Senac

Fecomércio

Sesc

Big Data

Prof. Marco Mialaret

Maio

2024

Big Data

Onde me encontrar:

https://www.linkedin.com/in/marco-mialaret-junior/

e

https://github.com/MatmJr

Aula Passada: PySpark

Big Data

PySpark é uma API em Python para executar o Spark e foi lançado para oferecer suporte à colaboração entre Apache Spark e Python. O PySpark também oferece suporte à interface do Apache Spark com conjuntos de dados distribuídos resilientes (RDDs) na linguagem de programação Python.

Big Data

O PySpark também pode ser utilizado no Colab. Vamos começar a explorar esse novo mundo juntos:

Crie um notebook novo

Big Data

Use a primeira célula para instalar a biblioteca do PySpark

```
!pip install pyspark requests
```

Big Data

Vamos iniciar uma sessão Spark

```
from pyspark.sql import SparkSession

spark = (SparkSession.builder
        .appName("PySpark App")
        .getOrCreate()
```

Big Data

Carregando o primeiro dataset

```
import requests

response=requests.get(
"https://ddragon.leagueoflegends.com/cdn/12.17.1/data/pt_BR/champion.json")

champions=response.json().get("data")
champions.keys()
```

Big Data

Limpeza dos dados

Antes de começarmos de fato com a análise, é necessário fazermos uma limpeza prévia nos dados. Vamos pegar apenas os que nos interessa, e remover os dicionários dentro de dicionários, deixando um único dicionário para cada campeão com os dados necessários.

Big Data

```
champions=[{'name': value['name'], 'title': value['title'], **value['info'],
**value['stats']} for key, value in champions.items()]
champions[2]
```

Big Data

Criando o DataFrame

Agora que os dados dos campeões estão limpos, podemos criar nosso DataFrame usando o Spark.

No entanto, o Spark é bastante específico quanto ao tipo de objeto que aceitamos para criar um DataFrame. Atualmente, nosso objeto "champions" é uma lista de dicionários, que não é compatível com o Spark.

Big Data

Mas existe uma solução! A biblioteca Pandas é muito mais flexível quando se trata de criar um DataFrame. Podemos criar um DataFrame do Pandas a partir do nosso objeto "champions" atual e, em seguida, usar esse DataFrame do Pandas para criar um DataFrame do Spark.

Big Data

```
import pandas as pd

df = spark.createDataFrame(pd.DataFrame(champions))

df.select("name", "title").show(5, False)
```

Big Data

Concatenação de colunas

Para facilitar a visualização dos dados, vamos criar uma nova coluna chamada `full_name` que concatena as colunas `name` e `title`. Utilizaremos o método `withColumn` para isso. Esse método recebe dois parâmetros: o nome da nova coluna e os dados para populá-la. Usaremos a função `concat` para juntar as colunas `name` e `title`, e a função `lit` para adicionar uma vírgula e um espaço entre elas.

Big Data

```
from pyspark.sql import functions as F

df = df.withColumn("full_name", F.concat(df.name, F.lit(", "), df.title))
df.select("full_name").show(5, False)
```

Big Data

Quem são os campeões mais poderosos de League of Legends?

```
base_columns = ["attackdamage", "armor", "hp", "mp"]

(df.orderBy(*base_columns, ascending=False)
 .select("full_name", *base_columns)
 .show(5, False)
 )
```

Big Data

Estatísticas dos níveis de poderes

```
(df2.agg({
    "attackdamage": "mean",
    "hp": "max",
    "mp": "max",
    "armor": "min"
})
    .show()
)
```

Dúvidas?

Marco Mialaret, MSc

Telefone:
81 98160 7018

E-mail:
marcomialaret@gmail.com