MODEL MANAGEMENT SYSTEMS: SCIKIT-LEARN & DJANGO

By Benjamin Bengfort, Laura Lorenz, and Rebecca Bilbro

PyCon 2017 Portland, Oregon

DistrictDataLabs

pycon.districtdatalabs.com

Data products are not single machine trained models, but are instead a rich tapestry of models that influence each other, interact and are ranked, age, and eventually perish.

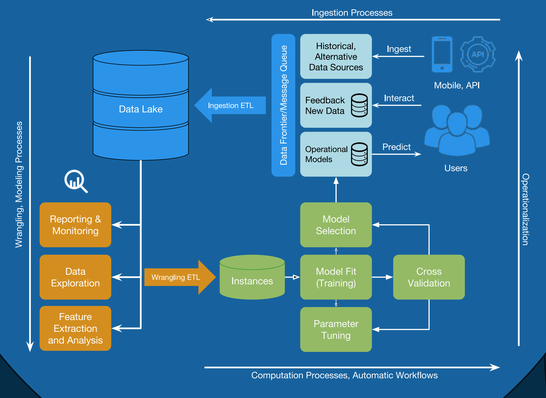
Managing and monitoring the model life cycle as data is obtained and models are automatically fit is critical to producing smooth interactions between users and the application.

MODEL STORAGE (ALSO "MODEL MANAGEMENT")

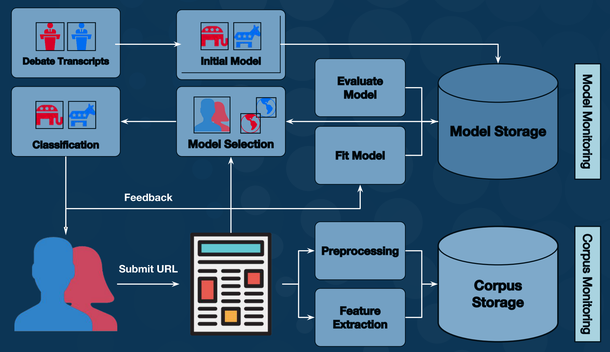
Additionally, models are also stored in the database as pickles, and can be retrieved and loaded by the web application.

ID	Model	Hyperparameters	Build Time	Score	Pickle
1	Naive Bayes	{"alpha": 1.0}	235.32	.832	BLOB
2	SVC	{"C": 1.0, "kernel": "linear"}	20.312	.861	BLOB
3	KNN	{"k": 5, "weights": "distance"}	482.129	.821	BLOB

So how do we architect data products?



In this poster, we propose a Model Management System (similar to a content management system) that integrates Scikit-Learn and Django to provide insight into how applications behave over time.



Step 1: Train an initial model on a relevant dataset.

Step 2: Employ the model and have users interact with it; users create additional data.

Step 3: Monitor corpus for growth and watch for triggers to retrain the model (e.g. size, time).

Step 4: Once a trigger happens, move to offline process: snapshot data, train one or more Scikit-Learn models using 12-fold cross-validation, test on full dataset, add new model to system.

Step 5: On query, Django selects best model for the user, makes predictions, and renders their views. Can be per-user, group-based (e.g. experts), or globally trained. Can be used application-side or with boosting.

Step 6: Monitor models over time to look for decay (using precision and recall, for example).