Comparing Scikit-Learn with PySpark

Villalon Francis, 8866016
University of Wollongong

Comparing Scikit-Learn with PySpark

PySpark is the python API for Apache Spark, a distributed computing engine designed for processing large-scale datasets across clusters. In contrast, scikit-learn is a widely used python machine learning library for classical machine learning on a single machine. Throughout CSCI316, we have seen extensive use of Scikit-learn and PySpark in our assignments. This brief report consolidates those experiences, compares the two ecosystems, and examines their strengths, limitations, similarities, and differences.

PySpark

Pros/Strengths

Scalability and Performance. PySpark has distributed processing across clusters which enables handling of very large datasets that exceed single-machine memory. This is a significant advantage over Scikit-learn which was designed to be single-node and in-memory. Therefore, making PySpark a necessity over Scikit-learn when handling datasets that are in the TB-scale.

Native Connectors to data lakes/warehouses PySpark has native connectors to data lakes and warehouses such as S3 (Simple Storage Service) and JDBC (Java Database Connectivity. It also has columnar formats such as Parquet, which was used in our group assignment task 2 to store the processed data, and ORC (Optimized Row Columnar) to keep processing close to the data. In contrast to Scikit-learn which requires the data to be parsed/extracted in pandas/NumPy.

Cons/Limitations

Debugging and Developer Experience. Debugging is harder as stack traces mix with Python and JVM/Scala errors, complicating diagnosis. Data quality issues can be harder to locate across partitions. In the group project, we had a docker container to host a single PySpark instance to run the necessary code for task 2. We had to keep a terminal open to track the logs of the spark instance when debugging as the errors shown in our IDE (Integrated Developer Environment) may only show “Connection was refused” as an error. Resulting, in a need for us to check the spark instance logs to determine what was happening. Furthermore, there are no built-in verbosity/progress options in PySpark that show the algorithm-level training progress. This overall was a more complicated experience to debug and track training as compared to scikit-learn.

Algorithm Coverage and ML Workflow. PySpark has a narrow set of built-in algorithms as compared to scikit-learns extensive classical ML catalog and utilities. More advanced models or customized neural networks are not natively available on Spark. As a result, the task 2 of the group assignment used 2 Tree Ensemble models out of 3. These models are

Gradient Boosted Trees and Random Forest. The 3rd model was simple Linear Regression. Trees handle distributed workflows and parallelization very well and as a result, tree-based approaches are prominent in the regression models natively available in PySpark's MLlib.

Scikit-Learn

Pros/Strengths

Simplicity and Developer Velocity. Scikit-learn runs locally without clusters and is ideal for notebooks and rapid iteration. Its clean and consistent API makes experimentation fast and readable. The errors are local and tracebacks are straightforward as compared to distributed job stacks. As a result, scikit-learn is the more flexible option as compared to PySpark. For the group assignment, the simplicity of Scikit-Learn enabled better collaboration as it was easier to setup and is more flexible than PySpark for the intended goal of the assignment.

Rich Algorithm and Utility Coverage. PySpark is designed primarily for distributed workloads and as a result, it does not have the same broad catalog of classical ML algorithms as Scikit-learn. Scikit-learn provides more options to developers to approach any ML problem as well as native feedback during model fitting in the console or notebook environment. For the group assignment in particular, the native feedback of Scikit-Learn in task 1 during model fitting proved a significant advantage over PySpark in task 2 as it allowed for greater visibility at the algorithm-level during training. We can more closely inspect the model while it was fitting as well as determine how much longer the model fitting would take.

Cons/Limitations

No Native Connectors to data lakes or data warehouses. Scikit-Learn does not have any native connectors to data lakes or data warehouses like PySpark. As a result, all data that is being handled by the library must first be extracted into pandas or NumPy. This requires the data to either be small enough to fit comfortably in RAM on one machine or additional effort must be invested to practically handle larger datasets. While this is a significant limitation of Scikit-Learn, it is by design and the library is intended for medium-scale supervised and unsupervised problems. For all assignments in CSCI316, Scikit-Learn is the most suitable and flexible option due to the scale of data we are working with.

No Native Distributed Training. Scikit-Learn does not natively parallelize model fitting across a cluster; scaling beyond one machine requires external frameworks or substantial engineering. In contrast, PySpark has built-in distributed execution.

Similarities & Differences

Similarities

Pipeline-based workflows and algorithms for a wide range of machine learning problems. Both PySpark and Scikit-Learn provide developers with a wide array of solutions to implement for machine learning paradigms like regression and classification. They both also have Pipeline abstractions that allow the developer to chain preprocessing and modeling steps with consistent fit/transform/predict semantics.

Feature Engineering. Both include transformers for scaling, encoding, and feature assembly, plus utilities for model selection and evaluation.

Common Metrics. Both have evaluators or built-in tools that support common metrics.

Differences

Scale of Data. PySpark purpose-built for large-scale datasets that don't fit in a single machine's memory, enabling distributed processing across clusters for industrial-grade workloads. In contrast, Scikit-Learn is optimized for small-to-medium datasets that fit comfortably in memory. Scikit-Learn does not natively support distributed training but is often more flexible and is faster than PySpark at a smaller scale due to how it avoids distributed overhead.

Execution Model. PySpark runs on a distributed engine across a cluster; operations are parallelized and executed lazily as Spark jobs/stages. On the other hand, Scikit-Learn is single-machine, in-memory computation; parallelism is typically limited to cores on one box unless significant engineering efforts are invested to allow for training across more than one machine.

Algorithm Coverage. PySpark has a focused set of scalable algorithms designed for distributed training. It does not provide nearly the same breadth of classical ML algorithms that Scikit-Learn natively provides.