
# **Projeto de Parceria** | Falta de Igualdade na Educação pelo Mundo

Aluna: Maria Luiza Cordeiro

---

# Tópicos

1. Coleta de Dados;
2. Modelagem;
3. Conclusões.

---

## Coleta de Dados

### Contexto

A educação é algo imprescindível para o desenvolvimento humano, porém, mesmo com essa importância, muitas regiões do mundo enfrentam problemas para garantir uma igualdade no acesso ao ensino.

Nesse projeto, foi utilizada a base de dados Inequality in Education Around the World, disponível no Kaggle. Nela, encontra-se o nível de desigualdade na educação entre os anos de 2010 e 2021 por país, bem como o seu Índice de Desenvolvimento Humano (IDH) em 2021 e outras informações relacionadas ao desenvolvimento dos países no mundo.

In [ ]:
```
!wget -q "https://raw.githubusercontent.com/malucor/Projeto_Educacao_ML/main/Ine
```

Essa atividade terá como foco a relação do rank do IDH com a evolução do índice relacionado a falta de igualdade da educação por ano. O modelo de aprendizagem de máquina, no final, terá, como objetivo, prever a posição que o país está no rank do IDH, por meio do seu índice de destribuição da educação.

## Manipulação dos Dados

In [ ]:
```
import sklearn
import numpy as np
import pandas as pd
import seaborn as sns
```

Para começar, é necessário tratar os dados no arquivo original, que está no formato `.csv`.

In [ ]:
```
# Conversão do arquivo de .csv para DataFrame

education_df = pd.read_csv('education.csv')
education_df.head()
```

Out[ ]:

| | ISO3 | Country | Human Development Groups | UNDP Developing Regions | HDI Rank (2021) | Inequality in Education (2010) | Inequality in Education (2011) | Inequal Educati (201 |
|---|---|---|---|---|---|---|---|---|
| **0** | AFG | Afghanistan | Low | SA | 180.0 | 42.809000 | 44.823380 | 44.8233 |
| **1** | AGO | Angola | Medium | SSA | 148.0 | NaN | NaN | N |
| **2** | ALB | Albania | High | ECA | 67.0 | 11.900000 | 11.900000 | 11.900( |
| **3** | AND | Andorra | Very High | NaN | 40.0 | 15.160302 | 15.160302 | 15.1603 |
| **4** | ARE | United Arab Emirates | Very High | AS | 26.0 | NaN | NaN | N |

In [ ]:
```
# Remoção das colunas que não são o foco do projeto

education = education_df.drop(columns=['ISO3', 'Country', 'Human Development Gro
education.head()
```

Out[ ]:

| | HDI Rank (2021) | Inequality in Education (2010) | Inequality in Education (2011) | Inequality in Education (2012) | Inequality in Education (2013) | Inequality in Education (2014) | Inequality in Education (2015) | Inequali Educatio (201 |
|---|---|---|---|---|---|---|---|---|
| **0** | 180.0 | 42.809000 | 44.823380 | 44.823380 | 44.823380 | 44.823380 | 45.365170 | 45.3651 |
| **1** | 148.0 | NaN | NaN | NaN | NaN | NaN | 34.171440 | 34.1714 |
| **2** | 67.0 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.9000 |
| **3** | 40.0 | 15.160302 | 15.160302 | 15.160302 | 15.160302 | 9.965681 | 10.083815 | 10.0081 |
| **4** | 26.0 | NaN | NaN | NaN | NaN | NaN | NaN | 18.2414 |

In [ ]:
```
education.info()
```

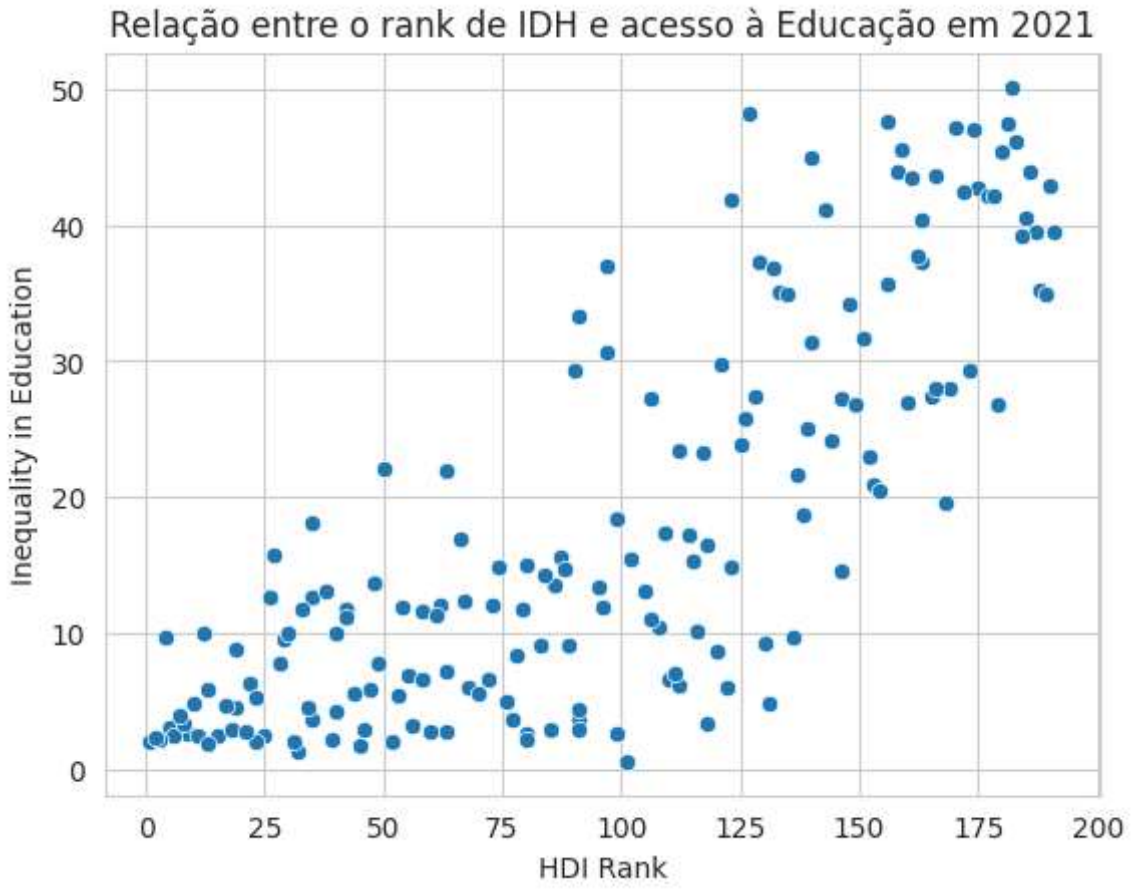
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 13 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   HDI Rank (2021)                 191 non-null    float64
 1   Inequality in Education (2010)  137 non-null    float64
 2   Inequality in Education (2011)  150 non-null    float64
 3   Inequality in Education (2012)  157 non-null    float64
 4   Inequality in Education (2013)  165 non-null    float64
 5   Inequality in Education (2014)  168 non-null    float64
 6   Inequality in Education (2015)  168 non-null    float64
 7   Inequality in Education (2016)  168 non-null    float64
 8   Inequality in Education (2017)  168 non-null    float64
 9   Inequality in Education (2018)  172 non-null    float64
 10  Inequality in Education (2019)  174 non-null    float64
 11  Inequality in Education (2020)  176 non-null    float64
 12  Inequality in Education (2021)  176 non-null    float64
dtypes: float64(13)
memory usage: 19.9 KB
```

## Visualização

Para se ter uma ideia da distribuição dos dados, os ranks de IDH serão comparados, em um gráfico, com o índice de desigualdade do ano que foi divulgado resultado do IDH, nesse caso, em 2021.

In [ ]:
```
with sns.axes_style('whitegrid'):

  grafico = sns.scatterplot(x=education['HDI Rank (2021)'], y=education['Inequal
  grafico.set(title='Relação entre o rank de IDH e acesso à Educação em 2021', x
```




Como visto no gráfico, a medida que a posição no rank é pior (aumenta na linha $x$), a desigualdade na educação é maior (aumenta na linha $y$). Então, de acordo com essas informações e o visual do gráfico, percebe-se que há uma relação de proporcionalidade.

## Tratamento dos Dados

É preciso tratar os dados para que eles estejam prontos para o machine learning.

### Dados Faltantes

In [ ]:
```
# Verificar se há valores nulos

print(education.isna().sum())
```

```
HDI Rank (2021)                    4
Inequality in Education (2010)    58
Inequality in Education (2011)    45
Inequality in Education (2012)    38
Inequality in Education (2013)    30
Inequality in Education (2014)    27
Inequality in Education (2015)    27
Inequality in Education (2016)    27
Inequality in Education (2017)    27
Inequality in Education (2018)    23
Inequality in Education (2019)    21
Inequality in Education (2020)    19
Inequality in Education (2021)    19
dtype: int64
```

Como há valores nulos, eles serão divididos em duas categorias:

- Os que impedem de continuar com o Aprendizado de Máquina; nesse caso, toda a linha de dados será descartada (o IDH na base de dados que está sendo trabalhada)
- Os que se podem inferir a partir de outros dados; nesse caso, será feita a média dos dados relacionados (os índices de desigualdade na base de dados que está sendo trabalhada)

In [ ]:
```
# Retirar os valores nulos da coluna de IDH

education_clean_df = education.dropna(subset=['HDI Rank (2021)'])
education_clean_df.head()
```

Out[ ]:

| | HDI Rank (2021) | Inequality in Education (2010) | Inequality in Education (2011) | Inequality in Education (2012) | Inequality in Education (2013) | Inequality in Education (2014) | Inequality in Education (2015) | Inequali Educatic (201 |
|---|---|---|---|---|---|---|---|---|
| **0** | 180.0 | 42.809000 | 44.823380 | 44.823380 | 44.823380 | 44.823380 | 45.365170 | 45.3651 |
| **1** | 148.0 | NaN | NaN | NaN | NaN | NaN | 34.171440 | 34.1714 |
| **2** | 67.0 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.9000 |
| **3** | 40.0 | 15.160302 | 15.160302 | 15.160302 | 15.160302 | 9.965681 | 10.083815 | 10.0081 |
| **4** | 26.0 | NaN | NaN | NaN | NaN | NaN | NaN | 18.2414 |

In [ ]:
```
print(education_clean_df.isna().sum())
```

```
HDI Rank (2021)                    0
Inequality in Education (2010)    55
Inequality in Education (2011)    42
Inequality in Education (2012)    34
Inequality in Education (2013)    26
Inequality in Education (2014)    23
Inequality in Education (2015)    23
Inequality in Education (2016)    23
Inequality in Education (2017)    23
Inequality in Education (2018)    19
Inequality in Education (2019)    17
Inequality in Education (2020)    15
Inequality in Education (2021)    15
dtype: int64
```

In [ ]:
```
# Criar um dataframe que só tenha as linhas com algum valor nulo

colunas_valores_nulos = education_clean_df.drop(columns=['HDI Rank (2021)'])
valores_nulos = colunas_valores_nulos.isna().any(axis=1)

valores_nulos = colunas_valores_nulos[valores_nulos]
valores_nulos.head()
```

Out[ ]:

| | Inequality in Education (2010) | Inequality in Education (2011) | Inequality in Education (2012) | Inequality in Education (2013) | Inequality in Education (2014) | Inequality in Education (2015) | Inequality in Education (2016) | Ine Edu |
|---|---|---|---|---|---|---|---|---|
| **1** | NaN | NaN | NaN | NaN | NaN | 34.17144 | 34.171440 | 34. |
| **4** | NaN | NaN | NaN | NaN | NaN | NaN | 18.241437 | 14. |
| **7** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **13** | NaN | 44.79226 | 44.79226 | 44.79226 | 43.65639 | 43.65639 | 43.656390 | 43. |
| **20** | NaN | NaN | 3.67781 | 3.67781 | 3.67781 | 3.67781 | 3.677810 | 3. |

In [ ]:
```
# Transformar as linhas em colunas

valores_nulos_transposed = valores_nulos.transpose()
valores_nulos_transposed.head()
```

Out[ ]:

| | 1 | 4 | 7 | 13 | 20 | 21 | 24 | 25 | 30 | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Inequality in Education (2010)** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | N |
| **Inequality in Education (2011)** | NaN | NaN | NaN | 44.79226 | NaN | 15.85449 | NaN | NaN | NaN | 33.66( |
| **Inequality in Education (2012)** | NaN | NaN | NaN | 44.79226 | 3.67781 | 15.85449 | 5.54685 | NaN | NaN | 33.66( |
| **Inequality in Education (2013)** | NaN | NaN | NaN | 44.79226 | 3.67781 | 15.85449 | 5.54685 | NaN | 2.34809 | 33.66( |
| **Inequality in Education (2014)** | NaN | NaN | NaN | 43.65639 | 3.67781 | 15.85449 | 5.54685 | NaN | 2.37852 | 32.95 |

5 rows × 65 columns

In [ ]:
```
# Fazer a média de cada coluna

valores_nulos_transposed_mean = valores_nulos_transposed.agg('mean')
valores_nulos_transposed_mean = pd.DataFrame([valores_nulos_transposed_mean], cc
valores_nulos_transposed_mean.head()
```

Out[ ]:

| | 1 | 4 | 7 | 13 | 20 | 21 | 24 | 25 | 30 | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 34.17144 | 13.875699 | NaN | 43.983064 | 3.406076 | 15.191584 | 5.54685 | NaN | 2.07139 | 3 |

1 rows × 65 columns

In [ ]:
```
# Transformar as colunas em linhas

valores_nulos_mean = valores_nulos_transposed_mean.transpose()
valores_nulos_mean.head()
```

Out[ ]:

| | 0 |
|---|---|
| **1** | 34.171440 |
| **4** | 13.875699 |
| **7** | NaN |
| **13** | 43.983064 |
| **20** | 3.406076 |

In [ ]:
```
# Adicionar a coluna de média ao dataframe com os valores nulos

valores_nulos['Mean'] = valores_nulos_mean
valores_nulos = valores_nulos.dropna(subset=['Mean'])

valores_nulos.head()
```

```
<ipython-input-15-6dc5a358e97c>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  valores_nulos['Mean'] = valores_nulos_mean
```

Out[ ]:

| | Inequality in Education (2010) | Inequality in Education (2011) | Inequality in Education (2012) | Inequality in Education (2013) | Inequality in Education (2014) | Inequality in Education (2015) | Inequality in Education (2016) | Ine Edu |
|---|---|---|---|---|---|---|---|---|
| **1** | NaN | NaN | NaN | NaN | NaN | 34.17144 | 34.171440 | 34. |
| **4** | NaN | NaN | NaN | NaN | NaN | NaN | 18.241437 | 14. |
| **13** | NaN | 44.79226 | 44.79226 | 44.79226 | 43.65639 | 43.65639 | 43.656390 | 43. |
| **20** | NaN | NaN | 3.67781 | 3.67781 | 3.67781 | 3.67781 | 3.677810 | 3. |
| **21** | NaN | 15.85449 | 15.85449 | 15.85449 | 15.85449 | 14.81278 | 14.812780 | 14. |

In [ ]:
```
# Juntar o dataframe que tem os valores nulos, com o que tem a média e depois re

education_merged = education_clean_df.merge(valores_nulos, how='left', left_inde
```

```
education_final = education_merged.apply(lambda coluna: coluna.fillna(coluna['Me
education_final = education_final.drop(columns=['Inequality in Education (2010)_
education_final = education_final.drop(columns=['Mean']).dropna(subset=['Inequal
education_final = education_final.rename(columns={'Inequality in Education (2010
                                                  'Inequality in Education (2011
                                                  'Inequality in Education (2012
                                                  'Inequality in Education (2013
                                                  'Inequality in Education (2014
                                                  'Inequality in Education (2015
                                                  'Inequality in Education (2016
                                                  'Inequality in Education (2017
                                                  'Inequality in Education (2018
                                                  'Inequality in Education (2019
                                                  'Inequality in Education (2020
                                                  'Inequality in Education (2021

education_completa = education_final
education_final.head()
```

Out[ ]:

| | HDI Rank (2021) | Inequality in Education (2010) | Inequality in Education (2011) | Inequality in Education (2012) | Inequality in Education (2013) | Inequality in Education (2014) | Inequality in Education (2015) | Inequali Educatio (201 |
|---|---|---|---|---|---|---|---|---|
| **0** | 180.0 | 42.809000 | 44.823380 | 44.823380 | 44.823380 | 44.823380 | 45.365170 | 45.3651 |
| **1** | 148.0 | 34.171440 | 34.171440 | 34.171440 | 34.171440 | 34.171440 | 34.171440 | 34.1714 |
| **2** | 67.0 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.9000 |
| **3** | 40.0 | 15.160302 | 15.160302 | 15.160302 | 15.160302 | 9.965681 | 10.083815 | 10.0081 |
| **4** | 26.0 | 13.875699 | 13.875699 | 13.875699 | 13.875699 | 13.875699 | 13.875699 | 18.2414 |

In [ ]:
```
print(education_final.isna().sum())
```

```
HDI Rank (2021)                    0
Inequality in Education (2010)     0
Inequality in Education (2011)     0
Inequality in Education (2012)     0
Inequality in Education (2013)     0
Inequality in Education (2014)     0
Inequality in Education (2015)     0
Inequality in Education (2016)     0
Inequality in Education (2017)     0
Inequality in Education (2018)     0
Inequality in Education (2019)     0
Inequality in Education (2020)     0
Inequality in Education (2021)     0
dtype: int64
```

# Modelagem

## Variáveis Numéricas

Para um melhor Machine Learning, é necessário trabalhar com os dados em escalas reduzidas.
Para isso, é necessário aplicar o método de padronização ($x' = \frac{x - x_m}{\sigma}$).

In [ ]:
```
# Padronização do IDH

media_idh = education_final['HDI Rank (2021)'].mean()
print(media_idh)

desvio_padrao_idh = education_final['HDI Rank (2021)'].std()
print(desvio_padrao_idh)

education_final['HDI Rank (2021) - std'] = education_final['HDI Rank (2021)'].ap
```

```
95.90055248618785
55.806023666543744
```

In [ ]:
```
# Padronização dos Índices de Desigualdade

  # 2010
media_10 = education_final['Inequality in Education (2010)'].mean()
desvio_padrao_10 = education_final['Inequality in Education (2010)'].std()
education_final['Inequality in Education (2010) - std'] = education_final['Inequ

  # 2011
media_11 = education_final['Inequality in Education (2011)'].mean()
desvio_padrao_11 = education_final['Inequality in Education (2011)'].std()
education_final['Inequality in Education (2011) - std'] = education_final['Inequ

  # 2012
media_12 = education_final['Inequality in Education (2012)'].mean()
desvio_padrao_12 = education_final['Inequality in Education (2012)'].std()
education_final['Inequality in Education (2012) - std'] = education_final['Inequ

  # 2013
media_13 = education_final['Inequality in Education (2013)'].mean()
desvio_padrao_13 = education_final['Inequality in Education (2013)'].std()
education_final['Inequality in Education (2013) - std'] = education_final['Inequ

  # 2014
media_14 = education_final['Inequality in Education (2014)'].mean()
desvio_padrao_14 = education_final['Inequality in Education (2014)'].std()
education_final['Inequality in Education (2014) - std'] = education_final['Inequ

  # 2015
media_15 = education_final['Inequality in Education (2015)'].mean()
desvio_padrao_15 = education_final['Inequality in Education (2015)'].std()
education_final['Inequality in Education (2015) - std'] = education_final['Inequ

  # 2016
media_16 = education_final['Inequality in Education (2016)'].mean()
desvio_padrao_16 = education_final['Inequality in Education (2016)'].std()
education_final['Inequality in Education (2016) - std'] = education_final['Inequ

  # 2017
media_17 = education_final['Inequality in Education (2017)'].mean()
desvio_padrao_17 = education_final['Inequality in Education (2017)'].std()
education_final['Inequality in Education (2017) - std'] = education_final['Inequ
```

```
  # 2018
media_18 = education_final['Inequality in Education (2018)'].mean()
desvio_padrao_18 = education_final['Inequality in Education (2018)'].std()
education_final['Inequality in Education (2018) - std'] = education_final['Inequ

  # 2019
media_19 = education_final['Inequality in Education (2019)'].mean()
desvio_padrao_19 = education_final['Inequality in Education (2019)'].std()
education_final['Inequality in Education (2019) - std'] = education_final['Inequ

  # 2020
media_20 = education_final['Inequality in Education (2020)'].mean()
desvio_padrao_20 = education_final['Inequality in Education (2020)'].std()
education_final['Inequality in Education (2020) - std'] = education_final['Inequ

  # 2021
media_21 = education_final['Inequality in Education (2021)'].mean()
desvio_padrao_21 = education_final['Inequality in Education (2021)'].std()
education_final['Inequality in Education (2021) - std'] = education_final['Inequ

education_final.head()
```

Out[ ]:

| | HDI Rank (2021) | Inequality in Education (2010) | Inequality in Education (2011) | Inequality in Education (2012) | Inequality in Education (2013) | Inequality in Education (2014) | Inequality in Education (2015) | Inequali Educatio (201 |
|---|---|---|---|---|---|---|---|---|
| **0** | 180.0 | 42.809000 | 44.823380 | 44.823380 | 44.823380 | 44.823380 | 45.365170 | 45.3651 |
| **1** | 148.0 | 34.171440 | 34.171440 | 34.171440 | 34.171440 | 34.171440 | 34.171440 | 34.1714 |
| **2** | 67.0 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.900000 | 11.9000 |
| **3** | 40.0 | 15.160302 | 15.160302 | 15.160302 | 15.160302 | 9.965681 | 10.083815 | 10.0081 |
| **4** | 26.0 | 13.875699 | 13.875699 | 13.875699 | 13.875699 | 13.875699 | 13.875699 | 18.2414 |

5 rows × 26 columns

In [ ]:
```
education_padrao = education_final.drop(columns=['HDI Rank (2021)', 'Inequality
education_padrao.head()
```

Out[ ]:

| | HDI Rank (2021) - std | Inequality in Education (2010) - std | Inequality in Education (2011) - std | Inequality in Education (2012) - std | Inequality in Education (2013) - std | Inequality in Education (2014) - std | Inequality in Education (2015) - std | Inequ Educ (20 |
|---|---|---|---|---|---|---|---|---|
| **0** | 1.506996 | 1.600717 | 1.754446 | 1.753449 | 1.776017 | 1.779577 | 1.820592 | 1.84 |
| **1** | 0.933581 | 1.000790 | 1.013840 | 1.017433 | 1.035942 | 1.038627 | 1.046603 | 1.06 |
| **2** | -0.517875 | -0.546087 | -0.534645 | -0.521455 | -0.511434 | -0.510576 | -0.493353 | -0.48 |
| **3** | -1.001694 | -0.319641 | -0.307963 | -0.296178 | -0.284915 | -0.645128 | -0.618933 | -0.61 |
| **4** | -1.252563 | -0.408864 | -0.397279 | -0.384940 | -0.374166 | -0.373146 | -0.356744 | -0.04 |

## Treino / Teste

Para criar um modelo de aprendizagem de máquina funcional, é necessário separar a base de dados em dois conjuntos, um maior e outro menor. O maior será usado para treinar o modelo e o menor será usado para testá-lo.

In [ ]:
```
# Separação da base de dados em 75% (treino) e 25% (teste)

from sklearn.model_selection import train_test_split

predictors_train, predictors_test, target_train, target_test = train_test_split(
    education_padrao.drop(['HDI Rank (2021) - std'], axis=1),
    education_padrao['HDI Rank (2021) - std'],
    test_size=0.25,
    random_state=123
)

print(f'predictors_train.shape = {predictors_train.shape}')
print(f'predictors_test.shape = {predictors_test.shape}')
print(f'target_train.shape = {target_train.shape}')
print(f'target_test.shape = {target_test.shape}')
```

```
predictors_train.shape = (135, 12)
predictors_test.shape = (46, 12)
target_train.shape = (135,)
target_test.shape = (46,)
```

### Treino

O treino serve para calcular os coeficiente, tanto linear quanto angular, da fórmula matemática que será usada no modelo de machine learning.

In [ ]:
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()

model = model.fit(predictors_train, target_train)

model.__dict__
```

Out[ ]:
```
{'fit_intercept': True,
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'feature_names_in_': array(['Inequality in Education (2010) - std',
        'Inequality in Education (2011) - std',
        'Inequality in Education (2012) - std',
        'Inequality in Education (2013) - std',
        'Inequality in Education (2014) - std',
        'Inequality in Education (2015) - std',
        'Inequality in Education (2016) - std',
        'Inequality in Education (2017) - std',
        'Inequality in Education (2018) - std',
        'Inequality in Education (2019) - std',
        'Inequality in Education (2020) - std',
        'Inequality in Education (2021) - std'], dtype=object),
 'n_features_in_': 12,
 'coef_': array([-0.43427931,  0.34166188,  0.14772933,  0.02866962, -0.5600631
1,
        -0.42809524,  0.2927623 ,  0.74302404,  3.1463301 , -4.51198674,
         0.9989692 ,  0.9989692 ]),
 'rank_': 11,
 'singular_': array([3.88664891e+01, 3.57011256e+00, 1.46497653e+00, 8.88581965
e-01,
        7.69241350e-01, 5.79583113e-01, 5.13725829e-01, 4.35992672e-01,
        3.65467637e-01, 3.10681914e-01, 1.83494986e-01, 3.17949257e-15]),
 'intercept_': -0.041588463285132146}
```

In [ ]:
```
# Coeficiente angular

a = model.coef_
print(a)
```

```
[-0.43427931  0.34166188  0.14772933  0.02866962 -0.56006311 -0.42809524
  0.2927623   0.74302404  3.1463301  -4.51198674  0.9989692   0.9989692 ]
```

In [ ]:
```
# Coeficiente linear

b = model.intercept_
print(b)
```

```
-0.041588463285132146
```

# Conclusões

## Avaliação

Cálculo da margem de erro da predição do modelo.

In [ ]:
```
# Cálculo do RMSE (root mean square error)

from sklearn.metrics import mean_squared_error

target_predicted = model.predict(predictors_test)

rmse = np.sqrt(mean_squared_error(target_test, target_predicted))
print(rmse)
```

```
0.620650562574459
```

## Predição

Com o modelo treinado, é possível colocar os dados de desigualdade na educação e prever em que rank o país estará em questão de IDH.

In [ ]:
```
# Receber os valores dos índice

indice_2010 = float(input("Índice de Desigualdade em 2010: "))
indice_2011 = float(input("Índice de Desigualdade em 2011: "))
indice_2012 = float(input("Índice de Desigualdade em 2012: "))
indice_2013 = float(input("Índice de Desigualdade em 2013: "))
indice_2014 = float(input("Índice de Desigualdade em 2014: "))
indice_2015 = float(input("Índice de Desigualdade em 2015: "))
indice_2016 = float(input("Índice de Desigualdade em 2016: "))
indice_2017 = float(input("Índice de Desigualdade em 2017: "))
indice_2018 = float(input("Índice de Desigualdade em 2018: "))
indice_2019 = float(input("Índice de Desigualdade em 2019: "))
indice_2020 = float(input("Índice de Desigualdade em 2020: "))
indice_2021 = float(input("Índice de Desigualdade em 2021: "))
```

In [ ]:
```
# Padronizar os valores dos índices

padrao_2010 = (indice_2010 - media_10) / desvio_padrao_10
padrao_2011 = (indice_2011 - media_11) / desvio_padrao_11
padrao_2012 = (indice_2012 - media_12) / desvio_padrao_12
padrao_2013 = (indice_2013 - media_13) / desvio_padrao_13
padrao_2014 = (indice_2014 - media_14) / desvio_padrao_14
padrao_2015 = (indice_2015 - media_15) / desvio_padrao_15
padrao_2016 = (indice_2016 - media_16) / desvio_padrao_16
padrao_2017 = (indice_2017 - media_17) / desvio_padrao_17
padrao_2018 = (indice_2018 - media_18) / desvio_padrao_18
padrao_2019 = (indice_2019 - media_19) / desvio_padrao_19
padrao_2020 = (indice_2020 - media_20) / desvio_padrao_20
padrao_2021 = (indice_2021 - media_21) / desvio_padrao_21
```

In [ ]:
```
# Criação de um array com dados para fazer a predição

idh_pais_novo = np.array([padrao_2010, padrao_2011, padrao_2012, padrao_2013, pa

print(idh_pais_novo)
```

In [ ]:
```
# Resultado do rank do IDH do novo país padronizado

idh_pais_novo_padrao = model.predict(idh_pais_novo.reshape(1, -1))
print(idh_pais_novo_padrao)
```

In [ ]:
```
# Rank do IDH do país hipotético sem estar padronizado

rank_idh_pais = idh_pais_novo_padrao * desvio_padrao_idh + media_idh
print(rank_idh_pais)
```

In [ ]:
```
print(f'Um país que teve nos anos de 2010 até 2021 os IDHs:\n{indice_2010}\n{inc
```

**Considerações finais:**

Com esse modelo, além de prever a posição que um país estará no rank do Indíce de Desenvolvimento Humano (IDH) de acordo com a evolução do nível de desigualdade na educação, também é possível constatar que, um melhor acesso ao ensino está relacionado com uma melhor posição na tebela de IDH's.

---