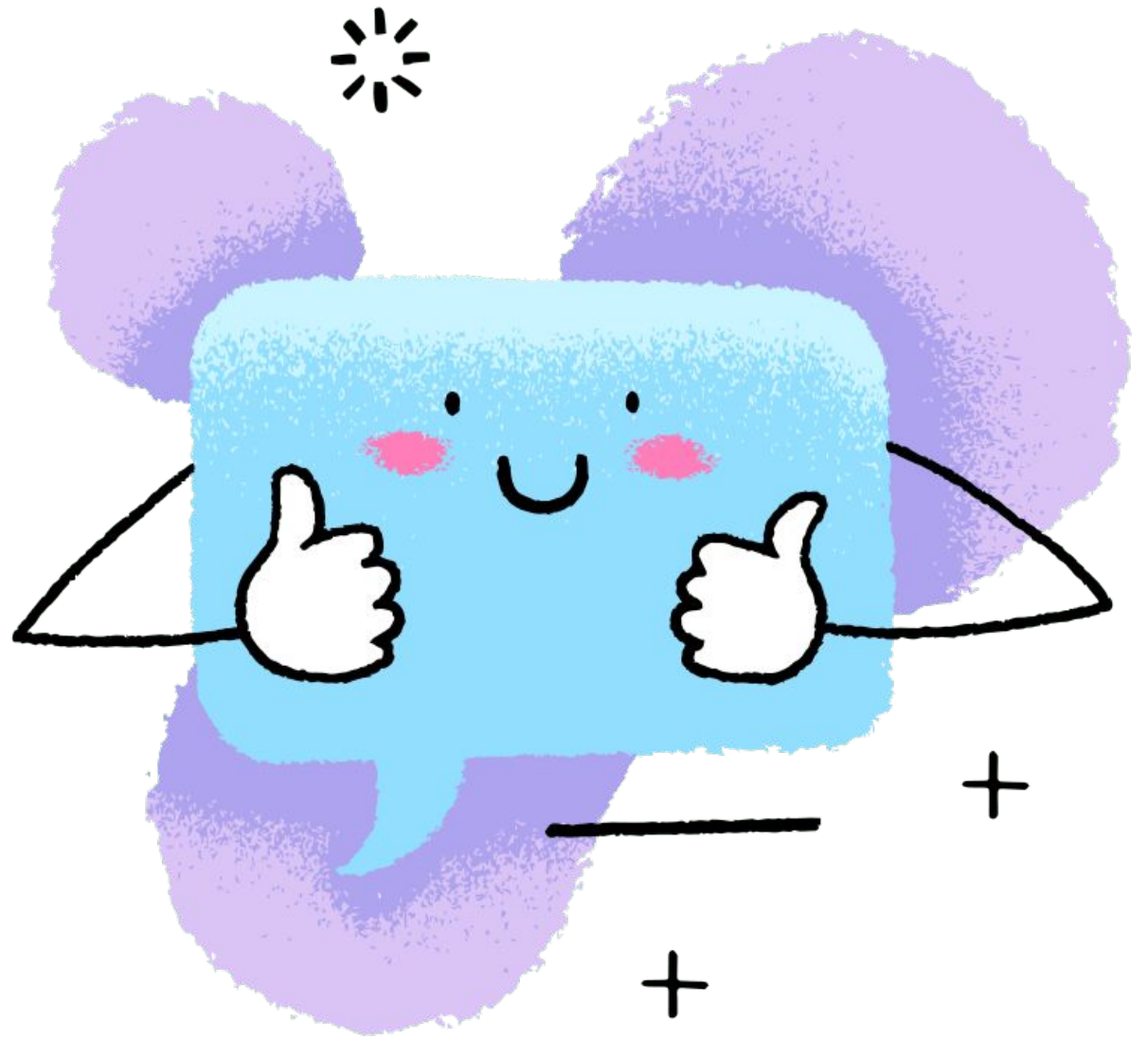


PySpark





Spark ML

Apache Spark

Spark ML vs MLlib



Spark MLlib

13

spark.mllib contains the legacy API built on top of RDDs.



Spark ML



spark.ml provides higher-level API built on top of DataFrames for constructing ML pipelines.

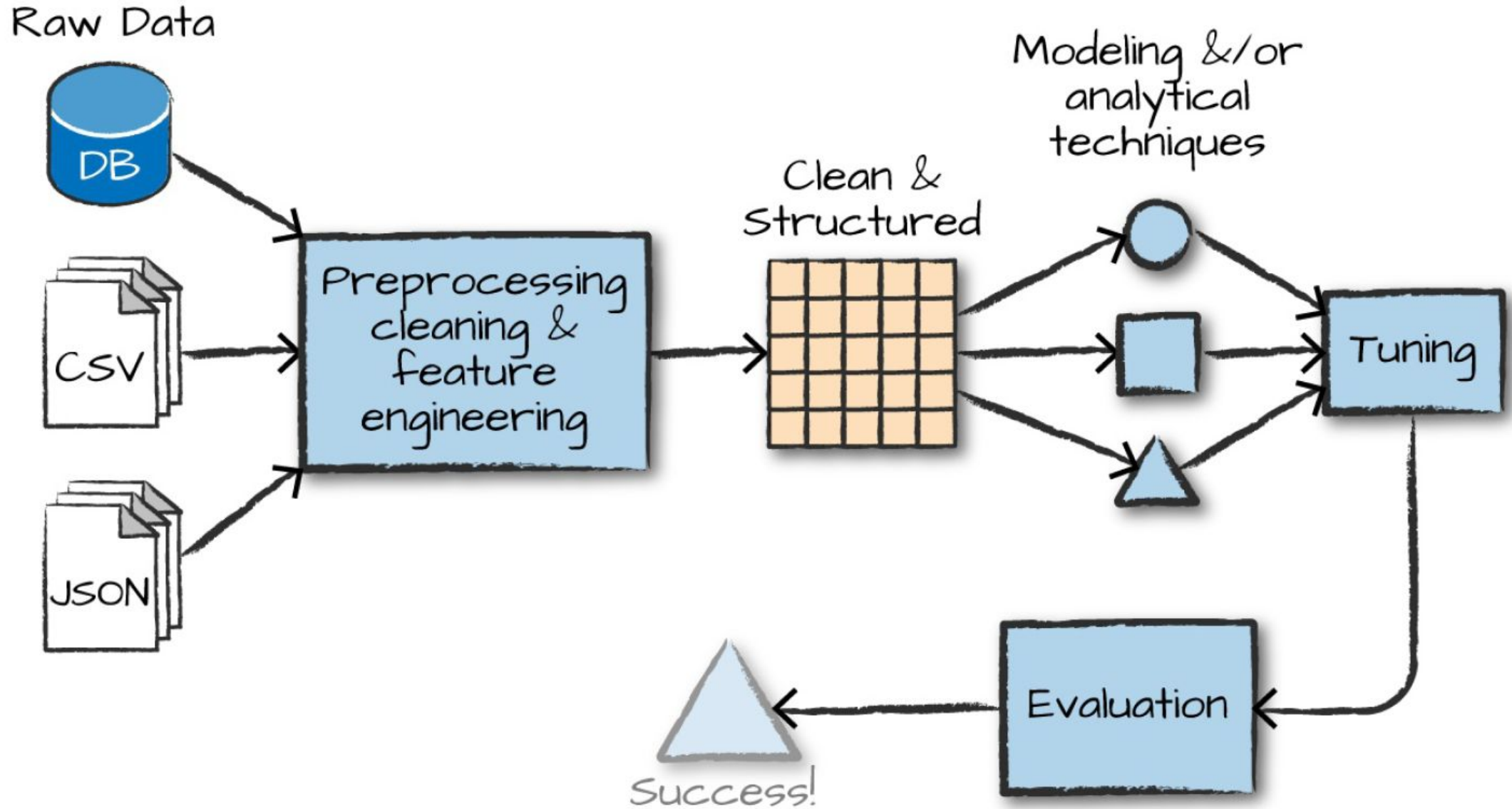
<https://stackoverflow.com/questions/38835829/whats-the-difference-between-spark-ml-and-mllib-packages>

4 причины строить ML-конвейеры, а не просто модели Machine Learning

- **чистый код** за счет автоматизации процедур подготовки данных – выборка, очистка, генерация предикторов (фичей, от англ. feature) и пр.;
- **сокращение ошибок** благодаря отработанной последовательности шагов, не получится пропустить или неправильно выполнить какой-то этап;
- **простота развертывания в production** – обычно преобразовать ML-модель от прототипа к масштабируемому и надежному решению для промышленной эксплуатации достаточно сложно, однако конвейеры помогут и здесь, облегчая тестирование и прочие MLOps-процедуры;
- **дополнительная проверка ML-модели** – можно применить перекрестную проверку (кросс-валидацию) и другие методы к этапам конвейера, пробуя различные параметры. Это ускоряет оптимизацию алгоритма и выбор наилучших конфигурационных настроек.

<https://www.bigdataschool.ru/blog/what-is-ml-pipeline-in-apache-spark-mllib.html>

The machine learning workflow



The machine learning workflow in Spark

Structured
APIs

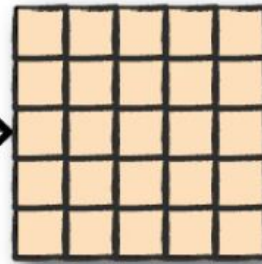
Raw Data



Preprocessing
cleaning &
feature
engineering

Transformers
& Estimators

Clean &
Structured



Estimators
& Models

Modeling &/or
analytical
techniques



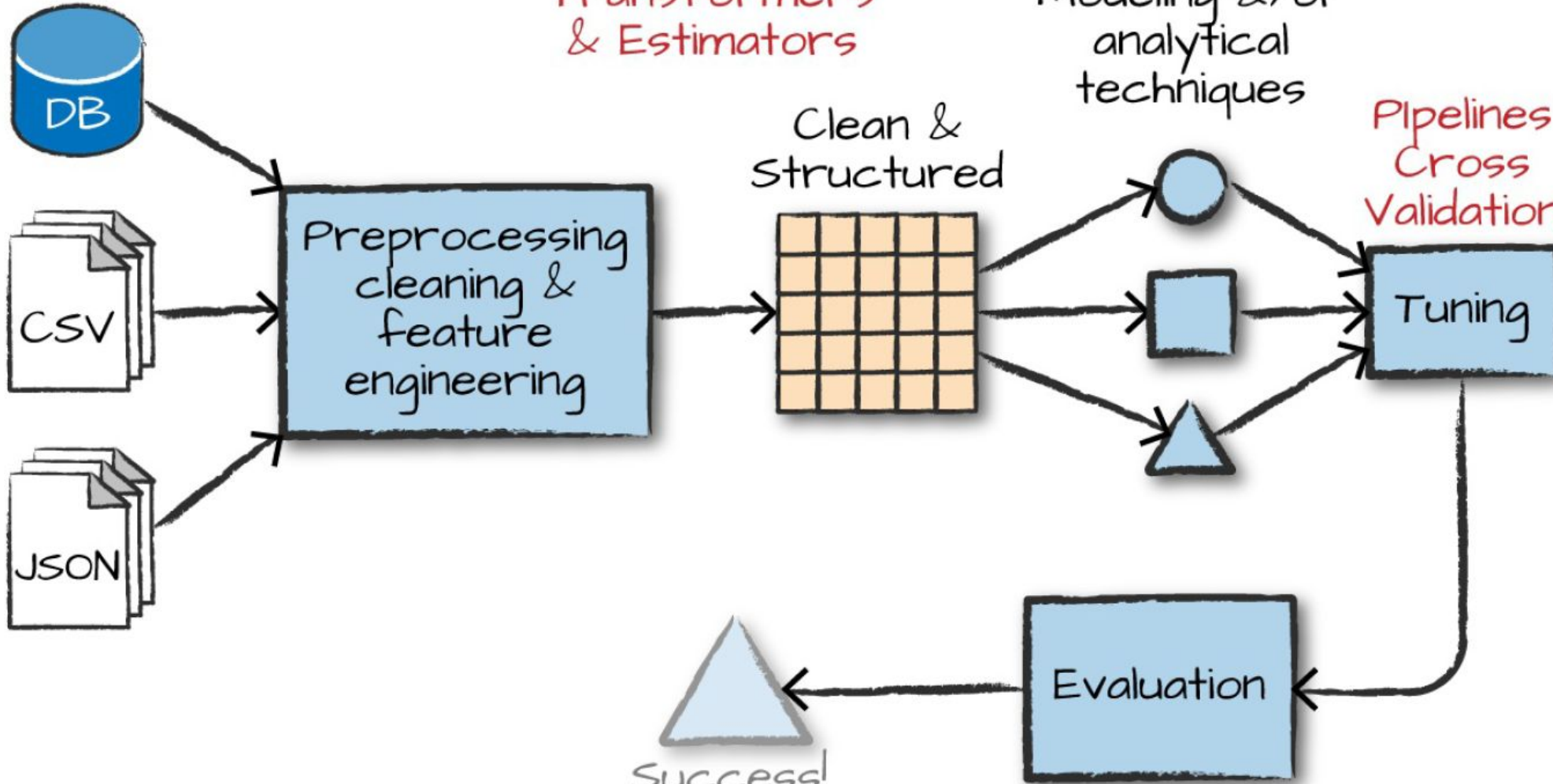
Pipelines
Cross
Validation

Tuning

Evaluation

Evaluators
Metrics

Success!

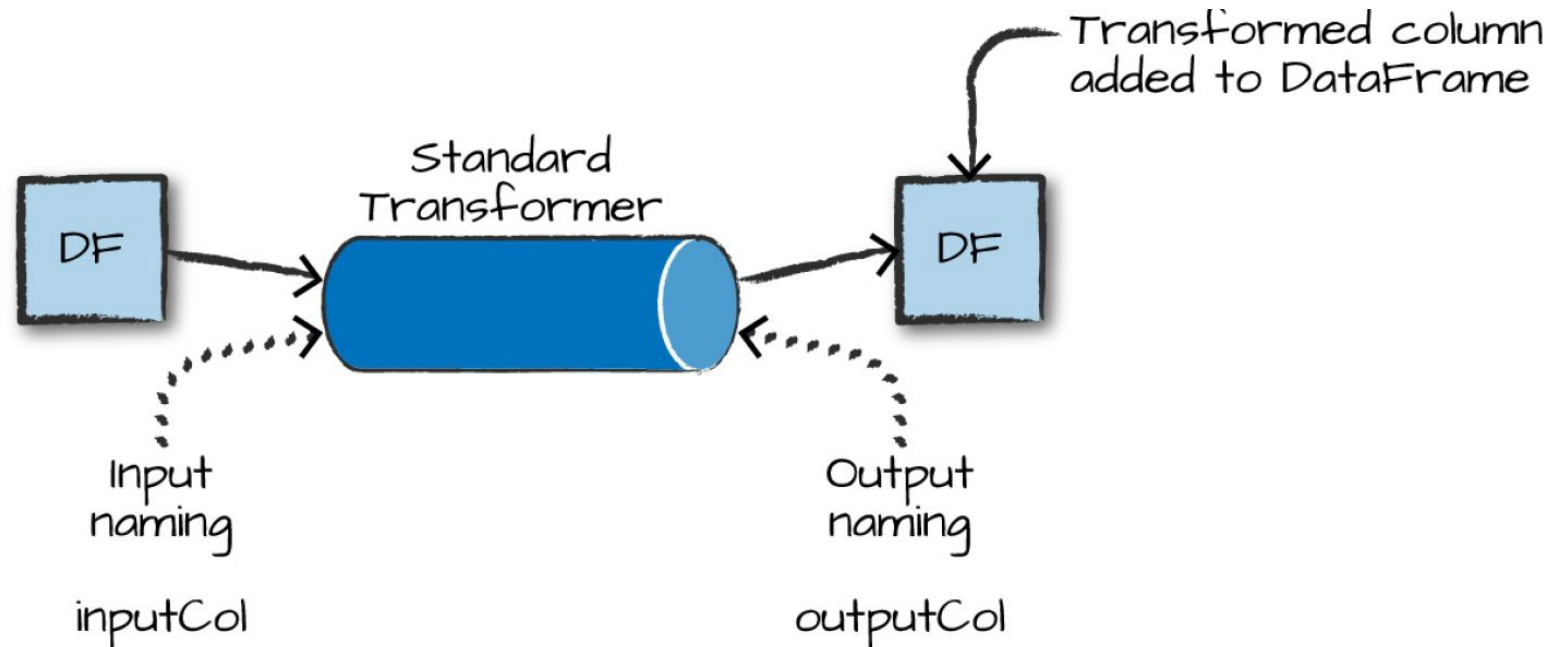


Pipeline components

Transformer and Estimator

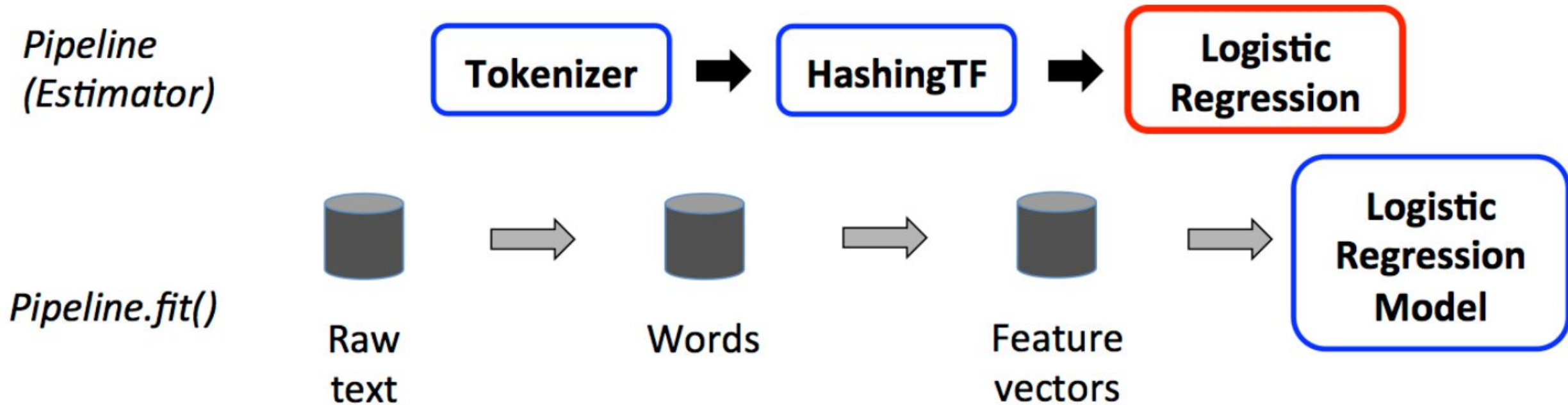
A Transformer is an abstraction that includes feature transformers and learned models

An Estimator abstracts the concept of a learning algorithm or any algorithm that fits or trains on data (`est.fit(df)` method)



A Pipeline

blue - transformations, red - estimator

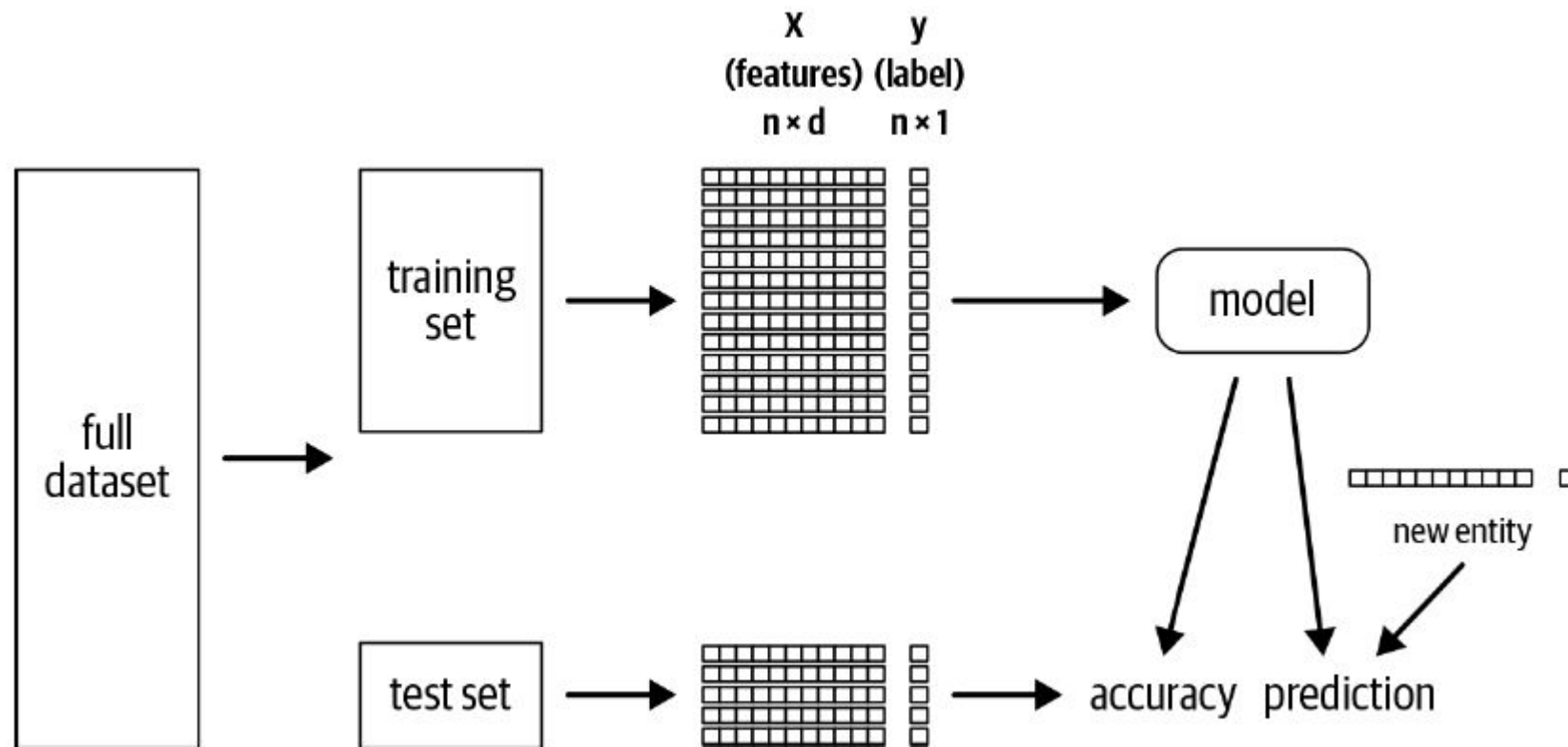


Sparse data types

Vector

```
1 from pyspark.ml.linalg import Vectors
2 denseVec = Vectors.dense(1.0, 2.0, 3.0)
3 size = 3
4 idx = [1, 2] # locations of non-zero elements in vector
5 values = [2.0, 3.0]
6 sparseVec = Vectors.sparse(size, idx, values)
```

Creating Training and Test Data Sets



Linear regression

The diagram shows the linear regression equation $Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$. Annotations include: a green arrow pointing from the text 'predictor, 'x-variable', independent variable, explanatory variable' to the variable x_1 ; an orange arrow pointing from the text 'coefficient' to the coefficient β_2 ; a blue bracket under the terms $\beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$ labeled 'linear predictor'; and a red arrow pointing from the variable Y to the text 'response, dependent variable, observation, 'y-variable''.

predictor, 'x-variable',
independent variable,
explanatory variable

coefficient

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

linear predictor

response, dependent variable,
observation, 'y-variable'

[Baby Steps Towards Data Science: Multiple Linear Regression in Python](#)

LogisticRegression

```
1 from pyspark.mllib.linalg import Vectors
2 from pyspark.ml.classification import LogisticRegression
3 from pyspark.ml.param import Param, Params
4
5 # Prepare training data from a list of (label, features) tuples.
6 training = sqlContext.createDataFrame([
7     (1.0, Vectors.dense([0.0, 1.1, 0.1])),
8     (0.0, Vectors.dense([2.0, 1.0, -1.0])),
9     (0.0, Vectors.dense([2.0, 1.3, 1.0])),
10    (1.0, Vectors.dense([0.0, 1.2, -0.5]))], ["label", "features"])
11
12 # Create a LogisticRegression instance. This instance is an Estimator.
13 lr = LogisticRegression(maxIter=10, regParam=0.01)
14 # Print out the parameters, documentation, and any default values.
15 print "LogisticRegression parameters:\n" + lr.explainParams() + "\n"
16
17 # Learn a LogisticRegression model. This uses the parameters stored in
18    lr.
19 model1 = lr.fit(training)
```


Спасибо!
Каждый день
вы становитесь
лучше :)

