Step 12: Design Your Deployment Solution Architecture

Intro:

Since my capstone code projects don't lend themselves to this, I will define the theoretical use case first and then answer the questions laid out at https://www.springboard.com/archeio/download/5d279d43b2e147c7b8251af7443082c2/

Use case:

We want to define a deployment architecture for a system that can predict loan default likelihood based on tabular categorical information pertaining to a particular user and loan application. The prediction must be near real time since our application commits to giving answers to end users on the spot, based on information gathered during our application session and prior knowledge about the user.

Components:

- Application. The loan application (web-based or mobile native) is interacting with user via a UI and gathering info during the loan application session about the loan request itself and the user. It interacts with the main system via REST API calls that will provide the loan acceptance decision (yes/no) based on info about user during the current session and prior information gathered about the user.
- Prediction Engine. It interfaces with the application while user is seeking a loan. It must respond in near-real-time (95^{th} percentile within 5 sec) via REST APIs. The evaluation metric is Kolmogorov–Smirnov statistic (KS), which is widely used for credit underwriting models.
- Modeling Process. This component is responsible for creating and updating the prediction engine listed above and ensures that the metrics used to gauge its effectiveness do not drop over time. It is the most complex component as it involves a cyclic model creation, evaluation, and productization.

Prediction Engine Details:

- It takes data from the application request, pre-processes it (e.g. normalization), merges it with additional info known about user from other storage areas and feed the consolidated data (with all needed features) into the actual Prediction Model, which will return a decision, eventually sent back to the end user.
    - Data from additional storage is fetched thru SQL or other interfaces depending on the nature and location of this additional user data.
- The Prediction Model is updated monthly. See details in Modeling Process section.
- An SLA monitoring unit, located in the public domain, sends requests to the Prediction Engine mimicking end users, to ensure its availability and latency. In case of breach of SLA, alerts are sent to operations for their investigation.
- A functional monitoring unit computes the daily percentage of approved loans, and if it departs from what is deemed acceptable, an alert is sent to operations.

Modeling Process:

This component is responsible for the creation and update of the Prediction Model and monitor its effectiveness over time. On a monthly basis, following sequence of event occurs:

- Most recent data (since last update) is run thru the current model (currently in production) and the KS metric is compared to the metrics that were produced when the last model was pushed to production.
- If outcome is similar or better than the current model, it is not updated for another month.
- If the metrics resulting from the latest (past month) data run thru the current model are significantly worse than the one realized the previous month, an investigation is started, aiming at understanding how the incoming data drifted. Based on outcome of investigation a new model is trained with appropriate data, validated in a QA/Sandbox system, then pushed to production.
- The whole model update can be orchestrated by Airflow and kicked off on a monthly basis.
- The model metadata is saved as versioned code (e.g. git) however the metadata may make reference to file locations, like the training/validation datasets. Goal is to be able to reconstruct any model that ever existed in production based on a version date or label.

Estimated Costs and Resources:

Capacity: The number of loan approval request peaks at a few requests a minute, so capacity is not a major concern.

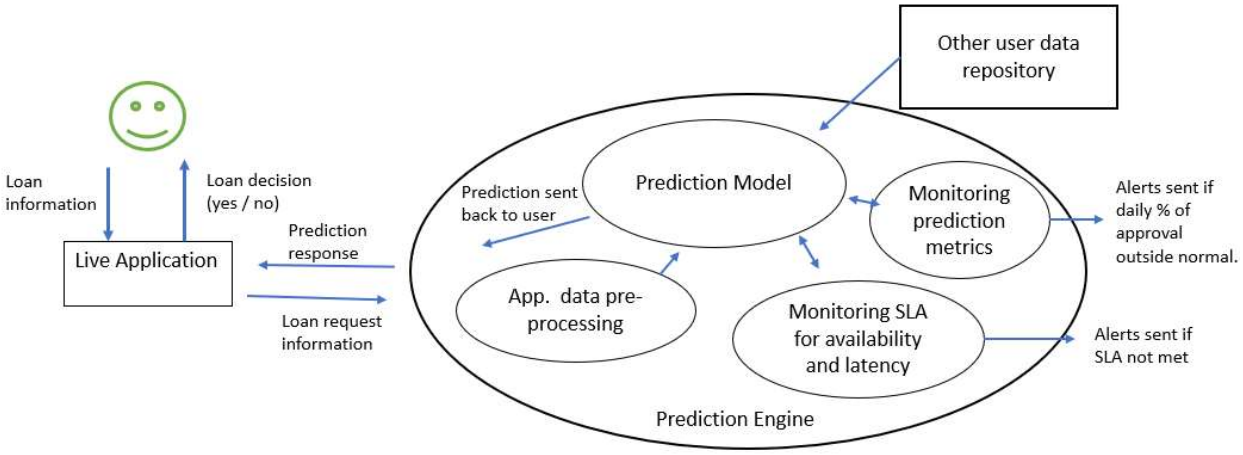
Latency is important, as the response must be provided as user is waiting. Since we want to provide final response to user thru the app within a few seconds, the different parts of the systems contributing to the final prediction must all comply to certain latency SLAs, for instance:

- Network time between application and our system: under 100 ms (US only, not in our direct control)
- Pre-processing incoming data (e.g. normalization, nan treatment): under 100 ms
- Fetching additional required information about user from DB or data warehouse: under one s
- Merging application data with user data to feed the model: under 100 ms
- Running resulting data thru model and get prediction: under one second (would vary on model)

Model predictive power is important, since both false negative (denied a loan we should have approved) and false positive (we approved a loan that defaulted) have negative business impact. We therefore monitor daily the rate of successful loan approval and re-train model every month on fresh data submitted by users.

Costs should be reasonable. Operational constraints are not so drastic (see above) and model training, performed once a month, does not involve huge amounts of data, typical order of magnitude is one million rows over about 100 features. Once the whole system is in place and new model generation workflow is automated thru Airflow, system should require only minimal maintenance. Once our operational and functional alerts are properly tuned, no additional manual supervision should be needed in between model-refresh.

Diagram of live overall system (offline Modeling Process not included)



- User initiate load application request from application
- Application sends user and load application to Prediction Engine
- Data from application is pre-processed and merged with additional user info
- Above data is fed to the Model, and makes a decision (yes/no)
- Decision is relayed to end user.

- A monitoring unit checks the Prediction Model for availability and latency, issues alerts if/when SLAs not met.

- Prediction Model is updated monthly based on most recent data