Machine Learning Library (MLlib) Guide

MLlib is Spark’s machine learning (ML) library. Its goal is to make practical machine learning scalable and easy. It consists of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as lower-level optimization primitives and higher-level pipeline APIs.

It divides into two packages:

- `spark.mllib` contains the original API built on top of RDDs.
- `spark.ml` provides higher-level API built on top of DataFrames for constructing ML pipelines.

Using `spark.ml` is recommended because with DataFrames the API is more versatile and flexible. But we will keep supporting `spark.mllib` along with the development of `spark.ml`. Users should be comfortable using `spark.mllib` features and expect more features coming. Developers should contribute new algorithms to `spark.ml` if they fit the ML pipeline concept well, e.g., feature extractors and transformers.

We list major functionality from both below, with links to detailed guides.

spark.mllib: data types, algorithms, and utilities

- Data types
- Basic statistics
  - summary statistics
  - correlations
  - stratified sampling
  - hypothesis testing
  - streaming significance testing
  - random data generation
- Classification and regression
  - linear models (SVMs, logistic regression, linear regression)
  - naive Bayes

spark.ml: high-level APIs for ML pipelines

Some techniques are not available yet in spark.ml, most notably dimensionality reduction Users can seamlessly combine the implementation of these techniques found in `spark.mllib` with the rest of the algorithms found in `spark.ml`.

Dependencies

MLlib uses the linear algebra package Breeze, which depends on netlib-java for optimised numerical processing. If natives libraries[1] are not available at runtime, you will see a warning message and a pure JVM implementation will be used instead.

« Due to licensing issues with runtime proprietary binaries, we do not include `netlib-java`'s native proxies by default. To configure `netlib-java` / Breeze to use system optimised binaries, include `com.github.fommil.netlib:all:1.1.2` (or build Spark with `-Pnetlib-lgpl`) as a dependency of your project and read the netlib-java documentation for your platform's additional installation instructions.

To use MLlib in Python, you will need NumPy version 1.4 or newer.

Migration guide

MLlib is under active development. The APIs marked `Experimental`/`DeveloperApi` may change in future releases, and the migration guide below will explain all changes between releases.

From 1.5 to 1.6

There are no breaking API changes in the `spark.mllib` or `spark.ml` packages, but there are deprecations and changes of behavior.

Deprecations:

- SPARK-11358: In `spark.mllib.clustering.KMeans`, the `runs` parameter has been deprecated.
- SPARK-10592: In `spark.ml.classification.LogisticRegressionModel` and `spark.ml.regression.LinearRegressionModel`, the `weights` field has been deprecated in favor of the new name `coefficients`. This helps disambiguate from instance (row) "weights" given to algorithms.

Changes of behavior:

- SPARK-7770: `spark.mllib.tree.GradientBoostedTrees`: `validationTol` has changed semantics in 1.6. Previously, it was a threshold for absolute change in error. Now, it resembles the behavior of `GradientDescent`'s `convergenceTol`: For large errors, it uses relative error (relative to the previous error); for small errors (`< 0.01`), it uses absolute error.
- SPARK-11069: `spark.ml.feature.RegexTokenizer`: Previously, it did not convert strings to lowercase before tokenizing. Now, it converts to lowercase by default, with an option not to. This matches the behavior of the simpler `Tokenizer` transformer.

«

Previous Spark versions

Earlier migration guides are archived on this page.

---

1. To learn more about the benefits and background of system optimised natives, you may wish to watch Sam Halliday's ScalaX talk on High Performance Linear Algebra in Scala. ↩