 1, 2, 2 )
sns.heatmap(aux5.corr(method='pearson'),annot=True,cmap="PiYG");
```



H7 - Paises mais inquinados representam uma expectativa de vida menor?

In [72]:

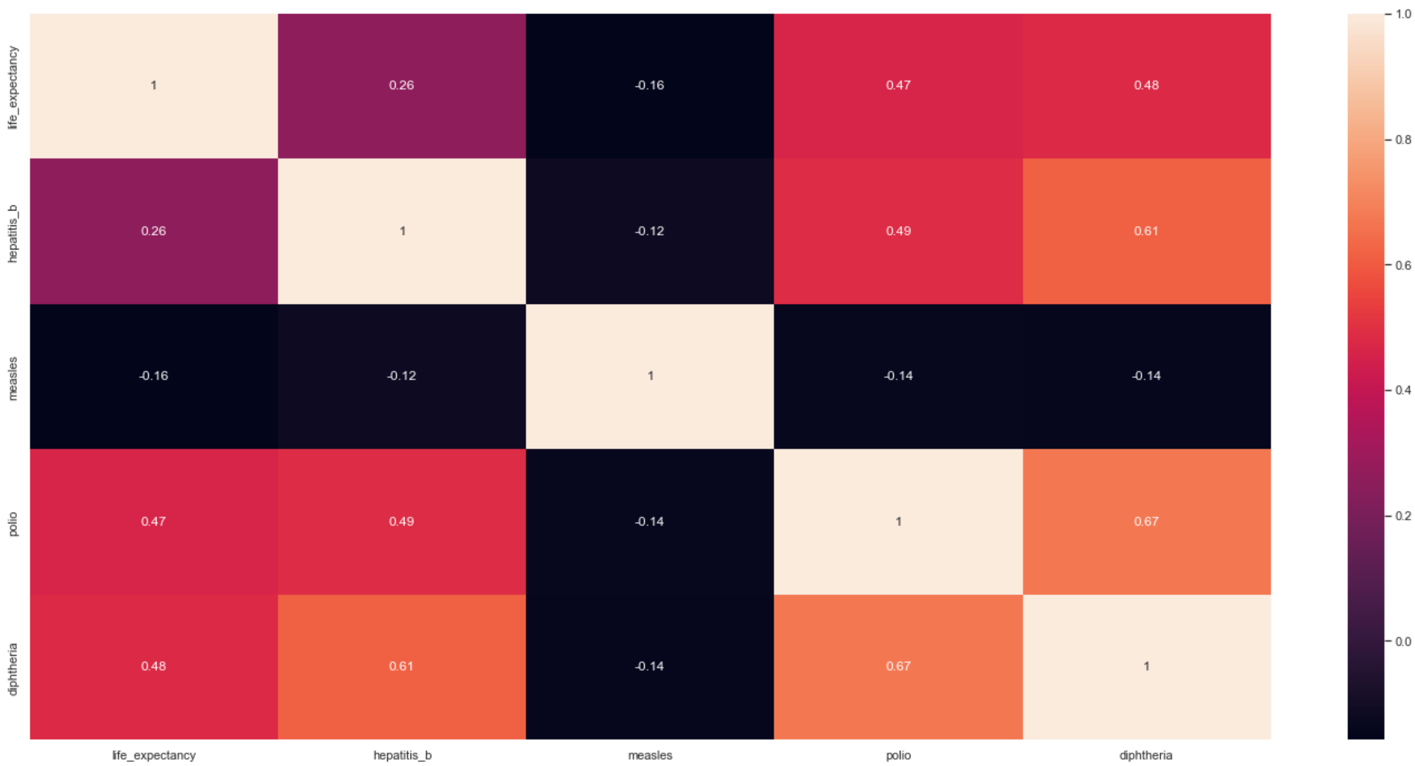
```
aux6 = df2[['life_expectancy','emission','continent']]

plt.subplot( 1, 2, 1 )
sns.scatterplot(x='emission',y='life_expectancy',hue='continent', data=df2);

plt.subplot( 1, 2, 2 )
sns.heatmap(aux6.corr(method='pearson'),annot=True,cmap="PiYG");
```



H8 - Qual é o impacto da cobertura de imunização na expectativa de vida?

In [186…

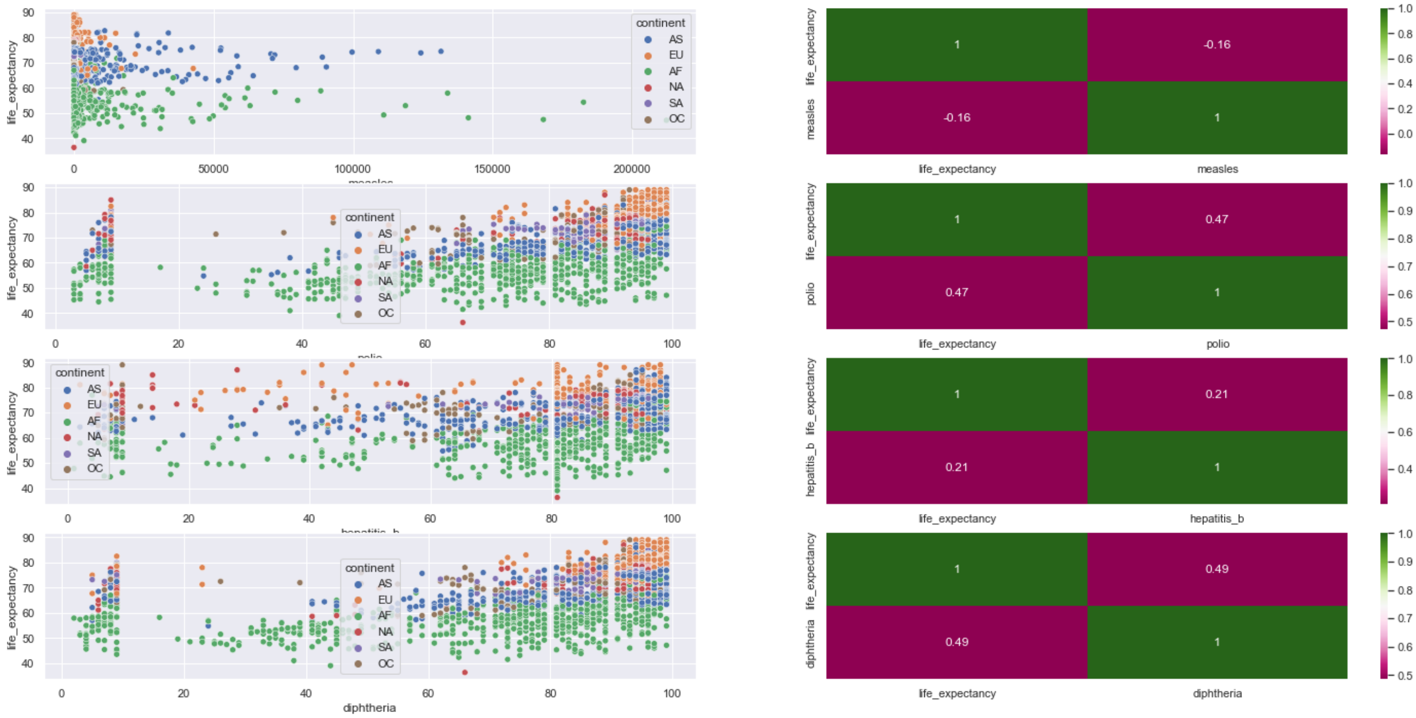
```
cols_drop = ['adult_mortality', 'infant_deaths',
       'alcohol', 'percentage_expenditure', 'bmi',
       'under_five_deaths', 'total_expenditure',
       'hiv_aids', 'gdp', 'pop_total', 'thinness_1_19_years',
       'thinness_5_9_years', 'income_composition_of_resources', 'schooling',
       'emission', 'lat', 'long','year','emission_pop','perc_female','density','pop_male',

num_attributes2=num_attributes.drop(cols_drop, axis=1 )
correlation = num_attributes2.corr( method='pearson' )
sns.heatmap( correlation, annot=True );
```



In [74]:

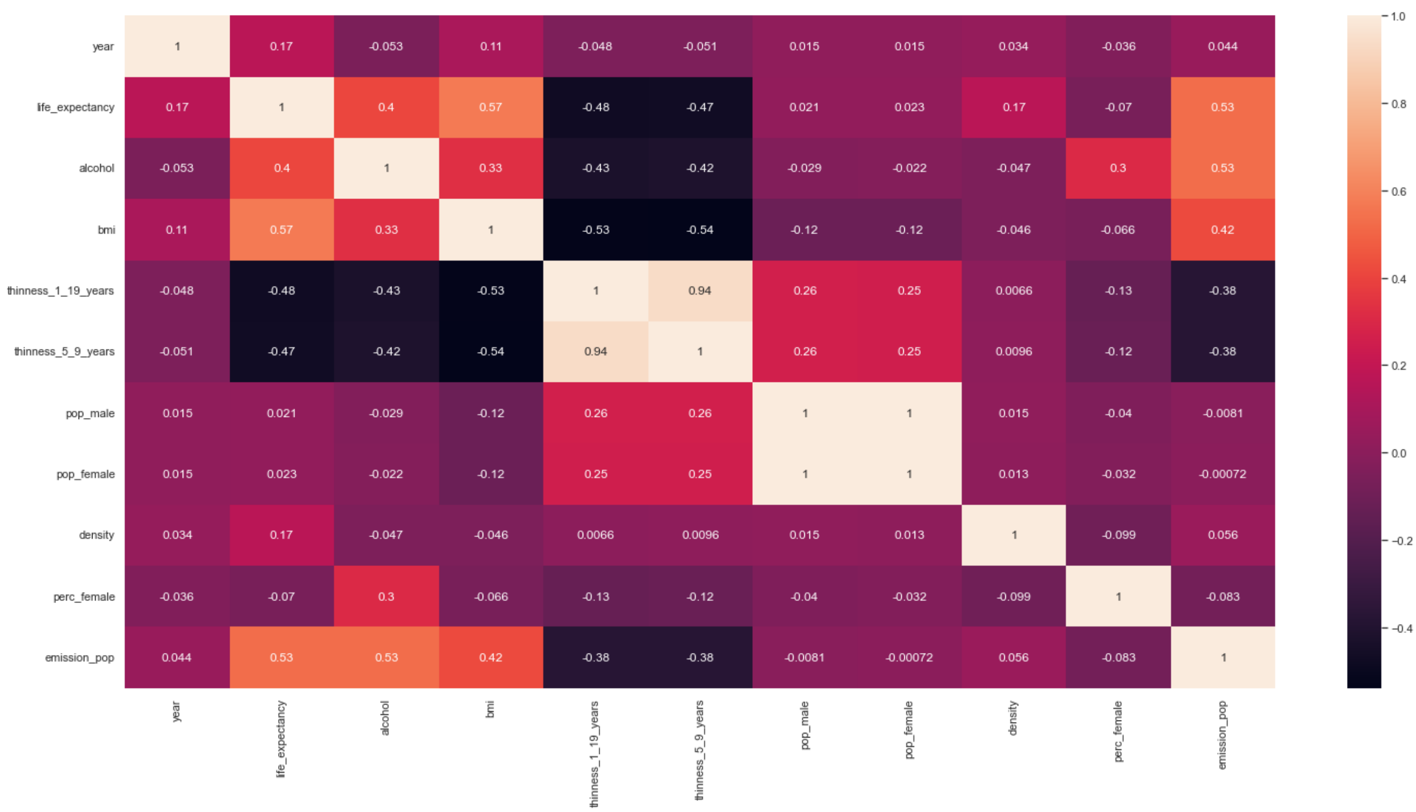
```
aux7 = df2[['life_expectancy','measles','continent']]
aux8 = df2[['life_expectancy','polio','continent']]
aux9 = df2[['life_expectancy','hepatitis_b','continent']]
aux0 = df2[['life_expectancy','diphtheria','continent']]

plt.subplot( 4, 2, 1 )
sns.scatterplot(x='measles',y='life_expectancy',hue='continent', data=df2);

plt.subplot( 4, 2, 2 )
sns.heatmap(aux7.corr(method='pearson'),annot=True,cmap="PiYG");

plt.subplot( 4, 2, 3 )
sns.scatterplot(x='polio',y='life_expectancy',hue='continent', data=df2);

plt.subplot( 4, 2, 4 )
sns.heatmap(aux8.corr(method='pearson'),annot=True,cmap="PiYG");

plt.subplot( 4, 2, 5 )
sns.scatterplot(x='hepatitis_b',y='life_expectancy',hue='continent', data=df2);

plt.subplot( 4, 2, 6)
sns.heatmap(aux9.corr(method='pearson'),annot=True,cmap="PiYG");

plt.subplot( 4, 2, 7 )
sns.scatterplot(x='diphtheria',y='life_expectancy',hue='continent', data=df2);
```

```
plt.subplot( 4, 2, 8)
sns.heatmap(aux0.corr(method='pearson'),annot=True,cmap="PiYG");
```



H9 - A expectativa de vida tem correlação positiva ou negativa com hábitos alimentares, estilo de vida, exercícios, fumo, bebida alcoólica etc.

In [75]:
```
cols_drop = ['adult_mortality','infant_deaths',
             'percentage_expenditure', 'hepatitis_b','measles', 'under_five_deaths', 'poli

num_attributes1=num_attributes.drop(cols_drop, axis=1 )
correlation = num_attributes1.corr( method='pearson' )
sns.heatmap( correlation, annot=True );
```



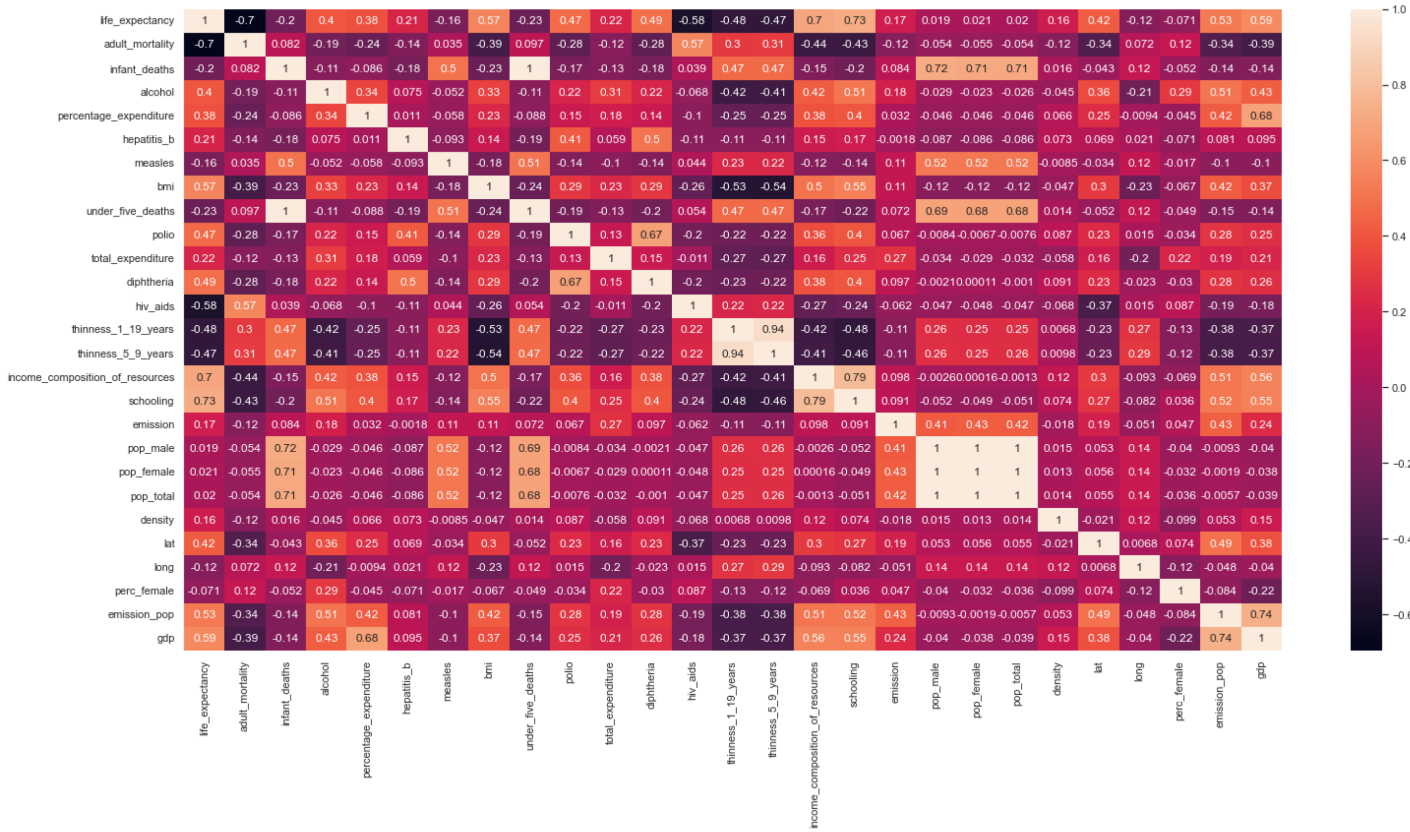
Numerical Heatmap

In [76]:

```
df3=df2.copy()
num_attributes4 = df3.select_dtypes( include=['int64', 'float64'] )
cat_attributes4 = df3.select_dtypes( exclude=['int64', 'float64', 'datetime64[ns]'] )
```

In [77]:

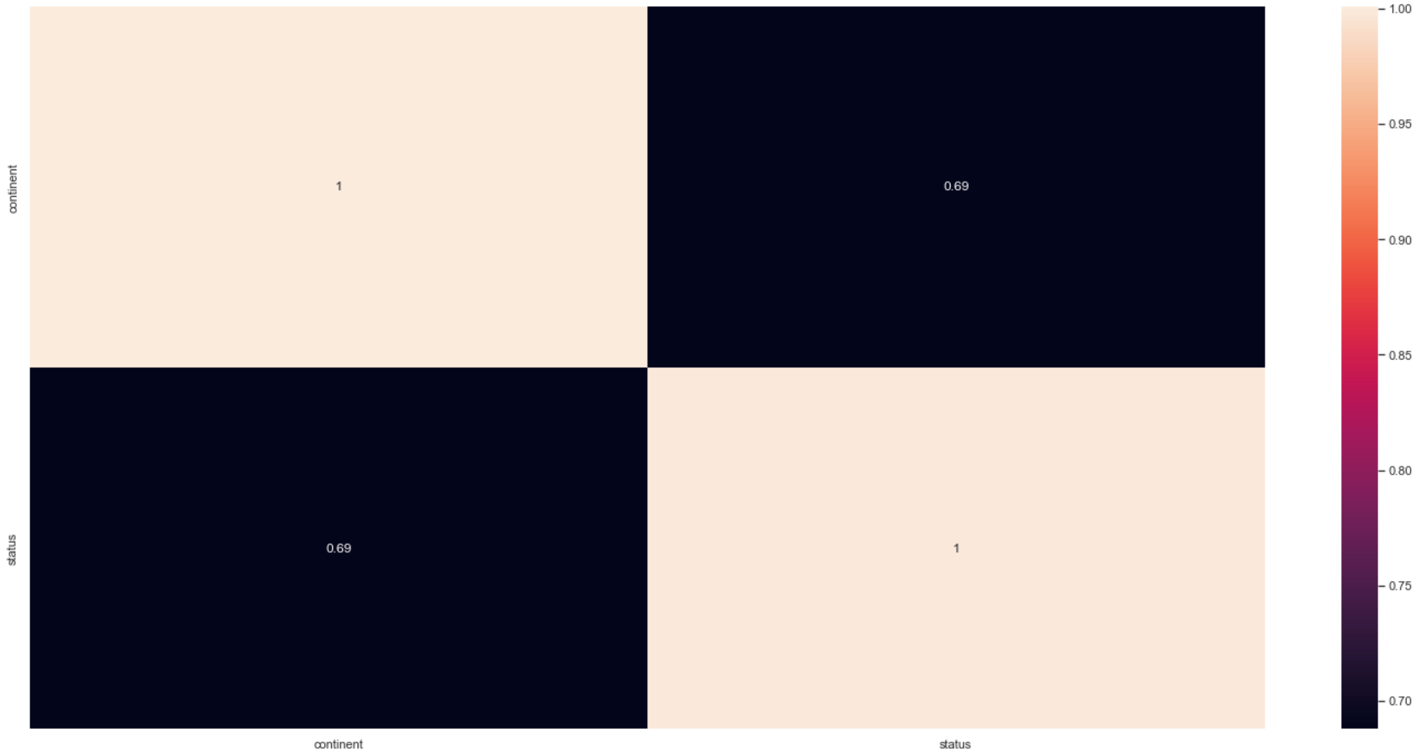
```
correlation = num_attributes4.corr( method='pearson' )
sns.heatmap( correlation, annot=True );
```



Categorical Heatmap

In [78]:

```
# # # Calculate cramer V
a = df2.select_dtypes( include='object' )

a1 = cramer_v( a['continent'], a['continent'] )
a2 = cramer_v( a['continent'], a['status'] )
a3 = cramer_v( a['status'], a['continent'] )
a4 = cramer_v( a['status'], a['status'] )



# # Final dataset
d = pd.DataFrame( {'continent': [a1, a2],'status': [a3, a4]})
d = d.set_index( d.columns )
sns.heatmap( d, annot=True );
```



Preprocessing Data

In [136…

```
jupyter_settings()
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [137…

```
df3=df2.copy()
```

In [138…

```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2872 entries, 0 to 2937
Data columns (total 32 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   country                          2872 non-null   object 
 1   year                             2872 non-null   object 
 2   status                           2872 non-null   object 
 3   life_expectancy                  2872 non-null   float64
 4   adult_mortality                  2872 non-null   float64
 5   infant_deaths                    2872 non-null   int64  
 6   alcohol                          2872 non-null   float64
 7   percentage_expenditure           2872 non-null   float64
 8   hepatitis_b                      2872 non-null   float64
 9   measles                          2872 non-null   int64  
 10  bmi                              2872 non-null   float64
 11  under_five_deaths                2872 non-null   int64  
 12  polio                            2872 non-null   float64
 13  total_expenditure                2872 non-null   float64
 14  diphtheria                       2872 non-null   float64
 15  hiv_aids                         2872 non-null   float64
 16  thinness_1_19_years              2872 non-null   float64
 17  thinness_5_9_years               2872 non-null   float64
 18  income_composition_of_resources  2872 non-null   float64
 19  schooling                        2872 non-null   float64
 20  emission                         2872 non-null   float64
```

```
 21   pop_male                          2872 non-null   float64
 22   pop_female                        2872 non-null   float64
 23   pop_total                         2872 non-null   float64
 24   density                           2872 non-null   float64
 25   lat                               2872 non-null   float64
 26   long                              2872 non-null   float64
 27   continent                         2872 non-null   object
 28   code                              2872 non-null   object
 29   perc_female                       2872 non-null   float64
 30   emission_pop                      2872 non-null   float64
 31   gdp                               2872 non-null   float64
dtypes: float64(24), int64(3), object(5)
memory usage: 805.0+ KB
```

In [139…

```
rs = RobustScaler()
mms = MinMaxScaler()
le=LabelEncoder()
```

Robust Scaler

Variaveis a serem rescaladas: (problema con range)

- emission
- gdp
- infant_deaths
- percentage_expenditure
- measles
- under_five_deaths
- pop_male
- pop_female
- pop_total
- density
- perc_female
- emission_pop

In [140…

```
df3['emission'] = rs.fit_transform( df3[['emission']].values )
df3['gdp'] = rs.fit_transform( df3[['gdp']].values )
df3['infant_deaths'] = rs.fit_transform( df3[['infant_deaths']].values )
df3['percentage_expenditure'] = rs.fit_transform( df3[['percentage_expenditure']].values )
df3['measles'] = rs.fit_transform( df3[['measles']].values )
df3['under_five_deaths'] = rs.fit_transform( df3[['under_five_deaths']].values )
df3['pop_male'] = rs.fit_transform( df3[['pop_male']].values )
df3['pop_female'] = rs.fit_transform( df3[['pop_female']].values )
df3['pop_total'] = rs.fit_transform( df3[['pop_total']].values )
df3['density'] = rs.fit_transform( df3[['density']].values )
df3['perc_female'] = rs.fit_transform( df3[['perc_female']].values )
df3['emission_pop'] = rs.fit_transform( df3[['emission_pop']].values )
```

Transformation

Order Encoder

- year

In [141…

```
a=['2000','2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','20
b=np.arange(1,17,1)
```

```
year_dict = dict(zip(a, b))
df3['year']=df3['year'].map(year_dict)
df3.head()
```

Out[141…

	country	year	status	life_expectancy	adult_mortality	infant_deaths	alcohol	percentage_expenditure	h
0	Afghanistan	16	Developing	65.00	263.00	2.81	0.01	0.01	
1	Afghanistan	15	Developing	59.90	271.00	2.90	0.01	0.01	
2	Afghanistan	14	Developing	59.90	268.00	3.00	0.01	0.01	
3	Afghanistan	13	Developing	59.50	272.00	3.14	0.01	0.02	
4	Afghanistan	12	Developing	59.20	275.00	3.24	0.01	-0.13	

Label Encoder

- status
- continent
- code
- country

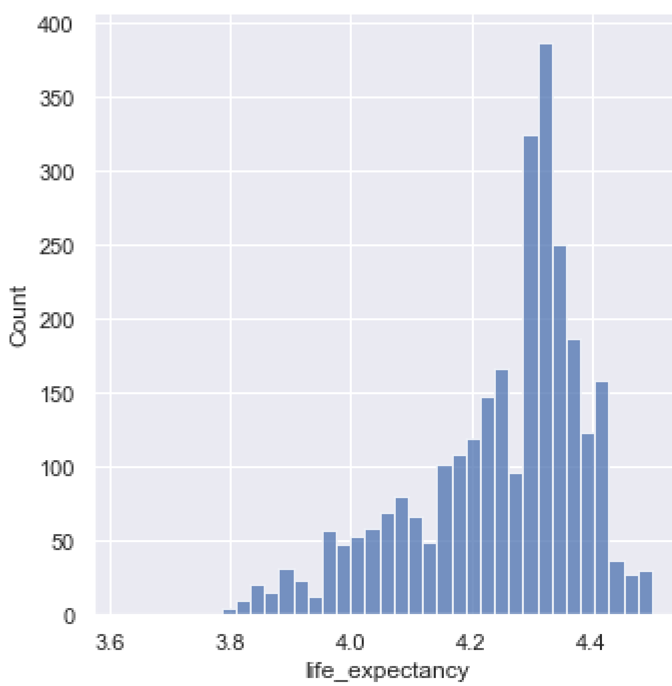
In [142…

```
# Label Encoder
df3['status']=le.fit_transform(df3['status'])
df3['continent']=le.fit_transform(df3['continent'])
df3['country']=le.fit_transform(df3['country'])
df3['code']=le.fit_transform(df3['code'])
```

Response varable Transformation

In [143…

```
df3['life_expectancy']=np.log1p(df3['life_expectancy'])
sns.displot(df3['life_expectancy']);
```



Features Selection

In [144…

```
df4=df3.copy()
```

```
df4.head()
```

Out[144…

	country	year	status	life_expectancy	adult_mortality	infant_deaths	alcohol	percentage_expenditure	hepatitis_l
0	0	16	1	4.19	263.00	2.81	0.01	0.01	65.0(
1	0	15	1	4.11	271.00	2.90	0.01	0.01	62.0(
2	0	14	1	4.11	268.00	3.00	0.01	0.01	64.0(
3	0	13	1	4.10	272.00	3.14	0.01	0.02	67.0(
4	0	12	1	4.10	275.00	3.24	0.01	-0.13	68.0(

Split dataframe into training and test dataset

In [145…

```
# features deletar variaveis derivdas
cols_drop = ['code','thinness_5_9_years','pop_male','pop_female']

df4=df4.drop(cols_drop, axis=1)
```

In [146…

```
# training dataset
X_train = df4[df4['year'] < 14]
y_train = X_train['life_expectancy']

# # test dataset(sem contaminaçao)
# X_test = df1[df1['year'] >= 14]
# y_test = X_test['life_expectancy']


# test dataset
X_test = df4[df4['year'] >= 14]
y_test = X_test['life_expectancy']

print( 'Training Min Date: {}'.format( X_train['year'].min() ) )
print( 'Training Max Date: {}'.format( X_train['year'].max() ) )

print( '\nTest Min Date: {}'.format( X_test['year'].min() ) )
print( 'Test Max Date: {}'.format( X_test['year'].max() ) )
```

```
Training Min Date: 1
Training Max Date: 13

Test Min Date: 14
Test Max Date: 16
```

Seleçao por subset -Boruta

In [147…

```
# ## training and test dataset for Boruta
X_train_n = X_train.drop(['life_expectancy','year'],axis=1).values
y_train_n = y_train.values.ravel()


# # ## define RandomForestRegressor
rf = RandomForestRegressor( n_jobs=-1 ) # cria as arvores em paralelo

# # define Boruta
boruta = BorutaPy( rf, n_estimators='auto', verbose=2, random_state=15 ).fit( X_train_n, y
```

```
Iteration:      1 / 100
Confirmed:      0
```

```
Tentative:      26
Rejected:       0
Iteration:      2 / 100
Confirmed:      0
Tentative:      26
Rejected:       0
Iteration:      3 / 100
Confirmed:      0
Tentative:      26
Rejected:       0
Iteration:      4 / 100
Confirmed:      0
Tentative:      26
Rejected:       0
Iteration:      5 / 100
Confirmed:      0
Tentative:      26
Rejected:       0
Iteration:      6 / 100
Confirmed:      0
Tentative:      26
Rejected:       0
Iteration:      7 / 100
Confirmed:      0
Tentative:      26
Rejected:       0
Iteration:      8 / 100
Confirmed:      23
Tentative:      3
Rejected:       0
Iteration:      9 / 100
Confirmed:      23
Tentative:      3
Rejected:       0
Iteration:      10 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      11 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      12 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      13 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      14 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      15 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      16 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      17 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
```

```
Iteration:      18 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      19 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      20 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      21 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      22 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      23 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      24 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      25 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      26 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      27 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      28 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      29 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      30 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      31 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      32 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      33 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      34 / 100
Confirmed:      23
```

```
Tentative:      2
Rejected:       1
Iteration:      35 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      36 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      37 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      38 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      39 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      40 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      41 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      42 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      43 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      44 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      45 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      46 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      47 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      48 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      49 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      50 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
```

```
Iteration:      51 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      52 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      53 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      54 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      55 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      56 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      57 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      58 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      59 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      60 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      61 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      62 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      63 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      64 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      65 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      66 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      67 / 100
Confirmed:      23
```

```
Tentative:      2
Rejected:       1
Iteration:      68 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      69 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      70 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      71 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      72 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      73 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      74 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      75 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      76 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      77 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      78 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      79 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      80 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      81 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      82 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      83 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
```

```
Iteration:      84 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      85 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      86 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      87 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      88 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      89 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      90 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      91 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      92 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      93 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      94 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      95 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      96 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      97 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      98 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
Iteration:      99 / 100
Confirmed:      23
Tentative:      2
Rejected:       1
```

```
BorutaPy finished running.

Iteration:      100 / 100
Confirmed:      23
Tentative:      1
Rejected:       1
```

In [148…

```
cols_selected = boruta.support_.tolist()

## best features
X_train_fs = X_train.drop( [ 'life_expectancy','year'], axis=1 )
cols_selected_boruta = X_train_fs.iloc[:, cols_selected].columns.to_list()

# ## not selected boruta
cols_not_selected_boruta = list( np.setdiff1d( X_train_fs.columns, cols_selected_boruta )
```

In [149…

```
cols_selected_boruta
```

Out[149…

```
['country',
 'adult_mortality',
 'infant_deaths',
 'alcohol',
 'percentage_expenditure',
 'bmi',
 'under_five_deaths',
 'polio',
 'total_expenditure',
 'diphtheria',
 'hiv_aids',
 'thinness_1_19_years',
 'income_composition_of_resources',
 'schooling',
 'emission',
 'pop_total',
 'density',
 'lat',
 'long',
 'continent',
 'perc_female',
 'emission_pop',
 'gdp']
```

In [150…

```
cols_not_selected_boruta
```

Out[150…

```
['hepatitis_b', 'measles', 'status']
```

In [151…

```
correlation = df4[['status','life_expectancy']].corr( method='pearson' )
sns.heatmap( correlation, annot=True );
```

In [152…

```
features_selection= [
 'country',
 'adult_mortality',
 'infant_deaths',
 'alcohol',
 'percentage_expenditure',
 'bmi',
 'under_five_deaths',
 'polio',
 'total_expenditure',
 'diphtheria',
 'hiv_aids',
 'thinness_1_19_years',
 'income_composition_of_resources',
 'schooling',
 'emission',
 'pop_total',
 'density',
 'lat',
 'long',
 'continent',
 'perc_female',
 'emission_pop',
 'gdp']

# columns to add
feat_to_add = ['life_expectancy','year']

features_selection_full = features_selection.copy()
features_selection_full.extend( feat_to_add )
```

In [153…

```
features_selection_full
```

Out[153…

```
['country',
 'adult_mortality',
 'infant_deaths',
 'alcohol',
 'percentage_expenditure',
```

```
 'bmi',
 'under_five_deaths',
 'polio',
 'total_expenditure',
 'diphtheria',
 'hiv_aids',
 'thinness_1_19_years',
 'income_composition_of_resources',
 'schooling',
 'emission',
 'pop_total',
 'density',
 'lat',
 'long',
 'continent',
 'perc_female',
 'emission_pop',
 'gdp',
 'life_expectancy',
 'year']
```

Machine Learning Modelling

In [154…

```
x_train = X_train[ features_selection ]
x_test = X_test[ features_selection ]

# Time Series Data Preparation
x_training = X_train[ features_selection_full ]
```

Linear regression model

In [155…

```
# model
lr = LinearRegression().fit( x_train, y_train )

# prediction
yhat_lr = lr.predict( x_test )
```

In [156…

```
print( 'Score Train: {}'.format( lr.score(x_train,y_train) ))
print( 'Score Test: {}'.format( lr.score(x_test,y_test) ))

# performance
lr_result = ml_error( 'Linear Regression', np.expm1( y_test ), np.expm1( yhat_lr ) )
lr_result
```

```
Score Train: 0.8398985047240835
Score Test: 0.809038190681365
```

Out[156…

	Model Name	MAE	MAPE	RMSE
0	Linear Regression	2.83	0.04	3.67

Linear Regression Model - Cross Validation

In [157…

```
lr_result_cv = cross_validation( x_training, 3, 'Linear Regression', lr, verbose=False )
lr_result_cv
```

Out[157…

	Model Name	MAE CV	MAPE CV	RMSE CV
0	Linear Regression	3.13 +/- 0.2	0.05 +/- 0.004	4.07 +/- 0.16

Linear Regression Regularized Model - Lasso

In [158…

```
# model
lrr = Lasso( alpha=0.01 ).fit( x_train, y_train )

# prediction
yhat_lrr = lrr.predict( x_test )
```

In [159…

```
print( 'Score Train: {}'.format( lrr.score(x_train,y_train) ))
print( 'Score Test: {}'.format( lrr.score(x_test,y_test) ))

# performance
lrr_result = ml_error( 'Linear Regression - Lasso', np.expm1( y_test ), np.expm1( yhat_lrr
lrr_result
```

```
Score Train: 0.8205853310278443
Score Test: 0.8043729632423837
```

Out[159…

	Model Name	MAE	MAPE	RMSE
0	Linear Regression - Lasso	2.78	0.04	3.68

In [160…

```
## Usar os ploters da Aula prática - Colab - Regressão linear, Ridge e LASSO
## Aula prática - Colab - Transformação de Dados
```

Linear Regression Regularized Model - Cross Validation

In [161…

```
lrr_result_cv = cross_validation( x_training, 3, 'Linear Regression - Lasso', lrr, verbose
lrr_result_cv
```

Out[161…

	Model Name	MAE CV	MAPE CV	RMSE CV
0	Linear Regression - Lasso	3.12 +/- 0.12	0.05 +/- 0.003	4.09 +/- 0.13

Linear Regression Regularized Model - Ridge

In [162…

```
# model
lrri = Ridge( alpha=0.01 ).fit( x_train, y_train )

# prediction
yhat_lrri = lrri.predict( x_test )
```

In [163…

```
print( 'Score Train: {}'.format( lrri.score(x_train,y_train) ))
print( 'Score Test: {}'.format( lrri.score(x_test,y_test) ))

# performance
lrri_result = ml_error( 'Linear Regression - Ridge', np.expm1( y_test ), np.expm1( yhat_lr
lrri_result
```

```
Score Train: 0.8398985045244436
Score Test: 0.8090367597346991
```

Out[163…

	Model Name	MAE	MAPE	RMSE
0	Linear Regression - Ridge	2.83	0.04	3.67

Linear Regression Regularized Model - Cross Validation

In [164…

```
lrri_result_cv = cross_validation( x_training, 3, 'Linear Regression - Ridge', lrri, verbo
lrri_result_cv
```

Out[164…

	Model Name	MAE CV	MAPE CV	RMSE CV
0	Linear Regression - Ridge	3.13 +/- 0.2	0.05 +/- 0.004	4.07 +/- 0.16

Random Forest Regressor

In [165…

```
RandomForestRegressor().get_params()
```

Out[165…

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'squared_error',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

In [166…

```
# model
rf = RandomForestRegressor( bootstrap= True,
                            criterion='mse',
                            min_samples_leaf= 1,
                            min_samples_split= 2,
                            n_estimators=100,
                            n_jobs=-1,
                           ).fit( x_train, y_train )

# prediction
yhat_rf = rf.predict( x_test )
```

In [167…

```
print( 'Score Train: {}'.format( rf.score(x_train,y_train) ))
print( 'Score Test: {}'.format( rf.score(x_test,y_test) ))

# performance
rf_result = ml_error( 'Random Forest Regressor', np.expm1( y_test ), np.expm1( yhat_rf ) )
rf_result
```

```
Score Train: 0.9959760780299197
Score Test: 0.9198008391292392
```

Out[167…

	Model Name	MAE	MAPE	RMSE
0	Random Forest Regressor	1.46	0.02	2.29

Ramdom Forest- Cross Validation

In [168…

```
rf_result_cv = cross_validation( x_training, 3, 'Random Forest Regressor', rf, verbose=Fal
rf_result_cv
```

Out[168...

	Model Name	MAE CV	MAPE CV	RMSE CV
0	Random Forest Regressor	1.57 +/- 0.07	0.02 +/- 0.001	2.42 +/- 0.09

XGBoost Regressor

In [169...

```
# model
model_xgb = xgb.XGBRegressor( objective='reg:squarederror',
                              n_estimators=100,
                              max_depth=5,
                              subsample=0.7
                             ).fit( x_train, y_train )

# prediction
yhat_xgb = model_xgb.predict( x_test )

# performance
xgb_result = ml_error( 'XGBoost Regressor', np.expm1( y_test ), np.expm1( yhat_xgb ) )
xgb_result
```

Out[169...

	Model Name	MAE	MAPE	RMSE
0	XGBoost Regressor	1.58	0.02	2.31

XGBoost - Cross Validation

In [170...

```
xgb_result_cv = cross_validation( x_training, 3, 'XGBoost Regressor', model_xgb, verbose=T
xgb_result_cv
```

```
KFold Number: 3

KFold Number: 2

KFold Number: 1
```

Out[170...

	Model Name	MAE CV	MAPE CV	RMSE CV
0	XGBoost Regressor	1.72 +/- 0.05	0.03 +/- 0.001	2.48 +/- 0.08

Performance Metrics

In [171...

```
modelling_result = pd.concat( [lr_result, lrr_result,lrri_result, rf_result, xgb_result] )
modelling_result.sort_values( 'RMSE' )
```

Out[171...

	Model Name	MAE	MAPE	RMSE
0	Random Forest Regressor	1.46	0.02	2.29
0	XGBoost Regressor	1.58	0.02	2.31
0	Linear Regression	2.83	0.04	3.67
0	Linear Regression - Ridge	2.83	0.04	3.67
0	Linear Regression - Lasso	2.78	0.04	3.68

Real Performance - Cross Validation

In [172…

```
modelling_result_cv = pd.concat( [lr_result_cv, lrr_result_cv, lrri_result_cv,rf_result_cv
modelling_result_cv.sort_values( 'RMSE CV' )
```

Out[172…

	Model Name	MAE CV	MAPE CV	RMSE CV
0	Random Forest Regressor	1.57 +/- 0.07	0.02 +/- 0.001	2.42 +/- 0.09
0	XGBoost Regressor	1.72 +/- 0.05	0.03 +/- 0.001	2.48 +/- 0.08
0	Linear Regression	3.13 +/- 0.2	0.05 +/- 0.004	4.07 +/- 0.16
0	Linear Regression - Ridge	3.13 +/- 0.2	0.05 +/- 0.004	4.07 +/- 0.16
0	Linear Regression - Lasso	3.12 +/- 0.12	0.05 +/- 0.003	4.09 +/- 0.13

Fine Tunning

Random search

In [173…

```
from sklearn.model_selection        import RandomizedSearchCV, GridSearchCV
from sklearn.ensemble               import RandomForestClassifier
```

In [174…

```
# Create the parameter grid based on the results of random search
bootstrap= [1,0],
criterion=['mse']
min_samples_leaf= [0.5, 0.75, 1, 2],
min_samples_split= [0.5, 0.75, 1, 2],
n_estimators= [10, 40, 80, 160, 250]


param_grid = {
    'bootstrap': bootstrap,
    'criterion': criterion,
    'min_samples_leaf': min_samples_leaf,
    'min_samples_split': min_samples_split,
    'n_estimators':n_estimators
}

n_combinations=1
for k in param_grid.keys():n_combinations*= len(param_grid[k])

print('numbers_of_combination =', n_combinations)

param_grid
```

```
numbers_of_combination = 5
```

Out[174…

```
{'bootstrap': ([1, 0],),
 'criterion': ['mse'],
 'min_samples_leaf': ([0.5, 0.75, 1, 2],),
 'min_samples_split': ([0.5, 0.75, 1, 2],),
 'n_estimators': [10, 40, 80, 160, 250]}
```

In [175…

```
# rf = RandomForestRegressor()

# model_rf = GridSearchCV(estimator = rf,
#                                 param_grid= param_grid,
#                                 cv = 3,
#                                 return_train_score=1,
```

```
#                                   n_jobs = -1,
#                                   verbose = 2)
#   # performance
# result = cross_validation( x_training, 3, 'Random Forest Regressor', model_rf, verbose=
# #final_result = pd.concat( [final_result, result] )

# #final_result
```

XGBoost

In [176…

```
param = {
    'n_estimators': [100, 500, 1000,10000],
    'max_depth': [1, 2, 5],
    'subsample': [0.1, 0.5, 0.7]
   # 'min_child_weight': [10, 30, 50]
        }


MAX_EVAL = 10
```

In [177…

```
final_result = pd.DataFrame()

for i in range( MAX_EVAL ):
    # choose values for parameters randomly
    hp = { k: np.random.choice( v, 1 )[0] for k, v in param.items() }
    print( hp )

    # model
    model_xgb = xgb.XGBRegressor( objective='reg:squarederror',
                                  n_estimators=hp['n_estimators'],
                                  max_depth=hp['max_depth'],
                                  subsample=hp['subsample'],
                                  #min_child_weight=hp['min_child_weight']
                                )

    # performance
    result = cross_validation( x_training, 3, 'XGBoost Regressor', model_xgb, verbose=True
    final_result = pd.concat( [final_result, result] )

final_result
```

```
{'n_estimators': 500, 'max_depth': 5, 'subsample': 0.5}

KFold Number: 3

KFold Number: 2

KFold Number: 1
{'n_estimators': 100, 'max_depth': 2, 'subsample': 0.7}

KFold Number: 3

KFold Number: 2

KFold Number: 1
{'n_estimators': 500, 'max_depth': 1, 'subsample': 0.1}

KFold Number: 3

KFold Number: 2

KFold Number: 1
```

```
{'n_estimators': 100, 'max_depth': 2, 'subsample': 0.5}

KFold Number: 3

KFold Number: 2

KFold Number: 1
{'n_estimators': 100, 'max_depth': 1, 'subsample': 0.1}

KFold Number: 3

KFold Number: 2

KFold Number: 1
{'n_estimators': 500, 'max_depth': 2, 'subsample': 0.1}

KFold Number: 3

KFold Number: 2

KFold Number: 1
{'n_estimators': 100, 'max_depth': 5, 'subsample': 0.5}

KFold Number: 3

KFold Number: 2

KFold Number: 1
{'n_estimators': 500, 'max_depth': 2, 'subsample': 0.5}

KFold Number: 3

KFold Number: 2

KFold Number: 1
{'n_estimators': 500, 'max_depth': 2, 'subsample': 0.7}

KFold Number: 3

KFold Number: 2

KFold Number: 1
{'n_estimators': 1000, 'max_depth': 2, 'subsample': 0.1}

KFold Number: 3

KFold Number: 2

KFold Number: 1
```

Out[177…

	Model Name	MAE CV	MAPE CV	RMSE CV
0	XGBoost Regressor	1.75 +/- 0.06	0.03 +/- 0.002	2.54 +/- 0.09
0	XGBoost Regressor	1.89 +/- 0.05	0.03 +/- 0.001	2.67 +/- 0.05
0	XGBoost Regressor	2.37 +/- 0.12	0.04 +/- 0.003	3.14 +/- 0.13
0	XGBoost Regressor	1.92 +/- 0.03	0.03 +/- 0.0	2.67 +/- 0.06
0	XGBoost Regressor	2.65 +/- 0.15	0.04 +/- 0.003	3.54 +/- 0.19
0	XGBoost Regressor	2.76 +/- 0.23	0.04 +/- 0.004	3.64 +/- 0.24
0	XGBoost Regressor	1.75 +/- 0.07	0.03 +/- 0.002	2.55 +/- 0.08
0	XGBoost Regressor	1.75 +/- 0.08	0.03 +/- 0.002	2.51 +/- 0.08

	Model Name	MAE CV	MAPE CV	RMSE CV
0	XGBoost Regressor	1.74 +/- 0.03	0.03 +/- 0.001	2.52 +/- 0.06
0	XGBoost Regressor	2.97 +/- 0.2	0.04 +/- 0.004	3.89 +/- 0.2

Final Model

Final tune XGBoost

In [178…

```
param_tuned = {
    'n_estimators': 500,
    'max_depth': 2,
    'subsample': 0.7,
        }
```

In [179…

```
# model
model_xgb_tuned = xgb.XGBRegressor( objective='reg:squarederror',
                                    n_estimators=param_tuned['n_estimators'],
                                    max_depth=param_tuned['max_depth'],
                                    subsample=param_tuned['subsample'],
                                    ).fit( x_train, y_train )

# prediction
yhat_xgb_tuned = model_xgb_tuned.predict( x_test )

# performance
xgb_result_tuned = ml_error( 'XGBoost Regressor', np.expm1( y_test ), np.expm1( yhat_xgb_t
xgb_result_tuned
```

Out[179…

	Model Name	MAE	MAPE	RMSE
0	XGBoost Regressor	1.67	0.02	2.39

Final Model Random Forest

In [180…

```
# model
rf = RandomForestRegressor( bootstrap= True,
                            criterion='mse',
                            min_samples_leaf= 1,
                            min_samples_split= 2,
                            n_estimators=100,
                            n_jobs=-1,
                           ).fit( x_train, y_train )

# prediction
yhat_rf = rf.predict( x_test )

print( 'Score Train: {}'.format( rf.score(x_train,y_train) ))
print( 'Score Test: {}'.format( rf.score(x_test,y_test) ))

# performance
rf_result = ml_error( 'Random Forest Regressor', np.expm1( y_test ), np.expm1( yhat_rf ) )
rf_result
```

```
Score Train: 0.9957837174309045
Score Test: 0.9206992371637881
```

Out[180…

	Model Name	MAE	MAPE	RMSE

	Model Name	MAE	MAPE	RMSE
0	Random Forest Regressor	1.47	0.02	2.28

Translation and interpretation of the error

In [181…

```
df5 = X_test[ features_selection_full ]

# rescale
df5['life_expectancy'] = np.expm1( df5['life_expectancy'] )
df5['predictions'] = np.expm1( yhat_rf )
```

Business Performance

In [ ]:

```
#sns.scatterplot( x='country', y='MAPE', data=df5);
```

In [ ]:

```
# sum of predictions
df51 = df5[['country', 'predictions']].groupby( 'country' ).mean().reset_index()

# # MAE and MAPE
df5_aux1 = df5[['country', 'life_expectancy', 'predictions']].groupby( 'country' ).apply(
df5_aux2 = df5[['country', 'life_expectancy', 'predictions']].groupby( 'country' ).apply(

# Merge
df5_aux3 = pd.merge( df5_aux1, df5_aux2, how='inner', on='country' )
df52 = pd.merge( df51, df5_aux3, how='inner', on='country' )

# # Scenarios
df52['worst_scenario'] = df52['predictions'] - df52['MAE']
df52['best_scenario'] = df52['predictions'] + df52['MAE']

# # order columns
df52 = df52[['country', 'predictions', 'worst_scenario', 'best_scenario', 'MAE', 'MAPE']]
```

In [ ]:

```
a=['2000','2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','20
b=np.arange(1,17,1)
year_dict = dict(zip(a, b))
df5['year']=df5['year'].map(year_dict)

df52.sort_values( 'MAPE', ascending=False ).head()
```

Total Performance

In [ ]:

```
df53 = df52[['predictions', 'worst_scenario', 'best_scenario']].apply( lambda x: np.mean(
                                                              axis=0 ).reset_index().rename
                                                              {'index': 'Scenario', 0:'Years
df53['Years'] = df53['Years'].map( '{:,.2f}'.format )
df53
```

Machine Learning Performance

In [ ]:

```
df5['error'] = df5['life_expectancy'] - df5['predictions']
df5['error_rate'] = df5['predictions'] / df5['life_expectancy']
```

```
a=np.arange(1,17,1)
b=['2000','2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','20
year_dict = dict(zip(a, b))
df5['year']=df5['year'].map(year_dict)
```

In [ ]:

```
plt.subplot( 2, 2, 1 )
sns.lineplot( x='year', y='life_expectancy', data=df5, label='life_expectancy' )
sns.lineplot( x='year', y='predictions', data=df5, label='PREDICTIONS' );

plt.subplot( 2, 2, 2 )
sns.lineplot( x='year', y='error_rate', data=df5)
plt.axhline( 1, linestyle='--')
```

In [ ]:

```
plt.subplot( 2, 2, 3 )
sns.distplot( df5['error'] )

plt.subplot( 2, 2, 4 )
sns.scatterplot( df5['predictions'], df5['error'] );
```

In [ ]:

In [ ]:

In [ ]: