Laianna Lana Virginio da Silva - llvs2@cin.ufpe.br
Lucas Natan Correia Couri - lncc2@cin.ufpe.br
Priscilla Amarante de Lima - pal4@cin.ufpe.br

Centro de Informática CIn - UFPE

Curso: PD em Larga Escala

Projeto Final

O Random Forest é um método de aprendizado ensemble que pode ser utilizado tanto para regressão como para classificação, ele constrói coleções de árvores de decisão no processo de aprendizado de forma a obter melhor desempenho que cada árvore de decisão poderia oferecer individualmente. No caso da classificação, como o problema abordado neste projeto, o resultado do random forest é a classe selecionada pela maioria das árvores de decisão. O método é bastante utilizado devido a sua flexibilidade, que permite trabalhar com problemas de regressão e classificação com desempenho satisfatório, outro ponto é a facilidade para determinar a importância das features e suas contribuições para o modelo. Apesar de muitas vantagens, o método geralmente é bem custoso, visto que está construindo muitas árvores de decisão por trás e isso pode ser problemático em conjuntos de dados maiores.

Tarefa 1 - Processamento ETL

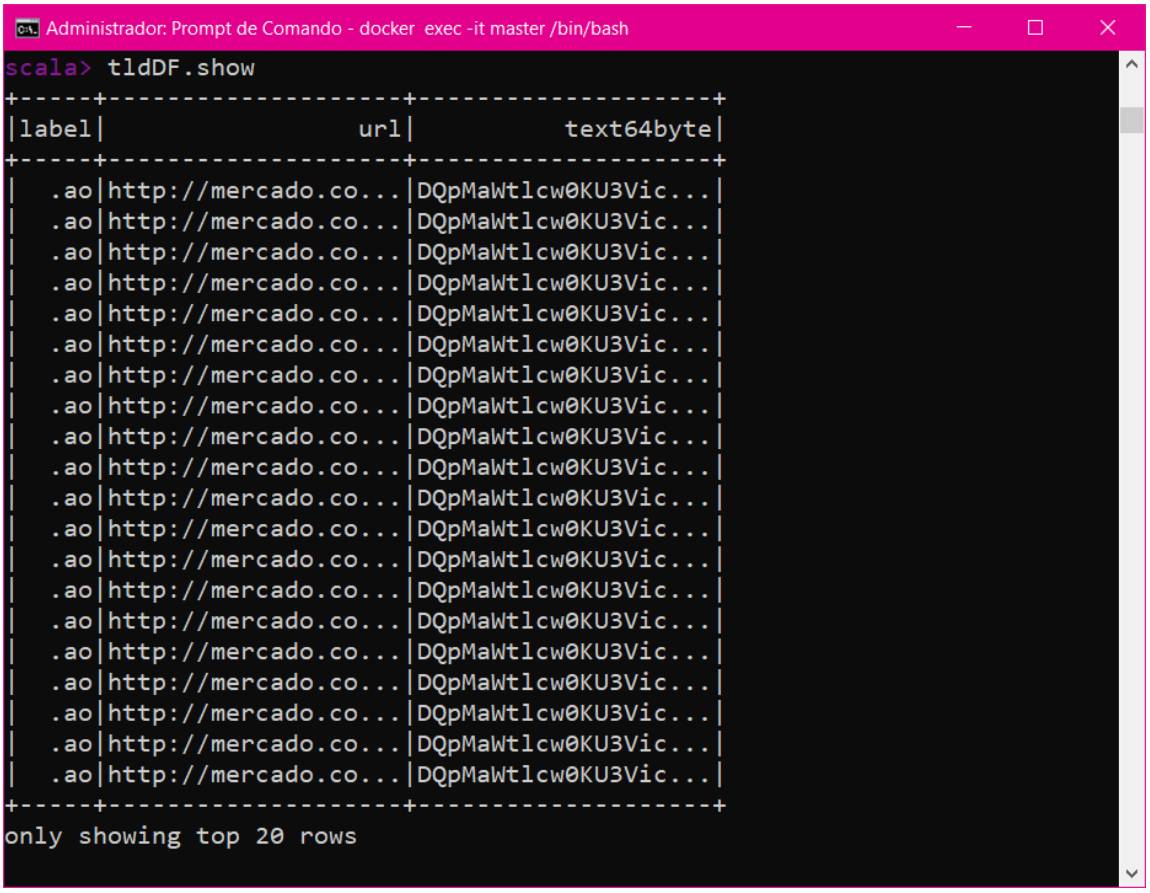
Executando os passos descritos, você terá no HDFS dados no formato:

- ★ **label** - ao, br, pt, mz, mo, gw (no conjunto reduzido há apenas seis países)
- ★ **features** - vetor esparso com a representação do texto de cada página

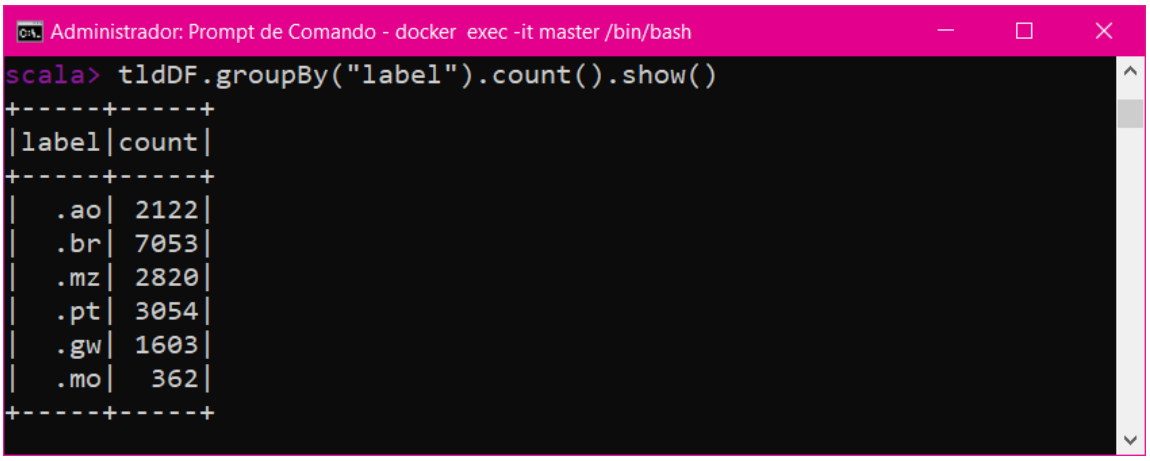
Passo-a-passo da Tarefa 1

- ★ Baixe o arquivo pt7-raw.zip
- ★ Copie a pasta descompactada para user_data/pt7-raw
- ★ Copie os arquivos do PT7 para o HDFS
  - *docker exec -it master /bin/bash*
  - *hadoop fs -mkdir -p /bigdata/*
  - *hadoop fs -put /user_data/pt7-raw hdfs://master:8020/bigdata/*
- ★ Processe o job labels-pt7-raw.scala
  - *spark-shell --master spark://master:7077 -i /user_data/labels-pt7-raw.scala*

Como resultado, será obtido um dataframe conforme imagens a seguir.



```
scala> tldDF.show
+-----+--------------------+--------------------+
|label|                 url|          text64byte|
+-----+--------------------+--------------------+
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
|  .ao|http://mercado.co...|DQpMaWtlcw0KU3Vic...|
+-----+--------------------+--------------------+
only showing top 20 rows
```

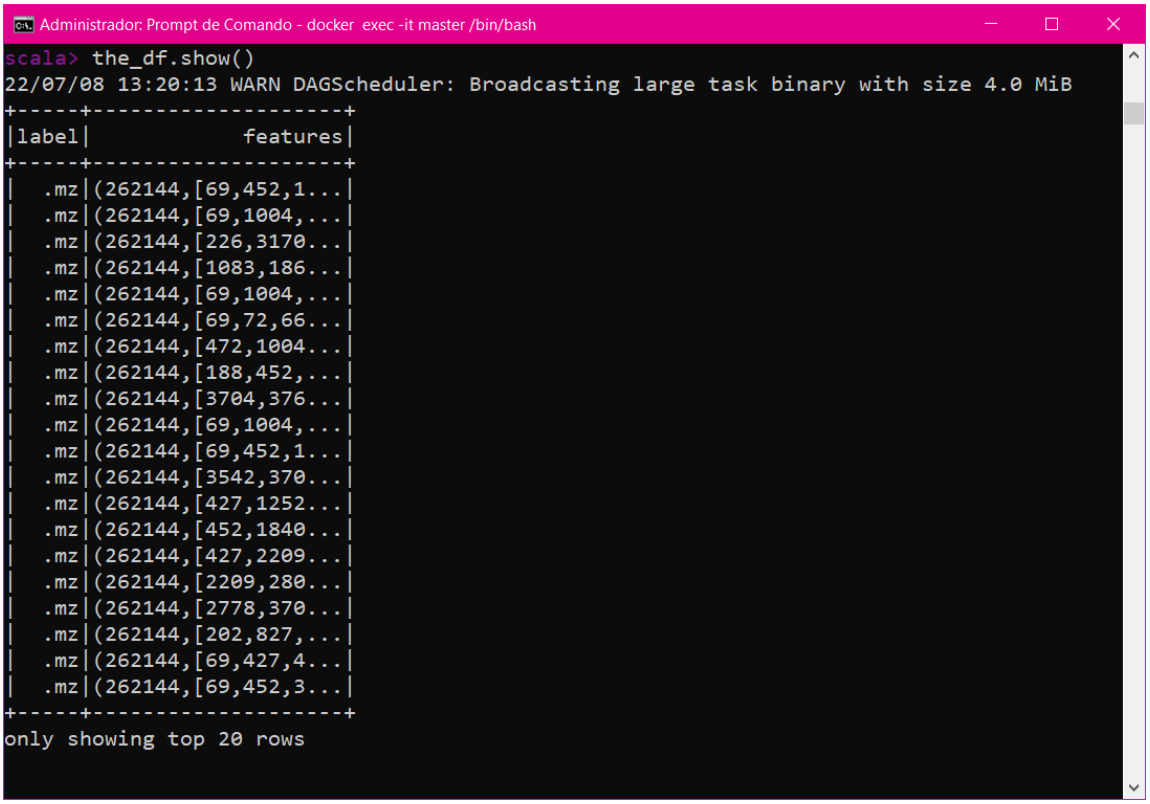


```
scala> tldDF.groupBy("label").count().show()
+-----+-----+
|label|count|
+-----+-----+
|  .ao| 2122|
|  .br| 7053|
|  .mz| 2820|
|  .pt| 3054|
|  .gw| 1603|
|  .mo|  362|
+-----+-----+
```

Processe o job etl-pt7.scala

★ *spark-shell --master spark://master:7077 -i /user_data/etl-pt7.scala*

Como resultado, será obtido um dataframe conforme imagens a seguir. Neste ponto, o dataframe multilabel com os vetores esparsos será gravado no seu HDFS no caminho hdfs://master:8020/bigdata/pt7-hash.parquet



Tarefa 2 - Treinar e Testar Um Modelo Supervisionado

Passo 1:

- ★ **colocar o arquivo “script.py” dentro da pasta “user_data”**

Passo 2:

- ★ **criar uma pasta “projeto” dentro da pasta “user_data”**

Passo 3: Instalar a biblioteca numpy no master e nos 3 workers

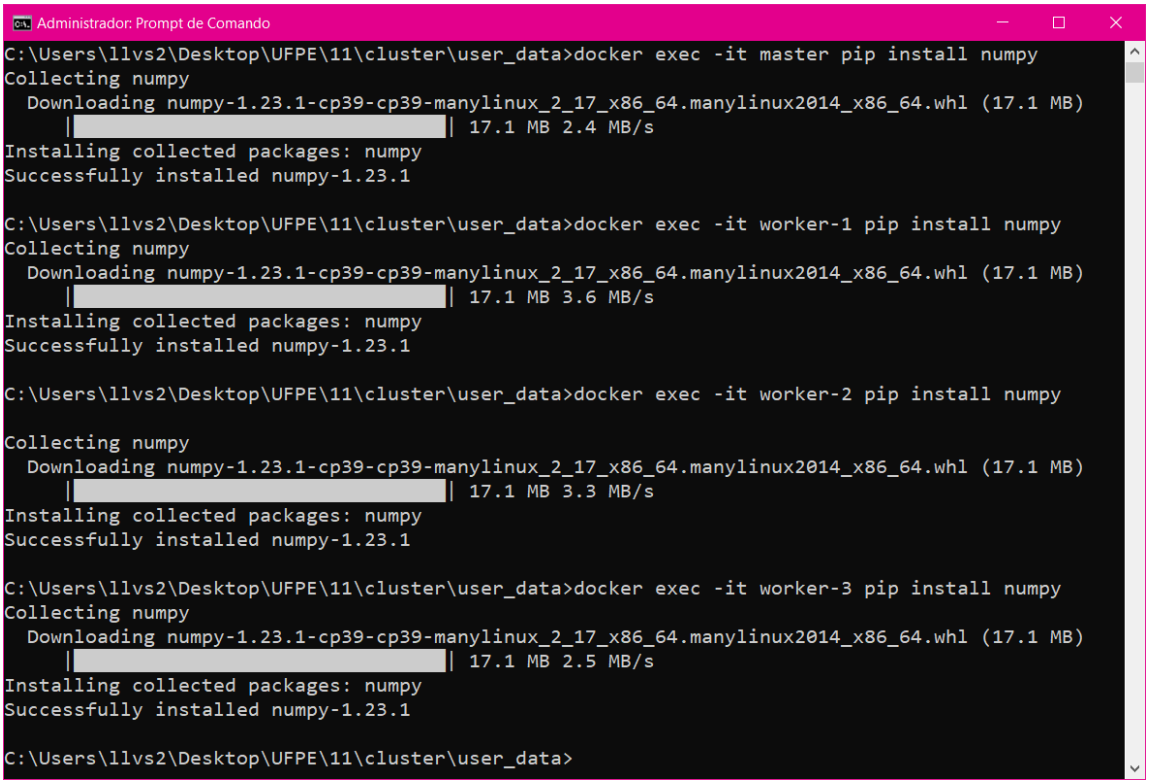
- ★ docker exec -it master pip install numpy
- ★ docker exec -it worker-1 pip install numpy
- ★ docker exec -it worker-2 pip install numpy
- ★ docker exec -it worker-3 pip install numpy

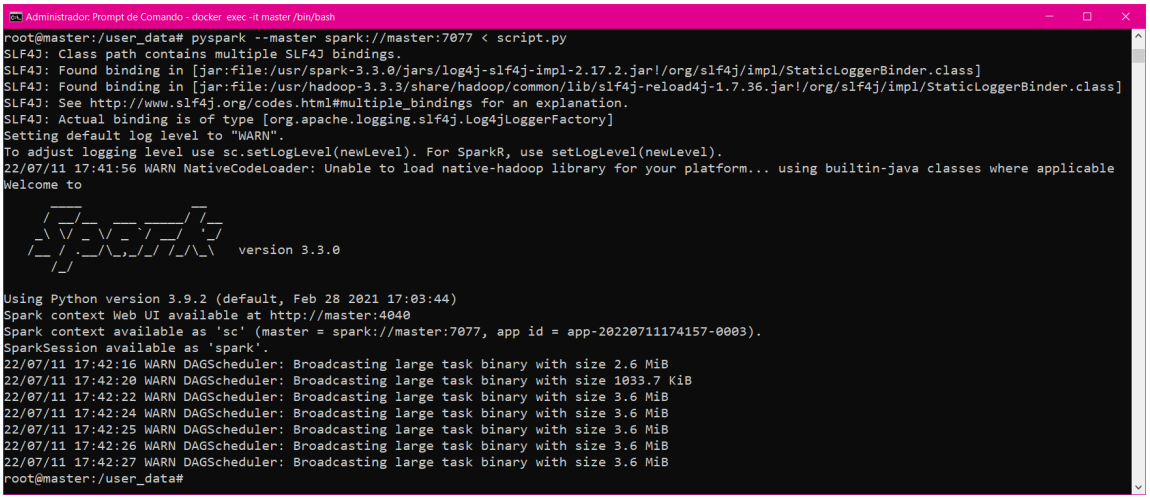
Passo 4: Rodar o arquivo script.py (Salva métricas em /user_data/projeto/metricas.txt)

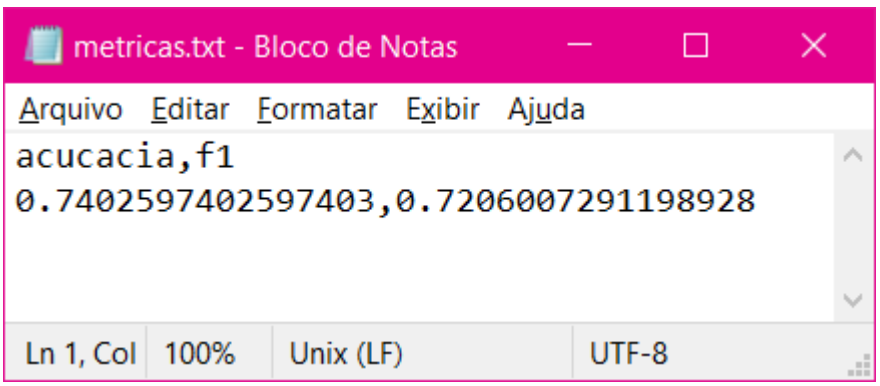
- ★ docker exec -it master /bin/bash
- ★ cd user_data
- ★ pyspark --master spark://master:7077 < script.py

Passo 5 (opcional): Exportar o modelo para o disco local (Salva o modelo em /user_data/projeto/modelo_rf)

- ★ hdfs dfs -copyToLocal hdfs://master:8020/bigdata/modelo_rf /user_data/projeto/


Administrador: Prompt de Comando
C:\Users\llvs2\Desktop\UFPE\11\cluster\user_data>docker exec -it master pip install numpy
Collecting numpy
Downloading numpy-1.23.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
17.1 MB 2.4 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.23.1
C:\Users\llvs2\Desktop\UFPE\11\cluster\user_data>docker exec -it worker-1 pip install numpy
Collecting numpy
Downloading numpy-1.23.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
17.1 MB 3.6 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.23.1
C:\Users\llvs2\Desktop\UFPE\11\cluster\user_data>docker exec -it worker-2 pip install numpy
Collecting numpy
Downloading numpy-1.23.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
17.1 MB 3.3 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.23.1
C:\Users\llvs2\Desktop\UFPE\11\cluster\user_data>docker exec -it worker-3 pip install numpy
Collecting numpy
Downloading numpy-1.23.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
17.1 MB 2.5 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.23.1
C:\Users\llvs2\Desktop\UFPE\11\cluster\user_data>


Administrador: Prompt de Comando - docker exec -it master /bin/bash
root@master:/user_data# pyspark --master spark://master:7077 < script.py
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/spark-3.3.0/jars/log4j-slf4j-impl-2.17.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hadoop-3.3.3/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/07/11 17:41:56 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to
version 3.3.0
Using Python version 3.9.2 (default, Feb 28 2021 17:03:44)
Spark context Web UI available at http://master:4040
Spark context available as 'sc' (master = spark://master:7077, app id = app-20220711174157-0003).
SparkSession available as 'spark'.
22/07/11 17:42:16 WARN DAGScheduler: Broadcasting large task binary with size 2.6 MiB
22/07/11 17:42:20 WARN DAGScheduler: Broadcasting large task binary with size 1033.7 KiB
22/07/11 17:42:22 WARN DAGScheduler: Broadcasting large task binary with size 3.6 MiB
22/07/11 17:42:24 WARN DAGScheduler: Broadcasting large task binary with size 3.6 MiB
22/07/11 17:42:25 WARN DAGScheduler: Broadcasting large task binary with size 3.6 MiB
22/07/11 17:42:26 WARN DAGScheduler: Broadcasting large task binary with size 3.6 MiB
22/07/11 17:42:27 WARN DAGScheduler: Broadcasting large task binary with size 3.6 MiB
root@master:/user_data#


metricas.txt - Bloco de Notas
Arquivo Editar Formatar Exibir Ajuda
acucacia,f1
0.7402597402597403,0.7206007291198928
Ln 1, Col 100% Unix (LF) UTF-8