

Centro de Informática CIn - UFPE

Curso: PD em Larga Escala

Projeto Final

Link do github: <https://github.com/lucascouri2/projeto-pdle>

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 esparsa 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.

```
Administrador: Prompt de Comando - docker exec -it master /bin/bash
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
```

```
Administrador: Prompt de Comando - docker exec -it master /bin/bash
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`

```
Administrador: Prompt de Comando - docker exec -it master /bin/bash
scala> the_df.show()
22/07/08 13:20:13 WARN DAGScheduler: Broadcasting large task binary with size 4.0 MiB
+-----+-----+
|label|          features|
+-----+-----+
|.mz|(262144,[69,452,1...|
|.mz|(262144,[69,1004,...|
|.mz|(262144,[226,3170...|
|.mz|(262144,[1083,186...|
|.mz|(262144,[69,1004,...|
|.mz|(262144,[69,72,66...|
|.mz|(262144,[472,1004...|
|.mz|(262144,[188,452,...|
|.mz|(262144,[3704,376...|
|.mz|(262144,[69,1004,...|
|.mz|(262144,[69,452,1...|
|.mz|(262144,[3542,370...|
|.mz|(262144,[427,1252...|
|.mz|(262144,[452,1840...|
|.mz|(262144,[427,2209...|
|.mz|(262144,[2209,280...|
|.mz|(262144,[2778,370...|
|.mz|(262144,[202,827,...|
|.mz|(262144,[69,427,4...|
|.mz|(262144,[69,452,3...|
+-----+-----+
only showing top 20 rows
```

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/*

Nota: O modelo e os resultados das métricas de avaliação encontram-se no github na pasta “resultados”.

```
Administrador: Prompt de Comando
C:\Users\l1vs2\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\l1vs2\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\l1vs2\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\l1vs2\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\l1vs2\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#
```

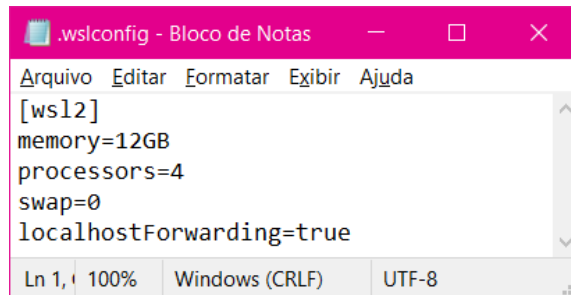
```
metricas.txt - Bloco de Notas
Arquivo Editar Formatar Exibir Ajuda
acucacia,f1,precisao_label,precisao_ponderada,recall_label,recall_ponderado
0.6432993808667865,0.5844985630322328,0.5444074737650371,0.782715732216662,0.9995300751879699,0.6432993808667865

Ln 1, Col 1      100%  Unix (LF)  UTF-8
```

Extra - Configuração do “.wslconfig” no Docker do Windows

Passo 1:

- ★ Abrir o bloco de notas e digitar o código abaixo:



```
[wsl2]
memory=12GB
processors=4
swap=0
localhostForwarding=true
```

- ★ A quantidade de memória e processadores alocados podem ser variados de acordo com a configuração da máquina

Passo 2:

- ★ Salvar o arquivo com o nome de “.wslconfig”

Passo 3:

- ★ Abrir o Explorador de arquivos e digitar “%USERPROFILE” na barra do diretório
 - Outra opção é ir manualmente no diretório “C:\Users\nome_do_usuario”

Passo 4:

- ★ Colocar o arquivo “.wslconfig” na pasta que abriu no passo anterior

Passo 5:

- ★ Fechar o Docker (encerrar o processo completamente)

Passo 6:

- ★ Reiniciar o computador

Passo 7:

- ★ Iniciar o Docker novamente

Passo 8:

- ★ Seguir os passos da Tarefa 1 e Tarefa 2 normalmente

