import

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import time

from sklearn import externals
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LinearRegression, SGDRegressor, Lasso, Ridge, ElasticNet
from sklearn.ensemble import GradientBoostingRegressor, BaggingRegressor, RandomForestRegressor, ExtraTreesRegressor, AdaBoostRegressor
from sklearn.model_selection import train_test_split, cross_val_score
import pickle

import lime
import lime.lime_tabular

from yellowbrick.regressor import ResidualsPlot
from fancyimpute import SoftImpute

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

```
Using TensorFlow backend.
```

functions & var

In [2]:
```
modelos = [
    GradientBoostingRegressor, BaggingRegressor, RandomForestRegressor, ExtraTreesRegressor, AdaBoostRegressor,
    SGDRegressor, Lasso, Ridge, ElasticNet, LinearRegression,
]

def fnc_Dummies(df):
    for cat_feature in df.select_dtypes(include=['object']).columns:
        df[cat_feature] = pd.Categorical(df[cat_feature]).codes
        df[cat_feature] = df[cat_feature].replace(-1,np.nan)
    return pd.DataFrame(df)

def fnc_Imputer(df):
    imp_cols = df.columns.values
    imputer = SoftImpute()
    return pd.DataFrame(imputer.fit_transform(df), columns= imp_cols)
```

datasets

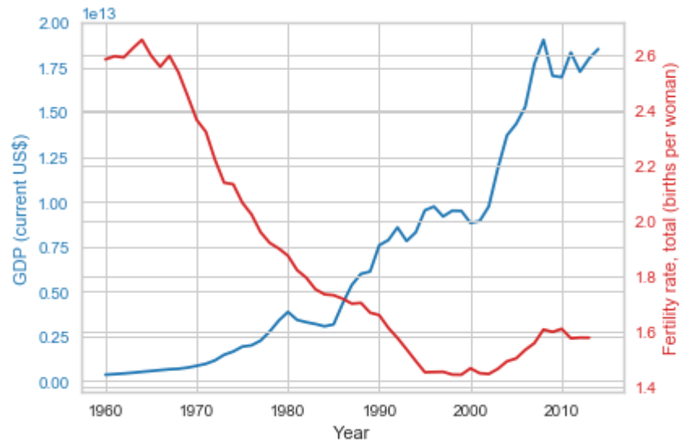
In [3]:
```
# importando os indicadores
indicators = pd.read_csv('Indicators.csv')

# dataset de países da união européia
eu_countries = ['European Union', 'Austria','Belgium', 'Bulgaria', 'Croatia', 'Cyprus',
                'Czech Republic', 'Denmark', 'Estonia', 'Finland', 'France', 'Germany', 'Greece',
                'Hungary', 'Ireland', 'Italy', 'Latvia', 'Lithuania', 'Luxembourg', 'Malta',
                'Netherlands', 'Poland', 'Portugal', 'Romania', 'Slovakia', 'Slovenia', 'Spain',
                'Sweden', 'United Kingdom']

# definindo apenas a base de indicadores da união européia
eu = indicators[indicators['CountryName'].isin(eu_countries)]
```

Questão 1

Mostre graficamente a evolução da taxa de fertilidade e do Produto Interno Bruto na União Européia.

Resposta:

In [4]:
```
# GDP (current US$)
# https://data.worldbank.org/indicator/NY.GDP.MKTP.CD
pib_cod = 'NY.GDP.MKTP.CD'
pib_nome = 'GDP (current US$)'

# Fertility rate, total (births per woman)
# https://data.worldbank.org/indicator/SP.DYN.TFRT.IN
fertilidade_cod = 'SP.DYN.TFRT.IN'
fertilidade_nome = 'Fertility rate, total (births per woman)'

# Filtro para UE
df_grafico = eu[eu['CountryName']=='European Union']

# Dados para gráfico
data1 = df_grafico[df_grafico['IndicatorCode']==pib_cod]['Value'].values
data1_year = df_grafico[df_grafico['IndicatorCode']==pib_cod]['Year'].values

data2 = df_grafico[df_grafico['IndicatorCode']==fertilidade_cod]['Value'].values
data2_year = df_grafico[df_grafico['IndicatorCode']==fertilidade_cod]['Year'].values

fig, ax1 = plt.subplots()

color = 'tab:blue'
ax1.set_xlabel('Year')
ax1.set_ylabel(pib_nome, color=color)
ax1.plot(data1_year, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel(fertilidade_nome, color=color)
ax2.plot(data2_year, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()
plt.show()
```



Questão 2

Podemos afirmar que existe uma correlação entre a taxa de fertilidade e do Produto Interno Bruto na União Européia? Explique.

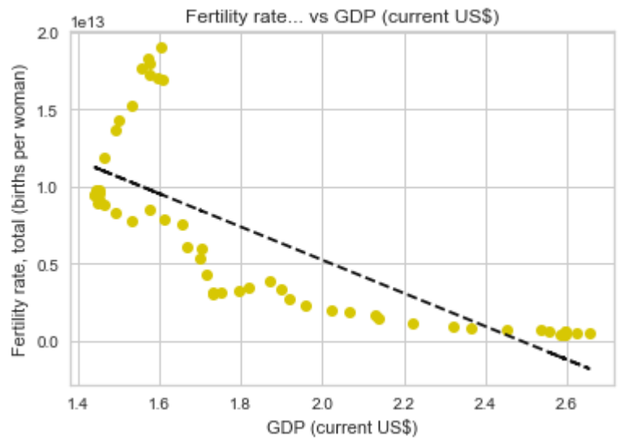
Resposta:

Sim, existe uma correlação, mas é negativa. A taxa de fertilidade da UE é inversamente proporcional ao PIB, ou seja, quanto maior é o PIB menor é a taxa.

In [5]:
```
# Anos com valores
# df_grafico[df_grafico['IndicatorCode'].isin([fertilidade_cod,pib_cod])].groupby('Year').count()
```

In [6]:
```
df_lr = df_grafico[df_grafico['Year']<2014]

x = df_lr[df_lr['IndicatorCode']==fertilidade_cod]['Value'].values
y = df_lr[df_lr['IndicatorCode']==pib_cod]['Value'].values

fit = np.polyfit(x,y,1)
fit_fn = np.poly1d(fit)

plt.title(fertilidade_nome[:14]+'... vs '+pib_nome)
plt.xlabel(pib_nome)
plt.ylabel(fertilidade_nome)

plt.plot(x,y, 'yo', x, fit_fn(x), '--k')

print('Correlação de Pearson: ',np.corrcoef(x,y)[1][0])
```

```
Correlação de Pearson:  -0.7551417562144183
```



Questão 3

Podemos afirmar com as informações das questões anteriores que *"o crescimento do PIB levou a uma redução da taxa de fertilidade na União Européia"*?. Explique.

Resposta:

Correlação não implica causa!

Pouco provável, não podemos concluir que a resposta para a redução é só o PIB. Precisamos utilizar mais variáveis para tentar responder esse evento de redução, pois ele é complexo e sabemos que existem muitas variáveis envolvidas. Seria interessante utilizar modelos de machine learning ou deep learning para tentar responder esse evento com mais precisão.

Questão 4

Consumo, segundo Keynes (https://en.wikipedia.org/wiki/John_Maynard_Keynes) (economista inglês), é uma função linear de um componente

autônomo invariável que é independente da renda disponível (*a*) e da proporção da renda alocada para consumo (*b*) multiplicada pela renda (*Y*), conforme abaixo:

$$C = a + b * Y$$

O parâmetro *b* é conhecido como a propensão marginal a consumir, ou seja, o crescimento no consumo decorrente de um aumento na renda disponível.

Faça a regressão da equação acima por `Mínimos Quadrados Ordinários` utilizando *Household final consumption expenditure (constant 2005 US$)* como proxy para o consumo, o *GDP at market prices (constant 2005 US$)* como proxy para a renda disponível utilizando os dados de cada um dos países membros da União Européia. Podemos rejeitar a hipótese de que o parâmetro *b* é igual a zero? Comente.

Resposta:

Sim, podemos. O consumo de fato subiu durante o crescimento da renda.

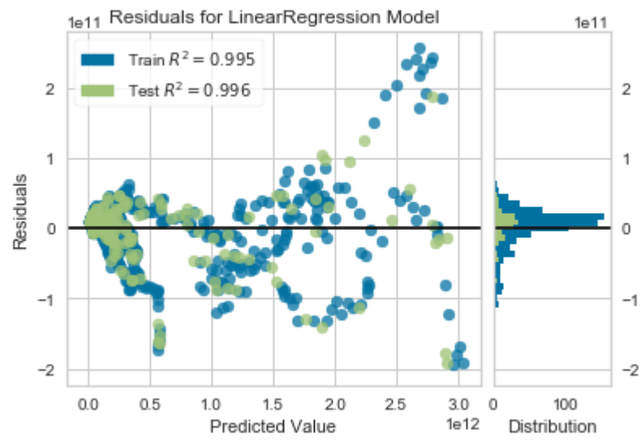
O modelo de regressão linear apresentou R^2 de 0.99, assim indicando forte relação entre as variáveis utilizadas. A distribuição de erros é uma curva normal e as variáveis apresentam relação linear, esses são pré-requisitos fundamentais para um modelo linear. Seguem as evidências:

Preparando Dataset

In [7]:
```
# Conversão do dataset
var_x = 'Household final consumption expenditure (constant 2005 US$)'
var_y = 'GDP at market prices (constant 2005 US$)'
df_mqo = eu[(~eu['CountryName'].isin(['European Union'])) & eu['IndicatorName'].isin([var_x,var_y])]

anos = df_mqo['Year'].unique()
paises = df_mqo['CountryName'].unique()
lista = [(x, y) for x in anos for y in paises]

df_mqo_v2 = pd.DataFrame(columns=['Year','CountryName',var_x,var_y],dtype=np.float)

df_mqo_v2['Year'] = np.asarray([ano for ano,pais in lista])
df_mqo_v2['CountryName'] = np.asarray([pais for ano,pais in lista])

for index, row in df_mqo.iterrows():
    if row['IndicatorName'] == var_x:
        df_mqo_v2.loc[(df_mqo_v2['Year'] == row['Year']) & (df_mqo_v2['CountryName'] == row['CountryName']), var_x] = row['Value']
    else:
        df_mqo_v2.loc[(df_mqo_v2['Year'] == row['Year']) & (df_mqo_v2['CountryName'] == row['CountryName']), var_y] = row['Value']
```

Modelo Linear - Resultados

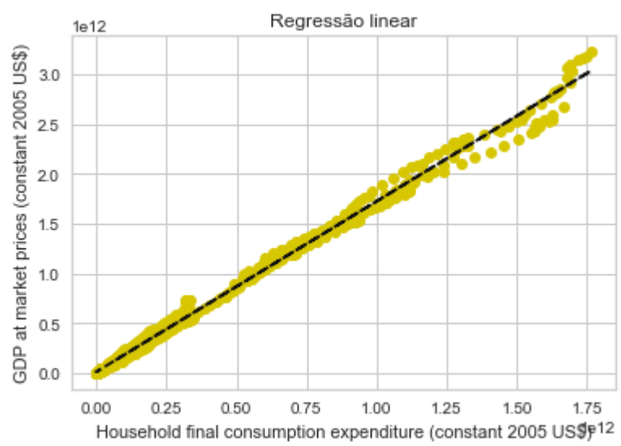
In [8]:
```
# Faz cópia de dataset
df_mqo_v3 = df_mqo_v2.dropna().copy()

# Converte variáveis categóricas
df_mqo_v3 = fnc_Dummies(df=df_mqo_v3)

# Variáveis para treino
feature_names = ['Year', 'CountryName', var_x]
target_name = var_y
X = df_mqo_v3[feature_names]
y = df_mqo_v3[target_name]

# Separa dados para treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Prepara modelo para gráfico
model = LinearRegression()
visualizer = ResidualsPlot(model)
visualizer.fit(X_train, y_train)
visualizer.score(X_test, y_test)
visualizer.poof()
```



Correlação

In [9]:
```
x = df_mqo_v3[var_x].values
y = df_mqo_v3[var_y].values

fit = np.polyfit(x,y,1)
fit_fn = np.poly1d(fit)

plt.title('Regressão linear')
plt.xlabel(var_x)
plt.ylabel(var_y)

plt.plot(x,y, 'yo', x, fit_fn(x), '--k')

print('Correlação de Pearson: ',np.corrcoef(x,y)[1][0])
```

```
Correlação de Pearson:  0.997676685154029
```



Questão 5

Utilizando todas variáveis disponíveis na base de dados, construa o melhor modelo para explicar a taxa de fertilidade com até 5 variáveis.

Resposta:

Data preparation & Missing Values

In [10]:
```
# Cópia de dataframe
df_new = eu[(~eu['CountryName'].isin(['European Union']))]
df_new.Year.astype('category')

# Pivot table para Indicadores
df_new_v2 = pd.pivot_table(df_new, values='Value', index=['Year','CountryName'], columns='IndicatorName')
df_new_v2.insert(loc=0, column='CountryName', value=df_new_v2.index.get_level_values('CountryName') )
df_new_v2.insert(loc=0, column='Year', value=df_new_v2.index.get_level_values('Year') )

# Converte variáveis categóricas
df_new_v2 = fnc_Dummies(df=df_new_v2)
```

In [ ]:
```
# Elimina features com menos de 30% de dados
df = df_new_v2.copy()
df_new_v3 = df[[column for column in df if df[column].count() / len(df) >= 0.3]]

print("Lista de fatos excluídos:\n\n Total excluído:", round(len(df_new_v3.columns)/len(df.columns),2)*100,'%\n\n', end=" ")

for c in df.columns:
    if c not in df_new_v3.columns:
        print(c, end="\n ")

# Missing Values
df_new_v3 = fnc_Imputer(df_new_v2)
```

In [12]:
```
# Matriz de Correlação da Fertilidade
df_corr = df_new_v3.corr()
df_corr[[fertilidade_nome]].sort_values(by=fertilidade_nome,ascending=False)
```

Out[12]:

	Fertility rate, total (births per woman)
Fertility rate, total (births per woman)	1.000000
Birth rate, crude (per 1,000 people)	0.949686
Age dependency ratio, young (% of working-age population)	0.852223
Population, ages 0-14 (% of total)	0.845400
Age dependency ratio (% of working-age population)	0.737768
Urban population growth (annual %)	0.489621
Mortality rate, adult, female (per 1,000 female adults)	0.467726
Mortality rate, infant (per 1,000 live births)	0.466446
Mortality rate, under-5 (per 1,000)	0.465638
Adolescent fertility rate (births per 1,000 women ages 15-19)	0.400526
Population, female (% of total)	0.393750
Rural population (% of total population)	0.391737
Life expectancy at birth, male (years)	0.389749
Life expectancy at birth, total (years)	0.383373
Population growth (annual %)	0.380989
Life expectancy at birth, female (years)	0.375188
Survival to age 65, female (% of cohort)	0.363533
Number of under-five deaths	0.360481
Number of infant deaths	0.359097
Survival to age 65, male (% of cohort)	0.315776
Death rate, crude (per 1,000 people)	0.267121
Agricultural raw materials imports (% of merchandise imports)	0.261226
Merchandise imports from developing economies in Sub-Saharan Africa (% of total merchandise imports)	0.260621
Merchandise exports to developing economies in Sub-Saharan Africa (% of total merchandise exports)	0.252337
Mortality rate, adult, male (per 1,000 male adults)	0.245531
Merchandise exports by the reporting economy, residual (% of total merchandise exports)	0.244817
Merchandise imports by the reporting economy, residual (% of total merchandise imports)	0.231759
Population, ages 15-64 (% of total)	0.224319
CO2 emissions from manufacturing industries and construction (% of total fuel combustion)	0.221011
CO2 emissions from residential buildings and commercial and public services (% of total fuel combustion)	0.213546
...	...
Labor force participation rate, total (% of total population ages 15-64) (modeled ILO estimate)	-0.491713
Labor force participation rate, male (% of male population ages 15-64) (modeled ILO estimate)	-0.492392
Ratio of female to male labor force participation rate (%) (modeled ILO estimate)	-0.493769

	Fertility rate, total (births per woman)
Bank capital to assets ratio (%)	-0.500658
Immunization, measles (% of children ages 12-23 months)	-0.501781
Labor force, female (% of total labor force)	-0.502054
Ease of doing business index (1=most business-friendly regulations)	-0.503063
Immunization, DPT (% of children ages 12-23 months)	-0.504797
Time to resolve insolvency (years)	-0.518458
Lifetime risk of maternal death (1 in: rate varies by country)	-0.542568
Improved sanitation facilities, rural (% of rural population with access)	-0.601791
Improved sanitation facilities (% of population with access)	-0.609923
Improved sanitation facilities, urban (% of urban population with access)	-0.612721
Improved water source, rural (% of rural population with access)	-0.616532
Improved water source (% of population with access)	-0.618691
Improved water source, urban (% of urban population with access)	-0.619488
Year	-0.704792
Commitments, IDA (COM, current US$)	NaN
Currency composition of PPG debt, SDR (%)	NaN
Debt forgiveness grants (current US$)	NaN
EBRD, private nonguaranteed (NFL, current US$)	NaN
IDA grants (current US$)	NaN
Net financial flows, IDA (NFL, current US$)	NaN
Net financial flows, IMF concessional (NFL, current US$)	NaN
PPG, IDA (AMT, current US$)	NaN
PPG, IDA (DIS, current US$)	NaN
PPG, IDA (DOD, current US$)	NaN
PPG, IDA (INT, current US$)	NaN
PPG, IDA (NTR, current US$)	NaN
PPG, IDA (TDS, current US$)	NaN

1286 rows × 1 columns

In [13]:
```
# Variável de interesse
var_interesse = fertilidade_nome

# Escopo de treino/test
colunas = list(df_new_v3.columns)
colunas.remove(fertilidade_nome)
feature_names = colunas
target_name = var_interesse
X = df_new_v3[feature_names]
y = df_new_v3[target_name]

# Separa dados para treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Lista de resultados
resultados = [['status','var','model','mean','std','time','scores','caminho']]

# Score dos modelos
print('='*100)
print(var_interesse)
print('='*100)

Y = df_new_v3[[var_interesse]].values.ravel()

for var in modelos:
    start = time.time()
    try:
        filename = str('../../models/'+var.__name__+'.model')
        print(var)
        clf = var()

        # Calcula o score
        scores = cross_val_score(clf, df_new_v3, Y, cv=10, error_score='raise')
        print('Mean score: ',np.mean(scores), '/ Std Score: ',np.std(scores))
        resultados.append(['ok',var_interesse,var.__name__,np.mean(scores),np.std(scores),time.time() - start,scores,filename])

        # Salva modelo
        model = clf.fit(X_train, y_train)
        externals.joblib.dump(model,filename)

    except(Exception):
        print('>> Validar parâmetros.')
        resultados.append(['erro',var_interesse,var.__name__,None,None,time.time() - start,None,None])
        pass
    finally:
        print('-'*100)
```

```
==========================================================================================================
Fertility rate, total (births per woman)
==========================================================================================================
<class 'sklearn.ensemble.gradient_boosting.GradientBoostingRegressor'>
Mean score:  0.9498657118073565 / Std Score:  0.1376871684155691
----------------------------------------------------------------------------------------------------------
<class 'sklearn.ensemble.bagging.BaggingRegressor'>
Mean score:  0.958153173047309 / Std Score:  0.11060923252366513
----------------------------------------------------------------------------------------------------------
<class 'sklearn.ensemble.forest.RandomForestRegressor'>
Mean score:  0.9426215306778645 / Std Score:  0.1504299415825261
----------------------------------------------------------------------------------------------------------
<class 'sklearn.ensemble.forest.ExtraTreesRegressor'>
Mean score:  0.9433292175633525 / Std Score:  0.10748931348624972
----------------------------------------------------------------------------------------------------------
<class 'sklearn.ensemble.weight_boosting.AdaBoostRegressor'>
Mean score:  0.9418747789470107 / Std Score:  0.07399990967872937
----------------------------------------------------------------------------------------------------------
<class 'sklearn.linear_model.stochastic_gradient.SGDRegressor'>
Mean score:  -2.4123377050424653e+71 / Std Score:  3.273163204464693e+71
----------------------------------------------------------------------------------------------------------
<class 'sklearn.linear_model.coordinate_descent.Lasso'>
Mean score:  -4051.464182044576 / Std Score:  12075.168106369343
----------------------------------------------------------------------------------------------------------
<class 'sklearn.linear_model.ridge.Ridge'>
Mean score:  -603402718197.5461 / Std Score:  1398275131351.2979
----------------------------------------------------------------------------------------------------------
<class 'sklearn.linear_model.coordinate_descent.ElasticNet'>
Mean score:  -790.4387726538359 / Std Score:  2298.147954062192
----------------------------------------------------------------------------------------------------------
<class 'sklearn.linear_model.base.LinearRegression'>
Mean score:  -294088734216.0228 / Std Score:  588229160126.1681
----------------------------------------------------------------------------------------------------------
```

In [14]:
```
# Salva planilhas com score de modelos
writer = pd.ExcelWriter('../../dist/resultados_modelos.xlsx', engine='xlsxwriter')
df_final = pd.DataFrame(resultados[1:])
df_final.columns = resultados[0]
df_final.to_excel(writer, sheet_name='Sheet1', index=False)
writer.save()
```

In [15]:
```
df_final.sort_values(by=['mean'], ascending=False)
```

Out[15]:

	status	var	model	mean	std	time	scores	caminho
1	ok	Fertility rate, total (births per woman)	BaggingRegressor	9.581532e-01	1.106092e-01	50.583918	[0.9638536535659511, 0.9975196663418858, 0.998...	../../models/BaggingRegressor.model
0	ok	Fertility rate, total (births per woman)	GradientBoostingRegressor	9.498657e-01	1.376872e-01	131.870818	[0.9668257313948193, 0.9982827521245184, 0.998...	../../models/GradientBoostingRegressor.model
3	ok	Fertility rate, total (births per woman)	ExtraTreesRegressor	9.433292e-01	1.074893e-01	19.065458	[0.8734091736600261, 0.9831857764887317, 0.990...	../../models/ExtraTreesRegressor.model
2	ok	Fertility rate, total (births per woman)	RandomForestRegressor	9.426215e-01	1.504299e-01	47.736675	[0.9450056586169125, 0.9971917988056322, 0.996...	../../models/RandomForestRegressor.model
4	ok	Fertility rate, total (births per woman)	AdaBoostRegressor	9.418748e-01	7.399991e-02	114.062698	[0.932219056841623, 0.98965140411541, 0.982667...	../../models/AdaBoostRegressor.model
8	ok	Fertility rate, total (births per woman)	ElasticNet	-7.904388e+02	2.298148e+03	14.034799	[0.06096002684390356, 0.4556464086168164, 0.62...	../../models/ElasticNet.model
6	ok	Fertility rate, total (births per woman)	Lasso	-4.051464e+03	1.207517e+04	11.567414	[-0.2230657803450975, 0.2322276941487278, 0.46...	../../models/Lasso.model
9	ok	Fertility rate, total (births per woman)	LinearRegression	-2.940887e+11	5.882292e+11	8.799115	[0.6980346972624497, 0.7569493032095755, -1736...	../../models/LinearRegression.model
7	ok	Fertility rate, total (births per woman)	Ridge	-6.034027e+11	1.398275e+12	11.631908	[0.4480556037313639, 0.6063387959622368, -2486...	../../models/Ridge.model
5	ok	Fertility rate, total (births per woman)	SGDRegressor	-2.412338e+71	3.273163e+71	0.342238	[-9.748696307358854e+69, -1.1831318055192161e+...	../../models/SGDRegressor.model

In [16]:
```
# Carrega melhor modelo
model = externals.joblib.load(df_final.sort_values(by=['mean'], ascending=False)['caminho'][0])

# Acurácia do modelo de regressão
model.score(X_test, y_test)
```

Out[16]: 0.9454037921403707

In [17]:
```
# Preparando modelo Lime
explainer = lime.lime_tabular.LimeTabularExplainer(
    X_train.values,
    feature_names=list(X_train.columns),
    class_names=[var_interesse],
    verbose=True,
    mode='regression'
)
```

In [21]:
```
# Define seed para modelo
def explain(instance, predict_fn, **kwargs):
  np.random.seed(16)
  return explainer.explain_instance(instance, predict_fn, **kwargs)

# Modelo de predição para teste
i = 302
exp = explain(X_test.values[i], model.predict, num_features=5)
```

```
Intercept 1.6969907389449366
Prediction_local [1.98394275]
Right: 1.9475947385082149
```

In [22]:
```
# Resultados para 5 features
exp.show_in_notebook(show_table=True)
```


Feature	Value
Birth rate, crude (per 1,000 people)	13.30
Age dependency ratio (% of working-age population)	57.94
Electricity production from hydroelectric sources (% of total)	0.42
Mortality rate, under-5 (per 1,000)	20.80
Health expenditure, public (% of government expenditure)	0.72

In [23]:
```
# Detalhe por variável
exp.as_list()
```

Out[23]:
```
[('12.70 < Birth rate, crude (per 1,000 people) <= 15.90',
  0.14414684780518253),
 ('Age dependency ratio (% of working-age population) > 54.32',
  0.09886783941658628),
 ('0.05 < Electricity production from hydroelectric sources (% of total) <= 3.08',
  -0.05685114334039232),
 ('Mortality rate, under-5 (per 1,000) > 19.50', 0.05159677806094871),
 ('0.04 < Health expenditure, public (% of government expenditure) <= 1.48',
  0.04919169186857802)]
```