MVC for data science is here

Abhishek Tiwari

Citation: *A. Tiwari*, ”MVC for data science is here”, Abhishek Tiwari, 2015.
doi:10.59350/39zj2-zx686

Published on: April 02, 2015

For those who are not familiar with MVC (Model–view–controller), MVC is a software architectural pattern primarily used for web development and to implement user interfaces. Ruby on Rails (RoR), Django, Spring MVC, ASP .net, Symfony, Zend, Express are some of the most popular MVC frameworks currently available for rapid web development. Over the time web development community has immensely benefited from these MVC frameworks. So much that a TODO list application has become something of a poster child for these MVC frameworks. In a nutshell, MVC frameworks have: 1) standardised the web development; 2) lowered the web development learning curve, and; 3) accelerated the web development and release cycle.

MVC for data science

Data science as it stands certainly need it's own MVC moment, and it starts with machine learning. PredictionIO - An open-source machine learning server made an early attempt in this direction using a DASE architecture which includes D (Data source/data preparator), A (algorithms), S (serving) and E (evaluator) components. Like MVC which has modularise the model, view and controller components - PredictionIO's DASE architecture helps you to modularise DASE components so developers and data scientist can build deployable prediction engines more quickly. In addition, in line with MVC components, DASE components maintain clear separation-of-concerns. This allows data scientists to swap and evaluate algorithms as they wish without impacting the rest of platform typically managed by the developers. DASE modularity is also appealing for developers. It offers developers flexibility to re-use the codebase and integrate prediction engine with a variety of applications using APIs.

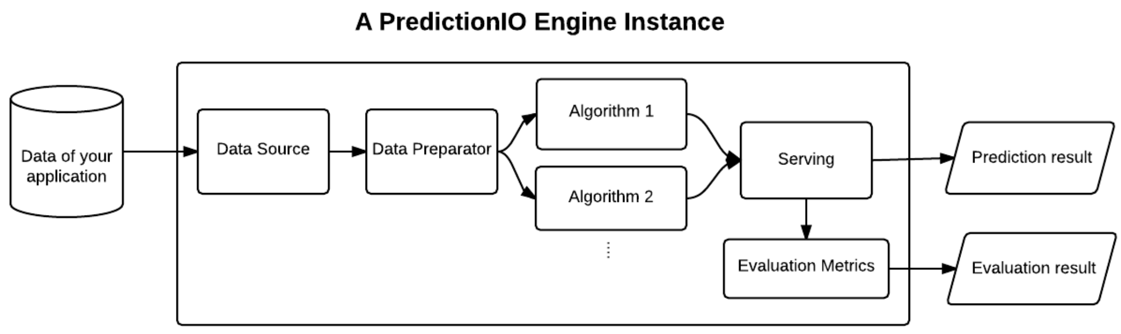

Figure 1: Prediction engine using modular DASE components. Image credits PredictionIO.

Engine

The engine is responsible for making predictions. A prediction engine includes all DASE components into a deployable state by specifying - A data source component - reads data from an input source and transforms it into a desired format - A data preparator component - preprocesses the data and forwards it to the algorithm for model training - One or more algorithm(s) - outputs trained models with fine tuned parameters - A serving component - takes prediction queries, feed into models and returns prediction results

An engine performs following key functions,

- Produces a trained model using the training data set
- Allows external application to interact via web services
- Respond to prediction query in real-time

One engine predicts only one thing.

> Each engine processes data and constructs predictive models independently. Therefore, every engine serves its own set of prediction results. For example, you may deploy two engines for your mobile application: one for recommending news to users and another one for suggesting new friends to users.

Overall ecosystem

Although the engine is a core part of PredictionIO and focus of this article, the overall ecosystem consists 3 key building blocks,

1. Event Server - collects data from your application, in real-time or in batch
2. Engine - one or more engines responsible for its own set of prediction results
3. Template Gallery - highly customisable engine templates for all kinds of prediction tasks

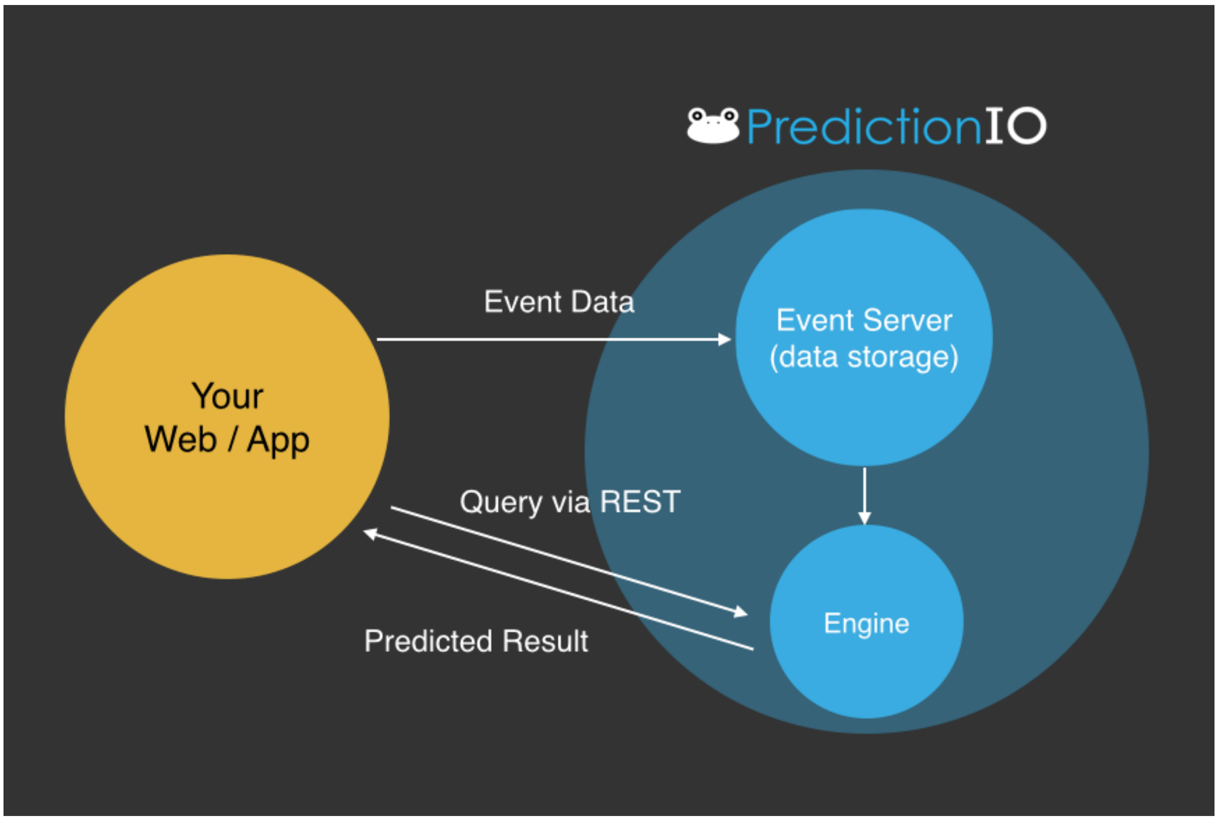

Figure 2: Application, event server and engine. Image credits PredictionIO.

External applications such as mobile or web apps,

- Push the event data into event server, which is used for model training and evaluation
- Poll the predicted results from engine by sending a prediction query to serving component

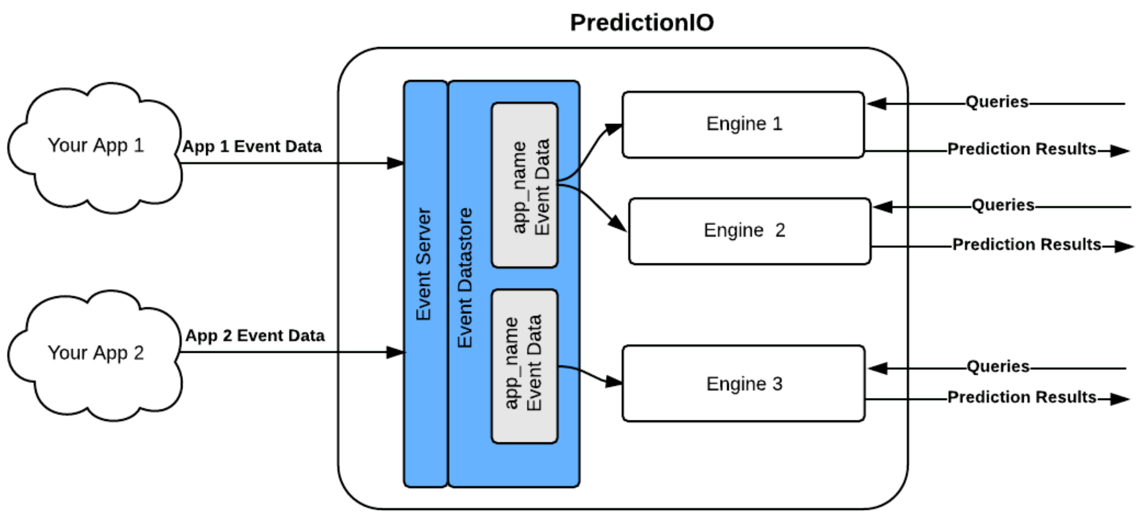

Figure 3: Integrating PredictionIO with your application. Image credits PredictionIO.

Technology stack

- The engine is written in Scala and APIs are built using Spray
- The engine is built on top of Apache Spark
- Event Server uses Apache HBase as the data store
- Trained model is stored in HDFS (part of Apache Hadoop)
- Model metadata is stored in ElasticSearch

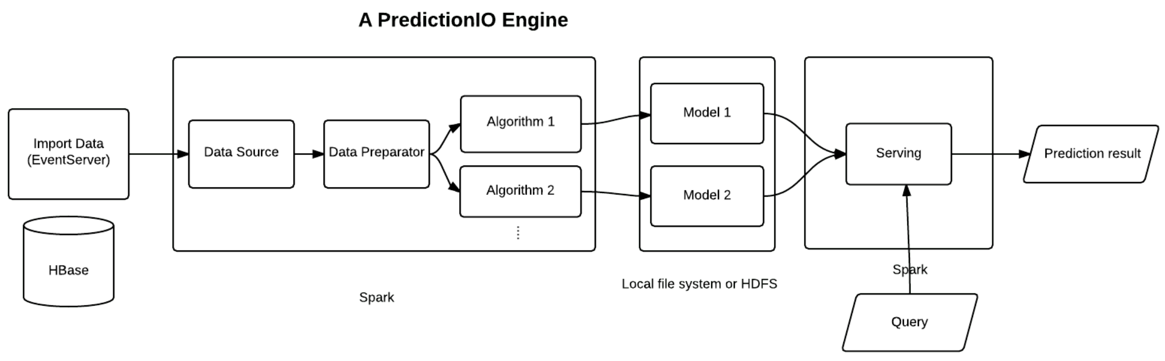

Figure 4: PredictionIO technology stack. Image credits PredictionIO.