Instituto Politécnico de Setúbal

Escola Superior de Tecnologia do Barreiro

Projeto de Big Data

Licenciatura em Bioinformática

Body Performance

Janeiro de 2023

Francisco Amaral (201900202)
Tiago Artilheiro (202000209)

Índice

1 Introdução **1**
1.1 Objetivos do Projeto: 1
1.2 Introdução ao Dataset: 1

2 Análise e tratamento dos dados **2**
2.1 Quantidade e tipo de dados 2
2.2 Retirar colunas não necessárias 2
2.3 Analisar variável idade(Age) 3
2.4 Verificar e alterar o tipo das variáveis 4
2.5 Verificar e corrigir existencia de valores nulos 5

3 Algoritmos de machine learning **6**
3.1 Processamento de dados 6
3.2 Aplicação dos algoritmos 7

4 Conclusões **9**

1 Introdução

1.1 Objetivos do Projeto:

Este projeto de BigData tem como objetivo usar algoritmos de Machine Learning, recorrendo ao uso do PySpark, pois o mesmo ajuda o processamento de grandes quantidades de dados e é incorpurado com ferramentas para prever resultados importantes em relação a um conjunto de dados. O Spark é uma ferramenta poderosa que permite o processamento distribuído de dados. Iremos usar o Spark para ler, limpar e preparar os dados, avaliar e ajustar modelos e, finalmente, fazer previsões precisas e relevantes. Este projeto é uma excelente oportunidade para aplicar e praticar habilidades de Machine Learning em um ambiente de grande escala e de alta performance.

1.2 Introdução ao Dataset:

O nosso dataset inclui dados de percentagem de gordura, dados de pressão sanguínea dialóstica e sistólica, alguns exercícios tais como broad jump, abdominais e amplitude no sit and bend; Temos como objetivo utilizar técnicas Machine Learning para identificar padrões e tendências nos dados e talvez prever a queda de performance com idade;

2 Análise e tratamento dos dados

2.1 Quantidade e tipo de dados

Começamos por analisar o nosso dataset e observar a quantidade e o tipo de dados do dataset

Figura 1:

```
print("Numero de Linhas:", df.count())
print("Numero de Colunas:", len(df.columns))
print("Esquema do DataFrame: ")
df.printSchema()
```

```
Numero de Linhas: 13393
Numero de Colunas: 12
Esquema do DataFrame:
root
 |-- age: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- height_cm: string (nullable = true)
 |-- weight_kg: string (nullable = true)
 |-- body fat_%: string (nullable = true)
 |-- diastolic: string (nullable = true)
 |-- systolic: string (nullable = true)
 |-- gripForce: string (nullable = true)
 |-- sit and bend forward_cm: string (nullable = true)
 |-- sit-ups counts: string (nullable = true)
 |-- broad jump_cm: string (nullable = true)
 |-- class: string (nullable = true)
```

Observação da quantidade e tipo de dados.

2.2 Retirar colunas não necessárias

Decidimos retirar a variável class pois era uma variável que não nos seria útil neste projeto

Figura 2:

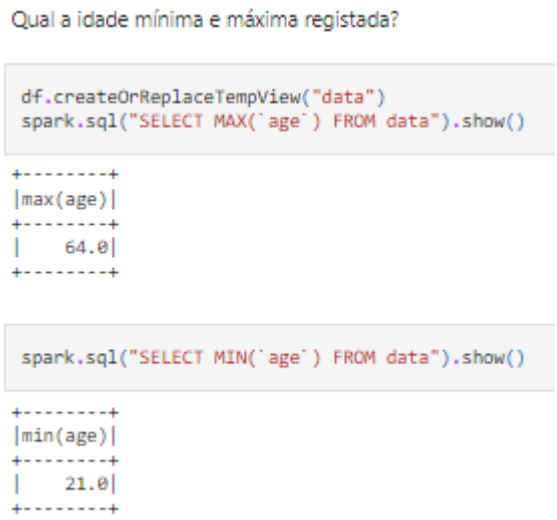
```
#retirar a variável "class" que não será útil para a nossa análise
df = df.drop("class")
```

Retirar variavél não necessária.

2.3 Analisar variável idade(Age)

Em seguida decidimos verificar e avaliar os máximos e mínimos da coluna "Age"para verificarmos que não havia nenhum valor fora do normal.

Figura 3:



Analise variável "Age".

Após verificarmos que não havia nenhum valor fora do padrão seguimos para o tratamento do tipo de variáveis

2.4 Verificar e alterar o tipo das variáveis

Agora vamos verificar o tipo de variáveis existentes;

Figura 4:

```
df.dtypes
```

```
[('age', 'string'),
 ('gender', 'string'),
 ('height_cm', 'string'),
 ('weight_kg', 'string'),
 ('body fat_%', 'string'),
 ('diastolic', 'string'),
 ('systolic', 'string'),
 ('gripForce', 'string'),
 ('sit and bend forward_cm', 'string'),
 ('sit-ups counts', 'string'),
 ('broad jump_cm', 'string')]
```

Verificar tipo das variáveis.

Depois disto percebemos que as variáveis estavam todas como 'string' e alterámos as necessárias

Figura 5:

```
from pyspark.sql.types import *
from pyspark.sql.types import IntegerType, DoubleType
df = df \
    .withColumn("age", df["age"].cast(IntegerType())) \
    .withColumn("gender", df["gender"].cast(StringType())) \
    .withColumn("height_cm", df["height_cm"].cast(DoubleType())) \
    .withColumn("weight_kg", df["weight_kg"].cast(DoubleType())) \
    .withColumn("body fat_%", df["body fat_%"].cast(DoubleType())) \
    .withColumn("diastolic", df["diastolic"].cast(IntegerType())) \
    .withColumn("systolic", df["systolic"].cast(IntegerType())) \
    .withColumn("gripForce", df["gripForce"].cast(DoubleType())) \
    .withColumn("sit and bend forward_cm", df["sit and bend forward_cm"].cast(DoubleType())) \
    .withColumn("sit-ups counts", df["sit-ups counts"].cast(IntegerType())) \
    .withColumn("broad jump_cm", df["broad jump_cm"].cast(IntegerType()))
```

```
df.dtypes
```

```
[('age', 'int'),
 ('gender', 'string'),
 ('height_cm', 'double'),
 ('weight_kg', 'double'),
 ('body fat_%', 'double'),
 ('diastolic', 'int'),
 ('systolic', 'int'),
 ('gripForce', 'double'),
 ('sit and bend forward_cm', 'double'),
 ('sit-ups counts', 'int'),
 ('broad jump_cm', 'int')]
```

Alterar tipo das variáveis.

2.5 Verificar e corrigir existencia de valores nulos

Verificamos também a existência de valores nulos

Figura 6:

```
missing_values = []
for col in df.columns:
    missing_values.append((col, df.filter(df[col].isNull()).count()))
for col, val in missing_values:
    if val == 0:
        print("{} : Nenhum valor nulo".format(col))
    else:
        print("{} : {} missing values".format(col, val))
```

```
Age : Nenhum valor nulo
Gender : Nenhum valor nulo
Height : Nenhum valor nulo
Weight : Nenhum valor nulo
BodyFat : 65 missing values
Dialostic : Nenhum valor nulo
Systolic : Nenhum valor nulo
GripForce : Nenhum valor nulo
SitBend : Nenhum valor nulo
SitUps : Nenhum valor nulo
BroadJump : Nenhum valor nulo
```

Verificar nulos.

Em seguida retiramos substituímos os valores nulos.

Figura 7:

```
df = df.fillna(0)
```

```
missing_values = []
for col in df.columns:
    missing_values.append((col, df.filter(df[col].isNull()).count()))
for col, val in missing_values:
    if val == 0:
        print("{} : Nenhum valor nulo".format(col))
    else:
        print("{} : {} missing values".format(col, val))
```

```
Age : Nenhum valor nulo
Gender : Nenhum valor nulo
Height : Nenhum valor nulo
Weight : Nenhum valor nulo
BodyFat : Nenhum valor nulo
Dialostic : Nenhum valor nulo
Systolic : Nenhum valor nulo
GripForce : Nenhum valor nulo
SitBend : Nenhum valor nulo
SitUps : Nenhum valor nulo
BroadJump : Nenhum valor nulo
```

Corrigir nulos.

3 Algoritmos de machine learning

3.1 Processamento de dados

Na aplicação de algoritmos de machine learning começamos por processar os nossos dados para que fosse possível aplicar o código.

Figura 8:

Data Pre-processing

```
from pyspark.ml.feature import StandardScaler, VectorAssembler
assembler = VectorAssembler(inputCols=['Height','Weight','BodyFat','GripForce','SitBend','SitUps','BroadJump'], outputCol='features')
assembler = assembler.transform(df)
assembler.select('features','Age')
```

```
DataFrame[features: vector, Age: int]
```

```
scale = StandardScaler(inputCol='features', outputCol='standard')
data_scale=scale.fit(assembler)
data_scale_output=data_scale.transform(assembler)
data_scale_output.select('standard').show(10, 0)
```

```
+------------------------------------------------------------------------------------------------------------------------------------------------+
|standard                                                                                                                                        |
+------------------------------------------------------------------------------------------------------------------------------------------------+
|[20.4471977775314,6.2964101123975835,2.5791810459503974,5.167124949435115,2.1757955258188457,4.202606929642596,5.442963160093119]  |
|[19.5808916615942,4.669586446993423,1.901086498658274,3.4259261959824805,1.9274710364590864,3.7123027878509602,5.743956514568315]  |
|[21.313503893468596,6.527378904399409,2.433875071530657,4.216524548901514,1.418997082055769,3.4321289925414535,4.539983096667533]  |
|[20.70827633301932,5.949956924394845,2.2280249411026904,3.8965204536723816,1.7973963039373073,3.7123027878509602,5.493128719172319]|
|[20.625205883545892,5.665430151638975,2.070610135481305,4.0941700419021405,3.2045684103092786,3.1519551972319473,5.442963160093119]|
|[19.628360489864733,4.636112709022145,2.6639428643619127,2.2400286666039295,2.483244893597596,1.8911731183391685,3.837665269558743]|
|[19.521555626256035,5.330692771926184,3.899043646929709,2.136497929912151,0.0945998054703846,1.2607820788927788,3.662085812781546] |
|[20.755745161289852,6.460431428456851,4.468158713407027,4.320055285593293,1.4544720091071632,2.9418248507498173,5.869370412266313] |
|[19.794501388811593,5.648693282653335,3.342037411654036,3.802401602134401,2.1994454771864422,2.3814772601308047,3.712251371860745] |
|[21.95433307512077,7.079695580925512,1.7436716930368883,5.449481504049055,1.430822057739567,3.852389685505713,5.342632041934721]  |
+------------------------------------------------------------------------------------------------------------------------------------------------+
only showing top 10 rows
```

Processamento de dados.

Figura 9:

```
modelo_df= assembler.select(['features', 'Age'])
modelo_df = modelo_df.withColumnRenamed("Age", "label")
modelo_df.printSchema()
```

```
root
 |-- features: vector (nullable = true)
 |-- label: integer (nullable = true)
```

Processamento de dados.

3.2 Aplicação dos algoritmos

Separamos inicialmente o dataset em dois 70% para treino e 30% para teste.

Figura 10:

```
from pyspark.ml.classification import LogisticRegression
train, test = modelo_df.randomSplit([0.7, 0.3])
#divisão do dataset em dois, 70% para treino e 30% para teste
```

Separação dataset.

Logo em seguida fomos testar a accuracy da regressão e verificamos que o nosso modelo não é fiável para prever se a idade afeta a performance nos exercícios do nosso data set, visto que pode existir indivíduos que tenham uma idade avançada e com boa performance e adolescentes com pouca aptidão fisica.

Figura 11:

```
Lr = LogisticRegression(featuresCol = 'features', labelCol = 'label').fit(train)
```

```
Lr_summary = Lr.summary
```

```
Lr_summary.accuracy
```

```
0.09433361629881154
```

Verificar accuracy.

O valor da accuracy foi de 0.0943.

Tentamos ainda testar outros metodos contudo obtivemos em todos uma accuracy extremamente baixa.

Figura 12:

```
from pyspark.ml.classification import NaiveBayes
```

```
Naiveby = NaiveBayes(modelType="gaussian")
nb = Naiveby.fit(train)
```

```
p_df = nb.transform(test)
p_df.show(5, True)
```

```
+--------------------+-----+--------------------+--------------------+----------+
|            features|label|       rawPrediction|         probability|prediction|
+--------------------+-----+--------------------+--------------------+----------+
|[140.5,49.6,32.1,...|   64|[-33.497763398171...|[2.51720189859315...|      35.0|
|[141.0,46.1,25.5,...|   60|[-34.274401356927...|[9.21474238658230...|      35.0|
|[144.7,43.1,28.3,...|   61|[-36.303054100024...|[2.27239099544285...|      41.0|
|[144.9,50.8,33.9,...|   62|[-42.321973137002...|[7.71655120603505...|      43.0|
|[145.5,47.4,29.4,...|   61|[-30.950194685838...|[3.37309023969197...|      41.0|
+--------------------+-----+--------------------+--------------------+----------+
only showing top 5 rows
```

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
e2 = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
print("Accuracy = ",e2.evaluate(p_df))
```

```
Accuracy =  0.003527336860670194
```

Outros metodos.

Como é possível observar nas figuras estes dados não são apropriados para a aplicação de algoritmos de machine learning.

4 Conclusões

Este projeto tinha como principal objetivo a aplicação de algoritmos de Machine Learning de modo a prever se a idade tem influencia na performance, utilizando o PySpark como principal ferramenta para a aplicação dos algoritmos e análise do conjunto de dados.
Com base no que foi apresentado no relatório, tivemos alguns problemas com a implementação dos algoritmos, nomeadamente, o facto de a Logistic Regression e o Naive Bayes nos apresentarem uma accuracy extremamente baixa o que nos incapacitou de realizar algumas funções.
Apesar deste contratempo, conseguimos alcançar o principal objetivo do projeto que era a aplicação dos algoritmos de machine learning e a utilização da ferramenta pyspark.
Concluímos então que o nosso dataset não era propício a utilização destes métodos de Machine Learning.