@Odemir Depieri Jr

Tour pelo PySpark

Trabalhando com BigData

- ✓ Tratamento de Dados
- ✓ Pré-processamento
- ✓ Regressão Linear *(ML - Linear)*
- ✓ Regressão Logística *(ML - Classificação)*
- ✓ Random Forest *(ML - Classificação)*
- ✓ Naive Bayes *(ML – Classificação)*
- ✓ KMeans *(ML – Clusterização)*

100% Português (Br)

@Odemir Depieri Jr

Um resumo do Spark

O que é Apache Spark ?

O Apache Spark é um mecanismo multilíngue para executar **engenharia de dados**, **ciência de dados** e **aprendizado de máquina** em máquinas ou clusters de nó único.

PySpark

PySpark é a interface alto nível que permite você conseguir acessar e usar o Spark por meio da linguagem Python. Usando o PySpark, você consegue escrever todo o seu código usando apenas o nosso estilo Python de escrever código.

Big Data e Python

A biblioteca PySpark permite você criar seu servidor Apache Spark, trabalhar com grandes volumes de dados e até mesmo fazer streaming em tempo real.

Spark é o dos melhores framework para trabalhar com Big Data.
Tenha certeza que o PySpark vai te ajudar muito ao criar uma interface Python que permita a comunicação entre seu projeto e o servidor.

Fonte da pesquisa:

https://spark.apache.org/
https://bityli.com/kysit

@Odemir Depieri Jr

Iniciando e Instalando o PySpark no Google Colab

Instalando o PySpark

```
[1]  1 # Instalando o PySpark
     2 !pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.7/dist-packages (3.2.1)
Requirement already satisfied: py4j==0.10.9.3 in /usr/local/lib/python3.7/dist-packages (from pyspark) (0.10.9.
```

Tratamento dos Daodos

```
[2]  1 # Importando a Lib
     2 from pyspark.sql import SparkSession
     3
     4 # Ignorando avisos
     5 import warnings
     6 warnings.filterwarnings("ignore")
```

Iniciar a sessão do Spark

```
1 # Criando a seção do Spark (API)
2 Spark = SparkSession.builder.appName('Dataframe').getOrCreate()
3
4 # Verificando
5 Spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.2.1
Master
local[*]
AppName
Dataframe

Carregando uma base de dados.
Estava usando uma base com mais de 1.5 milhões de registros.

```
[4]  1 # Lendo um Arquivo CSV
     2 Base_Dados_Spark = Spark.read.csv( 'Dados_TesteSpark.csv', header=True, inferSchema=True, sep=';' )
```

```
[5]  1 # Verificando a estrutura da Base
     2 Base_Dados_Spark.printSchema()
```

```
root
 |-- Numero_01: integer (nullable = true)
 |-- Numero_02: integer (nullable = true)
 |-- Numero_03: integer (nullable = true)
 |-- Numero_04: integer (nullable = true)
 |-- Classe: integer (nullable = true)
```

```
[6]  1 # Verificando o tipo da informação
     2 type( Base_Dados_Spark )
```

```
pyspark.sql.dataframe.DataFrame
```

```
[7]  1 # Verificando primeiros registros
     2 Base_Dados_Spark.head(5)
```

```
[Row(Numero_01=316, Numero_02=706, Numero_03=617, Numero_04=803, Classe=1),
 Row(Numero_01=306, Numero_02=580, Numero_03=452, Numero_04=91, Classe=1),
 Row(Numero_01=705, Numero_02=628, Numero_03=4, Numero_04=665, Classe=1),
 Row(Numero_01=303, Numero_02=48, Numero_03=746, Numero_04=956, Classe=2),
 Row(Numero_01=39, Numero_02=489, Numero_03=89, Numero_04=702, Classe=1)]
```

Verificando os registros

```
[8]  1 # Mostrando os valores
     2 Base_Dados_Spark.show()
```

```
+---------+---------+---------+---------+------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|
+---------+---------+---------+---------+------+
|      316|      706|      617|      803|     1|
|      306|      580|      452|       91|     1|
|      705|      628|        4|      665|     1|
|      303|       48|      746|      956|     2|
|       39|      489|       89|      702|     1|
|      791|      136|      974|       48|     3|
|      874|      931|      386|      963|     1|
|      723|      880|       88|      357|     2|
|      787|       81|      412|      793|     2|
|      221|      477|      568|      330|     2|
|      726|      430|      434|      632|     1|
|      146|        4|      224|       27|     3|
|      982|      140|       47|      878|     2|
|      553|      420|      184|      519|     1|
|      281|      555|      379|      862|     1|
|      396|      812|      678|      700|     3|
|      754|      635|        9|      679|     2|
|      970|      152|      191|      984|     3|
|      151|       98|      925|      262|     3|
|      691|      635|      756|       46|     2|
+---------+---------+---------+---------+------+
only showing top 20 rows
```

```
[9]  1 # Selecionado algumas colunas
     2 Base_Dados_Spark.select( ['Numero_01','Numero_02'] ).show()
```

```
+---------+---------+
|Numero_01|Numero_02|
+---------+---------+
|      316|      706|
|      306|      580|
|      705|      628|
|      303|       48|
|       39|      489|
|      791|      136|
|      874|      931|
|      723|      880|
|      787|       81|
|      221|      477|
|      726|      430|
|      146|        4|
|      982|      140|
|      553|      420|
|      281|      555|
|      396|      812|
|      754|      635|
|      970|      152|
|      151|       98|
|      691|      635|
+---------+---------+
only showing top 20 rows
```

```
[10]  1 # Veriricando a coluna
      2 Base_Dados_Spark['Numero_01']
```

```
Column<'Numero_01'>
```

```
[11]  1 # Analisando a esutrura dos tipos dos dados
      2 Base_Dados_Spark.dtypes
```

```
[('Numero_01', 'int'),
 ('Numero_02', 'int'),
 ('Numero_03', 'int'),
 ('Numero_04', 'int'),
 ('Classe', 'int')]
```

```
[12]  1 # Verificando dados estatisticos
      2 Base_Dados_Spark.describe().show()
```

```
+-------+------------------+------------------+------------------+------------------+------------------+
|summary|         Numero_01|         Numero_02|         Numero_03|         Numero_04|            Classe|
+-------+------------------+------------------+------------------+------------------+------------------+
|  count|                99|                99|                99|                99|                99|
|   mean|492.64646464646466|516.2929292929293|454.74747474747477|541.1616161616162|2.0303030303030303|
| stddev| 280.6632201146578|289.2558098308892|272.13915795889875|305.3000433521599|0.7485297896617954|
|    min|                 8|                 4|                 4|                24|                 1|
|    max|               998|               995|               974|               999|                 3|
+-------+------------------+------------------+------------------+------------------+------------------+
```

Operando nas colunas

[13]
```
# Adicionando uma coluna
Base_Dados_Spark = Base_Dados_Spark.withColumn('Numero_05', Base_Dados_Spark['Numero_04'] + 2 )

# Verificando
Base_Dados_Spark.head(5)
```

```
[Row(Numero_01=316, Numero_02=706, Numero_03=617, Numero_04=803, Classe=1, Numero_05=805),
 Row(Numero_01=306, Numero_02=580, Numero_03=452, Numero_04=91, Classe=1, Numero_05=93),
 Row(Numero_01=705, Numero_02=628, Numero_03=4, Numero_04=665, Classe=1, Numero_05=667),
 Row(Numero_01=303, Numero_02=48, Numero_03=746, Numero_04=956, Classe=2, Numero_05=958),
 Row(Numero_01=39, Numero_02=489, Numero_03=89, Numero_04=702, Classe=1, Numero_05=704)]
```

[14]
```
# Excluindo a coluna
Base_Dados_Spark = Base_Dados_Spark.drop('Numero_05')

# Veriricando
Base_Dados_Spark.head(5)
```

```
[Row(Numero_01=316, Numero_02=706, Numero_03=617, Numero_04=803, Classe=1),
 Row(Numero_01=306, Numero_02=580, Numero_03=452, Numero_04=91, Classe=1),
 Row(Numero_01=705, Numero_02=628, Numero_03=4, Numero_04=665, Classe=1),
 Row(Numero_01=303, Numero_02=48, Numero_03=746, Numero_04=956, Classe=2),
 Row(Numero_01=39, Numero_02=489, Numero_03=89, Numero_04=702, Classe=1)]
```

[15]
```
# Renomeando as colunas
Base_Dados_Spark.withColumnRenamed( 'Numero_01', 'Coluna_01').show()
```

```
+---------+---------+---------+---------+------+
|Coluna_01|Numero_02|Numero_03|Numero_04|Classe|
+---------+---------+---------+---------+------+
|      316|      706|      617|      803|     1|
|      306|      580|      452|       91|     1|
|      705|      628|        4|      665|     1|
|      303|       48|      746|      956|     2|
|       39|      489|       89|      702|     1|
|      791|      136|      974|       48|     3|
|      874|      931|      386|      963|     1|
|      723|      880|       88|      357|     2|
|      787|       81|      412|      793|     2|
|      221|      477|      568|      330|     2|
|      726|      430|      434|      632|     1|
|      146|        4|      224|       27|     3|
|      982|      140|       47|      878|     2|
|      553|      420|      184|      519|     1|
|      281|      555|      379|      862|     1|
|      396|      812|      678|      700|     3|
|      754|      635|        9|      679|     2|
|      970|      152|      191|      984|     3|
|      151|       98|      925|      262|     3|
|      691|      635|      756|       46|     2|
+---------+---------+---------+---------+------+
only showing top 20 rows
```

Removendo valores nulos

[16]
```
# Retornar eliminado as linhas vazias/nulas
Base_Dados_Spark.na.drop().show()
```

```
+---------+---------+---------+---------+------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|
+---------+---------+---------+---------+------+
|      316|      706|      617|      803|     1|
|      306|      580|      452|       91|     1|
|      705|      628|        4|      665|     1|
|      303|       48|      746|      956|     2|
|       39|      489|       89|      702|     1|
|      791|      136|      974|       48|     3|
|      874|      931|      386|      963|     1|
|      723|      880|       88|      357|     2|
|      787|       81|      412|      793|     2|
|      221|      477|      568|      330|     2|
|      726|      430|      434|      632|     1|
|      146|        4|      224|       27|     3|
|      982|      140|       47|      878|     2|
|      553|      420|      184|      519|     1|
|      281|      555|      379|      862|     1|
|      396|      812|      678|      700|     3|
|      754|      635|        9|      679|     2|
|      970|      152|      191|      984|     3|
|      151|       98|      925|      262|     3|
|      691|      635|      756|       46|     2|
+---------+---------+---------+---------+------+
only showing top 20 rows
```

[17]
```
# # Retornar eliminado as linhas vazias/nulas de uma coluna especifica
Base_Dados_Spark.na.drop(how='any', subset=['Numero_02']).show()
```

```
+---------+---------+---------+---------+------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|
+---------+---------+---------+---------+------+
|      316|      706|      617|      803|     1|
|      306|      580|      452|       91|     1|
|      705|      628|        4|      665|     1|
|      303|       48|      746|      956|     2|
|       39|      489|       89|      702|     1|
|      791|      136|      974|       48|     3|
|      874|      931|      386|      963|     1|
|      723|      880|       88|      357|     2|
|      787|       81|      412|      793|     2|
|      221|      477|      568|      330|     2|
|      726|      430|      434|      632|     1|
|      146|        4|      224|       27|     3|
|      982|      140|       47|      878|     2|
|      553|      420|      184|      519|     1|
|      281|      555|      379|      862|     1|
|      396|      812|      678|      700|     3|
|      754|      635|        9|      679|     2|
|      970|      152|      191|      984|     3|
|      151|       98|      925|      262|     3|
|      691|      635|      756|       46|     2|
+---------+---------+---------+---------+------+
only showing top 20 rows
```

[18]
```
# Preenchendo o valor ausente
Base_Dados_Spark.na.fill('Qualquer_Valor', ['Numero_01','Numero_02'] ).show()
```

```
+---------+---------+---------+---------+------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|
+---------+---------+---------+---------+------+
|      316|      706|      617|      803|     1|
|      306|      580|      452|       91|     1|
|      705|      628|        4|      665|     1|
|      303|       48|      746|      956|     2|
|       39|      489|       89|      702|     1|
|      791|      136|      974|       48|     3|
|      874|      931|      386|      963|     1|
|      723|      880|       88|      357|     2|
|      787|       81|      412|      793|     2|
|      221|      477|      568|      330|     2|
|      726|      430|      434|      632|     1|
|      146|        4|      224|       27|     3|
|      982|      140|       47|      878|     2|
|      553|      420|      184|      519|     1|
|      281|      555|      379|      862|     1|
|      396|      812|      678|      700|     3|
|      754|      635|        9|      679|     2|
|      970|      152|      191|      984|     3|
|      151|       98|      925|      262|     3|
|      691|      635|      756|       46|     2|
+---------+---------+---------+---------+------+
only showing top 20 rows
```

[19]
```
# Função para inserir
from pyspark.ml.feature import Imputer

# Estimador de imputação para completar valores faltantes, usando a média, mediana ou moda
# Parametros para preencher [ mean, median or mode ]
Inserir = Imputer(
  inputCols=['Numero_01', 'Numero_02', 'Numero_03', 'Numero_04'],
  outputCols=[ '{}_imputed'.format(Loop) for Loop in ['Numero_01', 'Numero_02', 'Numero_03', 'Numero_04'] ]
  ).setStrategy('mean')
```

[20]
```
# Adiciona colunas de imputação ao df
Inserir.fit( Base_Dados_Spark ).transform( Base_Dados_Spark ).show()
```

```
+---------+---------+---------+---------+------+----------------+----------------+----------------+----------------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|Numero_01_imputed|Numero_02_imputed|Numero_03_imputed|Numero_04_imputed|
+---------+---------+---------+---------+------+----------------+----------------+----------------+----------------+
|      316|      706|      617|      803|     1|             316|             706|             617|             803|
|      306|      580|      452|       91|     1|             306|             580|             452|              91|
|      705|      628|        4|      665|     1|             705|             628|               4|             665|
|      303|       48|      746|      956|     2|             303|              48|             746|             956|
|       39|      489|       89|      702|     1|              39|             489|              89|             702|
|      791|      136|      974|       48|     3|             791|             136|             974|              48|
|      874|      931|      386|      963|     1|             874|             931|             386|             963|
|      723|      880|       88|      357|     2|             723|             880|              88|             357|
|      787|       81|      412|      793|     2|             787|              81|             412|             793|
|      221|      477|      568|      330|     2|             221|             477|             568|             330|
|      726|      430|      434|      632|     1|             726|             430|             434|             632|
|      146|        4|      224|       27|     3|             146|               4|             224|              27|
|      982|      140|       47|      878|     2|             982|             140|              47|             878|
|      553|      420|      184|      519|     1|             553|             420|             184|             519|
|      281|      555|      379|      862|     1|             281|             555|             379|             862|
|      396|      812|      678|      700|     3|             396|             812|             678|             700|
|      754|      635|        9|      679|     2|             754|             635|               9|             679|
|      970|      152|      191|      984|     3|             970|             152|             191|             984|
|      151|       98|      925|      262|     3|             151|              98|             925|             262|
|      691|      635|      756|       46|     2|             691|             635|             756|              46|
+---------+---------+---------+---------+------+----------------+----------------+----------------+----------------+
only showing top 20 rows
```

Filtrando

[21]
```
# Filtrar um valor
Base_Dados_Spark.filter('Numero_01 >= 900 ').show()
```

```
+---------+---------+---------+---------+------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|
+---------+---------+---------+---------+------+
|      982|      140|       47|      878|     2|
|      970|      152|      191|      984|     3|
|      980|      924|      193|      895|     2|
|      938|      237|      333|      830|     2|
|      916|      229|      334|      557|     3|
|      968|      143|      458|      666|     2|
|      998|      856|       74|      464|     2|
|      973|      389|      102|      389|     1|
+---------+---------+---------+---------+------+
```

[22]
```
# Filtrar um valor e colunas
Base_Dados_Spark.filter('Numero_01 >= 1000 ').select(['Numero_01', 'Numero_02']).show()
```

```
+---------+---------+
|Numero_01|Numero_02|
+---------+---------+
+---------+---------+
```

[23]
```
# Filtrando usando o and
Base_Dados_Spark.filter(
    ( Base_Dados_Spark['Numero_02'] == 1000) &
    (Base_Dados_Spark['Numero_01'] == 494 ) ).show()
```

```
+---------+---------+---------+---------+------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|
+---------+---------+---------+---------+------+
+---------+---------+---------+---------+------+
```

[24]
```
# Filtrando usando o OR
Base_Dados_Spark.filter(
    ( Base_Dados_Spark['Numero_02'] == 1000) |
    (Base_Dados_Spark['Numero_01'] == 494 ) ).show()
```

```
+---------+---------+---------+---------+------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|
+---------+---------+---------+---------+------+
+---------+---------+---------+---------+------+
```

[25]
```
# Agrupando valores usando a Soma
Base_Dados_Spark.groupBy('Classe').sum().show()
```

```
+------+--------------+--------------+--------------+--------------+-----------+
|Classe|sum(Numero_01)|sum(Numero_02)|sum(Numero_03)|sum(Numero_04)|sum(Classe)|
+------+--------------+--------------+--------------+--------------+-----------+
|     1|         12588|         12792|         11856|         14825|         26|
|     3|         14077|         14180|         13601|         16782|         87|
|     2|         22107|         24141|         19563|         21968|         88|
+------+--------------+--------------+--------------+--------------+-----------+
```

```
# Agrupando valores usando a Media
Base_Dados_Spark.groupBy('Classe').avg().show()
```

```
+------+------------------+------------------+------------------+------------------+-----------+
|Classe|    avg(Numero_01)|    avg(Numero_02)|    avg(Numero_03)|    avg(Numero_04)|avg(Classe)|
+------+------------------+------------------+------------------+------------------+-----------+
|     1|484.15384615384613|             492.0|             456.0| 570.1923076923077|        1.0|
|     3|485.41379310344826|488.9655172413793|             469.0| 578.6896551724138|        3.0|
|     2| 502.4318181818182|548.6590909090909|444.6136363636364|499.27272727272725|        2.0|
+------+------------------+------------------+------------------+------------------+-----------+
```

```
# Agrupando valores usando a Media
Base_Dados_Spark.groupBy('Classe').mean().show()
```

```
+------+------------------+------------------+------------------+------------------+-----------+
|Classe|    avg(Numero_01)|    avg(Numero_02)|    avg(Numero_03)|    avg(Numero_04)|avg(Classe)|
+------+------------------+------------------+------------------+------------------+-----------+
|     1|484.15384615384613|             492.0|             456.0| 570.1923076923077|        1.0|
|     3|485.41379310344826|488.9655172413793|             469.0| 578.6896551724138|        3.0|
|     2| 502.4318181818182|548.6590909090909|444.6136363636364|499.27272727272725|        2.0|
+------+------------------+------------------+------------------+------------------+-----------+
```

Agrupando

```
[28]  1 # Agrupando valores contando
      2 Base_Dados_Spark.groupBy('Classe').count().show()
```

```
+------+-----+
|Classe|count|
+------+-----+
|     1|   26|
|     3|   29|
|     2|   44|
+------+-----+
```

```
[29]  1 # Agrupando o argumento agg
      2 Base_Dados_Spark.agg( {'Classe':'count'} ).show()
```

```
+-------------+
|count(Classe)|
+-------------+
|           99|
+-------------+
```

```
[30]  1 # Verificando as colunas
      2 Base_Dados_Spark.columns
```

```
['Numero_01', 'Numero_02', 'Numero_03', 'Numero_04', 'Classe']
```

Join

```
[31]  1 # Realizando um JOIN
      2
      3 # Divindindo os dados em 2 partes
      4 Parte_1 = Base_Dados_Spark.filter( ( Base_Dados_Spark['Numero_02'] > 500 ) )
      5 Parte_2 = Base_Dados_Spark.filter( ( Base_Dados_Spark['Numero_02'] <= 500 ) )
      6
      7 # Aplicando o comnado
      8 Base_Join = Parte_1.join(Parte_2, Parte_1.Classe == Parte_2.Classe, 'inner')
      9
     10 Base_Join.show()
```

```
+---------+---------+---------+---------+------+---------+---------+---------+---------+------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|Numero_01|Numero_02|Numero_03|Numero_04|Classe|
+---------+---------+---------+---------+------+---------+---------+---------+---------+------+
|      316|      706|      617|      803|     1|      973|      389|      102|      389|     1|
|      316|      706|      617|      803|     1|      833|      176|      233|      151|     1|
|      316|      706|      617|      803|     1|      580|      372|       23|      507|     1|
|      316|      706|      617|      803|     1|      427|      378|      798|      810|     1|
|      316|      706|      617|      803|     1|      526|      207|      517|      887|     1|
|      316|      706|      617|      803|     1|      737|      351|      660|      450|     1|
|      316|      706|      617|      803|     1|      171|      327|      242|      754|     1|
|      316|      706|      617|      803|     1|      216|       57|      829|      788|     1|
|      316|      706|      617|      803|     1|      366|       31|      182|      507|     1|
|      316|      706|      617|      803|     1|      518|      107|      840|       54|     1|
|      316|      706|      617|      803|     1|      553|      420|      184|      519|     1|
|      316|      706|      617|      803|     1|      726|      430|      434|      632|     1|
|      316|      706|      617|      803|     1|       39|      489|       89|      702|     1|
|      306|      580|      452|       91|     1|      973|      389|      102|      389|     1|
|      306|      580|      452|       91|     1|      833|      176|      233|      151|     1|
|      306|      580|      452|       91|     1|      580|      372|       23|      507|     1|
|      306|      580|      452|       91|     1|      427|      378|      798|      810|     1|
|      306|      580|      452|       91|     1|      526|      207|      517|      887|     1|
|      306|      580|      452|       91|     1|      737|      351|      660|      450|     1|
|      306|      580|      452|       91|     1|      171|      327|      242|      754|     1|
+---------+---------+---------+---------+------+---------+---------+---------+---------+------+
only showing top 20 rows
```

Ordenando

```
[32]  1 # Ordenando por coluna
      2 Base_Dados_Spark.sort('Classe').show()
```

```
+---------+---------+---------+---------+------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|
+---------+---------+---------+---------+------+
|      518|      107|      840|       54|     1|
|      737|      351|      660|      450|     1|
|       19|      667|      480|       65|     1|
|      306|      580|      452|       91|     1|
|       39|      489|       89|      702|     1|
|      874|      931|      386|      963|     1|
|      726|      430|      434|      632|     1|
|      553|      420|      184|      519|     1|
|      281|      555|      379|      862|     1|
|      316|      706|      617|      803|     1|
|      705|      628|        4|      665|     1|
|      366|       31|      182|      507|     1|
|      363|      733|      560|      760|     1|
|      734|      721|      619|      201|     1|
|      216|       57|      829|      788|     1|
|       48|      838|      518|      936|     1|
|      171|      327|      242|      754|     1|
|      580|      516|       51|      805|     1|
|      526|      207|      517|      887|     1|
|      552|      790|      956|      511|     1|
+---------+---------+---------+---------+------+
only showing top 20 rows
```

Pré-processamento

```
[33]  1 # Função de Vetor
      2 from pyspark.ml.feature import VectorAssembler
      3
      4 # Selecionado os dados
      5 Amostra = Base_Dados_Spark
      6
      7 # Vetorizando os dados
      8 Dados_Amostra = VectorAssembler(
      9     inputCols=['Numero_01', 'Numero_02', 'Numero_03', 'Numero_04', 'Classe'],
     10     outputCol='Correlação' )
     11
     12 # Aplicando a transformação
     13 Dados_Amostral = Dados_Amostra.transform( Amostra )
     14
     15 # Função da regressão Linear
     16 from pyspark.ml.feature import MinMaxScaler
     17
     18 # Definindo os parametros da função
     19 Funcao_MinMax = MinMaxScaler(outputCol='Valor_Esclada_MixMAx')
     20
     21 # Setando a coluna para ser aplicado
     22 Funcao_MinMax.setInputCol('Correlação')
     23
     24 # Fitando o modelo
     25 Modelo_MinMax = Funcao_MinMax.fit( Dados_Amostral )
     26
     27 Modelo_MinMax.transform(Dados_Amostral).show()
```

```
+---------+---------+---------+---------+------+--------------------+--------------------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|          Correlação|Valor_Esclada_MixMAx|
+---------+---------+---------+---------+------+--------------------+--------------------+
|      316|      706|      617|      803|     1|[316.0,706.0,617....|[0.31111111111111...|
|      306|      580|      452|       91|     1|[306.0,580.0,452....|[0.30101010101010...|
|      705|      628|        4|      665|     1|[705.0,628.0,4.0,...|[0.70404040404040...|
|      303|       48|      746|      956|     2|[303.0,48.0,746.0...|[0.29797979797979...|
|       39|      489|       89|      702|     1|[39.0,489.0,89.0,...|[0.03131313131313...|
|      791|      136|      974|       48|     3|[791.0,136.0,974....|[0.79090909090909...|
|      874|      931|      386|      963|     1|[874.0,931.0,386....|[0.87474747474747...|
|      723|      880|       88|      357|     2|[723.0,880.0,88.0...|[0.72222222222222...|
|      787|       81|      412|      793|     2|[787.0,81.0,412.0...|[0.78686868686868...|
|      221|      477|      568|      330|     2|[221.0,477.0,568....|[0.21515151515151...|
|      726|      430|      434|      632|     1|[726.0,430.0,434....|[0.72525252525252...|
|      146|        4|      224|       27|     3|[146.0,4.0,224.0,...|[0.13939393939393...|
|      982|      140|       47|      878|     2|[982.0,140.0,47.0...|[0.98383838383838...|
|      553|      420|      184|      519|     1|[553.0,420.0,184....|[0.55050505050505...|
|      281|      555|      379|      862|     1|[281.0,555.0,379....|[0.27575757575757...|
|      396|      812|      678|      700|     3|[396.0,812.0,678....|[0.39191919191919...|
|      754|      635|        9|      679|     2|[754.0,635.0,9.0,...|[0.75353535353535...|
|      970|      152|      191|      984|     3|[970.0,152.0,191....|[0.97171717171717...|
|      151|       98|      925|      262|     3|[151.0,98.0,925.0...|[0.14444444444444...|
|      691|      635|      756|       46|     2|[691.0,635.0,756....|[0.68989898989898...|
+---------+---------+---------+---------+------+--------------------+--------------------+
only showing top 20 rows
```

Correlação

```
[34]
# Calcurando a Correlação

# Gerando uma amostra devido termos +1mm de registros
Amostra = Base_Dados_Spark

# Vetorizando os dados
Dados_Correlacao = VectorAssembler(
    inputCols=['Numero_01', 'Numero_02', 'Numero_03', 'Numero_04', 'Classe'],
    outputCol='Correlação' )

# Aplicando a transformação
Saida_Correlcao = Dados_Correlacao.transform( Amostra )

# Função da Correlação
from pyspark.ml.stat import Correlation

# Aplicando a Função da correlação de Pearson
Correlacao_Pearson = Correlation.corr( Saida_Correlcao, 'Correlação', 'pearson').collect()
# Plotando o valor
print( str(Correlacao_Pearson).replace('nan', 'NaN') )

# Aplicando a Função da correlação de Spearman
Correlacao_Spearman = Correlation.corr(Saida_Correlcao, 'Correlação', method='spearman').collect()[0][0]
# Plotando o valor
print( '\n', str(Correlacao_Spearman).replace('nan', 'NaN') );
```

```
[Row(pearson(Correlação)=DenseMatrix(5, 5, [1.0, -0.0734, -0.163, 0.074, 0.0005, -0.0734, 1.0, 0.1117, ..., -0.1151

 DenseMatrix([[ 1.00000000e+00, -6.45503431e-02, -1.62613794e-01,
               7.04068128e-02,  4.80629708e-03],
             [-6.45503431e-02,  1.00000000e+00,  1.26782296e-01,
               1.05897232e-01, -7.16473292e-04],
             [-1.62613794e-01,  1.26782296e-01,  1.00000000e+00,
              -9.97495284e-02,  8.78672168e-03],
             [ 7.04068128e-02,  1.05897232e-01, -9.97495284e-02,
               1.00000000e+00,  2.28341986e-02],
```

Regressão Linear

```
[35]
# Regressão Linear
from pyspark.ml.feature import VectorAssembler

# Separar os dados de caracteristcas e previsor
Selecao_Dados = VectorAssembler(
    inputCols=['Numero_01', 'Numero_02', 'Numero_03'], outputCol='Variaveis_Caracteristicas' )

# Aplicando a transformação
Saida = Selecao_Dados.transform( Base_Dados_Spark )

# Veriricando
Saida.show()
```

```
+---------+---------+---------+---------+------+-------------------------+
|Numero_01|Numero_02|Numero_03|Numero_04|Classe|Variaveis_Caracteristicas|
+---------+---------+---------+---------+------+-------------------------+
|      316|      706|      617|      803|     1|      [316.0,706.0,617.0]|
|      306|      580|      452|       91|     1|      [306.0,580.0,452.0]|
|      705|      628|        4|      665|     1|        [705.0,628.0,4.0]|
|      303|       48|      746|      956|     2|       [303.0,48.0,746.0]|
|       39|      489|       89|      702|     1|        [39.0,489.0,89.0]|
|      791|      136|      974|       48|     3|      [791.0,136.0,974.0]|
|      874|      931|      386|      963|     1|      [874.0,931.0,386.0]|
|      723|      880|       88|      357|     2|       [723.0,880.0,88.0]|
|      787|       81|      412|      793|     2|       [787.0,81.0,412.0]|
|      221|      477|      568|      330|     2|      [221.0,477.0,568.0]|
|      726|      430|      434|      632|     1|      [726.0,430.0,434.0]|
|      146|        4|      224|       27|     3|        [146.0,4.0,224.0]|
|      982|      140|       47|      878|     2|       [982.0,140.0,47.0]|
|      553|      420|      184|      519|     1|      [553.0,420.0,184.0]|
|      281|      555|      379|      862|     1|      [281.0,555.0,379.0]|
|      396|      812|      678|      700|     3|      [396.0,812.0,678.0]|
|      754|      635|        9|      679|     2|        [754.0,635.0,9.0]|
|      970|      152|      191|      984|     3|      [970.0,152.0,191.0]|
|      151|       98|      925|      262|     3|       [151.0,98.0,925.0]|
|      691|      635|      756|       46|     2|      [691.0,635.0,756.0]|
+---------+---------+---------+---------+------+-------------------------+
only showing top 20 rows
```

Continuação da Regressão

```
[38]  1 # Treinamento do Modelo
      2
      3 # Função da regressão Linear
      4 from pyspark.ml.regression import LinearRegression
      5
      6 # Separando os dados de Teste e Treino
      7 Dados_Treino, Dados_Teste = Dados_Modelo.randomSplit( [0.75, 0.25] )
      8
      9 # Aplicando as features para o modelo linear
     10 Funcao_Linear = LinearRegression( featuresCol='Variaveis_Caracteristicas', labelCol='Numero_04')
     11
     12 # Fitando o Modelo
     13 Funcao_Linear = Funcao_Linear.fit(Dados_Treino)
     14
     15 # Veriricando os coeficientes
     16 Funcao_Linear.coefficients
```

```
DenseVector([0.1345, 0.2464, -0.1219])
```

```
[39]  1 # Fazendo a previsões dos dados de Teste
      2 Previoes = Funcao_Linear.evaluate( Dados_Teste )
      3
      4 # Mostrando as previsões
      5 Previoes.predictions.show()
```

```
+-------------------------+---------+------------------+
|Variaveis_Caracteristicas|Numero_04|        prediction|
+-------------------------+---------+------------------+
|        [39.0,489.0,89.0]|      702| 505.6934710791836|
|       [49.0,953.0,418.0]|      876|  581.269095129581|
|      [126.0,154.0,137.0]|      889|428.97986313855927|
|      [171.0,327.0,242.0]|      754|464.86228797793945|
|      [177.0,945.0,401.0]|       77| 598.5838567503675|
|      [262.0,763.0,680.0]|      289| 531.1379102159367|
|      [316.0,706.0,617.0]|      803| 532.0349298953695|
|      [329.0,796.0,733.0]|      600| 541.8175203144399|
|       [413.0,47.0,633.0]|      962| 380.7195629598334|
|      [619.0,207.0,445.0]|      926| 470.7797055052423|
|       [723.0,880.0,88.0]|      357| 694.1594591995411|
|      [734.0,721.0,619.0]|      201| 591.6997195553748|
|      [862.0,506.0,500.0]|      512| 570.4385433103265|
|      [968.0,143.0,458.0]|      666|500.35474763967875|
+-------------------------+---------+------------------+
```

Regressão Logística

```
[40]  1 # -- Construção Modelo Logistico
      2 from pyspark.ml.feature import VectorAssembler
      3
      4 # Separar os dados de caracteristcas e previsor
      5 Selecao_Dados = VectorAssembler(
      6     inputCols=['Numero_01', 'Numero_02', 'Numero_03'], outputCol='Variaveis_Caracteristicas' )
      7
      8 # Aplicando a transformação
      9 Saida = Selecao_Dados.transform( Base_Dados_Spark )
     10
     11 # Selecionado a coluna previsor
     12 Dados_Modelo = Saida.select('Variaveis_Caracteristicas', 'Classe')
     13
     14 # Veriricando
     15 Dados_Modelo.head(5)
```

```
[Row(Variaveis_Caracteristicas=DenseVector([316.0, 706.0, 617.0]), Classe=1),
 Row(Variaveis_Caracteristicas=DenseVector([306.0, 580.0, 452.0]), Classe=1),
 Row(Variaveis_Caracteristicas=DenseVector([705.0, 628.0, 4.0]), Classe=1),
 Row(Variaveis_Caracteristicas=DenseVector([303.0, 48.0, 746.0]), Classe=2),
 Row(Variaveis_Caracteristicas=DenseVector([39.0, 489.0, 89.0]), Classe=1)]
```

Continuação da Regressão

```
# Função Logistica
from pyspark.ml.classification import LogisticRegression

# Separando os dados de Teste e Treino
Dados_Treino, Dados_Teste = Dados_Modelo.randomSplit( [0.75, 0.25] )

# Aplicando as features para o modelo linea
Funcao_Logistica = LogisticRegression(featuresCol = 'Variaveis_Caracteristicas', labelCol = 'Classe', maxIter=1000)

# Fitando o Modelo
Funcao_Logistica = Funcao_Logistica.fit(Dados_Treino)

# Fazendo a previsões dos dados de Teste
Previoes = Funcao_Logistica.evaluate( Dados_Teste )

# Mostrando as previsões
Previoes.predictions.show()
```

```
+-------------------------+------+--------------------+--------------------+----------+
|Variaveis_Caracteristicas|Classe|       rawPrediction|         probability|prediction|
+-------------------------+------+--------------------+--------------------+----------+
|       [48.0,838.0,518.0]|     1|[-16.907459680451...|[4.95840595284761...|       2.0|
|       [49.0,953.0,418.0]|     3|[-16.824641484571...|[5.36377553415517...|       2.0|
|      [171.0,327.0,242.0]|     1|[-17.508217077354...|[2.41376346539577...|       3.0|
|      [262.0,763.0,680.0]|     2|[-17.004938076398...|[4.47928410764775...|       2.0|
|      [280.0,694.0,248.0]|     2|[-17.191531797229...|[3.55014073947933...|       2.0|
|      [287.0,679.0,919.0]|     2|[-17.029572889327...|[4.40589136530800...|       2.0|
|      [299.0,154.0,344.0]|     2|[-17.687451599033...|[1.90060591658064...|       3.0|
|       [303.0,48.0,746.0]|     2|[-17.682940920292...|[1.89950053022615...|       3.0|
|      [325.0,784.0,233.0]|     2|[-17.124144355503...|[3.82376451743130...|       2.0|
|      [355.0,817.0,894.0]|     1|[-16.926455150734...|[4.93245981800289...|       2.0|
|      [363.0,733.0,560.0]|     1|[-17.098064549541...|[4.00120290126989...|       2.0|
|      [392.0,951.0,950.0]|     3|[-16.795606951791...|[5.68584493256866...|       2.0|
|      [407.0,509.0,679.0]|     2|[-17.294359683659...|[3.18291697518008...|       2.0|
|       [413.0,47.0,633.0]|     2|[-17.749431659974...|[1.73574875736086...|       3.0|
|      [558.0,810.0,418.0]|     2|[-17.125243170880...|[3.83609978878100...|       2.0|
|      [578.0,459.0,481.0]|     3|[-17.449898237052...|[2.60692683122368...|       2.0|
|      [667.0,967.0,607.0]|     2|[-16.960275848684...|[4.62823953423381...|       2.0|
|      [691.0,635.0,756.0]|     2|[-17.245206820038...|[3.37731658987232...|       2.0|
|      [723.0,549.0,539.0]|     3|[-17.395334110280...|[2.79302648149763...|       2.0|
|       [723.0,880.0,88.0]|     2|[-17.199448815722...|[3.45438339493292...|       2.0|
+-------------------------+------+--------------------+--------------------+----------+
only showing top 20 rows
```

Random Forest

```
[42]
# -- Construção Modelo Logistico
from pyspark.ml.feature import VectorAssembler

# Separar os dados de caracteristcas e previsor
Selecao_Dados = VectorAssembler(
    inputCols=['Numero_01', 'Numero_02', 'Numero_03'], outputCol='Variaveis_Caracteristicas' )

# Aplicando a transformação
Saida = Selecao_Dados.transform( Base_Dados_Spark )

# Selecionado a coluna previsor
Dados_Modelo = Saida.select('Variaveis_Caracteristicas', 'Classe')

# Veriricando
Dados_Modelo.head(5)
```

```
[Row(Variaveis_Caracteristicas=DenseVector([316.0, 706.0, 617.0]), Classe=1),
 Row(Variaveis_Caracteristicas=DenseVector([306.0, 580.0, 452.0]), Classe=1),
 Row(Variaveis_Caracteristicas=DenseVector([705.0, 628.0, 4.0]), Classe=1),
 Row(Variaveis_Caracteristicas=DenseVector([303.0, 48.0, 746.0]), Classe=2),
 Row(Variaveis_Caracteristicas=DenseVector([39.0, 489.0, 89.0]), Classe=1)]
```

Continuação Random Forest

```
[43]  1 # Função Random Forest
      2 from pyspark.ml.classification import RandomForestClassifier
      3
      4 # Separando os dados de Teste e Treino
      5 Dados_Treino, Dados_Teste = Dados_Modelo.randomSplit( [0.75, 0.25] )
      6
      7 # Aplicando as features para o modelo linea
      8 Floresta_Decisao = RandomForestClassifier(numTrees=100, maxDepth=2,
      9                                          featuresCol = 'Variaveis_Caracteristicas',
     10                                          labelCol = 'Classe')
     11
     12 # Fitando o Modelo
     13 Floresta_Decisao = Floresta_Decisao.fit(Dados_Treino)
     14
     15 # Fazendo a previsões dos dados de Teste
     16 Previoes = Floresta_Decisao.evaluate( Dados_Teste )
     17
     18 # Mostrando as previsões
     19 Previoes.predictions.show()
```

```
+-------------------------+------+--------------------+--------------------+----------+
|Variaveis_Caracteristicas|Classe|       rawPrediction|         probability|prediction|
+-------------------------+------+--------------------+--------------------+----------+
|       [19.0,667.0,480.0]|     1|[0.0,18.780937689...|[0.0,0.1878093768...|       3.0|
|       [48.0,838.0,518.0]|     1|[0.0,12.606309650...|[0.0,0.1260630965...|       3.0|
|       [79.0,983.0,746.0]|     2|[0.0,17.572142649...|[0.0,0.1757214264...|       3.0|
|      [287.0,679.0,919.0]|     2|[0.0,47.072560590...|[0.0,0.4707256059...|       1.0|
|       [303.0,48.0,746.0]|     2|[0.0,45.244959173...|[0.0,0.4524495917...|       1.0|
|      [305.0,255.0,447.0]|     2|[0.0,27.808429281...|[0.0,0.2780842928...|       2.0|
|      [329.0,796.0,733.0]|     2|[0.0,43.052830279...|[0.0,0.4305283027...|       1.0|
|      [395.0,182.0,827.0]|     2|[0.0,60.200983714...|[0.0,0.6020098371...|       1.0|
|      [402.0,890.0,207.0]|     2|[0.0,18.625380283...|[0.0,0.1862538028...|       2.0|
|      [473.0,370.0,149.0]|     2|[0.0,27.649587153...|[0.0,0.2764958715...|       2.0|
|      [485.0,182.0,919.0]|     2|[0.0,47.323135848...|[0.0,0.4732313584...|       1.0|
|      [553.0,420.0,184.0]|     1|[0.0,27.619986331...|[0.0,0.2761998633...|       2.0|
|      [578.0,459.0,481.0]|     3|[0.0,26.484213424...|[0.0,0.2648421342...|       2.0|
|       [580.0,516.0,51.0]|     1|[0.0,25.790745015...|[0.0,0.2579074501...|       2.0|
|      [628.0,405.0,110.0]|     2|[0.0,28.064697869...|[0.0,0.2806469786...|       2.0|
|      [681.0,815.0,955.0]|     3|[0.0,46.651022687...|[0.0,0.4665102268...|       1.0|
|      [691.0,635.0,756.0]|     2|[0.0,43.061239730...|[0.0,0.4306123973...|       1.0|
|        [705.0,628.0,4.0]|     1|[0.0,22.224655135...|[0.0,0.2222465513...|       2.0|
|      [719.0,749.0,370.0]|     2|[0.0,22.523702978...|[0.0,0.2252370297...|       2.0|
|      [737.0,351.0,660.0]|     1|[0.0,27.432170227...|[0.0,0.2743217022...|       2.0|
+-------------------------+------+--------------------+--------------------+----------+
only showing top 20 rows
```

Naive Bayes

```
[44]  1 # -- Construção Modelo Logistico
      2 from pyspark.ml.feature import VectorAssembler
      3
      4 # Separar os dados de caracteristcas e previsor
      5 Selecao_Dados = VectorAssembler(
      6     inputCols=['Numero_01', 'Numero_02', 'Numero_03'], outputCol='Variaveis_Caracteristicas' )
      7
      8 # Aplicando a transformação
      9 Saida = Selecao_Dados.transform( Base_Dados_Spark )
     10
     11 # Selecionado a coluna previsor
     12 Dados_Modelo = Saida.select('Variaveis_Caracteristicas', 'Classe')
     13
     14 # Veriricando
     15 Dados_Modelo.head(5)
```

```
[Row(Variaveis_Caracteristicas=DenseVector([316.0, 706.0, 617.0]), Classe=1),
 Row(Variaveis_Caracteristicas=DenseVector([306.0, 580.0, 452.0]), Classe=1),
 Row(Variaveis_Caracteristicas=DenseVector([705.0, 628.0, 4.0]), Classe=1),
 Row(Variaveis_Caracteristicas=DenseVector([303.0, 48.0, 746.0]), Classe=2),
 Row(Variaveis_Caracteristicas=DenseVector([39.0, 489.0, 89.0]), Classe=1)]
```

Continuação Naive Bayes

[45]
```
# Função Random Forest
from pyspark.ml.classification import NaiveBayes

# Separando os dados de Teste e Treino
Dados_Treino, Dados_Teste = Dados_Modelo.randomSplit( [0.75, 0.25] )

# Aplicando as features para o modelo linea
Funcao_NaiveBayes = NaiveBayes(smoothing=1.0, modelType='multinomial',
                                      featuresCol = 'Variaveis_Caracteristicas',
                                      labelCol = 'Classe')

# Fitando o Modelo
Funcao_NaiveBayes = Funcao_NaiveBayes.fit(Dados_Treino)

# Fazendo a previsões dos dados de Teste
Previoes = Funcao_NaiveBayes.transform( Dados_Teste )

# Mostrando as previsões
Previoes.show()
```

```
+-------------------------+------+--------------------+--------------------+----------+
|Variaveis_Caracteristicas|Classe|       rawPrediction|         probability|prediction|
+-------------------------+------+--------------------+--------------------+----------+
|       [13.0,935.0,631.0]|     3|[-1750.5841243277...|[8.89630790655601...|       1.0|
|       [19.0,667.0,480.0]|     1|[-1292.8352707934...|[8.77046157546633...|       1.0|
|        [39.0,489.0,89.0]|     1|[-682.97208892578...|[3.12444944059513...|       1.0|
|       [88.0,753.0,446.0]|     3|[-1424.5895225806...|[1.31308277748837...|       1.0|
|      [125.0,391.0,196.0]|     3|[-786.28920135591...|[9.64617431452898...|       1.0|
|        [146.0,4.0,224.0]|     3|[-412.06704398172...|[2.20543726329240...|       2.0|
|       [187.0,435.0,24.0]|     3|[-710.66117265844...|[1.31937063528319...|       1.0|
|      [299.0,154.0,344.0]|     2|[-876.38108574521...|[9.79314175512917...|       2.0|
|      [305.0,255.0,447.0]|     2|[-1109.0335988374...|[8.07848567107883...|       2.0|
|      [336.0,707.0,643.0]|     2|[-1860.2039198272...|[3.38788850181533...|       1.0|
|      [377.0,567.0,411.0]|     2|[-1491.7424484167...|[1.90158789585352...|       1.0|
|      [395.0,182.0,827.0]|     2|[-1547.7432547716...|[1.46620880808286...|       2.0|
|      [619.0,207.0,445.0]|     3|[-1392.4524733568...|[9.81673898901764...|       2.0|
|      [636.0,824.0,626.0]|     3|[-2294.3276513193...|[3.71453205292662...|       1.0|
|      [667.0,967.0,607.0]|     2|[-2464.7997237973...|[5.54637317300686...|       1.0|
|      [693.0,995.0,280.0]|     2|[-2160.3651900849...|[1.22375338155470...|       1.0|
|       [723.0,880.0,88.0]|     2|[-1852.1443576105...|[1.01629297111159...|       1.0|
|      [756.0,507.0,618.0]|     3|[-2064.3042843831...|[6.51099866136853...|       2.0|
|      [780.0,751.0,180.0]|     3|[-1873.2218824617...|[5.99479613328689...|       1.0|
|      [862.0,506.0,500.0]|     3|[-2046.3930298398...|[0.00574385534861...|       2.0|
+-------------------------+------+--------------------+--------------------+----------+
only showing top 20 rows
```

Kmeans

[46]
```
# Construindo modelo de Cluster

# Função do Kmeans
from pyspark.ml.clustering import KMeans

# Parametro do modelo
Funcao_KMeans = KMeans(k=2)

# Ajustando Nome das Colunas de Treino
Dados_Treino_Cluster = Dados_Treino.withColumnRenamed( 'Variaveis_Caracteristicas', 'features')
Dados_Treino_Cluster = Dados_Treino_Cluster.withColumnRenamed( 'Classe', 'weighCol')

# Fitar o Modelo
Modelo = Funcao_KMeans.fit( Dados_Treino_Cluster )
```

[47]
```
# Ajustando Nome das Colunas de Treino
Dados_Teste_Cluster = Dados_Teste.withColumnRenamed( 'Variaveis_Caracteristicas', 'features')
Dados_Teste_Cluster = Dados_Teste_Cluster.withColumnRenamed( 'Classe', 'weighCol')

# Fazendo previsão do primeiro registro
print( 'Previsão do Modelo:', Modelo.predict( Dados_Teste_Cluster.head().features ) )

print( 'Valore Real:', Dados_Teste_Cluster.select('weighCol').head() )
```

```
Previsão do Modelo: 0
Valore Real: Row(weighCol=3)
```

Final

Esse guia foi elaborada para demostrar o uso do PySpark

Link do código
https://colab.research.google.com/drive/1_ZPm80kl7zs5EDSUmcvtGVTp6WrRZiGj?usp=sharing

Odemir Depieri Jr

Data Intelligence Analyst Sr
Tech Lead
Specialization AI