Machine learning PoC productionization

This page is being used to capture information around the approach that would need to be followed to productionize the RSS churn predictive model PoC and integrate it with the IT TEAM technology estate.

Ownership of the solution

Before productization starts, the long-term owners of the solution and technology need to be identified; This is usually the Data & Analytics team or it can be the IT team. This will likely determine which AWS ownership to host the productionized infrastructure.

This team will need to be able to provide the skills to maintain the predictive model and work with the wider technology organisation to manage the integrations.

A review process needs to completed to ensure these have been addressed.

ML go live checklist:

The purpose of checklist is to ensure that, as for software development assets, all the correct materials and documents are in place for machine learning algorithms to operate reliably in production and can be maintained.

Requirement	Response (Y/N/NA)	Details	If N or NA, remediation plan
ML artefacts			
Code and binaries for the ML algorithms, including dependencies or Jupyter notebooks			
Hyper-parameters used in training, and documentation of the required values during production operation/inference			
Trained model(s)			
Training data used by data scientists			
Test data used by data scientists to validate the model(s)			
Gold result set and any regression result set			
Access to the tools used for comparing result sets (needed for consistency of verifying results)			
Docker, or equivalent, files and configs			
Any other associated artefacts, anything needed to recreate the model, such as ontologies, given levels of a graph, word vector files, code used to transform data used in training			
Documentation			
To describe the model(s), data, interfaces/APIs and algorithms			

Results and metrics, eg F-scores, from testing for the production version of model/algorithm			
How to make changes to any of them and proceed through the training/validation cycle			
Other algorithms or approaches used during development and why they were rejected			
Description/diagrams of the hardware used to build the models and carry out the processing			
Processes in place			
Code build			
Model build			
Deployment			
If this will be deployed as a service with multiple consumers, then there should be service level agreements in place			
Appointed service and support personnel for the deployed data science service			

Taking Machine learning PoC to Production.

The full process of taking a problem that is solved through any form of data science or AI, through the development of an algorithm, training of that with curated material, to the deployment to production, and maintenance over time is complex. This best practice is a first step towards managing this complexity, laying out the activities that a team should address as part of that process. This best practice does not cover how a team should approach each of these stages. It only ensures that teams have considered their needs for each stage and have created a plan appropriate to those needs.

Intended Outcome

Improved consistency, repeatability and time-to-production of data science services.

Requirements

A project delivering a data science, AI, ML or NLP component SHOULD describe a plan for each of the stages of the data science lifecycle outlined below.

Implementation

The stages of preparing a data science component, with associated models, training data and gold sets are outlined below. A team using data science SHOULD document how they handle each of these stages, where they apply to that team. That documentation could be in the form of a wiki (Confluence) or documents in shared storage. It SHOULD be available to the team working on the development of the data science components,

The stages used are described in Sculley et al. https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf,

Definition of the problem to be solved using data science/ML with quality targets

- Configuration management (code, data, gold sets, infrastructure)
- Data collection, cleansing and management, including security and privacy aspects

- Feature extraction
- Machine learning coding
- Model training
- Output data analysis and verification
- Machine resource management (managing where workloads execute, who is using what machine etc)
- Analysis/measurement tools (for comparing result sets) and quality metrics (eg f-scores, numbers of characters classified)
- Process management tools/practices, including training workflows
- Serving (production) infrastructure
- Monitoring (in production, for drift etc)

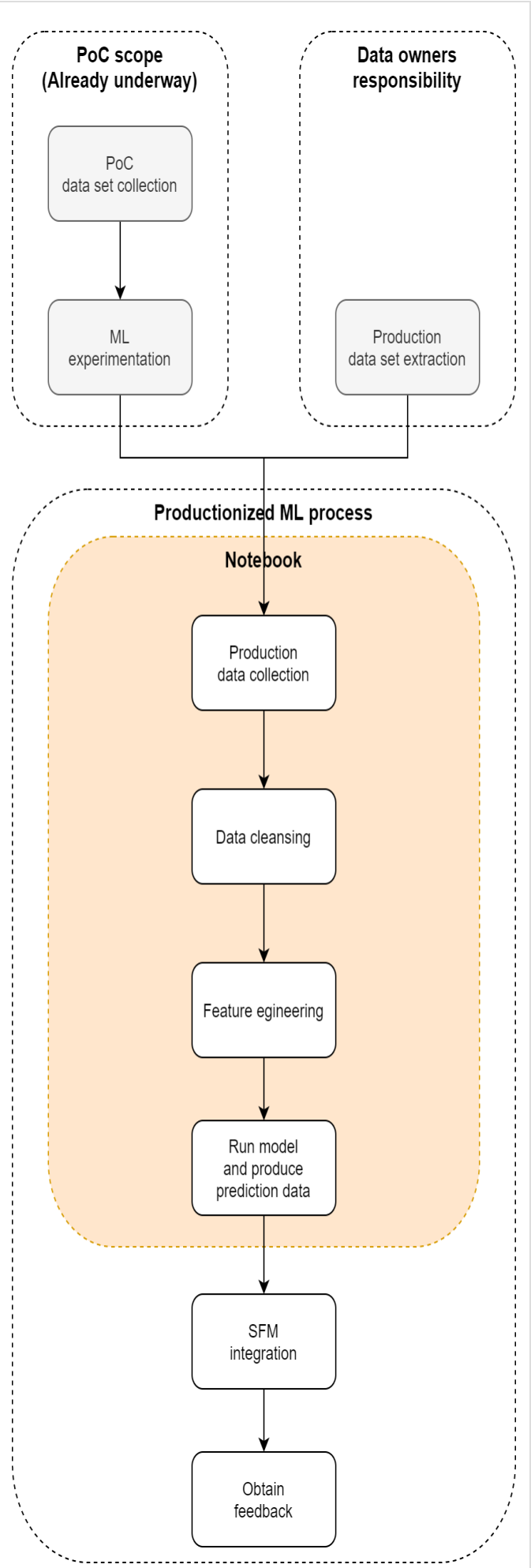
Benefits

- Provides data science teams with a checklist of activities in the lifecycle that they should have covered
- Improves the repeatability of the algorithms developed in
- Speeds incremental improvements to the services and hence the time to develop and deploy data science services
- Improves communication within the team and handoffs to other teams, for example to a production support team
- Clarifies roles of the team
- Gives clear guidance on how a team carries out each of the activities

Productionization of the PoC predictive model

Defining the problem

At a workshop on 11 Feb 2020 we worked through the high-level process steps and agreed what needed to be productionised for the RSS churn model. These since been elaborated with the relevant IT TEAM stakeholders.



Data set extraction

Data owners extract and publish data sets to data lake. Access to the data is managed through the data catalogue.

Delivery responsibility for each data set would be with the data owners, once it had been identified there was an enterprise need for the data. Productionization project would need to work with these teams to deliver the data sets.

Output: Data sets registered in the data lake catalogue

Data collection

Consumption of the relevant data sets, required by the ML model, from the data lake

Access to these will be to an AWS user.

An appropriate AWS account will need to be identified for running this model. This will need to be determined by which team is accepting the long-term ownership and support of the model.

Output: Raw data sets accessible

Data cleaning

This step is turning the raw data sets in to data that can be consumed by feature engineering. It may involve aggregating or filtering raw data sets to meet the needs of the models input format.

Feature engineering

Consuming the cleansed data, extracting features and applying this to the model

Run model

Produce predictions of churn risk for the previous year journal subscriptions.

Output: churn risk data set

SALESFORCE integration

Load the predictions into the SalesForce organisation so that the churn risks can be

raised to the appropriate sales team members.

- A business-agreed set of rules is created to interpret the output of the model and identify those renewals considered at enough risk that the sales owner should be alerted
    - This might be a threshold or cluster of thresholds that is felt strong enough to indicate a churn risk
- For each account where a risk is determined to exist, a SFDC "Risk" object is created and linked to the appropriate account
    - SFDC will then ensure it's sent to the correct sales person/team with a task to examine the risk and take appropriate steps
    - There are several ways in which these can be created:
        - SFDC data loader can ingest a file
        - APIs can be called to create the risks

Feedback

Provide a feedback mechanism from the consuming system to model provider.

Approach:

- Decide what information to collected to determine effectiveness of the risks being highlighted
- Determine how to extract this from the Account, Risk, Opportunity and Task objects that SFDC will use to capture the risk and how it's handled by the sales team.
- Use SALESFORCE (the Snowflake-based reporting and analytics warehouse connected to SALESFORCE) to extract the feedback information

Production architecture for churn model deployment

The proposal is to continue to maintain the model in a Jupyter notebook, as per the PoC. This will need some modification to be able to run in the proposed environment.

AWS EMR could be used to execute the notebook and run the stages enclosed in the orange boundary in the diagram above.

All artefacts should be version controlled in GitHub and deployment ideally managed through Jenkins.

The flow would work something like this:

- Jenkins is used to provision both the "Run model" and "Load Churn risks" lambda functions into an AWS account.
- These monitor specific folders within an churn model S3 bucket
- When a new notebook is written to the bucket (there could be other triggers), the "Run model" lambda function is triggered. This
    - Starts a new EMR cluster, with config from the S3 bucket
    - Pulls in the data extracts from the data lake
    - Executes the notebook in the cluster
    - The output is written back to a different folder in S3
    - The EMR instance is automatically torn down
- The output file then triggers the "Load RSS risks" lambda function
    - This would take the output file and either perform a bulk load into SALESFORCE or make API calls to create the risk objects
- Feedback, is collected through reporting from SALESFORCE
    - SALESFORCE is the existing Snowflake-based reporting platform
    - Query would need to be defined in business terms and then mapped to data queries that could be performed against the objects associated with the risks