Atividade de Aula – Trabalho Prático

Disciplina	FAM – Fundamentos de Aprendizado de Máquina

Objetivos

O Trabalho Prático em laboratório virtual possui como objetivo principal o uso do algoritmo K-Means, por meio do framework Apache Spark e da biblioteca de Machine Learning MLlib, aplicado a uma base de dados.

Ao final desta atividade, o aluno deverá ser capaz de executar um experimento com o algoritmo K-Means utilizando o Apache Spark e a linguagem Scala.

Algoritmo K-Means

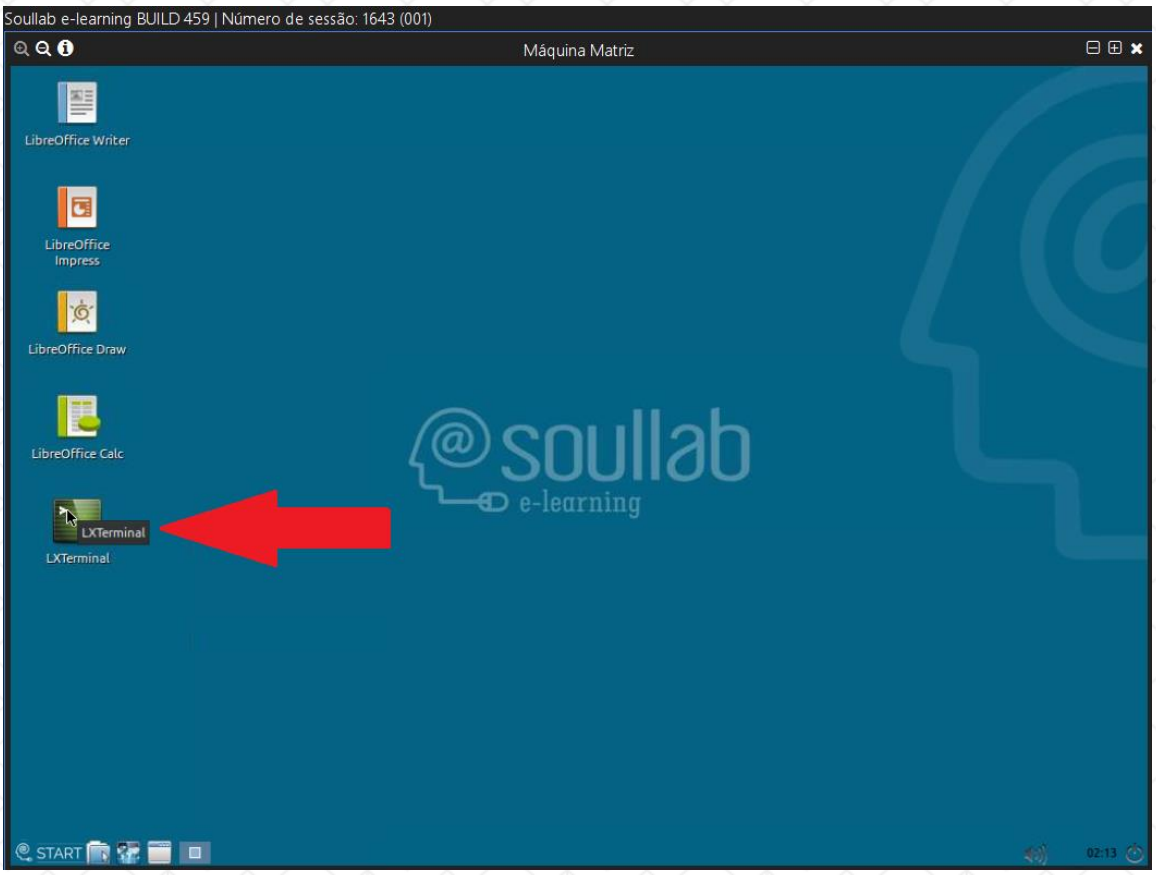
O algoritmo K-Means (também chamado de K-Médias), possui como objetivo fornecer uma clusterização de informações de acordo com o próprio conjunto de dados. O algoritmo é uma técnica poderosa para particionar um conjunto de dados em grupos separados, onde o valor de k deve ser predeterminado. Sendo assim, k é o número de grupos.

Mais informações sobre o algoritmo K-Means podem ser encontradas aqui: http://stanford.edu/~cpiech/cs221/handouts/kmeans.html

Enunciado

No momento de iniciar esta atividade é esperado que o aluno já tenha recebido seus dados de login para acesso ao laboratório virtual.

Após executar o login no laboratório virtual, inicie o Terminal do Linux, conforme a imagem abaixo:



Foi criado, no ambiente virtual, um usuário exclusivo para trabalhar com o Spark. O nome desse usuário é **spuser** e a sua senha é **spark**. Você deve fazer login com o usuário **spuser** e, para isso, execute o comando abaixo:

su **spuser**

Após digitar o comando acima, o sistema operacional irá solicitar a digitação da senha. Informe a senha "**spark**" (desconsidere as aspas).

A ferramenta Spark já foi previamente instalada no ambiente virtual e encontra-se no diretório `/usr/local/spark`.

Em seguida você deverá executar o Spark Shell. Para isso, utilize o seguinte comando:

```
/usr/local/spark/bin/spark-shell
```

Os dados que serão utilizados no experimento já foram baixados e se encontram no diretório `/usr/local/exemplosml/TrabalhoPratico/Dados/crime.csv`.

O objetivo desse experimento é analisar uma base de dados de estatísticas de crimes em alguns estados dos EUA, mostrando quais áreas são menos perigosas. É recomendado que você verifique o conteúdo desse arquivo acessando o diretório `/usr/local/exemplosml/TrabalhoPratico/Dados/` e utilizando o aplicativo vi.

Dentro do spark-shell, você deverá fazer referência às bibliotecas:

```
import org.apache.spark.mllib.clustering.{KMeans, KMeansModel}
import org.apache.spark.mllib.linalg.Vectors
import spark.sqlContext.implicits._
import org.apache.spark.sql.types._
```

Em seguida, crie um RDD com os dados:

```
var rdd = sc.textFile("/usr/local/exemplosml/TrabalhoPratico/Dados/crime.csv ")
```

Agora crie uma função que transforme os dados do arquivo, que estão em formato string, para o formato double:

```
def stoDouble (s : String): Double = {return
s.map(_.toByte.doubleValue()).reduceLeft( (x,y) => x + y)}
```

Como os dados de Estado foram anteriormente convertidos em double, precisaremos de algo que transforme os dados em formato String de volta. Criamos um dataframe chamado States.

```
case class StateCode(State:String, Code:Double)
var lines = rdd.map(l => l.split(","))
var states = lines.map(l => StateCode(l(0),stoDouble(l(0)))).toDF()
states.show()
states.createOrReplaceTempView("states")
```

Convertemos os dados do RDD original para um novo RDD chamado *crime*, usando a função makeDouble.

```
def makeDouble (s: String): Array[Double] = {
```

```
var str = s.split(",")
var a = stoDouble(str(0))
return Array(a,str(2).toDouble,str(3).toDouble,str(4).
toDouble,str(5).toDouble)
}
var crime = rdd.map(m => makeDouble(m))
```

Agora deve-se criar um Dense Vector (assim como fizemos em nossas atividades anteriores.

```
val crimeVector = crime.map(a => Vectors.dense(a(0),a(1),a(2),a(3),a(4)))
```

Vamos agora realizar o treinamento:

```
var crime = rdd.map(m => makeDouble(m))
```

Criamos agora outra case class (Crime), que irá apresentar os resultados finais:

```
case class Crime (Code:Double, Murder:Double, Assault:Double,
UrbanPop:Double,Rape:Double,
PredictionVector:org.apache.spark.mllib.linalg.Vector, Prediction:Double)
```

```
val crimeClass = crimeVector.map(a => Crime(a(0), a(1), a(2),
a(3), a(4), a ,clusters.predict(a))).toDF()

crimeClass.show()

crimeClass.createOrReplaceTempView("crimes")
```

Você criou duas tabelas temporárias que agora podem ser manipuladas via sentenças SQL.

Tente inserir o seguinte comando:

```
spark.sql("select State, Prediction, UrbanPop, from crimes inner join
states on crimes.Code = states.Code order by Prediction Desc").show(50,
false)
```

Referências

ROWE, Walker. *Spark Decision Tree Classifier*. Disponível em: <https://www.bmc.com/blogs/k-means-clustering-apache-spark/>. Acesso em: 09 abr. 2018.