```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
from scipy.stats.mstats import winsorize
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
import os
%matplotlib inline
```

```
df = pd.read_csv('/content/Life Expectancy Data.csv', header=0)
```

```
df.head()
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109

5 rows × 22 columns

```
orig_cols = list(df.columns)
new_cols = []
for col in orig_cols:
    new_cols.append(col.strip().replace('  ', ' ').replace(' ', '_').lower())
df.columns = new_cols
```

```
df.columns
```

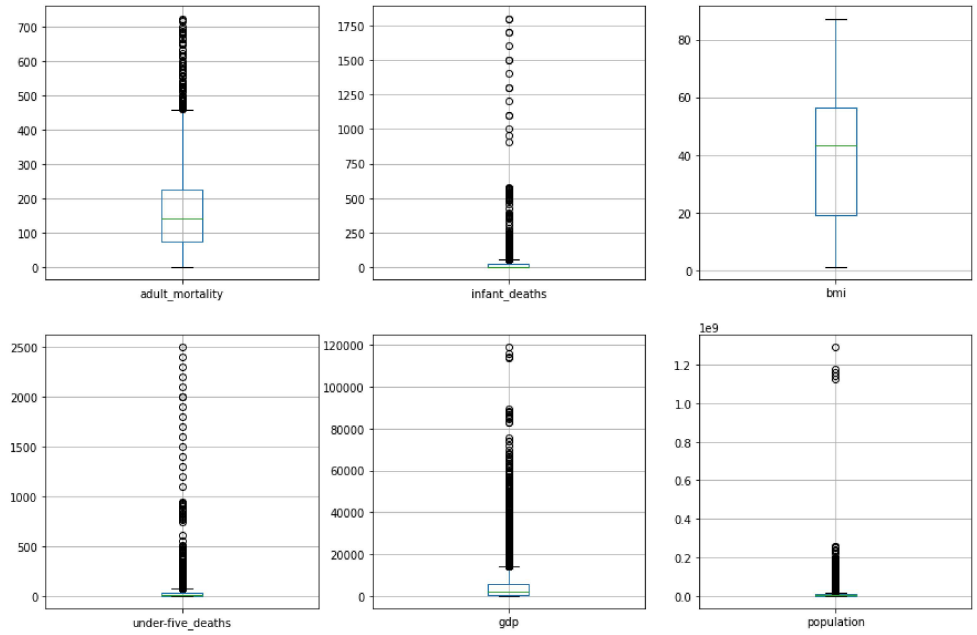
```
Index(['country', 'year', 'status', 'life_expectancy', 'adult_mortality',
       'infant_deaths', 'alcohol', 'percentage_expenditure', 'hepatitis_b',
       'measles', 'bmi', 'under-five_deaths', 'polio', 'total_expenditure',
       'diphtheria', 'hiv/aids', 'gdp', 'population', 'thinness_1-19_years',
       'thinness_5-9_years', 'income_composition_of_resources', 'schooling'],
      dtype='object')
```

```
df.rename(columns={'thinness_1-19_years':'thinness_10-19_years'}, inplace=True)
```

```
df.describe().iloc[:, 1:]
```

	life_expectancy	adult_mortality	infant_deaths	alcohol	percentage_exper
count	2928.000000	2928.000000	2938.000000	2744.000000	2938.
mean	69.224932	164.796448	30.303948	4.602861	738.
std	9.523867	124.292079	117.926501	4.052413	1987.
min	36.300000	1.000000	0.000000	0.010000	0.
25%	63.100000	74.000000	0.000000	0.877500	4.
50%	72.100000	144.000000	3.000000	3.755000	64.
75%	75.700000	228.000000	22.000000	7.702500	441.
max	89.000000	723.000000	1800.000000	17.870000	19479

```
plt.figure(figsize=(15,10))
for i, col in enumerate(['adult_mortality', 'infant_deaths', 'bmi', 'under-five_deaths', 'gdp', 'population'], start
    plt.subplot(2, 3, i)
    df.boxplot(col)
```




```
df.isna().sum()
```

```
country                             0
year                                0
status                              0
life_expectancy                    10
adult_mortality                    10
infant_deaths                       0
alcohol                           194
percentage_expenditure              0
hepatitis_b                       553
measles                             0
bmi                                34
under-five_deaths                   0
polio                              19
total_expenditure                 226
diphtheria                         19
hiv/aids                            0
gdp                               448
population                        652
thinness_10-19_years               34
thinness_5-9_years                 34
income_composition_of_resources   167
schooling                         163
dtype: int64
```

```
mort_5_percentile = np.percentile(df.adult_mortality.dropna(), 5)
df.adult_mortality = df.apply(lambda x: np.nan if x.adult_mortality < mort_5_percentile else x.adult_mortality, axis
df.infant_deaths = df.infant_deaths.replace(0, np.nan)
df.bmi = df.apply(lambda x: np.nan if (x.bmi < 10 or x.bmi > 50) else x.bmi, axis=1)
df['under-five_deaths'] = df['under-five_deaths'].replace(0, np.nan)
```

```
def nulls_breakdown(df=df):
    df_cols = list(df.columns)
    cols_total_count = len(list(df.columns))
    cols_count = 0
    for loc, col in enumerate(df_cols):
```

```
            null_count = df[col].isnull().sum()
            total_count = df[col].isnull().count()
            percent_null = round(null_count/total_count*100, 2)
            if null_count > 0:
                cols_count += 1
                print('[iloc = {}] {} has {} null values: {}% null'.format(loc, col, null_count, percent_null))
        cols_percent_null = round(cols_count/cols_total_count*100, 2)
        print('Out of {} total columns, {} contain null values; {}% columns contain null values.'.format(cols_total_cour
```

```
nulls_breakdown()
```

```
[iloc = 3] life_expectancy has 10 null values: 0.34% null
[iloc = 4] adult_mortality has 155 null values: 5.28% null
[iloc = 5] infant_deaths has 848 null values: 28.86% null
[iloc = 6] alcohol has 194 null values: 6.6% null
[iloc = 8] hepatitis_b has 553 null values: 18.82% null
[iloc = 10] bmi has 1456 null values: 49.56% null
[iloc = 11] under-five_deaths has 785 null values: 26.72% null
[iloc = 12] polio has 19 null values: 0.65% null
[iloc = 13] total_expenditure has 226 null values: 7.69% null
[iloc = 14] diphtheria has 19 null values: 0.65% null
[iloc = 16] gdp has 448 null values: 15.25% null
[iloc = 17] population has 652 null values: 22.19% null
[iloc = 18] thinness_10-19_years has 34 null values: 1.16% null
[iloc = 19] thinness_5-9_years has 34 null values: 1.16% null
[iloc = 20] income_composition_of_resources has 167 null values: 5.68% null
[iloc = 21] schooling has 163 null values: 5.55% null
Out of 22 total columns, 16 contain null values; 72.73% columns contain null values.
```

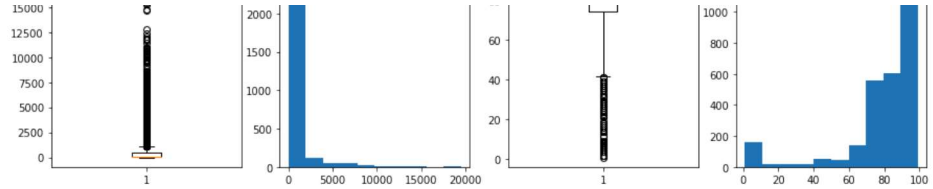
```
df.drop(columns='bmi', inplace=True)
```

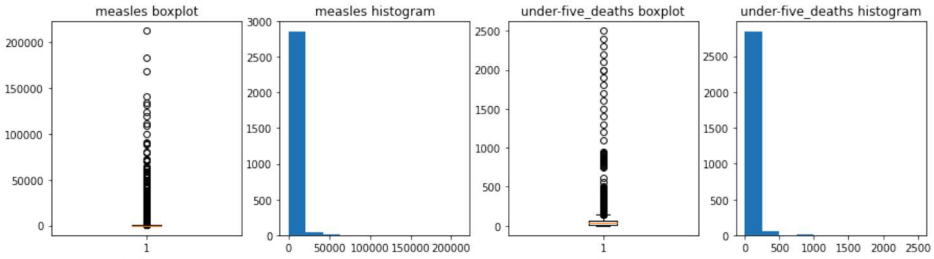
```
imputed_data = []
for year in list(df.year.unique()):
    year_data = df[df.year == year].copy()
    for col in list(year_data.columns)[3:]:
        year_data[col] = year_data[col].fillna(year_data[col].dropna().mean()).copy()
    imputed_data.append(year_data)
df = pd.concat(imputed_data).copy()
```

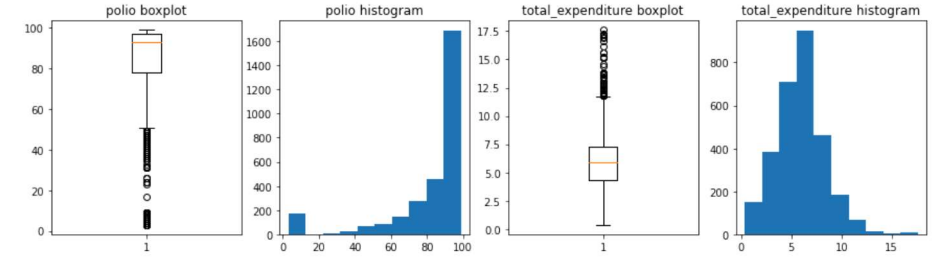
```
nulls_breakdown(df)
```

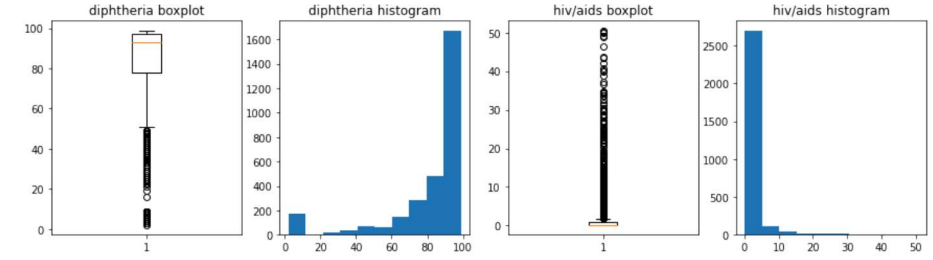
```
Out of 21 total columns, 0 contain null values; 0.0% columns contain null values.
```

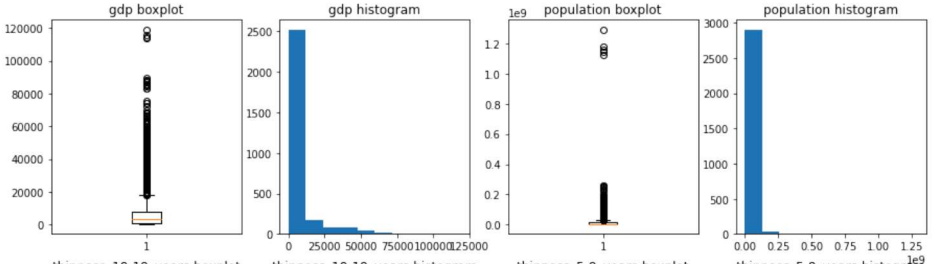
```
cont_vars = list(df.columns)[3:]
def outliers_visual(data):
    plt.figure(figsize=(15, 40))
    i = 0
    for col in cont_vars:
        i += 1
        plt.subplot(9, 4, i)
        plt.boxplot(data[col])
        plt.title('{} boxplot'.format(col))
        i += 1
        plt.subplot(9, 4, i)
        plt.hist(data[col])
        plt.title('{} histogram'.format(col))
    plt.show()
outliers_visual(df)
```

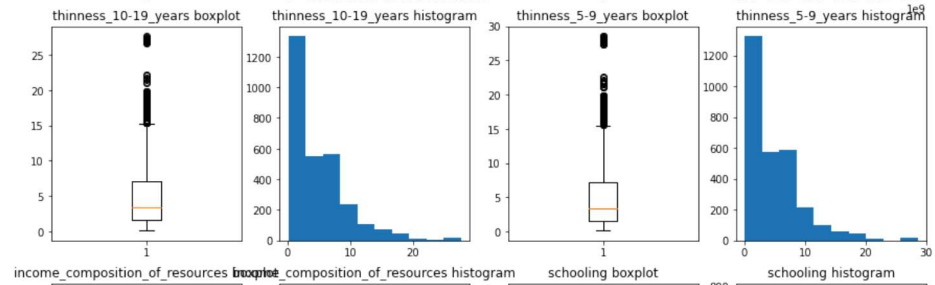



measles boxplot
measles histogram
under-five_deaths boxplot
under-five_deaths histogram


polio boxplot
polio histogram
total_expenditure boxplot
total_expenditure histogram


diphtheria boxplot
diphtheria histogram
hiv/aids boxplot
hiv/aids histogram


gdp boxplot
gdp histogram
population boxplot
population histogram


thinness_10-19_years boxplot
thinness_10-19_years histogram
thinness_5-9_years boxplot
thinness_5-9_years histogram


income_composition_of_resources boxplot
income_composition_of_resources histogram
schooling boxplot
schooling histogram

```
def outlier_count(col, data=df):
    print(15*'-' + col + 15*'-')
    q75, q25 = np.percentile(data[col], [75, 25])
    iqr = q75 - q25
    min_val = q25 - (iqr*1.5)
    max_val = q75 + (iqr*1.5)
    outlier_count = len(np.where((data[col] > max_val) | (data[col] < min_val))[0])
    outlier_percent = round(outlier_count/len(data[col])*100, 2)
    print('Number of outliers: {}'.format(outlier_count))
    print('Percent of data that is outlier: {}%'.format(outlier_percent))
```

```
for col in cont_vars:
    outlier_count(col)
```

```
---------------life_expectancy---------------
Number of outliers: 17
Percent of data that is outlier: 0.58%
---------------adult_mortality---------------
Number of outliers: 97
Percent of data that is outlier: 3.3%
---------------infant_deaths---------------
Number of outliers: 135
Percent of data that is outlier: 4.59%
---------------alcohol---------------
Number of outliers: 3
Percent of data that is outlier: 0.1%
---------------percentage_expenditure---------------
Number of outliers: 389
```

```
Percent of data that is outlier: 13.24%
---------------hepatitis_b---------------
Number of outliers: 222
Percent of data that is outlier: 7.56%
---------------measles---------------
Number of outliers: 542
Percent of data that is outlier: 18.45%
---------------under-five_deaths---------------
Number of outliers: 142
Percent of data that is outlier: 4.83%
---------------polio---------------
Number of outliers: 279
Percent of data that is outlier: 9.5%
---------------total_expenditure---------------
Number of outliers: 51
Percent of data that is outlier: 1.74%
---------------diphtheria---------------
Number of outliers: 298
Percent of data that is outlier: 10.14%
---------------hiv/aids---------------
Number of outliers: 542
Percent of data that is outlier: 18.45%
---------------gdp---------------
Number of outliers: 300
Percent of data that is outlier: 10.21%
---------------population---------------
Number of outliers: 203
Percent of data that is outlier: 6.91%
---------------thinness_10-19_years---------------
Number of outliers: 100
Percent of data that is outlier: 3.4%
---------------thinness_5-9_years---------------
Number of outliers: 99
Percent of data that is outlier: 3.37%
---------------income_composition_of_resources---------------
Number of outliers: 130
Percent of data that is outlier: 4.42%
---------------schooling---------------
Number of outliers: 77
Percent of data that is outlier: 2.62%
```

```
data=pd.get_dummies(df, columns=['country','status'])
X = data.drop('life_expectancy', axis=1)
y = data['life_expectancy']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, random_state=42)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((2350, 213), (588, 213), (2350,), (588,))
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor()
gbr.fit(X_train, y_train)
gbr_pred = gbr.predict(X_test)
print('R2 score is : {:.2f}'.format(r2_score(y_test, gbr_pred)))
```

```
R2 score is : 0.94
```

```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
outliers = pd.DataFrame(((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).sum())
outliers1= outliers[:60]
outliers2 = outliers[60:120]
outliers3 = outliers[120:180]
outliers4 = outliers[180:]
outliers1,outliers2,outliers3,outliers4
```

```
(                                0
 adult_mortality                97
 alcohol                         3
 country                         0
```

```
diphtheria                       298
gdp                              300
hepatitis_b                      222
hiv/aids                         542
income_composition_of_resources  130
infant_deaths                    135
life_expectancy                   17
measles                          542
percentage_expenditure           389
polio                            279
population                       203
schooling                         77
status                             0
thinness_10-19_years             100
thinness_5-9_years                99
total_expenditure                 51
under-five_deaths                142
year                               0,
Empty DataFrame
Columns: [0]
Index: [],
Empty DataFrame
Columns: [0]
Index: [],
Empty DataFrame
Columns: [0]
Index: [])
```

```
1  from sklearn.ensemble import RandomForestRegressor
2  rf = RandomForestRegressor()
3  rf.fit(X_train, y_train)
```

```
RandomForestRegressor()
```

```
1  rf_pred=rf.predict(X_test)
```

```
1  print('R2 score is : {:.2f}'.format(r2_score(y_test, rf_pred)))
```

```
R2 score is : 0.96
```

```
1  rf.feature_importances_
```

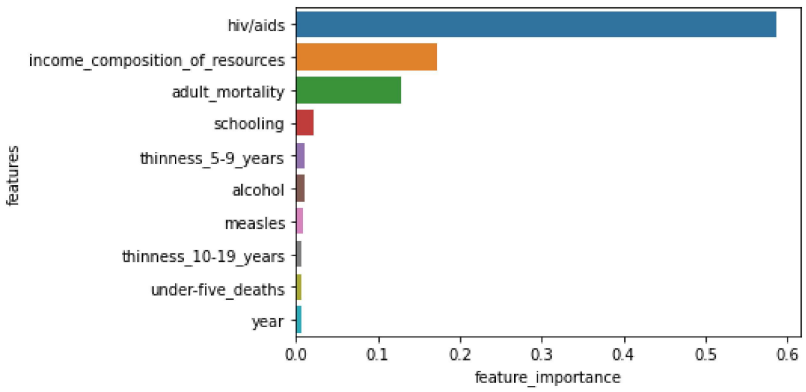
```
array([6.00791001e-03, 1.28939321e-01, 3.26100551e-03, 9.86419832e-03,
       5.02765740e-03, 2.17846056e-03, 7.98067254e-03, 6.42089596e-03,
       4.89367245e-03, 5.74286170e-03, 4.22662358e-03, 5.87124338e-01,
       3.07784093e-03, 2.91320046e-03, 6.43687308e-03, 1.05818258e-02,
       1.72407973e-01, 2.13619306e-02, 2.25015155e-06, 1.94925122e-05,
       1.76582409e-06, 4.78764533e-04, 1.36149705e-05, 4.69990003e-05,
       2.19254642e-06, 1.51342013e-06, 7.28547219e-06, 1.37030448e-04,
       5.21711505e-06, 2.56155271e-06, 2.18661046e-05, 5.43676652e-06,
       7.16938974e-06, 2.82880821e-05, 9.35300922e-05, 9.89013512e-05,
       3.29194530e-05, 1.20757384e-04, 1.45274223e-04, 4.69332958e-06,
       6.85465208e-06, 1.40357337e-06, 9.37322893e-06, 1.80101264e-05,
       7.79378542e-06, 9.19270371e-05, 4.13972522e-05, 3.72312543e-05,
       9.39730584e-06, 5.91096140e-05, 6.65797937e-06, 7.36905220e-05,
       3.05766430e-05, 2.08373939e-06, 2.52143787e-05, 9.24028761e-06,
       1.43492147e-18, 4.80110307e-04, 7.08832662e-06, 4.34145389e-04,
       2.04168651e-05, 5.59616892e-06, 1.68383345e-04, 2.55196993e-04,
       2.34534515e-05, 3.44684723e-05, 2.58300562e-06, 0.00000000e+00,
       7.12647185e-05, 1.07275610e-05, 1.58399623e-04, 8.42501978e-05,
       7.35638114e-05, 1.87815864e-04, 2.12012378e-04, 4.08420507e-05,
       3.33799867e-04, 6.90420075e-05, 1.42778259e-04, 5.24667186e-06,
       2.74377469e-06, 2.74956421e-06, 2.53721244e-05, 6.21561652e-05,
       3.00343468e-05, 2.21382298e-05, 8.69472732e-05, 2.26892792e-06,
       1.31222344e-05, 1.07447037e-04, 2.39480389e-04, 1.64019913e-04,
       1.72120129e-05, 9.10825738e-07, 8.54121708e-06, 2.42918648e-05,
       4.81062666e-06, 6.33302007e-05, 1.44279408e-05, 1.75157198e-06,
       1.20543728e-05, 3.25996824e-06, 1.98958731e-05, 8.65985084e-06,
       8.78540230e-05, 3.50413427e-05, 1.39538395e-04, 1.41644640e-06,
       3.62749982e-05, 4.73636724e-05, 3.75814896e-05, 7.24552178e-06,
       1.46118294e-06, 7.41727830e-06, 3.32103274e-05, 6.36193629e-05,
       4.10023964e-06, 1.39688291e-05, 1.91009576e-05, 5.75485476e-07,
       6.41810608e-05, 3.99226053e-05, 2.94688152e-06, 0.00000000e+00,
       3.93958080e-06, 2.59150607e-06, 5.55443301e-06, 1.16861507e-04,
       5.01828552e-07, 1.42957775e-04, 4.63998917e-05, 1.09486636e-05,
       4.93656253e-05, 8.80577268e-07, 7.57896529e-06, 7.38316333e-07,
```

```
            1.26082593e-05, 7.63630990e-06, 2.82937959e-05, 2.18731901e-04,
            3.94902420e-05, 3.59437103e-06, 1.78482387e-19, 1.47654975e-05,
            6.60010794e-06, 6.54870987e-06, 1.45627528e-05, 4.22691166e-05,
            1.35547549e-05, 1.26564771e-05, 1.79368171e-06, 1.96549877e-07,
            5.33550636e-07, 3.39924949e-05, 1.12394257e-05, 3.74546811e-04,
            2.09463981e-05, 3.74864091e-06, 5.32532023e-04, 1.07893947e-05,
            9.33975745e-06, 1.15673893e-06, 1.09866624e-04, 6.44550514e-05,
            2.66034948e-08, 2.95975598e-05, 2.48115142e-06, 3.42569285e-05,
            1.74519262e-06, 1.40541613e-05, 6.56316642e-04, 4.64952056e-04,
            3.48249275e-05, 1.20579807e-04, 3.62717167e-05, 1.44009772e-04,
            5.29627115e-05, 1.03011141e-05, 1.55478372e-05, 1.52107677e-06,
            2.71379234e-05, 1.65043524e-04, 9.32577536e-05, 3.94001195e-05,
            1.13501360e-05, 7.18372903e-05, 2.97604536e-05, 8.39017182e-06,
            4.42318053e-05, 5.28974766e-05, 2.12569497e-05, 3.61937618e-06,
            4.17039876e-05, 1.61234657e-05, 1.60769035e-05, 1.18590244e-05,
            0.00000000e+00, 1.46723315e-05, 1.90151200e-05, 7.54205765e-07,
            5.98131260e-04, 3.54710815e-05, 2.58147327e-05, 1.63865386e-06,
            1.12481484e-04, 2.67689062e-05, 1.80086731e-06, 7.17512818e-05,
            3.97379600e-06, 2.33277718e-05, 7.61936216e-05, 8.05033281e-05,
            9.31593893e-05])
```

```
import seaborn as sns
def plot_features(columns, importances, n=10):
    df = (pd.DataFrame({"features": columns,
                        "feature_importance": importances})
          .sort_values("feature_importance", ascending=False)
          .reset_index(drop=True))

    sns.barplot(x="feature_importance",
                y="features",
                data=df[:n],
                orient="h")
```

```
plot_features(X_train.columns, rf.feature_importances_)
```




```
new_data = data[['hiv/aids','adult_mortality','income_composition_of_resources','schooling',
      'thinness_5-9_years','under-five_deaths','infant_deaths',
      'thinness_10-19_years','year']]
X_train, X_test, y_train, y_test = train_test_split(new_data,y, test_size=0.2, random_state=42)
```

```
rf.fit(X_train, y_train)
```

```
RandomForestRegressor()
```

```
rf_pred_new = rf.predict(X_test)
```

```
print('R2 score is : {:.3f}'.format(r2_score(y_test, rf_pred_new)))
```

```
R2 score is : 0.964
```