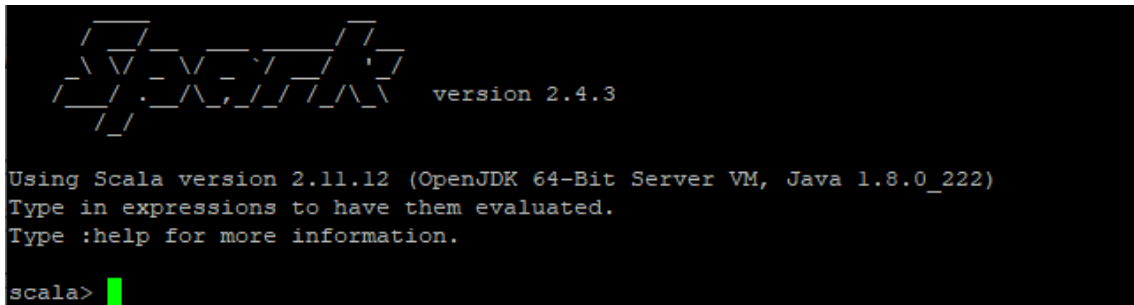


INTRODUÇÃO AO SPARK

//Comandos básicos

// Chamando o spark

```
spark-shell
```

A screenshot of a terminal window showing the Spark Shell interface. At the top, the word 'Spark' is displayed in a large, stylized, orange-outlined font, followed by 'version 2.4.3' in a smaller, white font. Below this, the text 'Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_222)' is shown in white. Further down, instructions are provided: 'Type in expressions to have them evaluated.' and 'Type :help for more information.' At the bottom, the prompt 'scala>' is visible with a green cursor bar to its right.

```
Spark version 2.4.3

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_222)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

// O Spark permite ler comandos a partir de scripts

//Lendo um script externo no scala

```
:load /home/bitnami/input.scala
```

// Trabalhando com dataframes

// Importando as bibliotecas necessárias

```
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.functions._
import java.text.SimpleDateFormat
import java.util.Calendar
import org.apache.spark.sql.Row
import org.apache.spark.sql.types.{StructType, StructField, StringType, IntegerType, FloatType, DoubleType}
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.regression.{LinearRegression, RandomForestRegressor}
import org.apache.spark.ml.evaluation.RegressionEvaluator

// Marcando a data e hora do início da operação
val starttime = Calendar.getInstance().getTime()
```

// Criando os dados de treinamento a partir de uma sequência de tuplas, e transformando-os em DATAFRAME

```
val dataframeTreinamento = spark.createDataFrame(Seq(
  (10L, 3L, 1L),
  (20L, 6L, 2L),
  (30L, 9L, 3L),
  (50L, 15L, 5L),
  (60L, 18L, 6L),
  (100L, 30L, 10L),
  (120L, 36L, 12L),
  (130L, 39L, 13L),
  (160L, 48L, 16L)
)).toDF("Coluna_1", "Coluna_2", "label")

//Mostrando o DATAFRAME de treinamento
dataframeTreinamento.show()
```

```
+-----+-----+-----+
|Coluna_1|Coluna_2|label|
+-----+-----+-----+
|      10|       3|    1|
```

	20	6	2
	30	9	3
	50	15	5
	60	18	6
	100	30	10
	120	36	12
	130	39	13
	160	48	16
	-----	-----	-----

// Criando os dados de teste a partir de uma sequencia de tuplas, e transformando-os em DATAFRAME

```
val dataframeTeste = spark.createDataFrame(Seq(
  (31L, 6L, 3L),
  (40L, 13L, 4L),
  (82L, 24L, 8L),
  (70L, 23L, 7L),
  (149L, 45L, 15L),
  (140L, 43L, 14L),
  (91L, 27L, 9L)
)).toDF("Coluna_1", "Coluna_2", "label")
```

```
//Mostrando o DATAFRAME de teste
dataframeTeste.show()
```

	Coluna_1	Coluna_2	label
	31	6	3
	40	13	4
	82	24	8
	70	23	7
	149	45	15
	140	43	14
	91	27	9
	-----	-----	-----

// Usando comandos SQL com Spark.SQL

//Criando Tabela 1 temporaria para visualizar o dataframeTreinamento usando Spark.SQL

```
dataframeTreinamento.createOrReplaceTempView("minhaTabela1")
```

//Usando comandos SQL para visualizações da tabela 1

```
spark.sql("SELECT * FROM minhaTabela1").show
```

```
+-----+-----+-----+
|Coluna_1|Coluna_2|label|
+-----+-----+-----+
|      10|       3|    1|
|      20|       6|    2|
|      ...|      ...|   12|
|     130|      39|   13|
|     160|      48|   16|
+-----+-----+-----+
```

```
spark.sql("SELECT Coluna_1 FROM minhaTabela1").show
```

```
+-----+
|Coluna_1|
+-----+
|      10|
|      20|
|      ...|
|     130|
|     160|
+-----+
```

```
spark.sql("SELECT Coluna_1, Coluna_2 FROM minhaTabela1 WHERE Coluna_1 > 100").show
```

```
+-----+-----+
|Coluna_1|Coluna_2|
+-----+-----+
|     120|      36|
|     130|      39|
|     160|      48|
+-----+-----+
```

// Usando o vector assembler

// VectorAssembler() para combinar 2 ou mais colunas em um único vetor de features.

```
//Instanciando um assembler para criar um único vetor de features
val assembler = new VectorAssembler().setInputCols(Array( "Coluna_1", "Coluna_2")).setOutputCol("features")

//Aplicando a transformacao com o assembler nos dados de treinamento
val dataframeTreinamento2 = assembler.transform(dataframeTreinamento)

//Mostrando os dados e esquema
dataframeTreinamento2.show()
+-----+-----+-----+-----+
|Coluna_1|Coluna_2|label|   features|
+-----+-----+-----+-----+
|      10|       3|    1| [10.0,3.0]|
|      20|       6|    2| [20.0,6.0]|
|      ...|      ...|   ...|      ...|
|     130|      39|   13|[130.0,39.0]|
|     160|      48|   16|[160.0,48.0]|
+-----+-----+-----+-----+

dataframeTreinamento2.printSchema()

// O resultado deve ser como abaixo
root
 |-- Coluna_1: long (nullable = false)
 |-- Coluna_2: long (nullable = false)
 |-- label: long (nullable = false)
 |-- features: vector (nullable = true)

//aplicando a transformacao com o assembler nos dados de teste
val dataframeTeste2 = assembler.transform(dataframeTeste)
```

// Aplicando regressão linear nos dados

// Criando um modelo de regressão linear

```
// Instanciando o modelo LR
val modelo = new LinearRegression().setLabelCol("label").setFeature
sCol("features")

//Treinando o modelo com os dados de treinamento
val modeloTreinado = modelo.fit(dataframeTreinamento2)

// Fazendo Predicoes com os dados de teste ou desconhecidas
val fazendoPredicoes = modeloTreinado.transform(dataframeTeste2)

// Selecionando as colunas para mostrar os resultados
fazendoPredicoes.select("prediction", "label", "features").show()
```

```
// Criando avaliadores para as metricas do modelo
val avaliador = new RegressionEvaluator().setLabelCol("label").setP
redictionCol("prediction").setMetricName("rmse")
val Test_rmse = avaliador.evaluate(fazendoPredicoes)

val avaliador2 = new RegressionEvaluator().setLabelCol("label").set
PredictionCol("prediction").setMetricName("r2")
val Test_R2 = avaliador2.evaluate(fazendoPredicoes)

//Imprimindo resultados na tela
println("A RAIZ DO ERRO MÉDIO QUADRÁTICO (RMSE) nos dados de teste
é = " + Test_rmse)
println("O R2 nos dados de teste é: " + Test_R2);
```

```
A RAIZ DO ERRO MÉDIO QUADRÁTICO (RMSE) nos dados de teste é = 0.48
79500364742662
O R2 nos dados de teste é: 0.9867424242424243
```

//Calculo do tempo total da operacao de regressao

```
val endtime = Calendar.getInstance().getTime()
val elapsedtime = ((endtime.getTime() - starttime.getTime())/1000)
.toString;

//Imprimindo tempo total na tela
println("Tempo total da operação = " + elapsedtime + " segundos.");
```

// Obtendo alguns parâmetros do modelo

// O intercepto é o valor do eixo Y quando X=0.

```
modeloTreinado.intercept
```

```
res11: Double = 3.629261699187618E-15
```

// Demais parâmetros do modelo, coeficientes 'a' e 'b'

```
modeloTreinado.coefficients
```

```
res12: org.apache.spark.ml.linalg.Vector = [-6.175859152393819E-17,  
0.3333333333333334]
```


// Exemplo de Machine Learning nos dados do Prouni

// Importando as bibliotecas necessárias

```
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.functions._
import java.text.SimpleDateFormat
import org.apache.spark.sql.Row
import org.apache.spark.sql.types.{StructType, StructField, StringType, IntegerType, FloatType, DoubleType}
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.regression.{LinearRegression, RandomForestRegressor, GBRegressor}
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.attribute.Attribute
import org.apache.spark.ml.feature.{IndexToString, StringIndexer}
```

// Esquema da tabela do nosso arquivo

```
val customSchema = StructType(Array(
    StructField("uf_busca", StringType, true),
    StructField("cidade_busca", StringType, true),
    StructField("universidade_nome", StringType, true),
    StructField("campus_nome", StringType, true),
    StructField("nome", StringType, true),
    StructField("grau", StringType, true),
    StructField("turno", StringType, true),
    StructField("mensalidade", DoubleType, true),
    StructField("bolsa_integral_cotas", IntegerType, true),
    StructField("bolsa_integral_ampla", IntegerType, true),
    StructField("bolsa_parcial_cotas", IntegerType, true),
    StructField("bolsa_parcial_ampla", IntegerType, true),
    StructField("nota_integral_ampla", DoubleType, true),
    StructField("nota_integral_cotas", DoubleType, true),
    StructField("nota_parcial_ampla", DoubleType, true),
    StructField("nota_parcial_cotas", DoubleType, true)))
```

// Lendo o arquivo para o Dataframe

```
val df = spark.read.format("csv").option("delimiter",",").option("quote","").option("header", "true").schema(customSchema).load("curso-s-prouni.csv")
```

// Verificando o conteúdo lido do arquivo

```
scala> df.show(5)
```

uf_busca	cidade_busca	universidade_nome	campus_nome	nome	grau	turno	mensalidade	bolsa_inte
AC	Acrelândia	Universidade Paul...	ACRELANDIA - Centro	Administração	Bacharelado	Curso a Distância	289.0	
AC	Cruzeiro do Sul	Centro Universitã...	PAP CRUZEIRO DO S...	Administração	Bacharelado	Curso a Distância	298.0	
AC	Cruzeiro do Sul	Faculdade Educaci...	AC - CRUZEIRO DO ...	Administração	Bacharelado	Curso a Distância	325.0	
AC	Cruzeiro do Sul	Universidade Paul...	CRUZEIRO DO SUL - ...	Administração	Bacharelado	Curso a Distância	319.0	
AC	Rio Branco	Centro Universitã...	PAP RIO BRANCO - ...	Administração	Bacharelado	Curso a Distância	298.0	

// Verificando o conteúdo lido do arquivo

```
df.printSchema()
```

```
root
 |-- uf_busca: string (nullable = true)
 |-- cidade_busca: string (nullable = true)
 |-- universidade_nome: string (nullable = true)
 |-- campus_nome: string (nullable = true)
 |-- nome: string (nullable = true)
 |-- grau: string (nullable = true)
 |-- turno: string (nullable = true)
 |-- mensalidade: double (nullable = true)
 |-- bolsa_integral_cotas: integer (nullable = true)
 |-- bolsa_integral_ampla: integer (nullable = true)
 |-- bolsa_parcial_cotas: integer (nullable = true)
 |-- bolsa_parcial_ampla: integer (nullable = true)
 |-- nota_integral_ampla: double (nullable = true)
 |-- nota_integral_cotas: double (nullable = true)
 |-- nota_parcial_ampla: double (nullable = true)
 |-- nota_parcial_cotas: double (nullable = true)
```

// Realizando consultas nos dados

```
scala> df.select("uf_busca").distinct().count()
res2: Long = 27
```

```
scala> df.select("cidade_busca").distinct().count()
res3: Long = 1158
```

```
scala> df.select("universidade_nome").distinct().count()
res4: Long = 1298
```

```
scala> df.select("campus_nome").distinct().count()
res5: Long = 4770
```

```
scala> df.select("nome").distinct().count()
res6: Long = 338
```

```
scala> df.select("grau").distinct().show()
```

```
+-----+
```

```
|          grau|
```

```
+-----+
```

```
|          Direito|
```

```
|Gestão de Recurso...|
```

```
|Engenharia de Pro...|
```

```
|Design de Interiores|
```

```
|    Engenharia Civil|
```

```
|  Gestão da Qualidade|
```

```
|Estética e Cosmética|
```

```
|  Pesquisa e Gestã...|
```

```
|Letras - Português...|
```

```
|          Agronegócio|
```

```
|Arquitetura e Urb...|
```

```
|Manutenção de Aer...|
```

```
|          Enfermagem|
```

```
|Letras - Português...|
```

```
|          Física|
```

```
|          Teologia|
```

```
|          Banco de Dados|
```

```
|Análise e Desenvol...|
```

```
| 333 - CENTRO - P...|
```

```
|          Agronomia|
```

```
+-----+
```

```
only showing top 20 rows
```

```
scala> df.select("turno").distinct().show()
```

```

+-----+
|          turno|
+-----+
|FACULDADE SÃO FID...|
|Engenharia de Pro...|
|          Medicina|
|Letras - Português...|
|          Odontologia|
|          Enfermagem|
|          Matutino|
|          Matemática|
|          KM 102  (101|
|    Curso a Distância|
|          Vespertino|
|  60 (NOVA SEDE) -...|
|UNIDADE SEDE - Mo...|
|  Ciências Biológicas|
|          Psicologia|
|    Administração|
|          Pedagogia|
|    Bacharelado|
|    Licenciatura|
|  1171 - CENTRO - ...|
+-----+

```

only showing top 20 rows

// Indexação

```
// Instanciando o Indexer
```

```
val indexer = new StringIndexer().setInputCol("uf_busca").setOutputCol("uf_buscaIndex").setHandleInvalid("skip").fit(df)
```

```
// Aplicando a transformação no Dataframe
```

```
val indexed = indexer.transform(df)
```

```
// Verificando o Dataframe com a coluna indexada
```

```
indexed.show()
```

[illegible]

```
// Verificando apenas as duas colunas envolvidas na transação
```

```
indexed.select("uf_busca", "uf_buscaIndex").distinct().show()
```

uf_busca	uf_buscaIndex
PA	8.0
AL	22.0
MG	1.0
DF	10.0
TO	20.0
MA	15.0
RS	3.0
RJ	6.0
BA	4.0
AC	25.0
PI	19.0
RO	17.0
PR	2.0
MT	11.0
RR	26.0
MS	14.0
SE	23.0
RN	21.0
SC	5.0
CE	13.0

```
only showing top 20 rows
```

// Repetir o procedimento nas outras colunas textuais

```
val indexer2 = new StringIndexer().setInputCol("cidade_busca").setOutputCol("cidade_buscaIndex").fit(indexed)

val indexed2 = indexer2.transform(indexed)

val indexer3 = new StringIndexer().setInputCol("universidade_nome").setOutputCol("universidade_nomeIndex").fit(indexed2)

val indexed3 = indexer3.transform(indexed2)

val indexer4 = new StringIndexer().setInputCol("campus_nome").setOutputCol("campus_nomeIndex").fit(indexed3)

val indexed4 = indexer4.transform(indexed3)

val indexer5 = new StringIndexer().setInputCol("nome").setOutputCol("nomeIndex").fit(indexed4)

val indexed5 = indexer5.transform(indexed4)

val indexer6 = new StringIndexer().setInputCol("grau").setOutputCol("grauIndex").fit(indexed5)

val indexed6 = indexer6.transform(indexed5)

val indexer7 = new StringIndexer().setInputCol("turno").setOutputCol("turnoIndex").fit(indexed6)

val indexed7 = indexer7.transform(indexed6)
```

// Verificar o resultado da transformação

```
indexed7.show()
```

// Listando apenas as colunas indexadas.

```
indexed7.select("uf_busca", "uf_buscaIndex", "universidade_nomeIndex", "nomeIndex", "grauIndex", "turnoIndex").distinct().show()
```

uf_busca	uf_buscaIndex	universidade_nomeIndex	nomeIndex	grauIndex	turnoIndex
AM	18.0	2.0	0.0	0.0	0.0
MA	15.0	2.0	0.0	0.0	0.0
MG	1.0	195.0	0.0	0.0	1.0
PR	2.0	2.0	0.0	0.0	0.0
RJ	6.0	5.0	0.0	0.0	0.0
RS	3.0	64.0	0.0	0.0	1.0
SP	0.0	421.0	0.0	0.0	1.0
SP	0.0	231.0	0.0	0.0	2.0
RS	3.0	214.0	67.0	0.0	3.0
RS	3.0	599.0	9.0	1.0	1.0
PE	9.0	61.0	25.0	0.0	2.0
SC	5.0	425.0	25.0	0.0	1.0
DF	10.0	0.0	28.0	2.0	0.0
SP	0.0	11.0	87.0	1.0	0.0
PR	2.0	760.0	33.0	0.0	1.0
RS	3.0	123.0	33.0	0.0	1.0
ES	12.0	16.0	31.0	2.0	0.0
SP	0.0	174.0	31.0	2.0	1.0
DF	10.0	75.0	2.0	0.0	2.0
ES	12.0	1099.0	2.0	0.0	2.0

only showing top 20 rows

// Criando as bases de treinamento e teste

//Instanciando um assembler para criar um único vetor de features

```
val assembler = new VectorAssembler().setInputCols(Array( "uf_buscaIndex", "nomeIndex", "grauIndex", "turnoIndex")).setOutputCol("features")
```

//Aplicando a transformacao com o assembler nos dados de treinamento

```
val indexed8 = assembler.transform(indexed7)
```

// Split do dataframe indexado

```
val splitDF = indexed8.randomSplit(Array(1,1))
```

//Aplicando o split para criar dataframes de treino e teste

```
val (dataframeTreinamento,dataframeTeste) = (splitDF(0),splitDF(1))
```


// Aprendizado de máquina com os dados

//Instanciando o modelo RandomForestRegressor

```
val modelo = new RandomForestRegressor().setLabelCol("mensalidade")
    .setFeaturesCol("features").setMaxBins(338).setNumTrees(100).setSeed(1)
```

//Treinando o modelo com os dados de treinamento

```
val modeloTreinado = modelo.fit(indexed8)
```

//Fazendo Predicoes com os dados de teste ou desconhecidas

```
val fazendoPredicoes = modeloTreinado.transform(dataframeTeste)
```

```
fazendoPredicoes.select("uf_busca", "nome", "grau", "turno", "prediction", "mensalidade").show()
```

// Criando avaliadores para as métricas do modelo, usando o RMSE e o R2

```
val avaliador = new RegressionEvaluator().setLabelCol("mensalidade")
    .setPredictionCol("prediction").setMetricName("rmse")

val Test_rmse = avaliador.evaluate(fazendoPredicoes)

val avaliador2 = new RegressionEvaluator().setLabelCol("mensalidade")
    .setPredictionCol("prediction").setMetricName("r2")

val Test_R2 = avaliador2.evaluate(fazendoPredicoes)

println("A RAIZ DO ERRO MÉDIO QUADRÁTICO (RMSE) nos dados de teste
= " + Test_rmse)

println("O R2 nos dados de teste : " + Test_R2);
```

//Criando uma tabela temporaria para fazer queries nos resultados usando SQL

```
fazendoPredicoes.createOrReplaceTempView("minhaTabela")
```

```
spark.sql("SELECT uf_busca, grau, turno, nome, prediction, mensalidade FROM minhaTabela WHERE uf_busca = 'RJ'").show
```

uf_busca	grau	turno	nome	prediction	mensalidade
RJ	Bacharelado	Curso a Distância	Ciências Contábeis	330.7527164427505	325.0
RJ	Tecnológico	Curso a Distância	Processos Gerenciais	304.01096607823195	325.0
RJ	Bacharelado	Curso a Distância	Administração	333.34754851326505	279.0
RJ	Bacharelado	Curso a Distância	Enfermagem	595.8265423619935	519.0
RJ	Bacharelado	Noturno	Administração	979.6470106030273	720.0
RJ	Bacharelado	Noturno	Biomedicina	1220.7292393503467	920.0
RJ	Bacharelado	Noturno	Direito	1222.2298069860888	820.0
RJ	Tecnológico	Curso a Distância	Gestão Hospitalar	293.8268250235524	279.0
RJ	Licenciatura	Curso a Distância	História	283.4026037201119	289.0
RJ	Licenciatura	Curso a Distância	Sociologia	269.552930247138	149.0
RJ	Bacharelado	Curso a Distância	Ciências Contábeis	330.7527164427505	402.65
RJ	Licenciatura	Curso a Distância	Pedagogia	297.7451310451306	402.65
RJ	Bacharelado	Noturno	Administração	979.6470106030273	632.28
RJ	Bacharelado	Noturno	Direito	1222.2298069860888	815.15
RJ	Tecnológico	Curso a Distância	Gestão de Recurso...	304.2915754562678	296.0
RJ	Bacharelado	Curso a Distância	Administração	333.34754851326505	259.0
RJ	Tecnológico	Curso a Distância	Análise e Desenv...	301.83506572768994	239.0
RJ	Bacharelado	Curso a Distância	Arquitetura e Urb...	674.864699462568	579.0
RJ	Licenciatura	Curso a Distância	Artes Visuais	275.44266879296515	209.0
RJ	Licenciatura	Curso a Distância	Ciências Biológicas	297.1462073482199	249.0

only showing top 20 rows

// Aprendizado de máquina com os dados – segundo exemplo

//Instanciando o modelo GBRegressor

```
val modelo = new GBRegressor().setLabelCol("nota_integral_ampla").  
setFeaturesCol("features").setMaxBins(338).setMaxIter(10)
```

//Removendo dados nulos

```
val dfFiltrada = indexed8.filter($"nota_integral_ampla".isNotNull)
```

//Treinando o modelo com os dados de treinamento

```
val modeloTreinado2 = modelo.fit(dfFiltrada)
```

// Split do dataframe indexado

```
val splitDF = dfFiltrada.randomSplit(Array(1,1))  
  
val (dataframeTreinamento,dataframeTeste) = (splitDF(0),splitDF(1))
```

// Fazendo Predicoes com os dados de teste ou desconhecidas

```
val fazendoPredicoes = modeloTreinado2.transform(dataframeTeste)
```

// Criando avaliadores para as metricas do modelo, usando o RMSE e o R2

```
val avaliador = new RegressionEvaluator().setLabelCol("nota_integral_ampla").setPredictionCol("prediction").setMetricName("rmse")
```

```
val Test_rmse = avaliador.evaluate(fazendoPredicoes)  
val avaliador2 = new RegressionEvaluator().setLabelCol("nota_integral_ampla").setPredictionCol("prediction").setMetricName("r2")
```

```
val Test_R2 = avaliador2.evaluate(fazendoPredicoes)
```

```
println("A RAIZ DO ERRO MÉDIO QUADRÁTICO (RMSE) nos dados de teste  
= " + Test_rmse)
```

```
println("O R2 nos dados de teste : " + Test_R2);
```

// Avaliando os resultados

```
fazendoPredicoes.createOrReplaceTempView("minhaTabela")
```

```
spark.sql("SELECT uf_busca, grau, turno, nome, prediction, nota_integral_ampla FROM minhaTabela WHERE uf_busca = 'RJ'").show
```

uf_busca	grau	turno	nome	prediction	nota_integral_ampla
RJ	Bacharelado	Curso a Distância	Ciências Contábeis	594.8813062569851	606.18
RJ	Licenciatura	Curso a Distância	Pedagogia	574.1427519039341	545.04
RJ	Bacharelado	Curso a Distância	Enfermagem	613.7621665644414	590.64
RJ	Bacharelado	Curso a Distância	Serviço Social	570.5937317708701	543.84
RJ	Bacharelado	Noturno	Administração	614.6686201328874	599.24
RJ	Bacharelado	Noturno	Direito	653.5714345322231	625.16
RJ	Tecnológico	Curso a Distância	Gestão Hospitalar	579.4382011417333	534.32
RJ	Licenciatura	Curso a Distância	Letras - Português	577.0800113389572	590.58
RJ	Bacharelado	Curso a Distância	Ciências Contábeis	594.8813062569851	578.98
RJ	Bacharelado	Noturno	Administração	614.6686201328874	595.0
RJ	Bacharelado	Noturno	Direito	653.5714345322231	674.08
RJ	Bacharelado	Noturno	Enfermagem	634.1907770992116	631.24
RJ	Tecnológico	Curso a Distância	Gestão de Recurso...	566.0697522518115	574.9
RJ	Bacharelado	Curso a Distância	Administração	594.2880578679697	611.1
RJ	Licenciatura	Curso a Distância	Artes Visuais	579.3227745908341	609.5
RJ	Tecnológico	Curso a Distância	Embelezamento e I...	574.9896853953522	568.54
RJ	Tecnológico	Curso a Distância	Gestão Pública	574.9896853953522	561.04
RJ	Tecnológico	Curso a Distância	Gestão de Recurso...	566.0697522518115	600.12
RJ	Licenciatura	Curso a Distância	História	599.3967928230439	619.72
RJ	Licenciatura	Curso a Distância	Letras - Português	577.0800113389572	550.4

only showing top 20 rows

