Instituto Politécnico de Setúbal

Escola Superior de Tecnologia do Barreiro

Projeto de "BIG DATA"

Licenciatura em Bioinformática

Análise da População Mundial usando PySpark

Janeiro de 2023

Diogo Cabrita (202000212)
Alexandre Duarte (202000198)

Índice

1 Introdução

A análise da população mundial é um campo as vezes complexo pois consegue abranger uma ampla gama de disciplinas, incluindo demografia, economia, sociologia e epidemiologia. Nos últimos anos, o uso de técnicas de *Machine Learning* tornou-se cada vez mais utilizado na análise populacional, pois permitem o processamento eficiente de grandes quantidades de dados e a identificação de padrões e tendências.
No âmbito da UC de Big Data temos como objetivo selecionar e explorar uma fonte de dados aberto, neste caso foi o *The world bank* para assim, ser escolhida uma base de dados, que neste projeto está relacionado com a *Population, total - from The World Bank: Data*.
Para assim ser possível realizar-mos a *customer segmentation* no nosso conjunto de dados utilizando o programa PySpark e isto usando *Machine Learning*. Para que depois, seja possivel comparar os resultados obtidos, usando algoritmos de *Machine Learning*.
O Spark é uma ferramenta poderosa que permite o processamento distribuído de dados, o que é essencial quando se trata de lidar com grandes volumes de informações. Desta forma,então, vamos utilizar o Spark para limpar, ler e preparar dados, atravez de a aplicação de técnicas de Machine learning, de forma a avaliar e ajustar modelos e, finalmente, fazer previsões precisas e relevantes.
O conjunto de dados inclui informações como a População total, por pais e região, entre os anos de 1960-2021 para depois ser possivel utilizar técnicas de Machine learning para identificar padrões e tendências nos dados.

Para a realização do trabalho foram usados os softwares nas seguintes versões:
- PySpark - PySpark
- LaTeX - LaTeX2e

Figura 1: PySpark e LaTeX

2 Projeto

Para começarmos a analisar os nossos dados, temos de entender quantos dados/variáveis existem, e de que forma estão distribuídos ou divididos pelo próprio *data set*, de forma a que a visualização e a recolha de informação seja mais prática.
Criação de uma *SparkSession*:
Para se trabalhar com dados no *Spark*, é essencial criar uma *SparkSession*, este passo é fundamental sendo o primeiro a ser realizado. A *Session* é responsável por gerenciar a configuração do *Spark*, criar RDDs *(Resilient Distributed Datasets)* e *DataFrames*, para se registar tabelas temporárias e gerenciar os recursos dos *clusters* utilizados.
A *SparkSession* é o ponto de partida para trabalhar com dados estruturados e relacionais no *Spark SQL*, o módulo do *Spark* que fornece suporte para essa funcionalidade.
Este permite executar consultas *SQL*, e manipular *DataFrames* e extrair metadados, oferecendo acesso às bibliotecas de processamento de fluxo de dados (*DataFrame API e o SQL*) e às bibliotecas de *Machine Learning*(como o MLlib) do *Spark*.
A *SparkSession*(criada usando *SparkSession.builder*) é uma classe fundamental para trabalhar com dados no *Spark*, sendo através dela que é possível configurar e gerenciar a sessão, como tambem fazer conexões com o *clusters* e definir opções de configuração e gerenciar recursos como o número de núcleos. É a porta de entrada para trabalhar com dados estruturados e relacionais no *Spark SQL*.

2.1 Requisitos

Este tópico do projeto está relacionado com as bibliotecas necessarias para a realização do projeto, como tambem os devidos *imports*.

```
!pip install pyspark py4j
!pip install findspark

from pyspark import SparkContext, SparkConf
from pyspark.sql.session import SparkSession
from pyspark.sql.types import *
from pyspark.sql import functions as F
from pyspark.ml.feature import VectorAssembler, StandardScaler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.stat import Correlation
from google.colab import drive
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

2.2 Recolha dos dados

Esta parte do trabalho está relacionado com o começo da importação dos dados através dos datasets em estudo.
Link dataset:
`https://data.worldbank.org/indicator/SP.POP.TOTL`

- Inicialmente, este foi o código necessário para a recolha dos dados como tambem,respetivamente, a criação e iniciação da sessão spark, a montagem da pasta no drive, a preparação para a leitura do ficheiro csv e finalmente a leitura do ficheiro csv.

```
conf= SparkConf().setAppName('WorldPopulationPrediction').
    setMaster('local[*]')
sc= SparkContext.getOrCreate(conf=conf)
spark = SparkSession(sc)
--------------------
drive.mount('/content/drive')
--------------------
df = spark.read.format("csv").option("header","true")\
.option("inferSchema","true").load("../content/drive/MyDrive/
    Colab Notebooks/pop_tot.csv")
-----------------
df=spark.read.csv('../content/drive/MyDrive/Colab Notebooks/pop_
    tot.csv',inferSchema=True,header=True)
```

Com a importação destes dados, é onde conseguimos identificar as variáveis,topicos e classificações do dataset, neste caso, seja a população, seja o pais em questão, o ano em estudo etc.

2.3 Descrição do Dataset

Nesta parte do trabalho decidimos mostrar as variáveis e informações do dataset de uma forma mais rapida e facil de visualização, através de uma tabela com as variáveis em estudo com um exemplo de um pais.

NomePaís	CódigoPaís	NomeIndicador	CodigoIndicador	2021
Angola	AGO	PopulaçãoTotal	SP.POP.TOTL	34503774

Tabela 1: Exemplo do dataset (Angola)

Nesta próxima tabela temos como objetivo mostrar o tipo de variáveis para que se consiga perceber melhor para como poderiamos analisar este dataset e como está dividido.
Na segunda coluna a variável denominada de "Descrição"esta engloba as 4 variaveis referidas acima, sendo elas o NomePais, CodigoPais, NomeIndicador e CodigoIndicador.

	Descrição	1960-1989	1990-2021
Tipo de Variável	String	String	Long/Bigint

Tabela 2: Tipos de variáveis

2.4 Tratamento dos dados

Nesta parte do trabalho passamos para o tratamento dos dados do *dataset* em estudo, para começarmos a organizar e corrigir erros, de forma a que o *dataset* esteja pronto para se fazer análises, corretas e fiáveis.

- Em primeiro lugar, fizemos uma verificação dos tipos de variáveis de forma a que os valores e as variáveis em estudo estivessem compatíveis.

```
1 df.dtypes
```

- Em segundo lugar, alterámos alguns tipos de variáveis, neste caso de *string* para *integer*. (Apenas 1960-1989)

```
year_string = df.columns[4:34]

for c in year_string:
  df = df.withColumn(c, F.col(c).cast('Integer'))
```

- Em terceiro lugar, tinhamos como objetivo fazer a verificação dos valores nulos para que não existissem erros na nossa análise.

```
missing_values = []

for col in bd.columns:
    missing_values.append((col, bd.filter(bd[col].isNull()).count
       ()))

for col, val in missing_values:
    if val == 0:
        print("{} : Nenhum missing values encontrado".format(col)
           )
    else:
        print("{} : {} missing values".format(col, val))
```

- Em quarto lugar, foi onde ocorreu a remoção das 4 primeiras colunas (pois a análise está virada para a predição relativamente aos anos), e também a exclusão de quaisquer valores nulos no *dataset*.

```
bd=df.drop('Country Name', 'Country Code', 'Indicator Name', '
   Indicator Code').dropna(how ='any')
```

2.5 Análise e exploração do Dataset

Depois da recolha dos dados, a sua descrição, o processamento e tratamento dos dados, será possivel passarmos para o passo principal onde começamos a exploração do dataset, para que assim consigamos fazer análises sobre os nossos dados em estudo.

Então desta forma, conseguimos fazer uma análise basica do nosso dataset, como por exemplo:

- Numero de linhas: 259
- País com maior população em 2021: China - 1.412.360.000
- País com menor população em 2021: Tuvalu - 11.204
- População mundial em 2021: 7.888.409.342

2.5.1 VectorAssembler

- Neste proximo passo vamos recorrer a uma ferramenta disponibilizada na biblioteca *PySpark MLlib* onde o *VectorAssembler* é um transformador que é usado principalmente para combinar uma determinada lista de colunas em uma única coluna de vetor.
Isto geralmente é necessário como entrada para modelos de *Machine Learning*, que normalmente exigem um único vetor de recursos como entrada, como é o caso do nosso dataset.
O *VectorAssembler* seleciona as colunas de entrada, onde estas podem ser de diferentes tipos, e combina-as numa única coluna de saída do tipo "vetor".

- Então, para a criação deste *Vector*, em primeiro lugar,onde foi feita uma seleção das colunas que farão parte do vetor e criação do *VectorAssembler*.

```
1 vector_col = 'features'
2
3 years=['1960','1961','1962','1963','1964','1965','1966','1967','
    1968','1969',
4        '1970','1971','1972','1973','1974','1975','1976','1977','
           1978','1979',
```

```
5       '1980','1981','1982','1983','1984','1985','1986','1987','
            1988','1989',
6       '1990','1991','1992','1993','1994','1995','1996','1997','
            1998','1999',
7       '2000','2001','2002','2003','2004','2005','2006','2007','
            2008','2009',
8       '2010','2011','2012','2013','2014','2015','2016','2017','
            2018','2019',
9       '2020','2021']
10
11  assembler = VectorAssembler(inputCols=years, outputCol=vector_col
        )
```

- Neste segundo passo, vamos então consiliar a aplicação do *Assembler* no nosso dataset.

```
1  df_vector = assembler.transform(bd)
```

- Neste terceiro passo, serviu para verificar os dados do vasto e denso vetor.

```
1  df_vector.select('features').show(5,False)
```

2.5.2 Matriz de Correlação

Uma matriz de correlação é uma tabela que nos consegue mostrar os coeficientes de correlação entre múltiplas variáveis onde cada uma das células na tabela representa a correlação entre duas variáveis.
Esta matriz apresenta um valor sendo este o coeficiente de correlação onde, é um número entre -1 e 1 que descreve a força e a direção da relação linear entre duas variáveis. Um coeficiente de correlação com valor 1 indica que esta é uma correlação positiva perfeita, um coeficiente de valor -1 indica que é uma correlação negativa perfeita e um coeficiente de valor 0 indica nenhuma correlação.
Em suma, esta matriz é uma ferramenta muito útil no que toca a análise exploratória de dados, pois permite identificar rapidamente quais variáveis estão fortemente correlacionadas e quais não estão.Além disso, pode ser usado como uma ferramenta de diagnóstico e analises, para se verificar as suposições de certos modelos estatísticos.

- Primeiramente, para a criação da matriz necessitamos de calcular a correlação entre cada variável no nosso dataset.

```
matrix = Correlation.corr(df_vector,vector_col).collect()[0][0]
```

- De seguida, é onde ocorre um passo importante relacionado a transformaçao dos valores anteriormente recolhidos para depois serem um array e por fim, uma lista.

```
corrmatrix = matrix.toArray().tolist()
```

- No proximo passo, existe a criação de um dataframe/tabela com os valores de correlação entre os anos em estudo.

```
df_corr=spark.createDataFrame(corrmatrix, years)
```

Com a realização do nosso *VectorAssembler* e de seguida a Matriz de Correlação, isto permite-nos passar para o objetivo principal do trabalho onde iremos utilizar, já referido acima, metodos de *Machine Learning*.

Então, aqui está uma tabela que exemplifica a nossa matriz de correlação:

	1960	1961	1962	1963	1964
1960	1	0.99993	0.99986	0.99983	0.99977
1961	0.99993	1	0.99998	0.99995	0.99989
1962	0.99986	0.99998	1	0.99998	0.99994
1963	0.99983	0.99995	0.99998	1	0.99998
1964	0.99977	0.99989	0.99994	0.99998	1

Tabela 3: Matriz de Correlação (5 primeiras colunas)

- Então, com este exemplo é possivel observar que a correlação entre variáveis do nosso dataset é quase nula devido aos valores estarem proximos de 1,ou seja, as variaveis não apresentam um valor de correlação significativo.

3 Machine Learning

O *Machine Learning* é um ramo que está em rápido crescimento onde tem sido aplicados com sucesso e em uma ampla gama de áreas, incluindo a bioinformática.
No contexto da previsão populacional, métodos de *Machine Learning* conseguem ajudar muito bem quando falamos sobre previsões ou estatisticas que poderão vir a acontecer.
Então desta forma decidimos escolher os 2 metodos mais adequados para este tipo de dataset e analise, neste caso a regressão linear e o *Clustering*, podem ser usados para analisar dados populacionais históricos e fazer previsões sobre tendências populacionais futuras.
Tanto a regressão linear quanto o *Clustering* são técnicas poderosas de *Machine Learning* que podem ser usadas para analisar dados populacionais e fazer previsões sobre tendências populacionais futuras.Estas técnicas podem ser usadas em combinação com outras técnicas de análise de dados, como visualização e análise estatística, para obter uma melhor compreensão da dinâmica populacional.

3.1 Regressão Linear

Nesta parte do trabalho decidimos por começar importação dos dados em estudo através
A regressão linear é um dos método de *Machine Learning* que vamos utilizar para prever uma variável de destino contínua.
No contexto da previsão populacional mundial, podemos usar para modelar a relação entre uma população e fatores como crescimento econômico, taxa de fertilidade e expectativa de vida e outros variaveis que estejam disponiveis. Como ja referido acima, podemos então usar este metodo para fazer previsões sobre a população futura com base em dados históricos e também para identificar os fatores mais importantes que influenciam o crescimento populacional.

Passamos então para o começo da realização deste modelo:

- Primeiramente ocorre a criação do modelo para depois ser dividido em dados para teste e dados para treino, de forma a serem usados no nosso algoritmo de *Machine Learning*.

```
1 model_df=df_vector.select('features','2021')
```

- A proxima fase é onde se divide os dados em conjunto de treinamento e teste

```
train_df,test_df=model_df.randomSplit([0.7,0.3])
```

- Depois dos processos anteriores é possivel, respetivamente, em relação a cada linha de código, primeiro criar objeto *StandardScaler*, depois fazer um "treinamento"dos dados de treino, para assim ser possivel a aplicação do *scaler* no conjunto de treinamento e por fim a plicação do *scaler* no conjunto de teste.

```
scaler = StandardScaler(inputCol='features', outputCol='standard_
    features')
scalerModel = scaler.fit(train_df)
train = scalerModel.transform(train_df)
test = scalerModel.transform(test_df)
```

- Desta forma passamos a fase seguinte onde existe a criação do objeto *LinearRegression* para ser possivel treinar o modelo com os dados de treinamento e fazer a realização de previsões com os dados de *test*, cada linha de código representa os 3 passos.

```
lin_Reg=LinearRegression(featuresCol='standard_features',
    labelCol='2021')
model = lin_Reg.fit(train)
predictions = model.transform(test)
```

- Neste passo é onde criamos o gráfico dos resultados da Regressão Linear

```
y_actual = predictions.select('2021').toPandas()
y_pred = predictions.select('prediction').toPandas()

plt.figure(figsize=(15, 12))

plt.scatter(y_actual, y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Linear Regression Results')
plt.show()
```

- No proximo comando permite calcular quão grande é a inclinação dos dados, ou seja, se eles tiverem muita correlação vai ser perto de 0 e vice-versa.

```
model.intercept
```

- Neste passo usamos um modelo de regressão linear, onde os coeficientes representam a inclinação da linha de melhor ajuste para cada. *input*.
  O coeficiente para uma determinada característica representa a mudança na variável de destino para uma mudança de uma unidade nessa característica.

```
model.coefficients
```

- No proximo passo, criamos a variável que corresponde à avaliação, ou seja, quão bom são as *predictions* com os dados de treino, para depois se calcular a média dos erros quadrados.

```
training_predictions=model.evaluate(train)
training_predictions.meanSquaredError
```

- De seguida,como fizemos para os dados de treino, desta vez vamos avaliar os resultados dos dados de teste e ainda fazer uma verificação dos erros residuais, baseados nas predições.

```
test_results=model.evaluate(test)
test_results.residuals.show(10)
```

- No proximo passo é onde se calcula a media dos erros quadrados para assim depois, se calcular a raiz da media dos erros quadrados.

```
test_results.meanSquaredError
test_results.rootMeanSquaredError
```

3.1.1 Resultados

Depois da execução de todos os codigos acima relacionados com o nosso metodo é então possivel fazer-se a análise dos dados.

- Então este é o gráfico que representa a Regressão Linear:

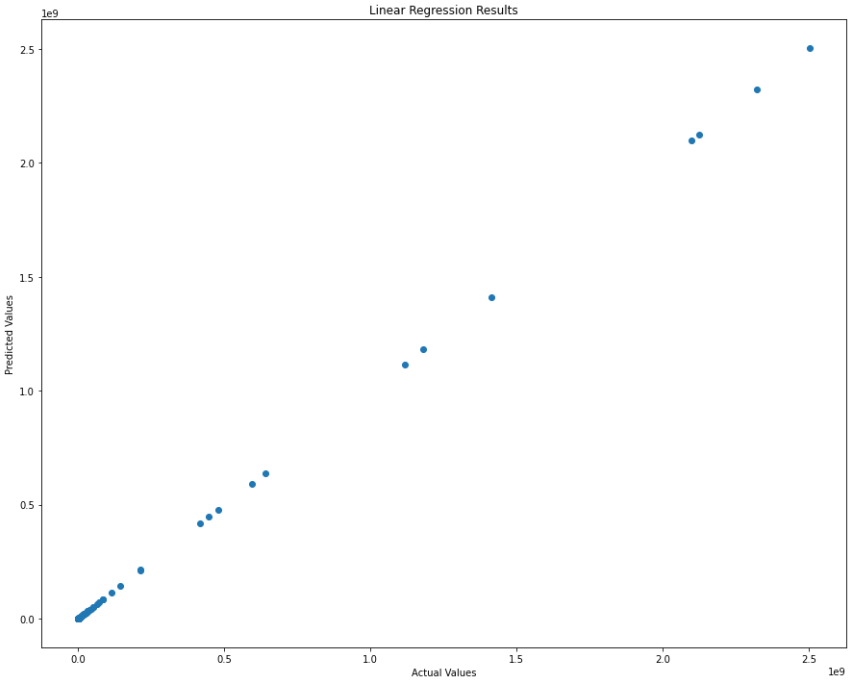

Figura 2: Gráfico Regressão Linear

Este gráfico apresenta os resultados da regressão linear, onde cada um dos ponto representa uma previsão feita pelo modelo para um determinado valor real. Então desta forma,podemos observar que a maioria dos pontos estão concentrados em torno da diagonal do gráfico, indicando que o modelo está a fazer previsões precisas.
Podemos observar o gráfico e ver que há uma certa correlação entre os valores valores reais e os valores previstos. Também detetamos que existem alguns pontos que estão fora do norma, isto quer dizer que o modelo tem erros de previsão.
Além disto, conseguimos o valor de RMSE que tem valor de aproximadamente 0.327, o que é considerado um valor aceitável para um modelo de previsão.
Portanto, podemos concluir que o nosso modelo de regressão linear pode e deve ser utilizado como um modelo de previsão de confiança, mas como referido anteriormente este pode cometer alguns erros.

3.2 Clustering

O *Clustering*, por outro lado, é um método de *Machine Learning* não supervisionado onde será usado para agrupar observações que sejam semelhantes.
No contexto da previsão populacional, o *Clustering* pode ser usado para identificar padrões e tendências em dados populacionais como tambem para agrupar regiões ou países com padrões de crescimento populacional semelhantes.
O *Clustering* pode ser usado para entender os fatores que impulsionam o crescimento populacional em diferentes regiões e para fazer previsões sobre futuras tendências populacionais nessas regiões.

Passamos então para o começo da realização deste modelo:

- Primeiramente começamos com a criação de uma lista vazia, onde vão ser inseridos a soma dos erros quadrados.
  Depois desse passo, criamos um *loop*(for) para definir o numero de k, para que nesse mesmo *loop* ser possivel criarmos o modelo *Kmeans*, para no fim apresentarmos com prints qual o valor de k a que o loop vai e qual a soma dos erros quadrados desse cluster.

```
errors=[]

for k in range(2,10):
    kmeans = KMeans(featuresCol='features',k=k)
    model = kmeans.fit(train)
    sse = model.summary.trainingCost
    errors.append(sse)
    print("With K={}".format(k))
    print("Within Set Sum of Squared Errors = " + str(sse))
    print('--'*30)
```

- No proximo passo, é onde fazemos a criação do gráfico de Cotovelo, defenindo o nº de *Clusters* e o custo dos erros.

```
plt.figure(figsize=(12, 6))

plt.plot(range(2,10), errors)
plt.xlabel('N mero de clusters')
plt.ylabel('Custo dos Errors')
plt.title('Gr fico de Cotovelo')
plt.show()
```

- De seguida, é onde ocorre a verificação dos dados que estão inseridos dentro de cada cluster.

```
centers = model.clusterCenters()
i=1
print("Cluster Centers: ")
for center in centers:
  print("Cluster " + str(i))
  print(center)
  i+=1
```

3.2.1 Resultados

Novamente, depois da execução de todos os codigos acima relacionados com este metodo, é então possivel fazer-se a análise dos dados.

- Com o codigo acima, é então possivel verificarmos o número de K com o respetivo valor da soma dos erros quadrados do nosso *dataset*:

```
With K=2
Within Set Sum of Squared Errors = 3.847326600115323e+20
------------------------------------------------------------
With K=3
Within Set Sum of Squared Errors = 1.8597556936888848e+20
------------------------------------------------------------
With K=4
Within Set Sum of Squared Errors = 6.004908317885996e+19
------------------------------------------------------------
With K=5
Within Set Sum of Squared Errors = 3.834910224250044e+19
------------------------------------------------------------
With K=6
Within Set Sum of Squared Errors = 3.0680555214319387e+19
------------------------------------------------------------
With K=7
Within Set Sum of Squared Errors = 2.1583962574696727e+19
------------------------------------------------------------
With K=8
Within Set Sum of Squared Errors = 1.7811907121401848e+19
------------------------------------------------------------
With K=9
Within Set Sum of Squared Errors = 1.9625454034482557e+19
------------------------------------------------------------
```

Figura 3: Output dos K´s

- Depois dos valores da soma dos erros quadrados para cada *cluster(K)*, podemos realizar o passo final onde é criado o gráfico de cotovelo.

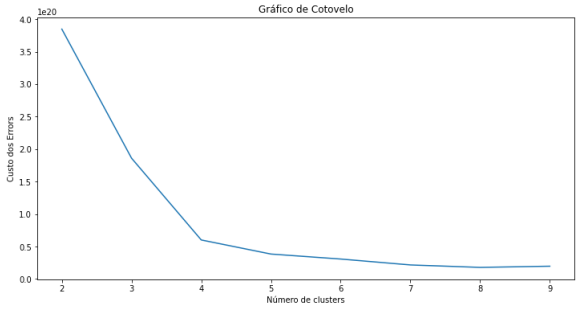

Figura 4: Gráfico Cotovelo

- Através do gráfico conseguimos verificar que o melhor valor de K para o nosso dataset seria o K= 5, pois este é o valor que seria mais aconselhado de usar, devido ao seu custo de erro(Y) e o nº de *clusters* serem os mais corretos, pois se a escolha for por exemplo quando k =6 o valor dos custos de erro são muito proximos dos valores em K=5, mas neste caso estamos a usar 6 clusters, logo o mais correto seria para K=5.

4 Conclusões

Como conclusão do nosso projeto após toda a analise realizada, conseguimos retirar que o nosso *dataset*, no geral, tem um nº de linhas equivalente a 259 onde estes representam os países, e sabendo que todos estes valores da população estão entre 1960-2021. Através desta interpretação inicial da população conseguimos retirar conclusões básicas onde apresentamos no ponto [2.5].
Depois desta análise passamos para um ponto importante onde fazemos uma matriz de correlação que nos vai ajudar a entender a correlação entre as nossas variáveis, desta forma, a conclusão é que as nossas variáveis tem uma grande correlação e a diferença não é muito significativa representado na matriz [3].
Após esta análise inicial, passamos para a parte principal do projeto onde iremos aplicar os metodos de *Machine Learning*. No nosso primeiro metodo utilizado denominado de Regressão Linear, foi possivel criarmos o gráfico desta regressão, de forma a tirarmos uma conclusão através da análise da figura [2], onde se conclui que a maioria das variáveis/valores estão em concordância, ou seja, em torno da diagonal principal, onde isto representa que o modelo poderá fazer previsões precisas. Com isto, entendemos que este modelo é uma boa opção para o nosso tipo de dados/*dataset*.
Passando para o proximo modelo de estudo denominado de *Clustering*, foi possivel calcularmos o valor da soma dos erros quadrados para cada valor de K, para depois se criar o gráfico de cotovelo [4]onde tiramos a conclusão que o valor mais adequando de K seria o K=5.

Alguns problemas que tivemos na realização do nosso projeto estão relacionados com o nosso dataset sendo um deles as variáveis de 1960 até 1989 serem do tipo string, sendo que os valores das mesmas são números, logo tivemos que mudar o tipo para *integer*, de seguida tambem tivemos de eliminar os valores nulos presentes e eliminamos as 4 primeiras colunas, pois estas não seriam precisas para o estudo do nº da população.

5 Bibliografia

https://spark.apache.org/docs/latest/api/python/

https://data.worldbank.org/indicator/SP.POP.TOTL

https://colab.research.google.com/drive/1JGNAgSePfZBdXRQ5myP4J5hlrQ0w1bEW?usp=sharing