MODEL MANAGEMENT SYSTEMS: SCIKIT-LEARN & DJANGO

By Benjamin Bengfort, Laura Lorenz, and Rebecca Bilbro

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.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Score 1</th>
<th>Score 2</th>
<th>Score 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Model 1</td>
<td>0.83</td>
<td>0.75</td>
<td>0.65</td>
</tr>
<tr>
<td>2</td>
<td>Model 2</td>
<td>0.90</td>
<td>0.85</td>
<td>0.78</td>
</tr>
<tr>
<td>3</td>
<td>Model 3</td>
<td>0.87</td>
<td>0.79</td>
<td>0.68</td>
</tr>
</tbody>
</table>

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 model applications between users.

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.

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

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

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

Step 4: Once a trigger happens, move to offline process: upload data, train one or more SciKit-Learn models using 5-fold cross-validation, test on full dataset, and return model to system.

Step 5: On query, Django creates a best model for the user, makes predictions, and renders their views. Can be person, group based (e.g. experts), or globally trained. Can be used application-wise or with bundling.

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