**Olá, aluno(a)!**
**Seja bem-vindo(a) à aula interativa!**

Você entrará na reunião com a câmera e o microfone desligados.

Sua presença será computada através da enquete.
Fique atento(a) e não deixe de respondê-la!

# Desenvolvimento de Soluções Utilizando Spark

Segunda Aula Interativa

Prof. Pedro Calais

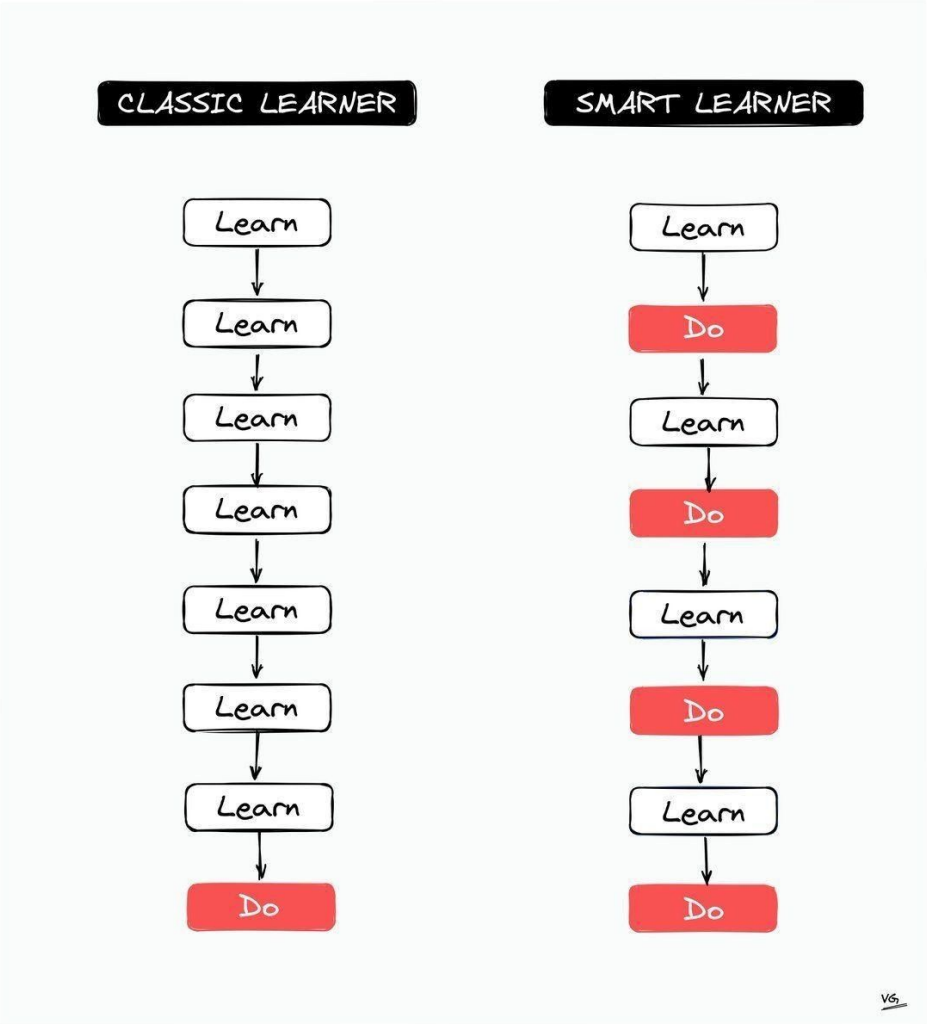
# O que vimos na primeira semana?

- Introdução o Apache Spark.
  - Big Data.
  - Computação distribuída.
- Conceitos Fundamentais do Spark.
  - RDDs / Dataframe.
- Estatística Descritiva com Spark.
  - df.describe().

# Plano de hoje

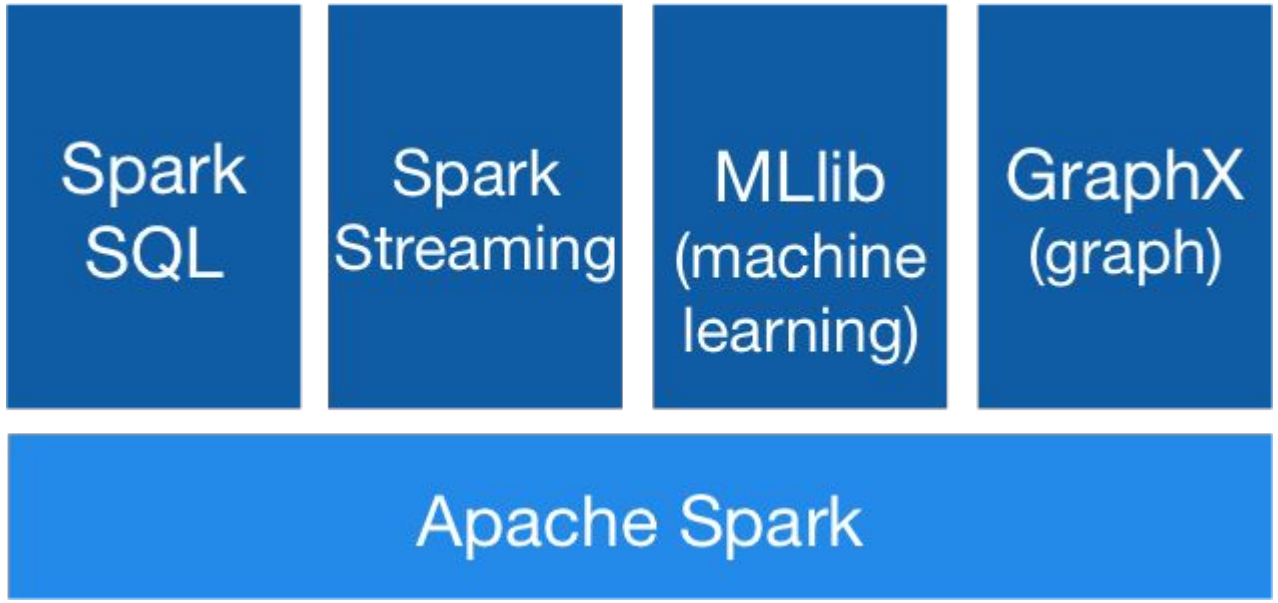
- Revisar a segunda semana.
- Escrever código juntos!

# Como aprender?



# Segunda semana:

# componentes do Spark



# Spark SQL

# Spark SQL

- Módulo do Apache Spark que integra o processamento de dados estruturados e relacionais com a API do Spark.

# Spark SQL usa DataFrames

- Coleção distribuída de registros com o mesmo esquema.
- É como um banco relacional.

| dept | age | name |
|---|---|---|
| Bio | 48 | H Smith |
| CS | 54 | A Turing |
| Bio | 43 | B Jones |
| Chem | 61 | M Kennedy |

Data grouped into named columns

# Spark SQL usa DataFrames

*RDD API*

```
pdata.map(lambda x: (x.dept, [x.age, 1])) \
  .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
  .map(lambda x: [x[0], x[1][0] / x[1][1]]) \
  .collect()
```

*DataFrame API*

```
data.groupBy("dept").avg("age")
```

# Operações com Dataframe

```
employees
  .join(dept, employees("deptId") === dept("id"))
  .where(employees("gender") === "female")
  .groupBy(dept("id"), dept("name"))
  .agg(count("name"))
```

# Suporte à linguagem SQL

- Em algumas situações, o SQL “puro” é mais conveniente.
- Útil quando você não conhece a API do Spark.
- Útil quando você já é experiente em SQL.

```
users.where(users("age") < 21)
     .registerTempTable("young")
ctx.sql("SELECT count(*), avg(age) FROM young")
```

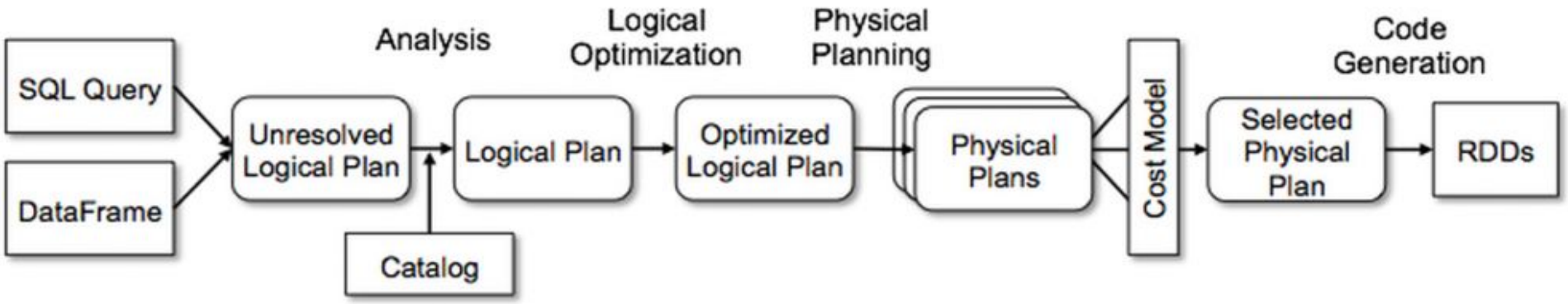
# Benefícios do Spark SQL

- Integração:
  - Spark SQL “mistura” consultas SQL em programas Spark.
  - Integrar consultas SQL com *analytics* complexos.

# Benefícios do Spark SQL

Otimização de consultas:

- Plano lógico e físico para cada consulta.
- *Catalyst Optimizer:*



# O que são UDFs?

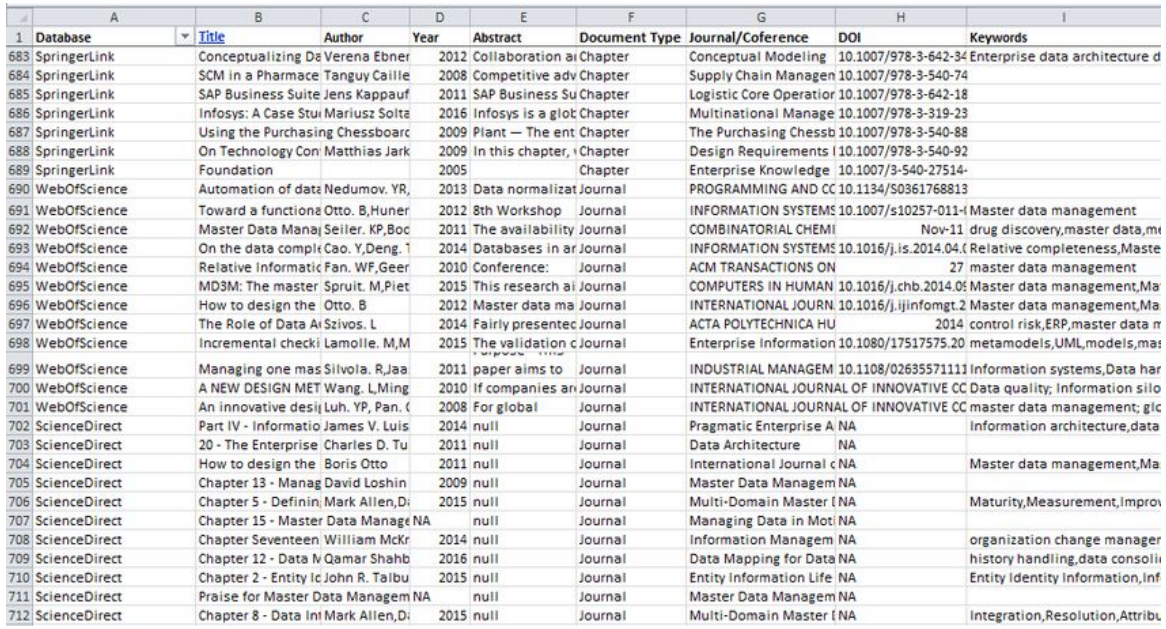
```
val model: LogisticRegressionModel = ...

ctx.udf.register("predict",
  (x: Float, y: Float) => model.predict(Vector(x, y)))

ctx.sql("SELECT predict(age, weight) FROM users")
```

Combinação poderosa:

- Operadores relacionais.
- Funções analíticas complexas.

# Spark SQL suporta vários formatos de dados

- CSV.
- JSON.
- texto.
- Parquet.
- tabelas do Hive.



| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Database | Title | Author | Year | Abstract | Document Type | Journal/Coference | DOI | Keywords |
| 683 | SpringerLink | Conceptualizing Da | Verena Ebner | 2012 | Collaboration ai | Chapter | Conceptual Modeling | 10.1007/978-3-642-34 | Enterprise data architecture d |
| 684 | SpringerLink | SCM in a Pharmace | Tanguy Caille | 2008 | Competitive adv | Chapter | Supply Chain Managen | 10.1007/978-3-540-74 | |
| 685 | SpringerLink | SAP Business Suite | Jens Kappauf | 2011 | SAP Business Su | Chapter | Logistic Core Operatior | 10.1007/978-3-642-18 | |
| 686 | SpringerLink | Infosys: A Case Stu | Mariusz Solta | 2016 | Infosys is a glot | Chapter | Multinational Manage | 10.1007/978-3-319-23 | |
| 687 | SpringerLink | Using the Purchasing | Chessboarc | 2009 | Plant — The ent | Chapter | The Purchasing Chessb | 10.1007/978-3-540-88 | |
| 688 | SpringerLink | On Technology Con | Matthias Jark | 2009 | In this chapter, | Chapter | Design Requirements | 10.1007/978-3-540-92 | |
| 689 | SpringerLink | Foundation | | 2005 | | Chapter | Enterprise Knowledge | 10.1007/3-540-27514- | |
| 690 | WebOfScience | Automation of data | Nedumov. YR, | 2013 | Data normalizat | Journal | PROGRAMMING AND CC | 10.1134/S0361768813 | |
| 691 | WebOfScience | Toward a functiona | Otto. B,Huner | 2012 | 8th Workshop | Journal | INFORMATION SYSTEMS | 10.1007/s10257-011-( | Master data management |
| 692 | WebOfScience | Master Data Mana | Seiler. KP,Boc | 2011 | The availability | Journal | COMBINATORIAL CHEMI | Nov-11 | drug discovery,master data,m |
| 693 | WebOfScience | On the data compl | Cao. Y,Deng. T | 2014 | Databases in ar | Journal | INFORMATION SYSTEMS | 10.1016/j.is.2014.04.( | Relative completeness,Maste |
| 694 | WebOfScience | Relative Informatic | Fan. WF,Geer | 2010 | Conference: | Journal | ACM TRANSACTIONS ON | 27 | master data management |
| 695 | WebOfScience | MD3M: The master | Spruit. M,Piet | 2015 | This research ai | Journal | COMPUTERS IN HUMAN | 10.1016/j.chb.2014.05 | Master data management,Ma |
| 696 | WebOfScience | How to design the | Otto. B | 2012 | Master data ma | Journal | INTERNATIONAL JOURN | 10.1016/j.ijinfomgt.2 | Master data management,Ma |
| 697 | WebOfScience | The Role of Data Ai | Szivos. L | 2014 | Fairly presentec | Journal | ACTA POLYTECHNICA HU | 2014 | control risk,ERP,master data m |
| 698 | WebOfScience | Incremental checki | Lamolle. M,M | 2015 | The validation c | Journal | Enterprise Information | 10.1080/17517575.20 | metamodels,UML,models,mas |
| 699 | WebOfScience | Managing one mas | Silvola. R,Jaa | 2011 | paper aims to | Journal | INDUSTRIAL MANAGEM | 10.1108/02635571111 | Information systems,Data har |
| 700 | WebOfScience | A NEW DESIGN MET | Wang. L,Ming | 2010 | If companies ar | Journal | INTERNATIONAL JOURNAL OF INNOVATIVE CC | | Data quality; Information silo |
| 701 | WebOfScience | An innovative desi | Luh. YP, Pan. ( | 2008 | For global | Journal | INTERNATIONAL JOURNAL OF INNOVATIVE CC | | master data management; glc |
| 702 | ScienceDirect | Part IV - Informatio | James V. Luis | 2014 | null | Journal | Pragmatic Enterprise A | NA | Information architecture,data |
| 703 | ScienceDirect | 20 - The Enterprise | Charles D. Tu | 2011 | null | Journal | Data Architecture | NA | |
| 704 | ScienceDirect | How to design the | Boris Otto | 2011 | null | Journal | International Journal c | NA | Master data management,Ma |
| 705 | ScienceDirect | Chapter 13 - Manag | David Loshin | 2009 | null | Journal | Master Data Managem | NA | |
| 706 | ScienceDirect | Chapter 5 - Definin | Mark Allen,Di | 2015 | null | Journal | Multi-Domain Master I | NA | Maturity,Measurement,Improv |
| 707 | ScienceDirect | Chapter 15 - Master | Data Manage | NA | null | Journal | Managing Data in Mot | NA | |
| 708 | ScienceDirect | Chapter Seventeen | William McKr | 2014 | null | Journal | Information Managem | NA | organization change manager |
| 709 | ScienceDirect | Chapter 12 - Data M | Qamar Shahb | 2016 | null | Journal | Data Mapping for Data | NA | history handling,data consoli |
| 710 | ScienceDirect | Chapter 2 - Entity Ic | John R. Talbu | 2015 | null | Journal | Entity Information Life | NA | Entity Identity Information,Inf |
| 711 | ScienceDirect | Praise for Master Data | Managem | NA | null | Journal | Master Data Managem | NA | |
| 712 | ScienceDirect | Chapter 8 - Data Int | Mark Allen,Di | 2015 | null | Journal | Multi-Domain Master I | NA | Integration,Resolution,Attribu |

https://spark.apache.org/docs/latest/sql-data-sources.html

# JSON

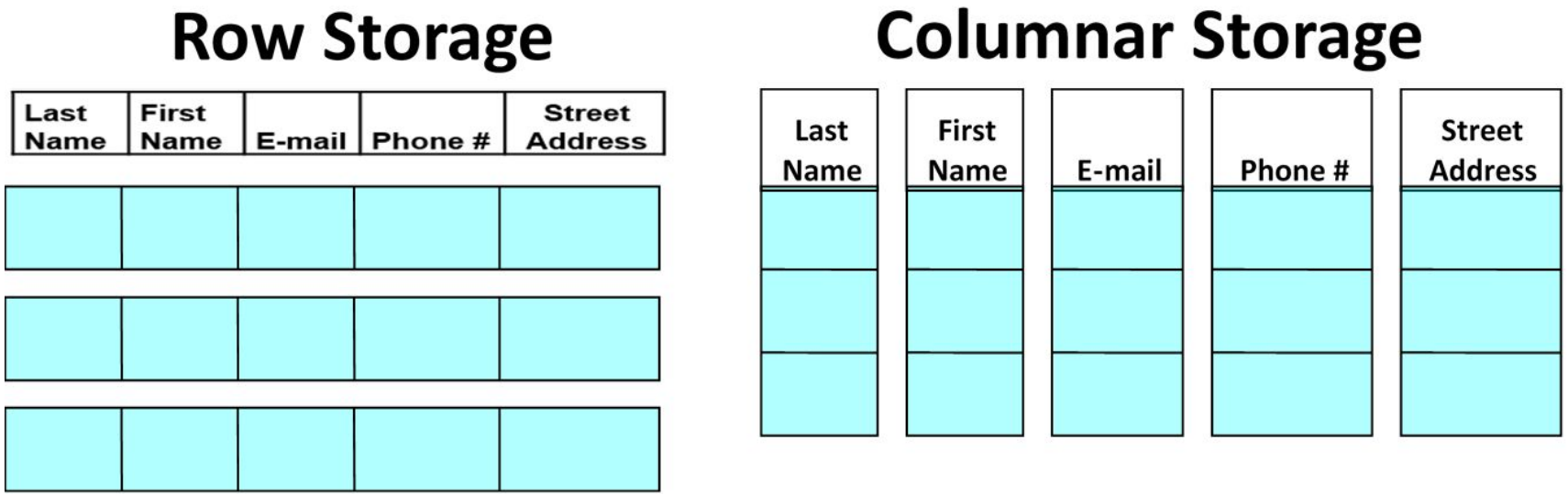
```scala
// Primitive types (Int, String, etc) and Product types (case classes) encoders are
// supported by importing this when creating a Dataset.
import spark.implicits._

// A JSON dataset is pointed to by path.
// The path can be either a single text file or a directory storing text files
val path = "examples/src/main/resources/people.json"
val peopleDF = spark.read.json(path)

// The inferred schema can be visualized using the printSchema() method
peopleDF.printSchema()
// root
//  |-- age: long (nullable = true)
//  |-- name: string (nullable = true)
```

```json
{
    "empid": "SJ011MS",
    "personal": {
        "name":"Smith Jones",
        "gender": "Male",
        "age": 28,
        "address":
        {
            "streetaddress": "7 24th Street",
            "city": "New York",
            "state": "NY",
            "postalcode": "10038"
        }
    },
    "profile": {
        "designation": "Deputy General",
        "department": "Finance"
    }
}
```



https://spark.apache.org/docs/latest/sql-data-sources-json.html

# Parquet: armazenamento por colunas

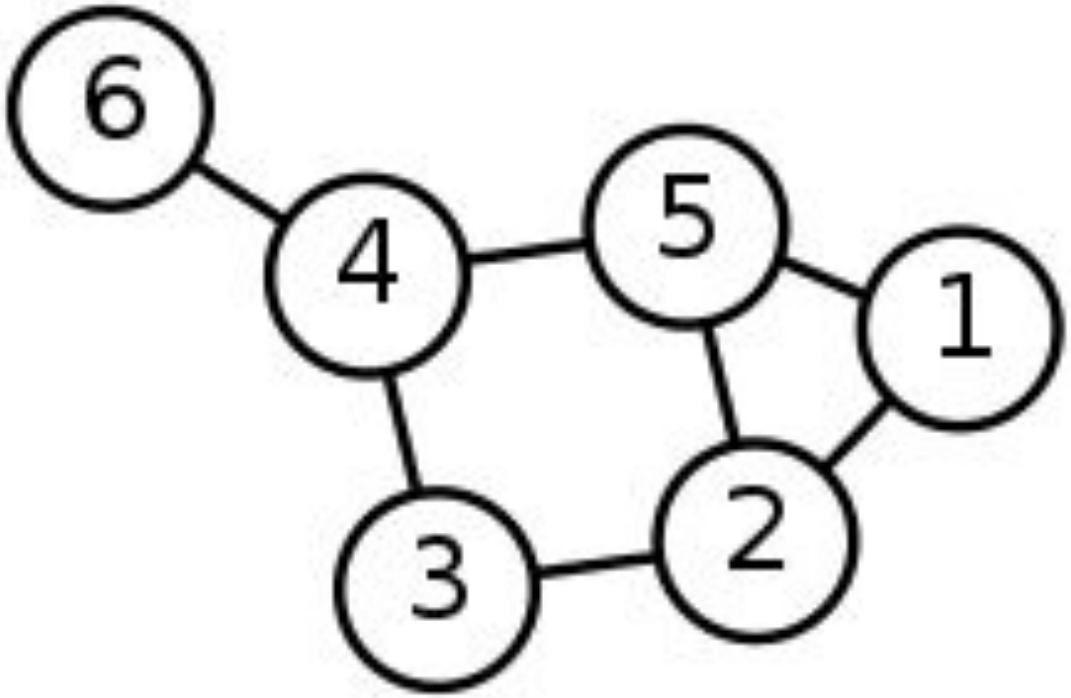



https://spark.apache.org/docs/latest/sql-data-sources-parquet.html

# Spark GraphX

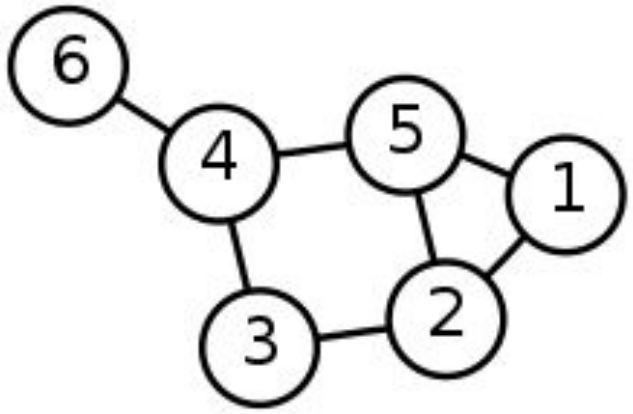
# O que são grafos?

- Uma estrutura matemática das mais importantes em Computação!
- Conjunto de vértices e arestas.



# Onde encontramos grafos?

- Redes sociais.
- Sistemas de telecomunicação.
- Páginas da web.
- Citação de artigos.

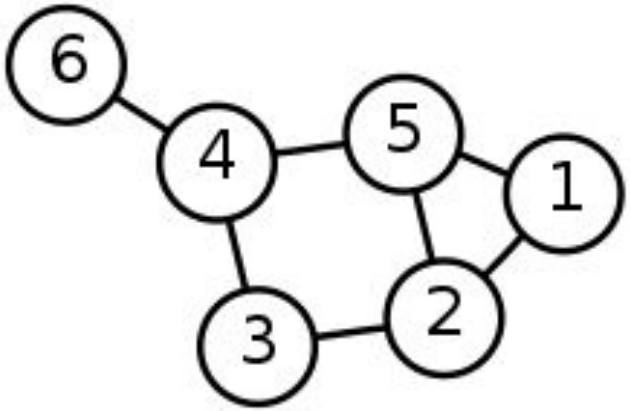


# Aplicações

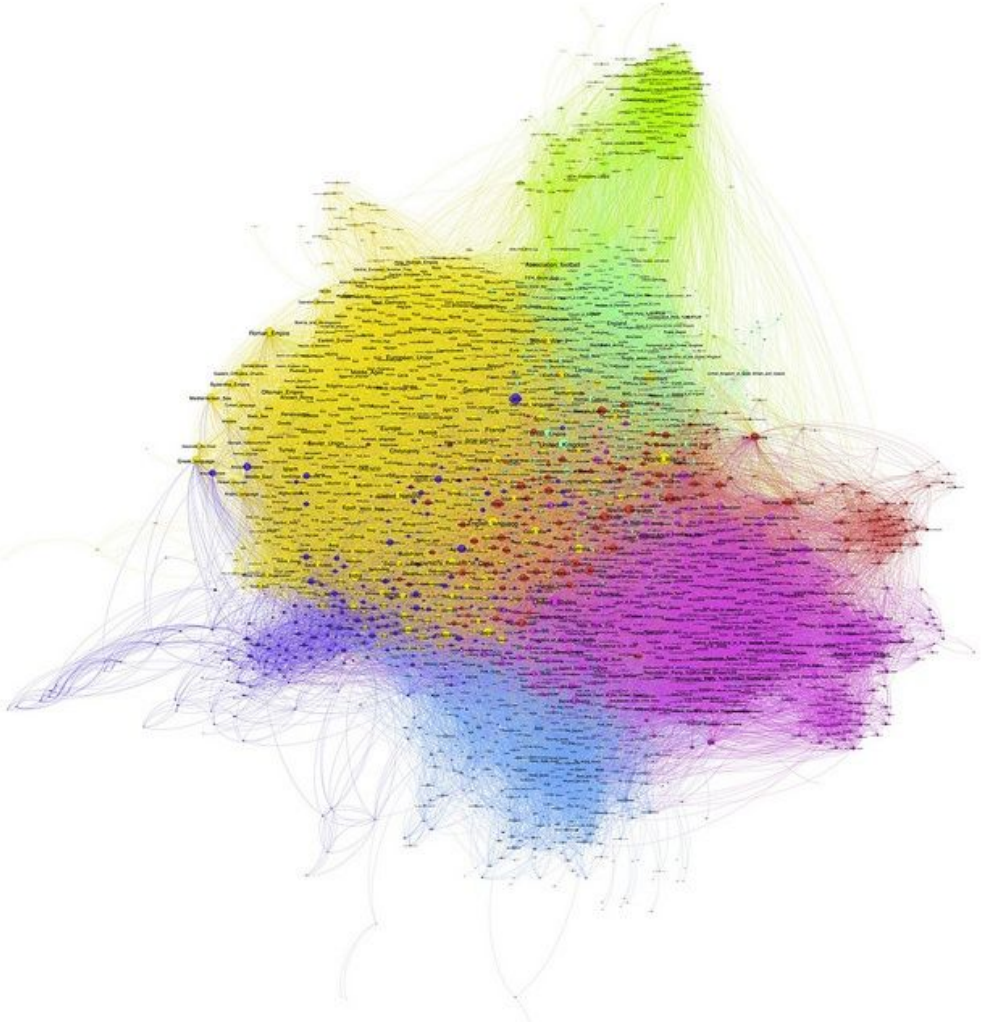
*Datapoints* são importantes individualmente, mas os grafos exploram as conexões entre eles!

Aplicações:

- Detecção de fraude.
- Recomendação.
- Bioinformática.
- Ranking de documentos da Web.

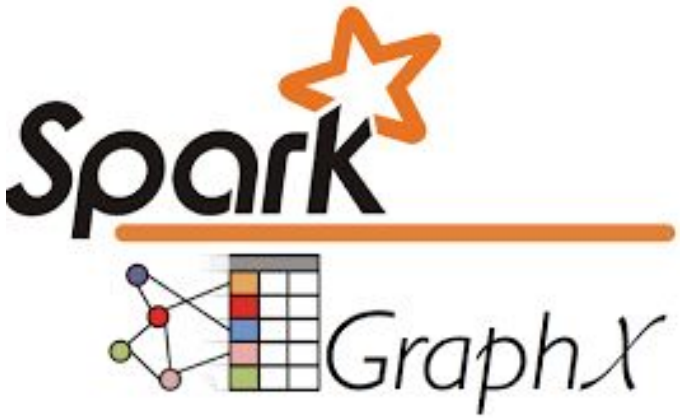


# O que são grafos?



# O que é o GraphX?

- Camada de processamento de grafos que funciona sobre o Spark.
- Útil para grafos que não cabem na memória de uma máquina.



# Spark Streaming

# Exemplos de dados em *stream*

- Finanças: preços de ações, identificar oportunidades de arbitragem.
- Monitoramento de logs e detecção de fraudes / hacking / DDOS.
- Sites de e-commerce: Clickstream.

# Big Data e *streams*

B2C BRANDVIEWS ⟩ UPWORK

## Streaming Data: Big Data at High Velocity

Tyler Keenan — August 9, 2016

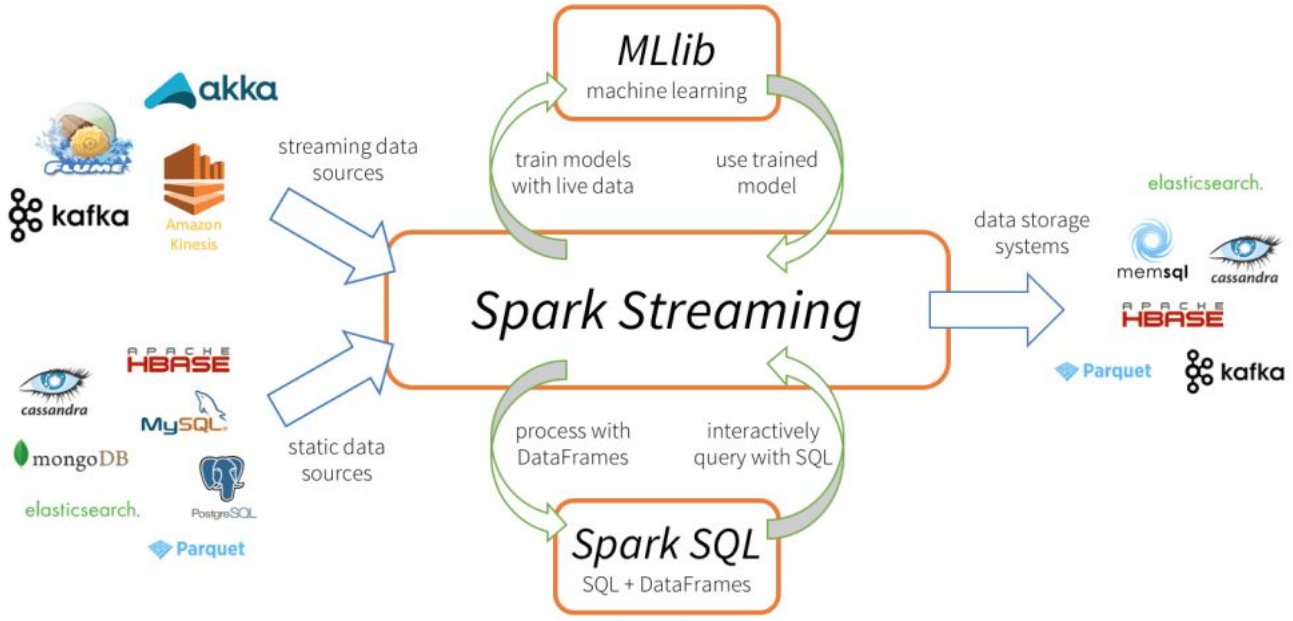
# Big Data e *streams*

# Dados em batch vs Dados em stream

- Dados em *batch*:
  - São coletados ao longo do tempo.
  - Depois de coletados, dados são enviados para processamento.
  - Processamento pode ser demorado.
- Dados em *stream*:
  - São produzidos continuamente.
  - São processados um a um.
  - Processamento deve ser rápido e o resultado é necessário imediatamente.

# Spark Streaming

- Extensão da API *core* do Spark, que permite que Cientistas e Engenheiros de Dados lidem com

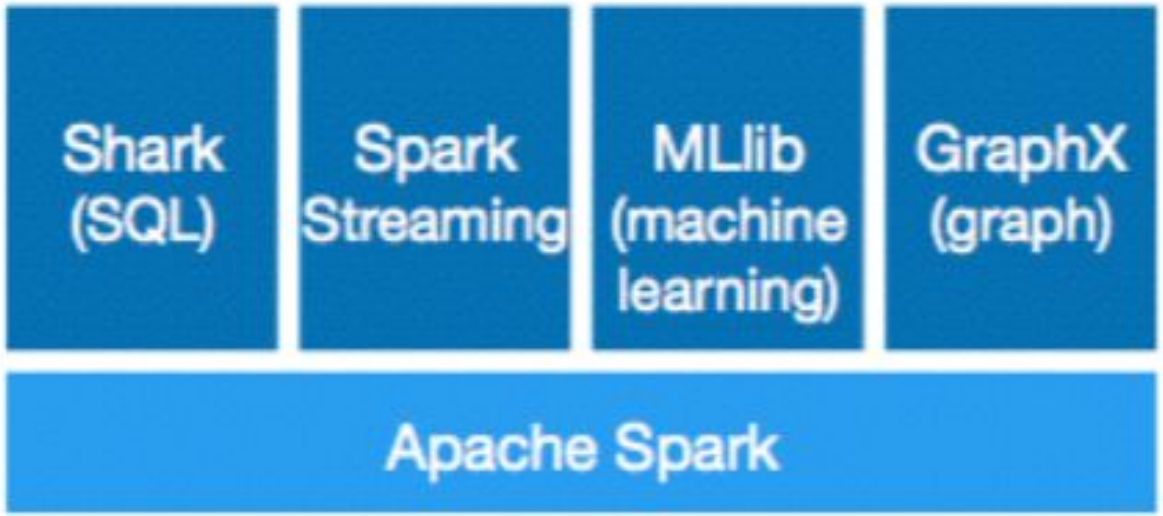


# Quatro principais características

- Recuperação rápida de falhas e lentidões.
- Balanceamento de carga e otimização dos recursos.
- Combinação de dados em stream e dados estáticos.
- Integração com Spark SQL, GraphX e MLLib.

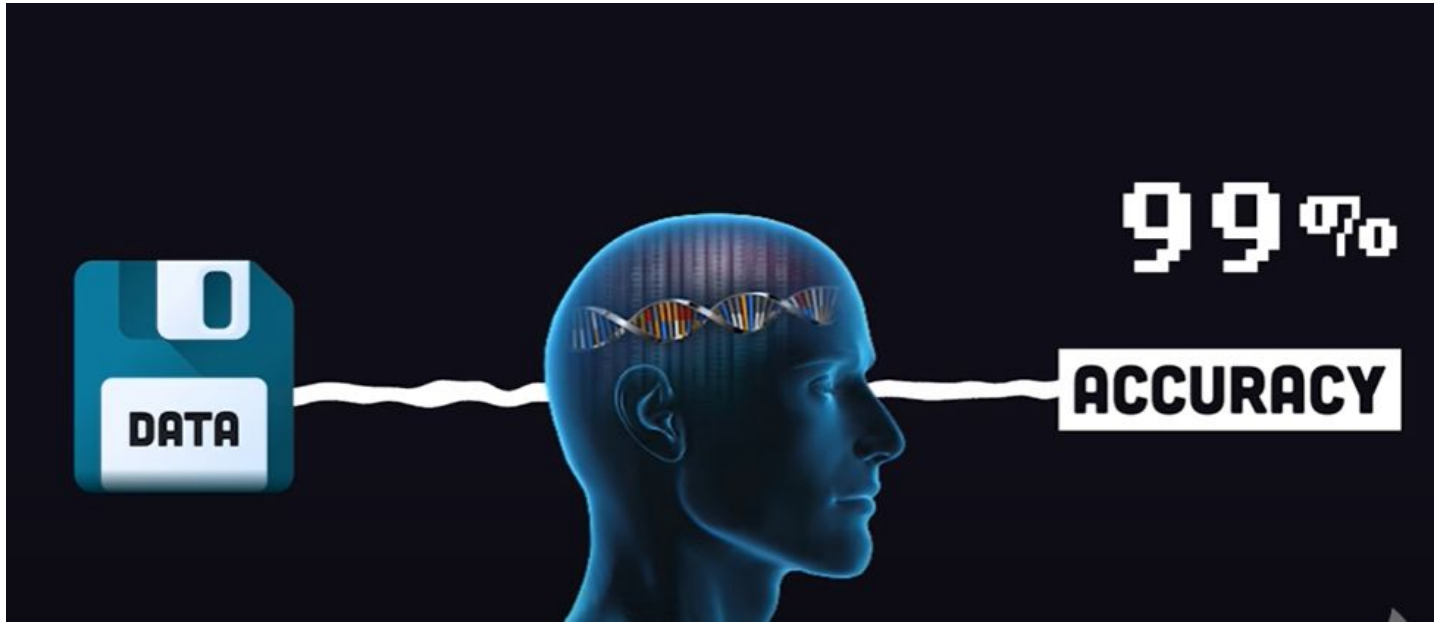
# Spark MLLib

# Spark MLLib


Shark (SQL)
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Apache Spark

# O que é *Machine Learning (ML)?*

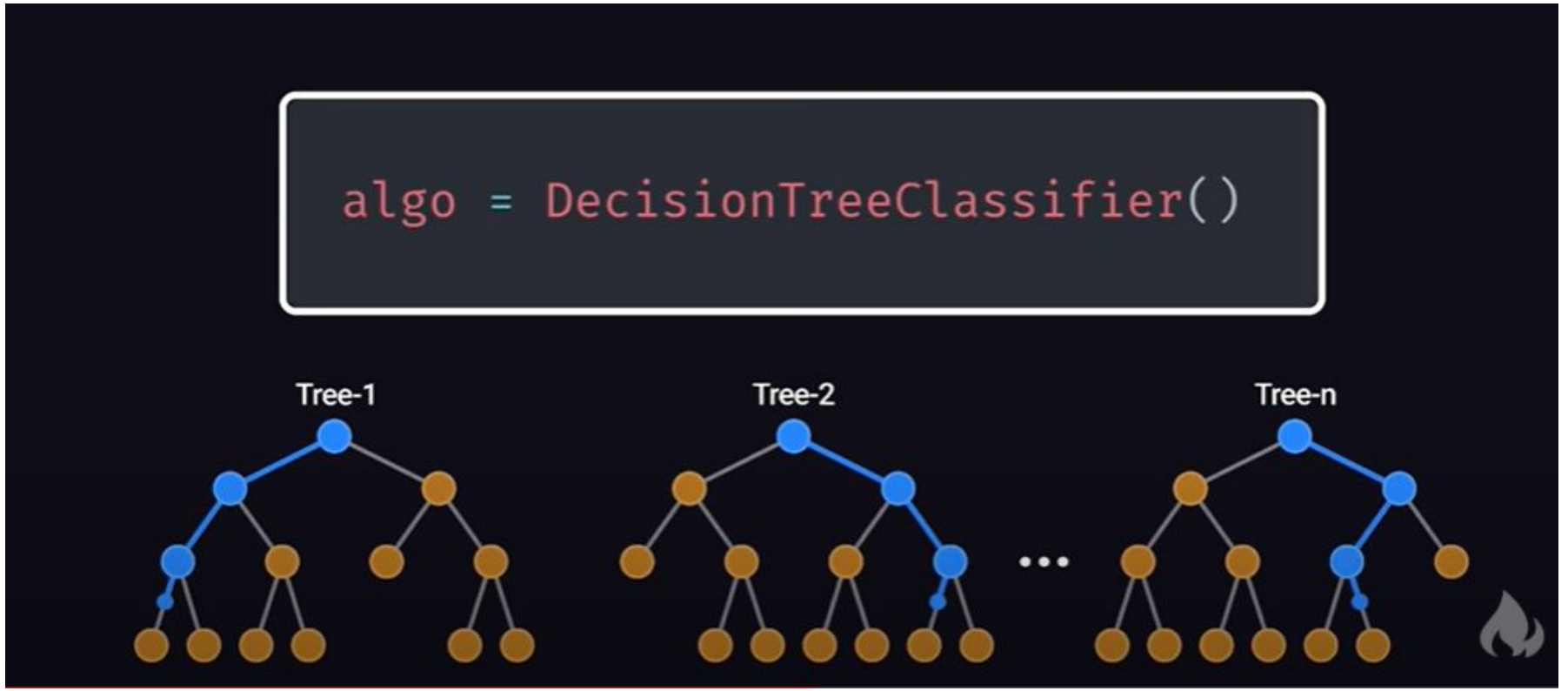
- Sub-área da inteligência artificial que estuda como máquinas podem **imitar** o comportamento humano na **execução de tarefas**:
- Sem **programação explícita**!



# O que é *Machine Learning (ML)?*

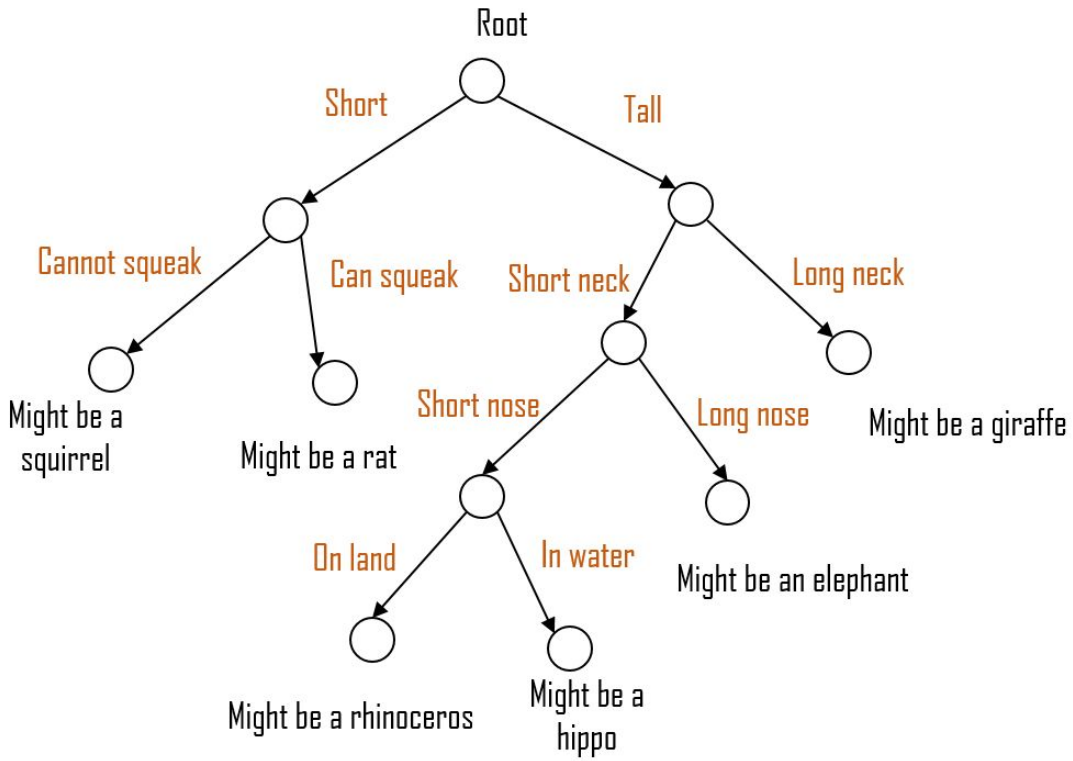
exemplo: prever o preço de mercado de um apartamento.

# O que é *Machine Learning (ML)?*



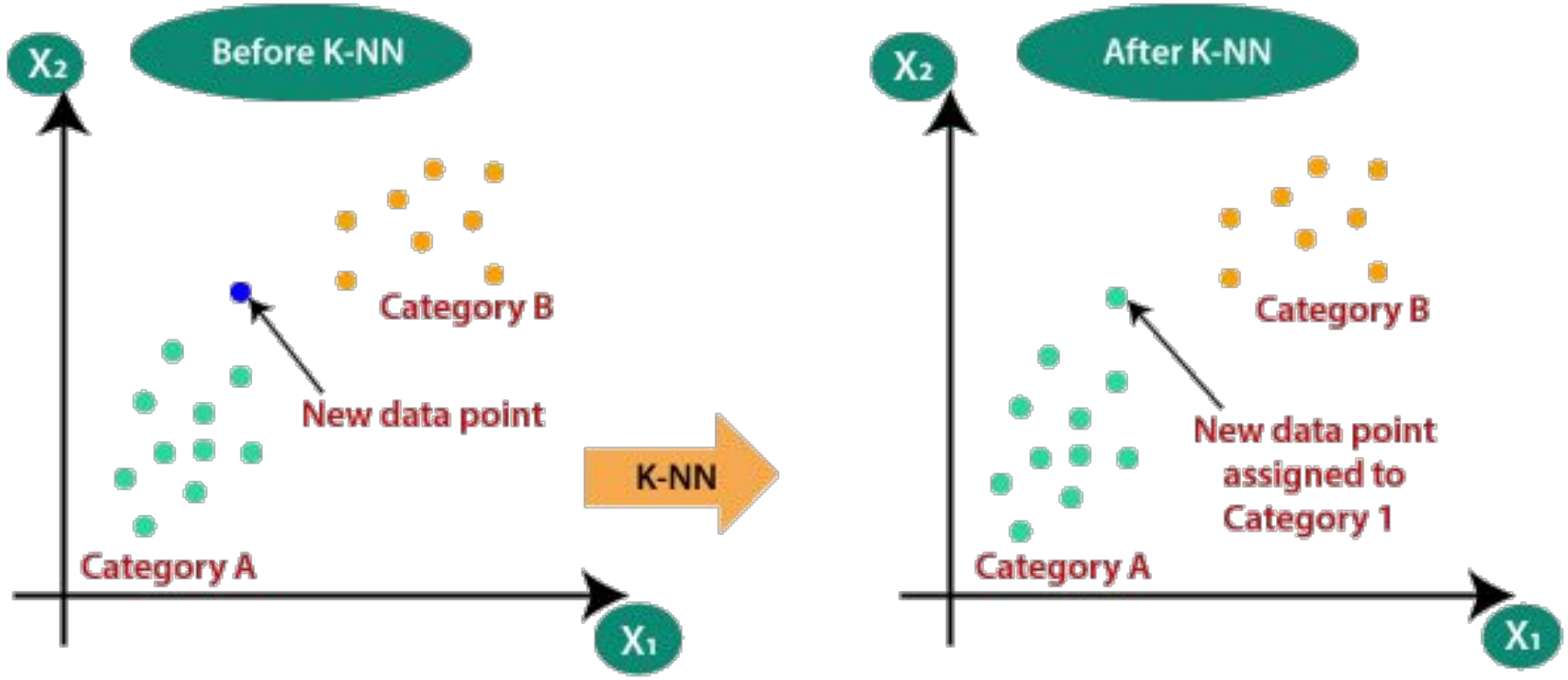
# O que é *Machine Learning (ML)?*

exemplo de algoritmo: árvore de decisão.

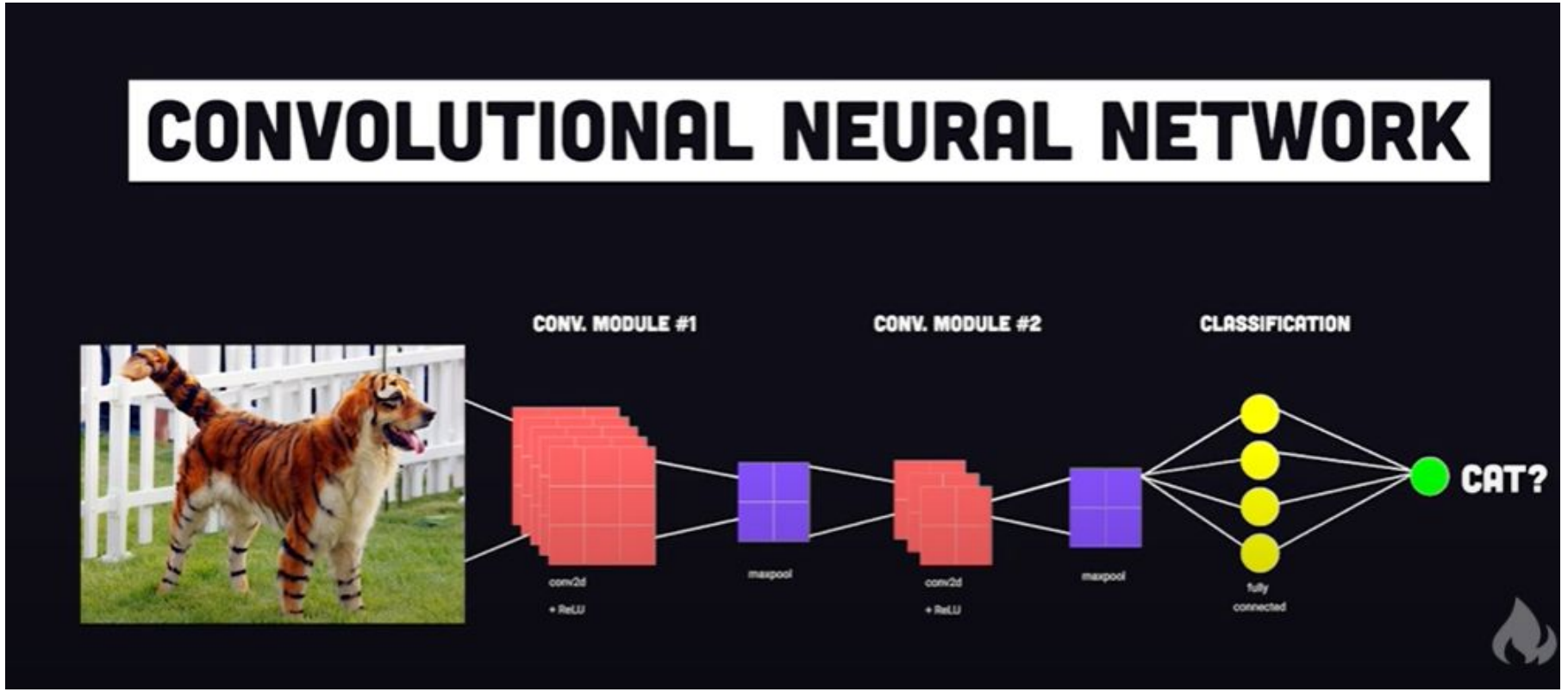



# O que é *Machine Learning (ML)?*

exemplo de algoritmo: KNN.



# O que é *Machine Learning (ML)?*

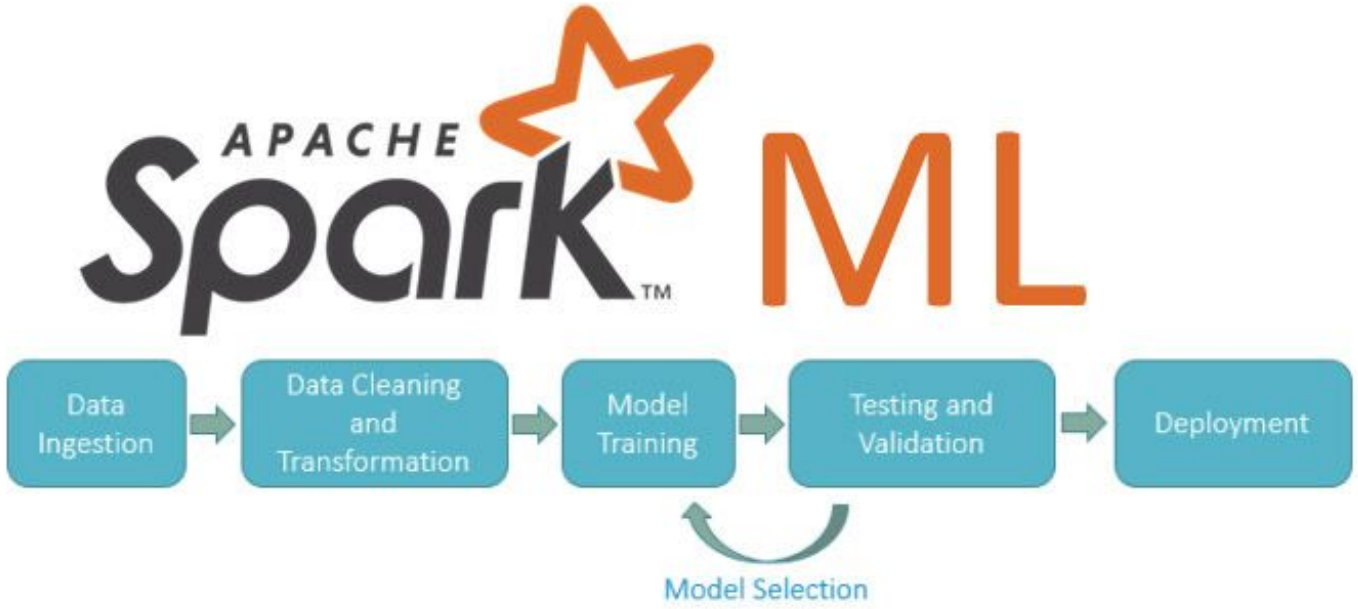


# Ciência de Dados vs ML

- Ciência de Dados:
  - estuda dados e como extrair significado deles
  - campo amplo e multidisciplinar
- Aprendizado de Máquina:
  - construção de modelos que aprendem a partir dos dados
  - É uma ferramenta para **ciência de dados**

# Características principais da Spark ML

- Implementações distribuídas dos principais algoritmos de aprendizado de máquina.
- Integração com os outros módulos do Spark.
- Suporta criação de Pipelines de ML.



# Algoritmos

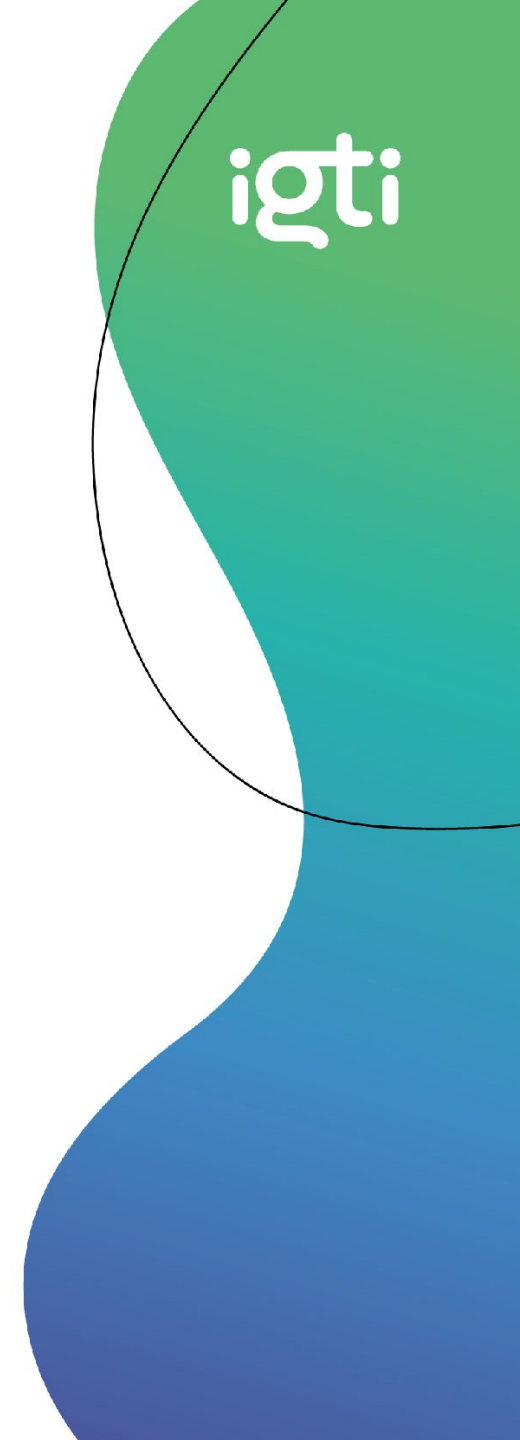
- Classificação.
- Regressão.
- Filtragem Colaborativa.
- Agrupamento.

# Spark SQL + Spark MLLib

```
// Data can easily be extracted from existing sources,
// such as Apache Hive.
val trainingTable = sql("""
  SELECT e.action,
         u.age,
         u.latitude,
         u.longitude
  FROM Users u
  JOIN Events e
  ON u.userId = e.userId""")

// Since `sql` returns an RDD, the results of the above
// query can be easily used in MLlib.
val training = trainingTable.map { row =>
  val features = Vectors.dense(row(1), row(2), row(3))
  LabeledPoint(row(0), features)
}

val model = SVMWithSGD.train(training)
```

# Streaming + MLLib

```
// collect tweets using streaming

// train a k-means model
val model: KMmeansModel = ...

// apply model to filter tweets
val tweets = TwitterUtils.createStream(ssc, Some(authorizations(0)))
val statuses = tweets.map(_.getText)
val filteredTweets =
  statuses.filter(t => model.predict(featurize(t)) == clusterNumber)

// print tweets within this particular cluster
filteredTweets.print()
```



# GraphX + MLLib

```
// assemble link graph
val graph = Graph(pages, links)
val pageRank: RDD[(Long, Double)] = graph.staticPageRank(10).vertices

// load page labels (spam or not) and content features
val labelAndFeatures: RDD[(Long, (Double, Seq((Int, Double))))] = ...
val training: RDD[LabeledPoint] =
  labelAndFeatures.join(pageRank).map {
    case (id, ((label, features), pageRank)) =>
      LabeledPoint(label, Vectors.sparse(features ++ (1000, pageRank))
}

// train a spam detector using logistic regression
val model = LogisticRegressionWithSGD.train(training)
```

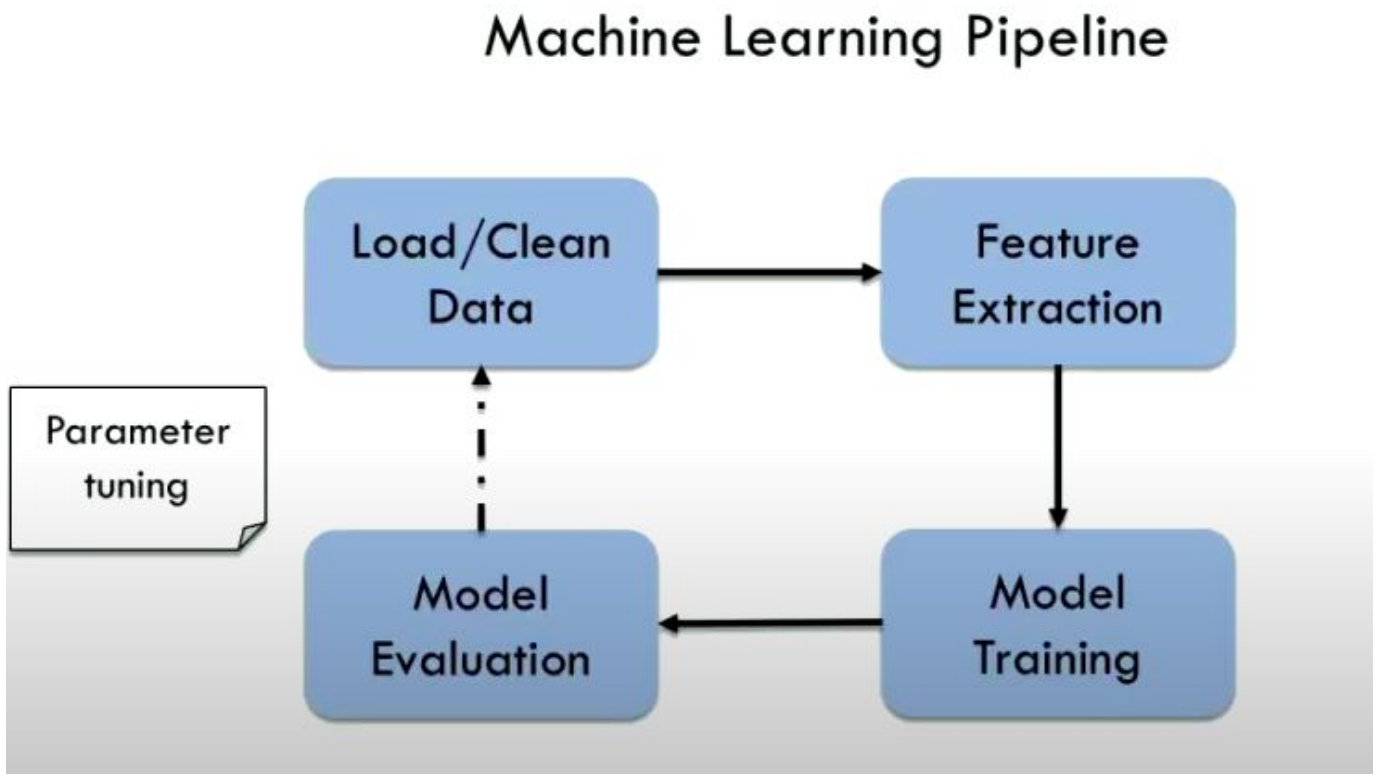
# Spark MLLib

-Problemas típicos de ML envolvem um fluxo que tem várias etapas:

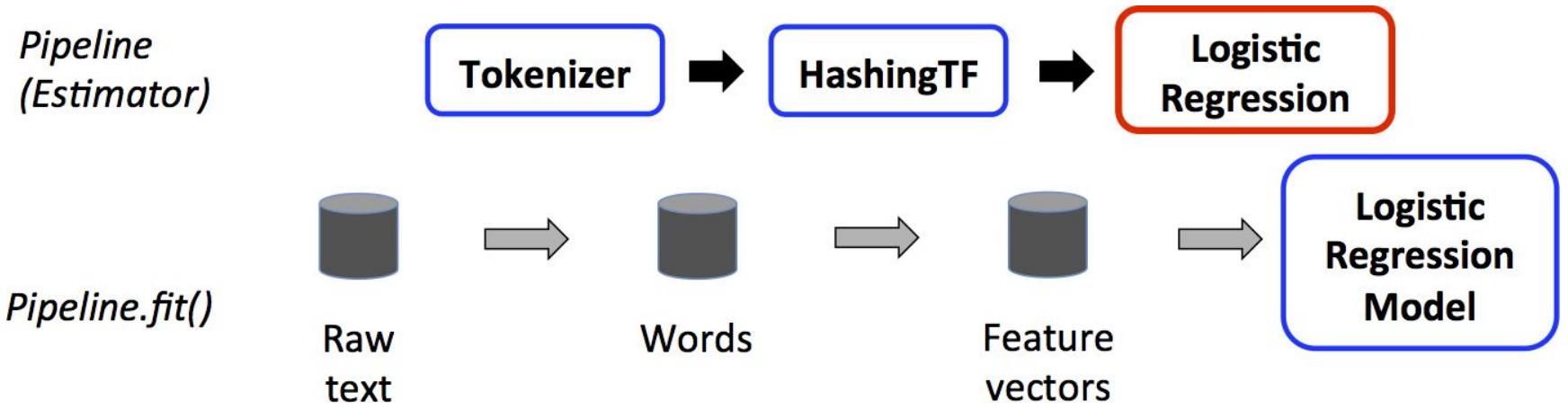
- Limpeza dos dados.
- Extração dos atributos.
- Ajuste dos parâmetros do modelo.
- Avaliação do modelo.

- A API spark.ml simplifica a criação de pipelines com múltiplos estágios:
    - API uniforme.
    - Possibilidade de personalizar etapas.

# ML e pipelines



# Exemplo típico de pipeline de ML

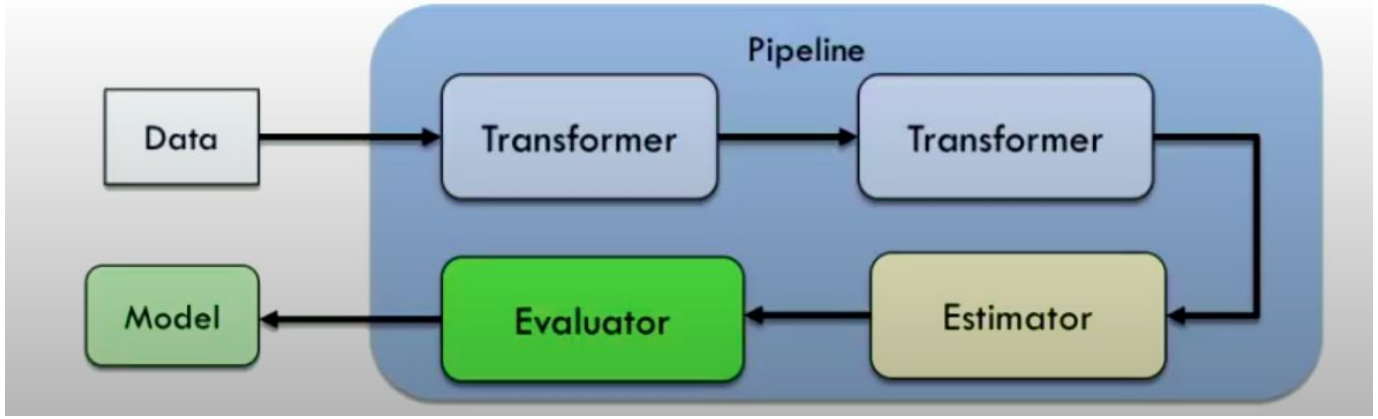


# Exemplo de Pipeline de ML

```
# Configure an ML pipeline, which consists of three stages: tokenizer, hashingTF, and lr.
tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
lr = LogisticRegression(maxIter=10, regParam=0.001)
pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])
```

# Um Pipeline contém *Transformers*

- Implementa *transform()*
- Transforma um *Dataframe* em outro *Dataframe*



# Vamos escrever código?