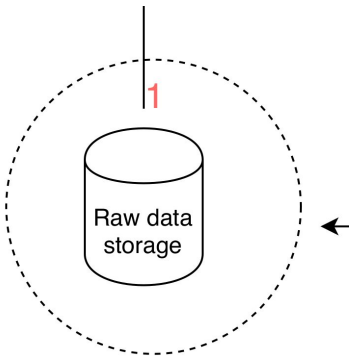
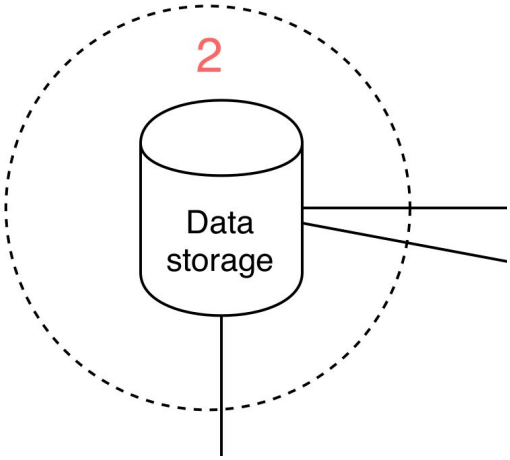
Hi,

The question about architecture is really open-ended and doesn't give much details about the features, output or how will the predictions be used. I have taken some liberty and it might not be what you expect. However, I will be more than happy to really drill down into the specifics of this or any other ml system problem if given an opportunity.



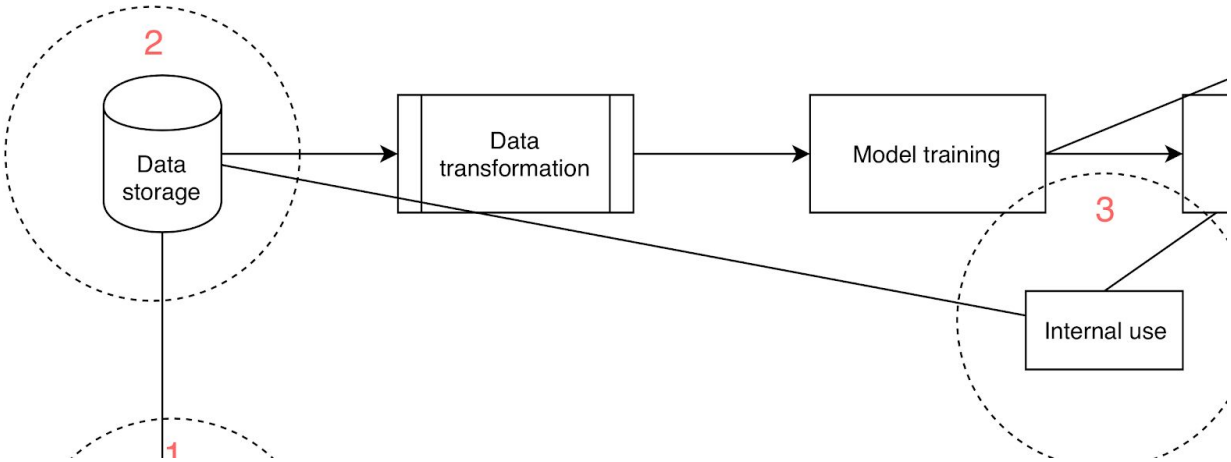
1. **Raw Data Storage**

Store on Amazon. Formats will be different for different types of dataset. No real structure is expected. Example - Common Crawl data, http://commoncrawl.org/the-data/get-started/



2. Data storage

This is the processed version of raw data. It is more structured and sophisticated than the raw data. However, we haven't made features out of them yet. This will be the main data source for machine learning models. We can store it on a separate bucket on AWS. We may use mysql, mongo, cassandra, etc. for this, depending upon the constraints of the type of data.



2B. Feature pipeline

Once we have the data, we need to transform it into features and target. I have used only Scikit-learn for building pipelines. However, large scale systems might use spark. Depending upon the size of the data, we may need to build distributed system to process the data.

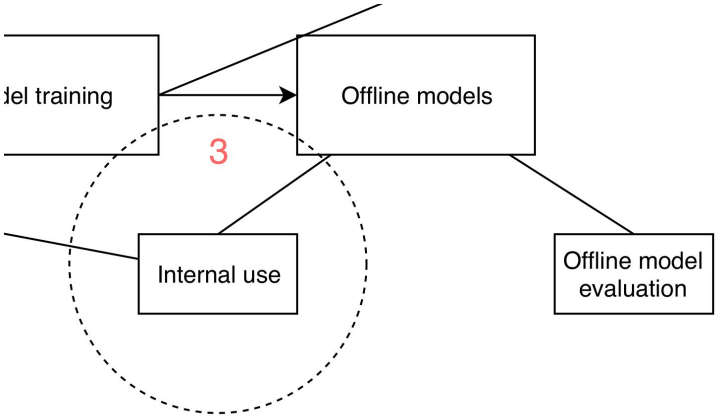
Here I will assume size of data to be manageable without distributed system and that we don't need online learning (we can use batch processing).

Training of the model takes place on an AWS training instance. This training instance loads the data, transforms it, trains the model and then put it on a separate AWS bucket which could be downloaded later for internal use as well.

For training, we can use PyTorch, MxNet, Tensorflow or any other mainstream python library.

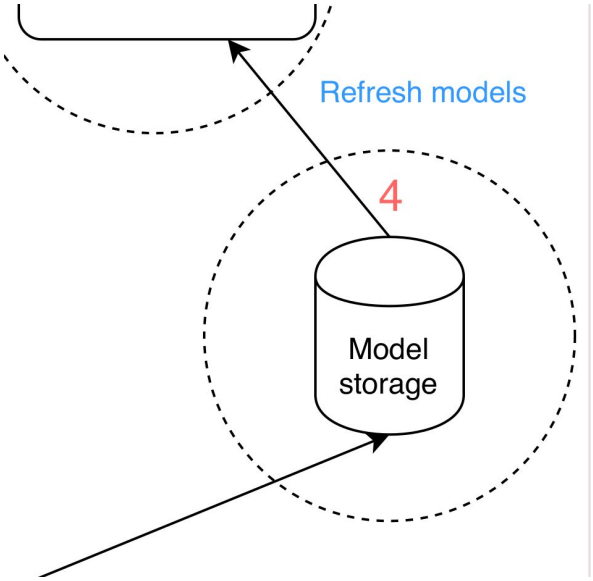
Each training/retraining is triggered by another process using airflow. Intervals between retraining depends upon time taken to train the model, within how much time the previous data/ models become useless, cost of training model, etc.

Each trained model is versioned. In case something goes wrong during training, previous model could be used. Versioning can be done on timestamp for example.



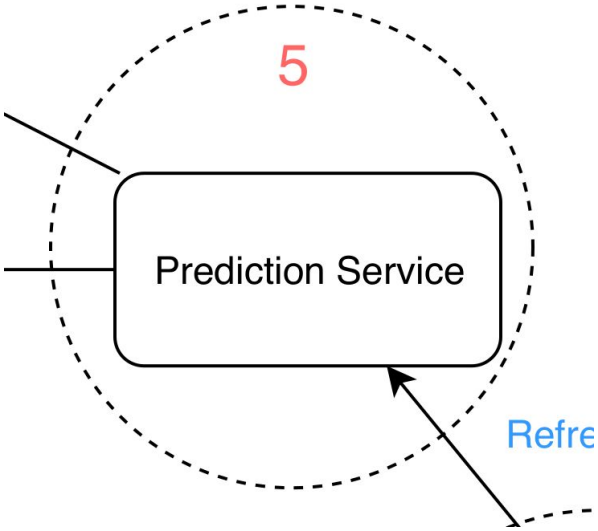
3. Using trained model

Once the models are trained, they will be used for making real-time predictions. But they can also be used for internal usage, example predictive analytics, marketing, customer care, etc. Offline models are also used as benchmarks for future model improvements.



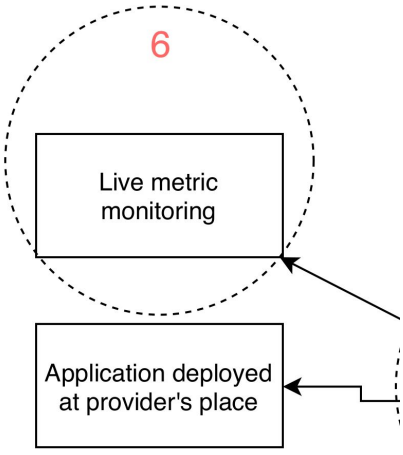
4. Prediction model storage

As mentioned before, training instance dumps the model into an AWS bucket for the prediction service to always choose the latest model. As soon as a new model is dumped, prediction service loads that model and starts making predictions using the new model. Models can be dumped as pickle and loaded later by the prediction service.



5. Prediction service

Prediction service is responsible for handling user's request to access the results from the model. Based on the input received from the users, it makes a prediction and returns the result. It also makes sure that the most updated model is used to make predictions. The prediction service includes a data transformation pipeline as well to make sure that features are processed before making a prediction.



6. Live system

We can monitor the system live on different levels - quantity of prediction requests, quality of model, real-world behavior of data and model. We can use scalyr to monitor these things. However, we need to be smart about logging in the prediction service.