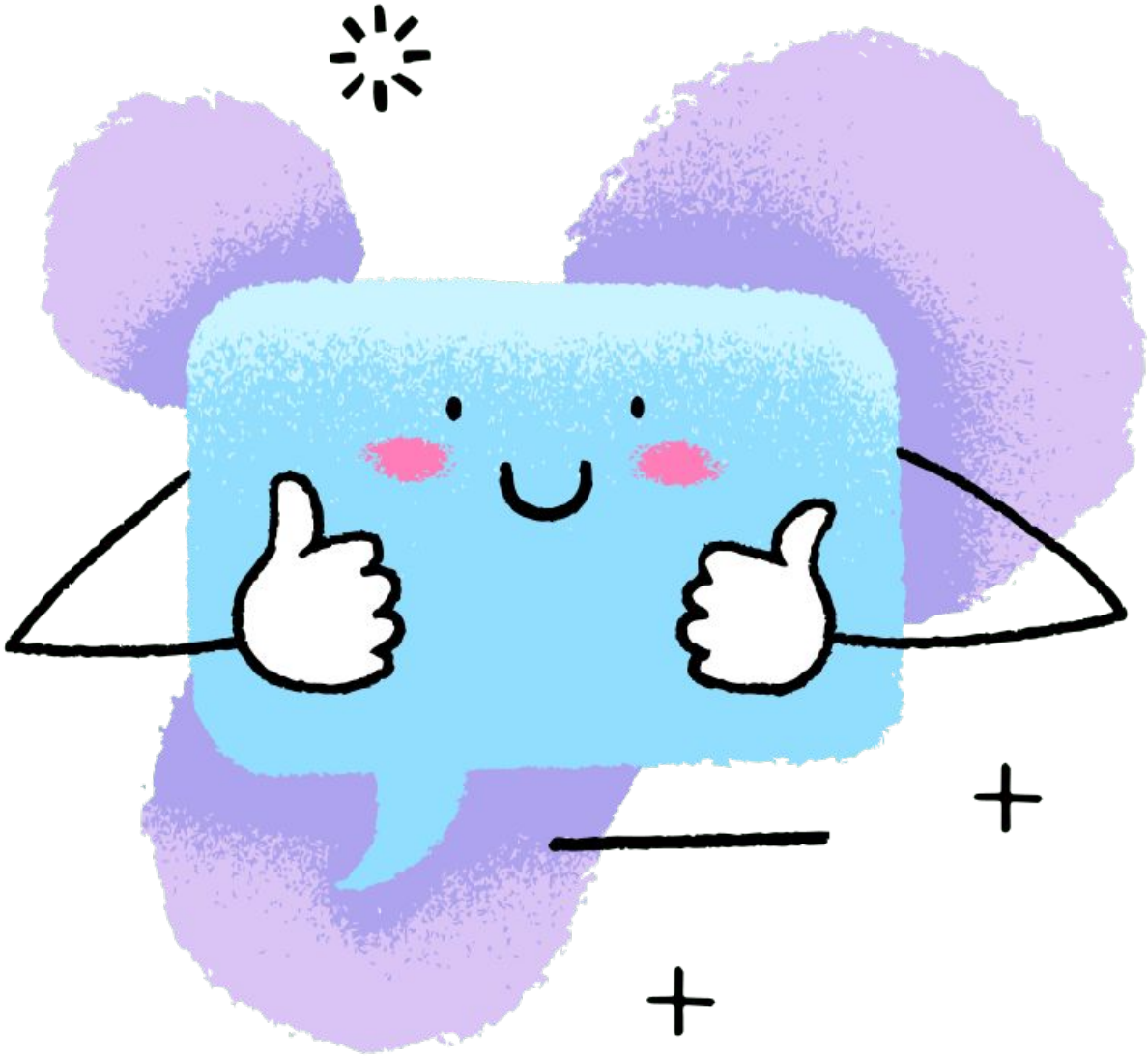
PySpark



GeekBrains

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.

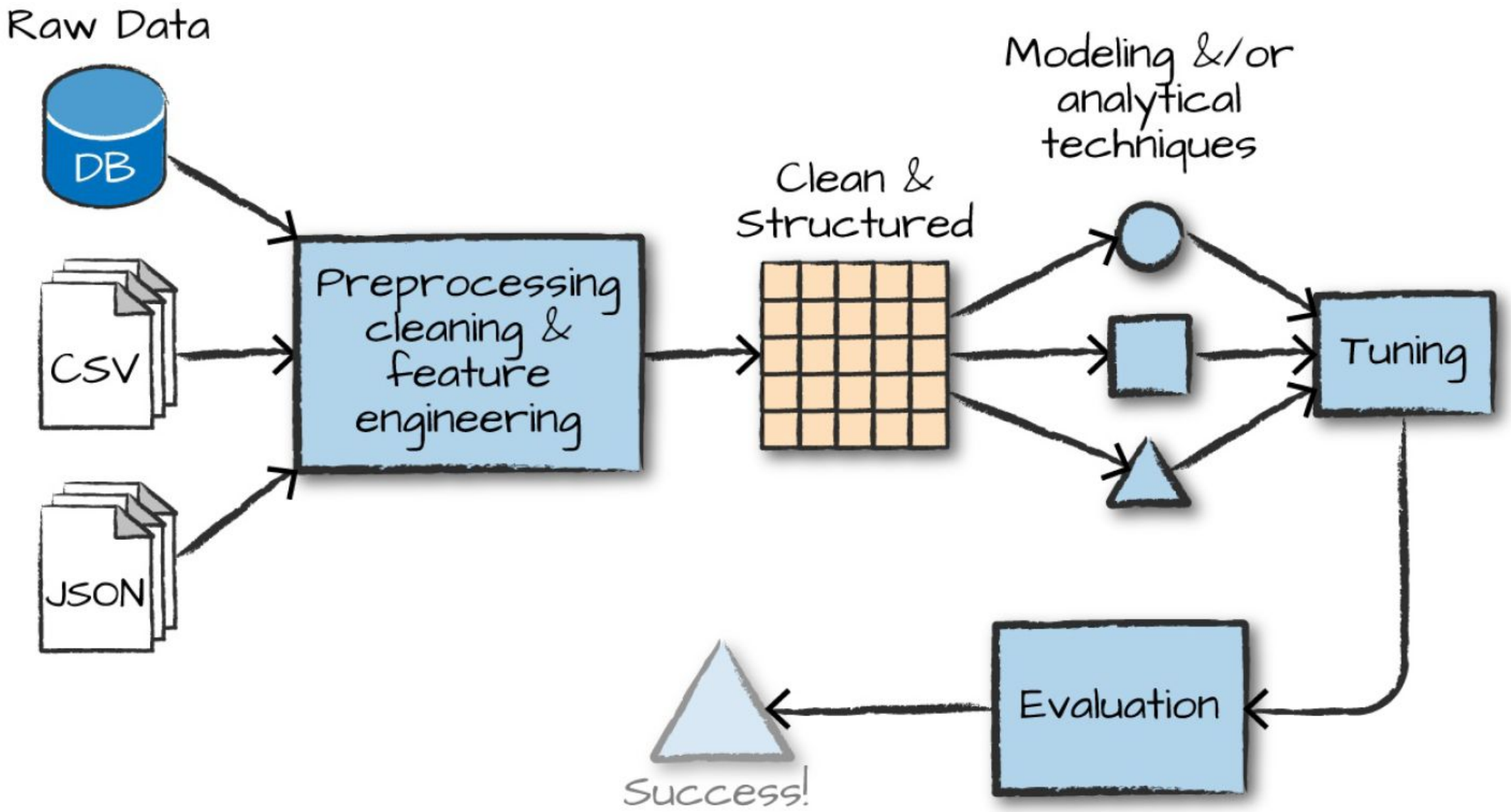
https://stackoverflow.com/questions/38835829/whats-the-difference-between-spark-ml-and-mllib-packages

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

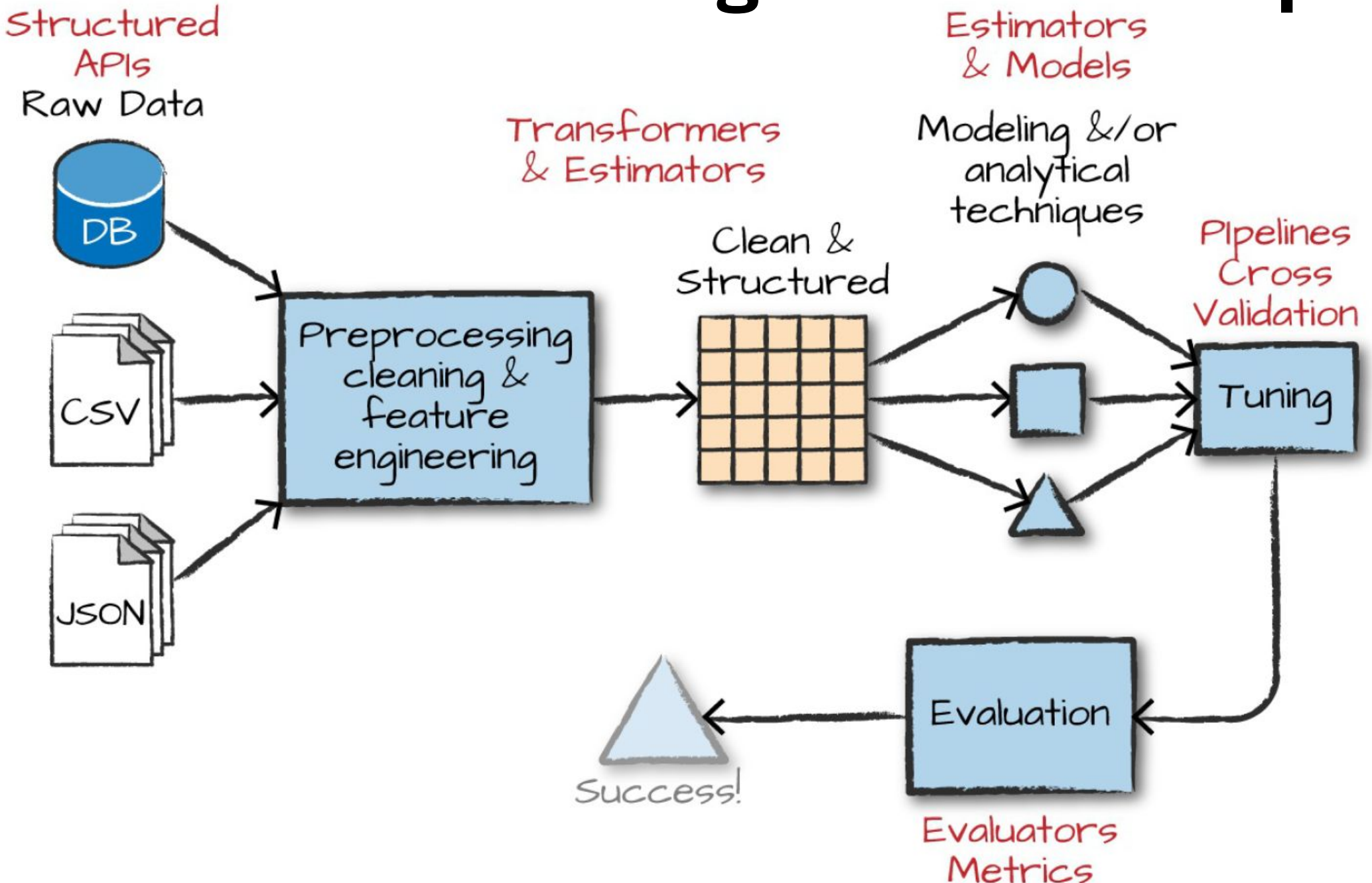
- ***чистый код*** за счет автоматизации процедур подготовки данных – выборка, очистка, генерация предикторов (фичей, от англ. feature) и пр.;
- ***сокращение ошибок*** благодаря отработанной последовательности шагов, не получится пропустить или неправильно выполнить какой-то этап;
- ***простота развертывания 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

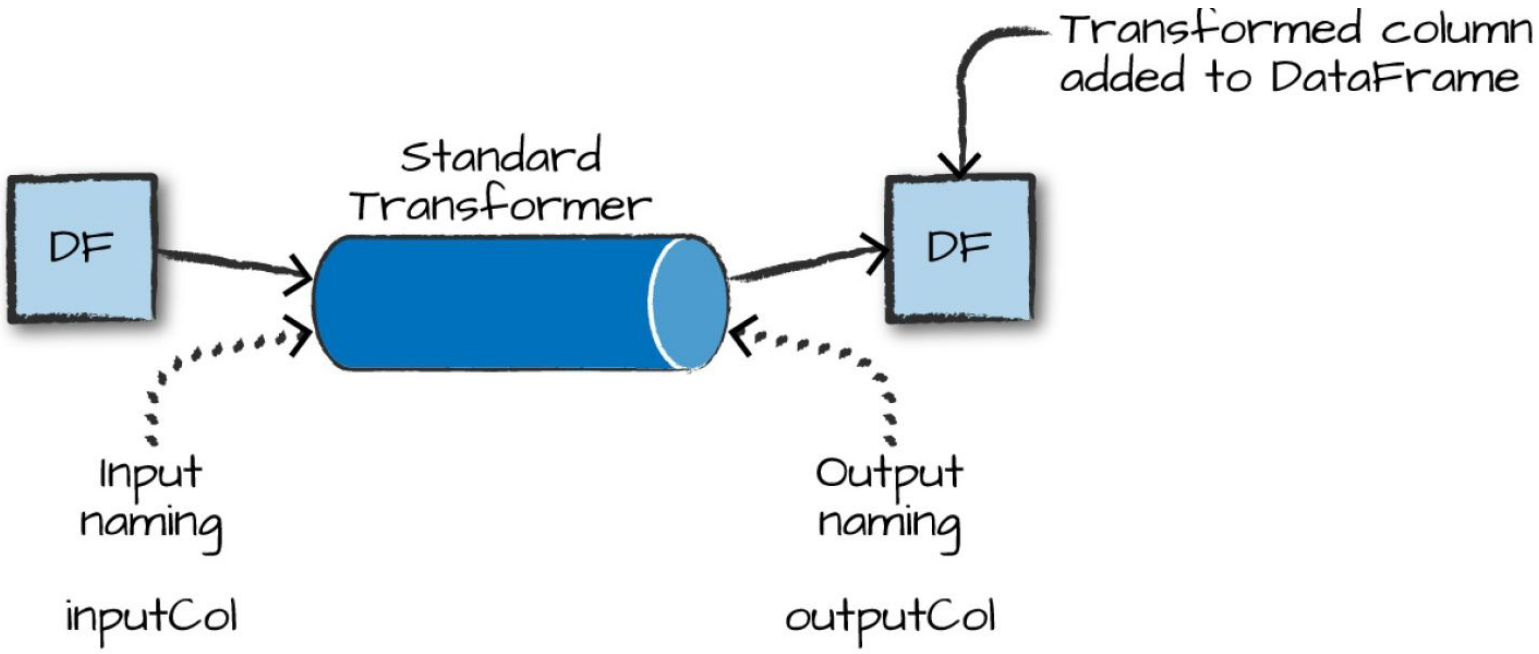


Pipeline components

Transformer and Estimator

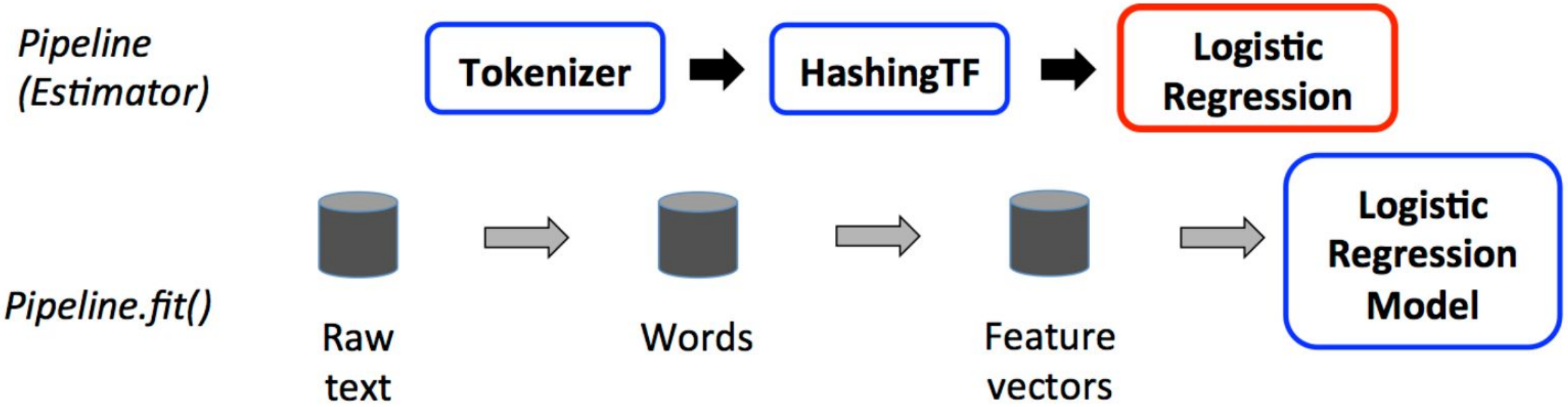
A Transformer is an abstraction that includes feature transformers and learned models

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

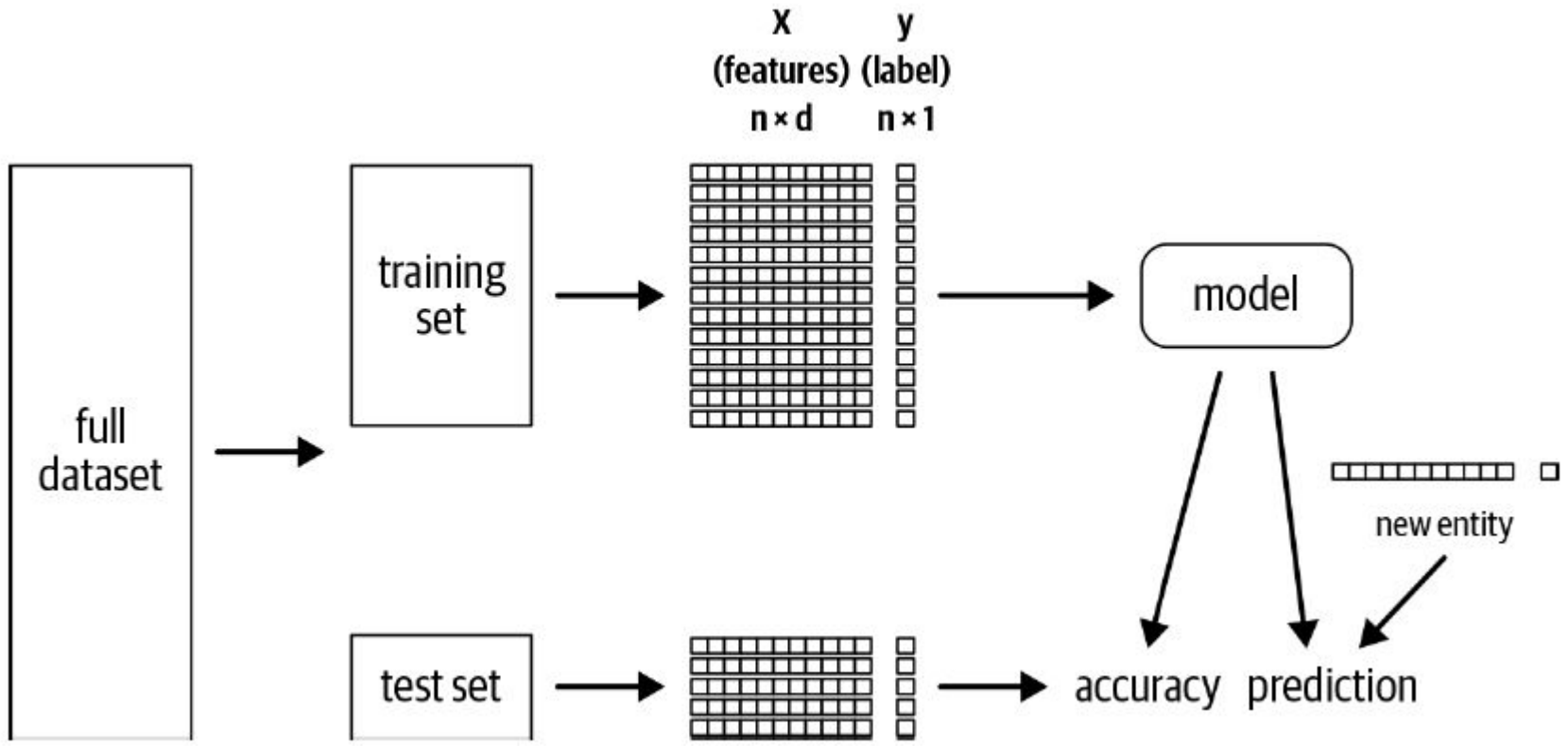


Sparse data types

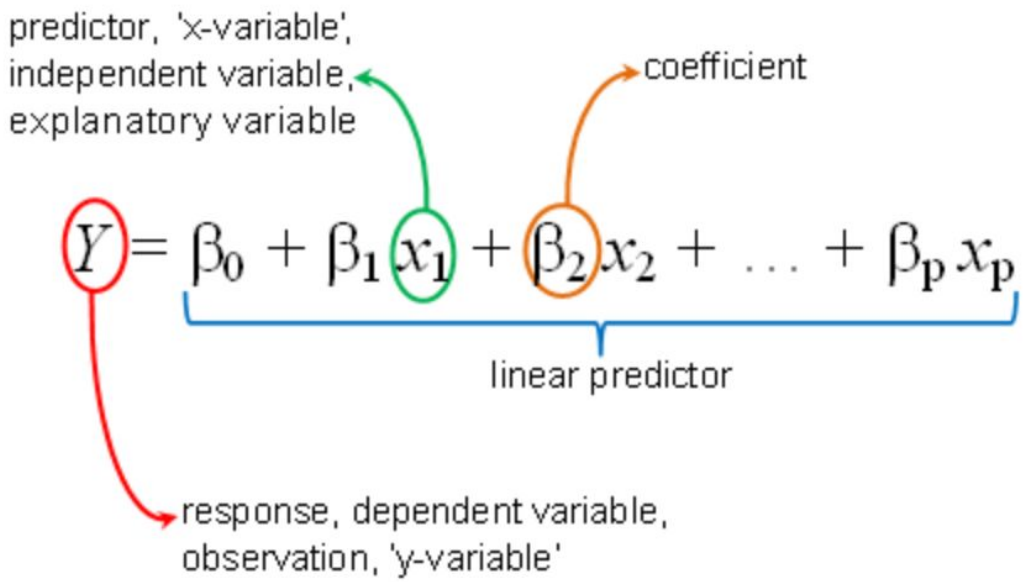
Vector

```
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



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

LogisticRegression

```
from pyspark.mllib.linalg import Vectors
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.param import Param, Params

# Prepare training data from a list of (label, features) tuples.
training = sqlContext.createDataFrame([
    (1.0, Vectors.dense([0.0, 1.1, 0.1])),
    (0.0, Vectors.dense([2.0, 1.0, -1.0])),
    (0.0, Vectors.dense([2.0, 1.3, 1.0])),
    (1.0, Vectors.dense([0.0, 1.2, -0.5]))], ["label", "features"])

# Create a LogisticRegression instance. This instance is an Estimator.
lr = LogisticRegression(maxIter=10, regParam=0.01)
# Print out the parameters, documentation, and any default values.
print "LogisticRegression parameters:\n" + lr.explainParams() + "\n"

# Learn a LogisticRegression model. This uses the parameters stored in
  lr.
model1 = lr.fit(training)
```

Спасибо!

Каждый день вы становитесь лучше :)