Data Science Platform

“Principled Data Engineering/Science”

Product Requirements Document

Syngenta

Product Requirements Document

Table of Contents

[1. Objective 2](#_Toc17477624)

[2. Technical Debt 3](#_Toc17477625)

[3. Division of Labor 3](#_Toc17477626)

[4. Architecture 4](#_Toc17477627)

[5. User flow and design 4](#_Toc17477628)

[6. Release 4](#_Toc17477629)

[7. Features 5](#_Toc17477630)

[7.1 Authoring 5](#_Toc17477639)

[7.2 Environment Management 6](#_Toc17477640)

[7.3 Data Management 8](#_Toc17477641)

[7.4 Model Management 10](#_Toc17477642)

[7.5 Graduation 14](#_Toc17477643)

[7.6 Scaling Hardware 15](#_Toc17477644)

[7.7 Auditing 17](#_Toc17477645)

[7.8 Model Monitoring: Data Science perspective 17](#_Toc17477646)

[7.9 Model Monitoring: OPS Perspective 18](#_Toc17477647)

[8. Analytics 20](#_Toc17477648)

[9. Future work 20](#_Toc17477649)

# 

# Objective

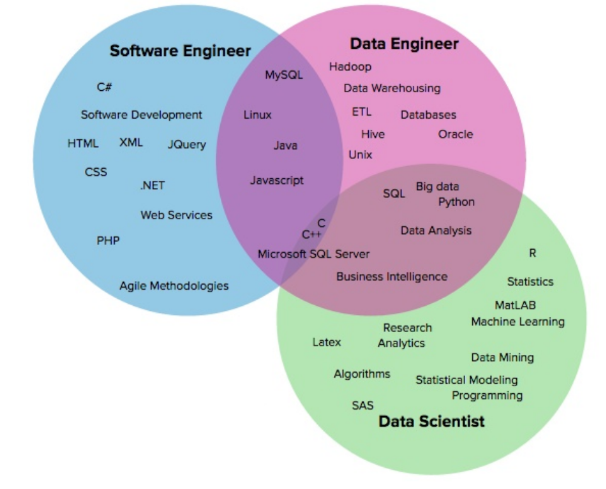
|  |  |
| --- | --- |
| Vision | The Data Science Sandbox is a flexible, cloud-based software environment for the **Data Scientists| Data Engineers| Data Analyst** for development, testing, rapidly prototype, save, share, scale, and publish data science tools, models, pipelines, and ultimately end-to-end analytics solutions.  This product is intended for software, data and cloud engineers to develop data products and iterate.  This document is intended for Data Scientists (users) to communicate needs, requirements, and expectations. |
| Goals | *List product goals*   * Common Interface and Execution Patterns * Coverage - Platform to support all the key interfaces, frameworks and public cloud environment * Scalability & Performance - How well the platforms’ architecture uses its available resources (CPU and GPUs & Parallel Processing). * Productivity – Improvise   + Amount of training required to use the platform   + Effort required to develop and deploy a solution   + Availability of collaboration and configuration management tools   + Platform complexity   + Skill level of resources * Cloud Capable - Completely cloud based (PaaS – Platform as a Service) * Continuous Deployment –   + Stream line and automate product ionization activities.   + Organize data, code, and model outputs into projects, making it easier to locate relevant files. * Interface –   + Providing a way to visualize machine learning/analytic processes, select algorithms, shape data, and define parameters.   + Easier way to integrate platform functionality into custom applications * Security –   + The need for security will be driven by data, compliance (GDPR, etc.) and audit requirements.   + The ability to define security by role may be necessary.   + Information and business outcomes must be balanced to ensure that they do not conflict. * Cost   *To be include their timeframe and success metric* |
|  | Central data science team way of working and supporting internal corporate stakeholders. High number of models, highly technical.  While we build up capability for ML/Deep Learning and AI strategy this would help   * Reduce duplication of efforts across teams * Identify and surface the best internal and external models * Move models and A/B test models in production * Run high-throughput analytics in production * Monitor and enforce regulation( e.g. Machine learning in the GDPR), security, and corporate compliance measures |
| Persona(s) |  |

# Technical Debt

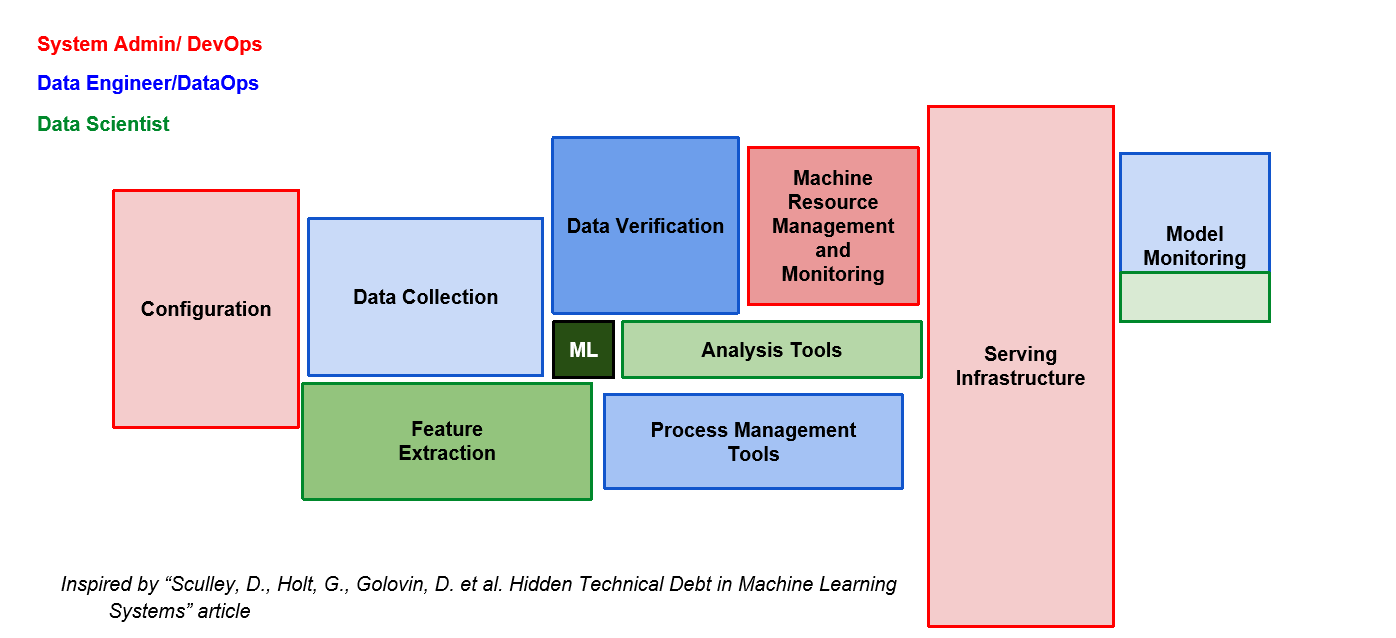
Current technical debt with respect to model productionization are as following:

* Lack of version control
* Manual and Complicated Deployment Tasks
* Artefact Management
* Metadata Management
* Lack of Production Standards
* Lack of tests and Portability of Code
* Way to choose between candidate models and Identify harmful models in production
* How to model is doing in production.

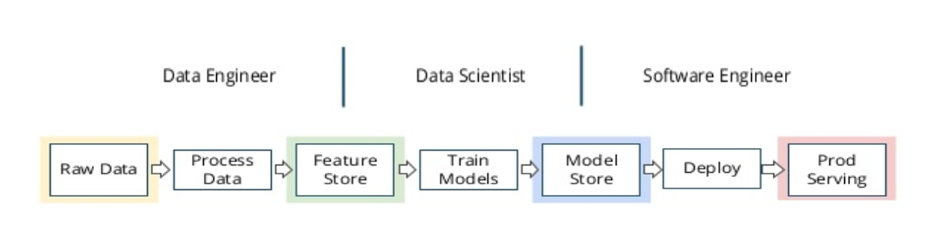
# Division of Labor



Overlap of Data Engineer and Scientist Technical Skills like would not be the source for “Division of Labor”.

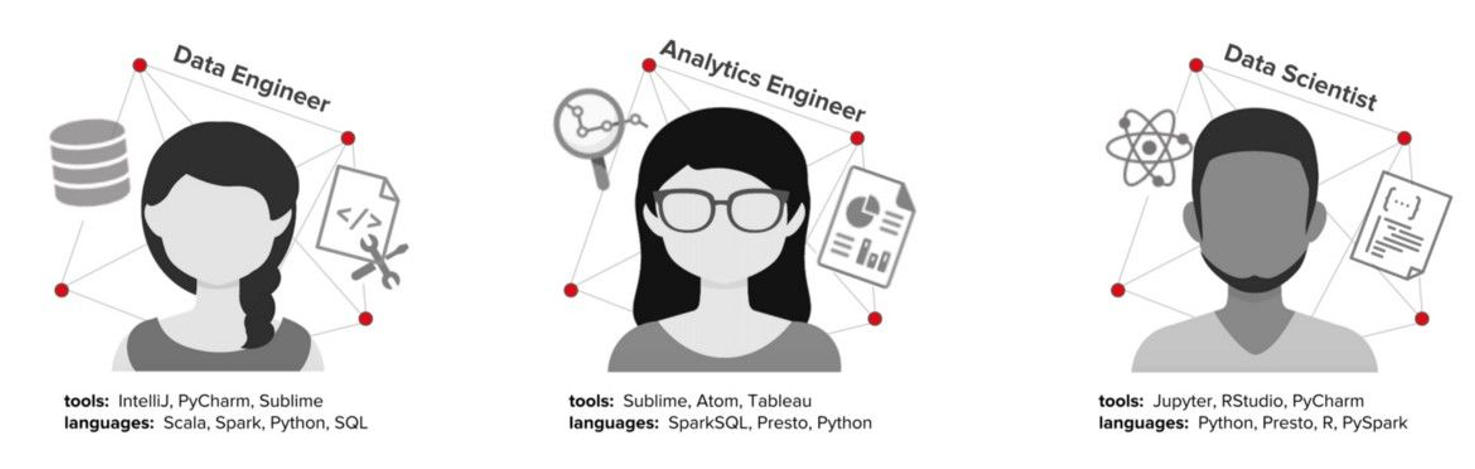
******

***Role Separation*** - Based on Tasks to be executed during the life cycle of machine learning model evolution, which platform is going to eliminate.



# Architecture

User Experience (Workbench)



Centralized Logging

Monitoring & Alerting

Life Cycle Management

Security

Data Platforms

Data Lake Data Catalog

Upgrades & Patching

Backup & Restoring

Infrastructure services

Leveraged for elasticity and redundancy

Public Cloud Platforms Containers

To be discussed: Tool selection and data retention procedure?

John 🡪 Survey mechanism to understand user needs and also evaluate toolsets.

Preetam 🡪Do a trade-off analysis and standardize them.

John 🡪 Local data, needs back up. All data should go back to Platform data architecture – while we have a mechanism to build ETL/ELT pipelines. Some of the process this might difficult.

Preetam 🡪 we need to discuss and standardize as the process matures in future, personally I would like to maintain a confluence/wiki why I selected a certain tool.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Description** | **Toolset** |
|  |  |  |
|  |  |  |
|  |  |  |

# User flow and design

*Insert wireframes and mockups.*

# Release

|  |  |
| --- | --- |
| Release | *Release name* |
| Date | *Release date* |
| Initiative | *Initiative that the release relates to* |
| Milestones | *Release milestones* |
| Features | *Features included in the release* |
| Dependencies | *Release dependencies* |

# Features



## Authoring

|  |  |
| --- | --- |
| Feature | Workbench |
| Description | Web based Integrated development environment for exploration, collaboration, and visualization  [Hue, the self service open source Analytics Workbench for browsing, querying and visualizing data interactively](http://gethue.com/)   * Jupyter kernels would be configured as per the supported programming languages * Hue would be helping us with data-driven decision making with SQL. |
| Purpose | Achieve the Goal of Common Interface and Execution Patterns. |
| User problem | Access to a pre-configured computational environments and resources without burdening the users with installation and maintenance tasks. |
| User value | Standardization of computational environments |
| Assumptions | Programming languages supported as mentioned in the document. Any addition or deletion need to go thru a change control board acceptance.  Users might have different choice of desktop editors example:   * PyCharm/ Spyder * RStudio * Atom/Sublime text   Above mentioned editor or its plugin installation, updates is not part of the data science sandbox.  In order to run code seamlessly in the platform while using the desktop editors the user/s need to submit their code via CLI to cloud managed services like AWS sage maker, Also note here in order to have reproducible projects runs CLI commands would be defined and standardized.  Interactive notebooks need to support multi-tenant environment.  TBD: Jupyter extensions (example: Gitlab (commit from the notebook), airflow (data pipeline status verification within notebook), s3, etc.) and scheduling notebook execution (papermill helps us executing a notebook workflow).  John 🡪 Currently we use SQL extension, code folding, Jupyter-extensions. We need to restart JupyterHub for few of the installations |
| Not doing | Other web based notebook environments like Zeppelin would be introduced If we have decent user base in subsequent releases. |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Support to diverse set of languages |
| Description | Ability to code with programming language of choice |
| Purpose | Achieve the Goal of Common Interface and Execution Patterns. |
| User problem | Access to a pre-configured computational environments and resources without burdening the users with installation and maintenance tasks. |
| User value | Standardization of computational environments |
| Assumptions | User might use some or all of these programming languages while defining the pipeline of data or machine learning  Version of programming language available as part of the platform would be based on the support they are going to have from the open source community or commercial vendor. |
| Not doing | Optimization of the platform to educate the user if the programming language and Interfaces/Framework compatibility would not be part of the scope. |
| Acceptance criteria | *Conditions of acceptance* |

## Environment Management

|  |  |
| --- | --- |
| Feature | Frameworks/Interfaces |
| Description | Ability to use Zoo(diverse set) of eco system frameworks/interfaces for deep learning and machine learning |
| Purpose | To support various ML/AI or Deep learning use cases for example   * Apache Spark for PCA(Principal Component Analysis) * XGBoost for Decision trees * TensorFlow and Apache MXNet for Deep Neural Networks * Keras convolutional neural network(CNN) for classification |
| User problem | Current environment does not allow user to access internet based installation or upgrade of various frameworks and interfaces. |
| User value | User would have virtual environment and install packages of their choice while doing   * data transformation * feature extraction, generation * exploratory data analysis * visualization of the results * expose the model as an API , to train and enhance the model in near real time   TBD: do we need pre-built images  John🡪 Future stage, we don’t need it currently. |
| Assumptions | Use cases might need distinct version of these frameworks or interfaces hence we might have multiple repositories hosting varied versions.  Would be hosting a local proxy for supporting package managers like   * Conda * Python Package Index * CRAN * Maven/Gradle |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Public Cloud infrastructure/s |
| Description | Ability to host on varied set of public cloud providers, Only those supported by Syngenta Cloud team. |
| Purpose | *Task or action the user wants to accomplish* |
| User problem | *Pain point or challenge* |
| User value | *How the proposed solution helps the user* |
| Assumptions | *Business, user, or technical assumptions* |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

## Data Management

Most expensive part of machine learning pipeline – Here we automate the processes involved in extracting, transforming, combining, validating, and loading data for further analysis by defining what, where, and how data is collected.

**Data Sourcing ⮊ Data Cleaning⮊ Data Labeling ⮊ Data Versioning**

|  |  |
| --- | --- |
| Feature | Data Sourcing and Building Pipeline |
| Description | Ability to automate the process of how data is collected, transformed and loaded with help of   * Ingestion * Identification | Filtration | Validation | Noise Reduction * Transformation | Compression * Integration |
| Purpose | Data pipeline is to provide concise data, making it easier to report, analyse, and use.  Here we can make the process reproducible, auditable, smooth flow of data from one station to the next. |
| User problem | As sourcing right data from different sources with variety of formats in bulk. Can be time-consuming & costly if money not spent on ETL/ELT automation.  Where source of data might be more than one and we might need to  manually pick every ﬁeld, table, data source, transformation, combine, join, etc. |
| User value | Automates the processes involved in extracting, transforming, combining, validating, and loading data for further analysis and visualization. |
| Assumptions | Sometimes user is lucky and have historical data readily available. Sometimes user need to search for raw data or use a simulated dataset.  Pipelines to be used in   * Batch – Movement of large data volumes * Real-time – From Streaming sources * Cloud Native – Cloud based data, such as from Object Store(S3)   Storage (Avro, parquet, csv, json, etc.,) format is open for platform user choice.  TBD: Abstract API to simplify things or data source management with Airflow hooks or custom UI  John 🡪 May be a Generic data source (Redshift)  To be discussed: What all data would be stored as part of platform while model training, test, result .To help us define what goes back to data platforms and what stays with the data science platform |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | Should have a mechanism to   * Audit * Schedule * Monitor * Alert * Messaging and Notification   Pipeline execution |

|  |  |
| --- | --- |
| Feature | Data Labelling |
| Description | Ability to understand what is shown on a photograph, said in a voice recording, or written in a text, among many other things. |
| Purpose | Dark data is valuable, Tagging dark data can improve model’s precision and expand use cases and help us in supervised learning, algorithms learn from labeled data. |
| User problem | Data scientists often spend up to 80% of their time cleaning and prepping data. |
| User value | By labeling data, models can improve their learning and evolution. |
| Assumptions | TBD : if this is done already by any other team – Specially with respect to Images / Text /Security and Compliance  John 🡪 Standardize the tool, so that we can have persistent data strategy  Preetam 🡪 Might need to finalize the mechanism to schedule (cron or airflow) the program which does data labelling. We might also use a labelling(labelbox or etc) tool |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

Hardest Part of machine learning pipeline is feature engineering

|  |  |
| --- | --- |
| Feature | Feature Store |
| Description | Centralized vault for collecting documented, curated and access controlled features and sharing features |
| Purpose | [*According to Uber*](https://www.globenewswire.com/Tracker?data=JRY6ZYaPVHZY7UzcGhN5cYk8oPjkNRSy49AXvXxNuaaiJuNXAxVrlgaADnVMDIplNQhyvmGEB_PLxBvdyupVcrNq1f5fBMvjDmVH6ZFocvWg35c9rrqOwCgWCFtYwFci)*,* “dealing with data access, integration, feature management, and pipelines can often waste a huge amount of a data scientist’s time”.  A **feature store** is a data management layer for **machine learning** that allows Data Scientist s and Data Engineers to share and discover **features**, better understand **features** over time, and effectives the **machine learning** workflow  Data scientists and Data Engineers to reuse versioned features and review feature metrics by models. |
| User problem | Machine learning and deep learning emphasizes ad-hoc feature engineering and training pipelines to experiment with ML models. Such pipelines have a tendency to become complex over time and do not allow features to be easily re-used across different pipelines.  Duplicating features can even lead to correctness problems when features have different implementations for training and serving. |
| User value | [*According to LogicalClocks*](https://www.globenewswire.com/Tracker?data=JRY6ZYaPVHZY7UzcGhN5cYk8oPjkNRSy49AXvXxNuaaiJuNXAxVrlgaADnVMDIplNQhyvmGEB_PLxBvdyupVcrNq1f5fBMvjDmVH6ZFocvWg35c9rrqOwCgWCFtYwFci)*,* The Feature Store(**Feature Store - the Data Warehouse for Machine Learning**) solves the data access and feature management problem for Data Science by removing the need for Data Scientists to constantly re-implement feature pipelines for collecting and transforming data to feed their machine learning models. Instead, Data Scientists can select features from the Feature Store to generate clean training data that can then be consumed directly by machine learning models. |
| Assumptions | * Platform- (John --> Partnership between Data Science and Data Engineering team) team curates core set of widely applicable features * Modelers contribute more features as part of ongoing model building * Modelers select features by name & join key. Oﬄine & online pipelines auto conﬁgured * Need of meta-data for each feature to track ownership, how computed, where used, etc.   TBD: Current feature extraction tools  John 🡪 Currently not doing any Auto Engineering or Auto ML , may be future scope |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

## Model Management

Have a mechanism to be reproducible and verifiable while going thru the following life cycle

**Model Architecture ⮊ Model Evaluation and Training ⮊** **Model Versioning ⮊ Model Deployment ⮊ Model Serving**

|  |  |
| --- | --- |
| Feature | Model Knowledge Repository |
| Description | Mechanism of sharing reports and analytics but also ML experiments so that knowledge can be pervasive. |
| Purpose | Centralized knowledge repository which is focused on facilitating the sharing of knowledge between data scientists and other technical roles. |
| User problem | No mechanism available in *scaling knowledge*, share insight uncovered by one person or effectively transfer beyond the targeted recipients. |
| User value | A capability to communicate with business stakeholders with lesser technical knowledge |
| Assumptions | User should also be able to share their Interactive notebooks apart from experiment knowledge  TBD: Do we need a ability to convert notebook into wide variety of output types, including PDFs, Emails, etc. |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Model Evaluation and Training |
| Description | Train your model with many different sets of hyper parameters and store run history  Compare model execution between executions (Between QA vs PROD, different data). |
| Purpose | *Task or action the user wants to accomplish* |
| User problem | • It takes many iterations to produce a good model  • Keeping track of how a model was built is important  • Evaluating and comparing models is hard |
| User value | *How the proposed solution helps the user* |
| Assumptions | TBD: How we currently test  John 🡪 Unit test cases are available currently and are currently we do not have hierarchy of coding standards, we need to establish one. |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Data Versioning |
| Description | Ability to store intermittent data while creating a model and help us have a self-descriptive way to understand how a model was produced. Note this is not similar to Data Pipeline (ELT/ETL) |
| Purpose | Light weight process taking inputs and producing outputs where Data Scientist/Engineer might able to   * Tag a new set of data with a new version * Return to old data versions or Switch between different data versions very easy. |
| User problem | The dataset on which Data Scientist/Engineer is working does not evolve in time. At times when our data is divided into train, test and validation folders by default, with the amount of data increasing over time either through an active learning cycle or by manually adding new data.  Here Data Scientist/Engineer might want to just not only split the data but also want to split the process in several sub-processes to be able to separate different concerns: while working either on one or the other part of the project.  Input Data → Process → Results  Becomes  Input Data → Process1 → Intermediate Data → Process2 → Results  Here there is no ground truth to remember - not-versioning our data makes it tougher to know what was the state (Identify which input data produced an intermediate data which then also produced a result data.) of the data at a previous milestone. |
| User value | Data Scientist/Engineer to have a mechanism (effortless way to split a project into atomic steps.) by which they can switch between versions of data during active machine learning cycle. |
| Assumptions | User don't need to change file suffixes\prefixes\hashes all the time from your code.  Sometimes we don't need a local copy and prefer to read from S3 into memory directly  *Kaggle Data Science Community:*  When should you version your data?   * When making schema/metadata changes, like adding or deleting columns or changing the units that information is stored in. * When you’re training experimental machine learning models. The smallest reproducible unit for machine learning models is training data + model specification code.   When should you consider not versioning your data?   * When your data isn’t being used to train models. For example, it’s more space efficient to just save the SQL query you used to make a chart than it is to save all the transformed data. * When your data is large enough that storing a versioned copy would be prohibitively expensive. In this case, I’d recommend versioning both the scripts you used to extract the data and enough descriptive statistics that you could re-generate a very similar dataset. * When your project lives entirely on GitHub. Versioning large datasets via GitHub can quickly become unwieldy. (GitLFS can help with this, but if you’re storing very large datasets, in general GitHub probably isn’t the best tool for the job. A database or blog storage hosting service specifically designed for large data will generally give you fewer headaches. Most cloud services will generally already have some form of versioning built in.)   Of course, whether or not you should version data eventually comes down to a judgement call on your part. |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | Versioning of datasets and machine learning models. Where data is saved in S3, Google cloud, Azure, Alibaba cloud, SSH server, HDFS or even local HDD RAID. |

|  |  |
| --- | --- |
| Feature | Model Versioning |
| Description | Ability to version Raw Model code and its Artefacts (Parameters learned via training). |
| Purpose | *Task or action the user wants to accomplish* |
| User problem | *Pain point or challenge* |
| User value | * Once we have an improved model and registers it store for future * We can also tag them for discovery * Maintain different versions of model and expose them as versioned API , when we expose them as web service/s |
| Assumptions | Should have an ability to store intermediate data and model version. |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Model Validation |
| Description | *Description of what the new feature will do* |
| Purpose | Carefully trained and evaluate a model in our Data Science sandbox, needs additional work is required to check that it will work correctly in your production environment |
| User problem | *Pain point or challenge* |
| User value | *How the proposed solution helps the user* |
| Assumptions | *Business, user, or technical assumptions* |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Model Deployment and Serving |
| Description | *Description of what the new feature will do* |
| Purpose | * Expose model via a REST API * Deploy new model to small subset of users to ensure everything goes smoothly, then roll out to all users * Maintain the ability to roll back model to previous versions   TBD: Integration testing , A/B kind of deployment  John 🡪 Integration testing with respect to MVP solutions. For testing we might use Python test modules. |
| User problem | *Pain point or challenge* |
| User value | *How the proposed solution helps the user* |
| Assumptions | TBD: Exposing the API behind a load balancer or with a API gateway  Ability to deploy two versions of the Model deployment for A/B Testing and to find out the best model. |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

## Graduation

|  |  |
| --- | --- |
| Feature | Continuous Integration |
| Description | Ability to do concurrent experiments on a single model |
| Purpose | **Continuous Integration** is merging all code from all developers to one central branch of the repo many times a day trying to avoid conflicts in the code in the future. The concept here is to have multiple dev’s on a project to keep the main branch of the repo to the most current form of the source code, so each dev can check out or pull from the latest code to avoid conflicts.  Here the developer's changes are validated by creating a build and running automated tests against the build, With a great emphasis on automated tests |
| User problem | *Pain point or challenge* |
| User value |  |
| Assumptions | *Business, user, or technical assumptions* |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Continuous Delivery |
| Description | Ability to do continuous delivery of ML models to production |
| Purpose | **Continuous Delivery** requires building, testing, and releasing faster and more frequently  Apart from extension to continuous integration, here we automate our release process and can deploy our application quickly in a sustainable way at any point of time.  We can decide to release daily, weekly, fortnightly, or whatever suits our business requirements. |
| User problem | Currently data scientist/analyst/engineer need to go thru a tedious manual process |
| User value | * User would be able to productionize his models in an automated fashion. * Where user would be able entire pipeline of training, validating, testing, deploying to staging environment and production. * Would allow user to fail fast in case he wants to roll out a better model * User can also roll back to a better model if he wants to. |
| Assumptions | Delivery pipeline should not create the production infrastructure on each execution.  TBD: Canary deployment – To help us know if we are deploying a model without any surprises. |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

## Scaling Hardware

|  |  |
| --- | --- |
| Feature | Powerful Computational infrastructure |
| Description | Ability to request additional servers over a public cloud / Change instance type  [Related image](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=imgres&cd=&cad=rja&uact=8&ved=2ahUKEwiazrKJoevjAhWFNY8KHcUqBWsQjRx6BAgBEAU&url=https://venturebeat.com/2018/08/16/how-ai-is-decommoditizing-the-chip-industry/&psig=AOvVaw1W26h5XPzrtWFqYk_CAIMq&ust=1565078245642767) |
| Purpose | Data Scientist to be able to execute his current model with more resources like VCores or RAM(Primary Storage) |
| User problem | Data Scientist or an Analyst currently |
| User value | User would be able to improve the model execution performance by adding additional cores or RAM either by changing the instance type or by addition of more instances |
| Assumptions | While requesting cloud computing and storage resources , If we are exhausted with allocated quota of cloud resources or instances, we need to raise the request with help of any Syngenta tool (JIRA) get an approval as a process. |
| Not doing | Instances which can be provisioned with the help of Syngenta Cloud team. |
| Acceptance criteria | User should be able to run his model or data pipeline. |

|  |  |
| --- | --- |
| Feature | Computation load distribution |
| Description | Achieve Data and Model parallelism, In order to enable computational load distribution we might need to use distributed computing. |
| Purpose | [*Data parallelism*](https://en.wikipedia.org/wiki/Data_parallelism) is a parallelization technique that is enabled by partitioning data. In data parallel distributed computing, we first divide the data into a few partitions, with the number of partitions equal to the number of worker machines (i.e. computational nodes). Then, we let each worker own one independent partition and let them perform computation over that data. Since we have multiple nodes scanning the data in parallel, we should be able to scan more data than when using a single node — we increase throughput through distributed parallel computing.  *Model Parallelism* is a parallelization technique that instead of partitioning data, we try to partition the machine learning model itself to distribute the workload to multiple computational workers. For example, let’s say we are solving a matrix factorization problem where the matrix is super huge and we want to learn every parameter of this huge matrix. To apply model parallelism, we have to partition the matrix into many small blocks (sub-matrices), and then let each worker take care of a few. |
| User problem | Currently user needs to request a Distributed computing cluster with the help of JIRA and it doesn’t have a common mechanism to build the same. |
| User value | Platform would help us in building a distributed computing (Spark/Hadoop) cluster with help of Platform while doing a Distributed Machine Learning. |
| Assumptions | Chris 🡪Initially we would use the script provided by the cloud team and eventually the script would be integrated with the platform.  Computational system would support us with   * Consistency * [Fault tolerance](https://en.wikipedia.org/wiki/Fault_tolerance) * Resource management   Tuning the cluster for optimal performance is a shared responsibility between the platform team and the platform end user. |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Scalable Deployment |
| Description | Ability to deploy with many production environments (e.g. with Docker and Kubernetes) |
| Purpose | *Task or action the user wants to accomplish* |
| User problem | *Pain point or challenge* |
| User value | *How the proposed solution helps the user* |
| Assumptions | *Business, user, or technical assumptions*  Chris 🡪 If we decide to use docker for the platform itself or for scaling the platform users we might need to use Open Shift environment of Syngenta, which is hosted on AWS platform. |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

## Auditing

|  |  |
| --- | --- |
| Feature | Logging |
| Description | Aggregate logs to track all errors and exceptions in your model creation pipeline. |
| Purpose | *Task or action the user wants to accomplish* |
| User problem | *Pain point or challenge* |
| User value | *How the proposed solution helps the user* |
| Assumptions | *Business, user, or technical assumptions* |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

## Model Monitoring: Data Science perspective

|  |  |
| --- | --- |
| Feature | Managing experiments |
| Description | Experiments are integrated with standard role-based access controls to set sharing permissions. |
| Purpose | *Task or action the user wants to accomplish* |
| User problem | *Pain point or challenge* |
| User value | *How the proposed solution helps the user* |
| Assumptions | *Business, user, or technical assumptions* |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Monitoring Data drift |
| Description | Monitoring their machine learning models, they are primarily checking for one thing: drift.  Drift means that the data is no longer relevant or useful to the problem at hand. Because data is always changing, drift occurs naturally. |
| Purpose | Data scientists must monitor the machine learning models to ensure that the model inputs look similar to those used in training. |
| User problem | If Data scientists do not monitor, the data may be drifting, which signifies that the data is out of date or no longer relevant.  This can produce incorrect results |
| User value | Monitoring the machine learning model helps us alerting the Data Scientists for corrective action. |
| Assumptions | *Business, user, or technical assumptions* |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Monitoring between Models |
| Description | Ability to record of experiments and recreate the results by reproducing the experiment and analyze previous execution or intermediate models. |
| Purpose | Continuously monitor model accuracy over time, and retrain or modify the model as needed. |
| User problem | *Pain point or challenge* |
| User value | *How the proposed solution helps the user* |
| Assumptions | *Business, user, or technical assumptions* |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

## Model Monitoring: OPS Perspective

|  |  |
| --- | --- |
| Feature | Monitoring the machine learning model from OPS perspective |
| Description | *Description of what the new feature will do* |
| Purpose | Monitor performance metrics, collect, visualize, and alert on all performance metric data using pre-configured monitoring tools. Gain full visibility into your training and inference jobs.   * The operational perspective. * A service perspective * A cost perspective |
| User problem | *Pain point or challenge* |
| User value | Operational : Consumption that is occurring, including the CPU/GPU etc., memory, disk, and network I/O and also latency and throughput.  Service: SLAs that are agreed for business success. while monitoring entire analytic workflow - maximum time it takes to create a model, deploy a model, and / or iterate on a model  Cost:  TBD: Control costs and manage your resources efficiently, Are we going to use shared environment or dedicated environments monitored  John 🡪 We have the need, but need to figure out the mechanism to do.  Harish 🡪 |
| Assumptions | *Business, user, or technical assumptions* |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Resource utilization Reports(example: Cloud Watch) |
| Description | Ability to understand   * Costs incurred during the execution of ML and Data Pipeline’s. * Usage Report with respect to current and also historically for the tagged resources. |
| Purpose | *Task or action the user wants to accomplish* |
| User problem | *Pain point or challenge* |
| User value | *How the proposed solution helps the user* |
| Assumptions | *Business, user, or technical assumptions* |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

|  |  |
| --- | --- |
| Feature | Usage Reports |
| Description | Ability to understand usage statistics by   * Viewed * Days since last accessed * Currently running apps/projects/instances/containers * by user * by project |
| Purpose | *Task or action the user wants to accomplish* |
| User problem | *Pain point or challenge* |
| User value | *How the proposed solution helps the user* |
| Assumptions | *Business, user, or technical assumptions* |
| Not doing | *Anything that is out of scope for this feature* |
| Acceptance criteria | *Conditions of acceptance* |

# Analytics

*Hypothesis: We believe <this feature> will achieve <this outcome>*.

|  |  |  |  |
| --- | --- | --- | --- |
| **Key performance indicator** | **Baseline** | **Target** | **Timeframe** |
| Percentage decrease in time-to-publish a model |  |  |  |
| Increased number of published models where production schedule adherence had no issues |  |  |  |
| Ease of use or Usability of the product |  |  |  |
| Speed to On boarding |  |  |  |
| Serviceability : Ease with which system can be maintained and repaired |  |  |  |
| System throughput under a given workload for a specific timeframe |  |  |  |
| Reliability, Availability (High Availability-HA) and Scalability  (RAS) |  |  |  |

# Future work

|  |  |  |  |
| --- | --- | --- | --- |
| Future features | Purpose | Priority | Timeframe |
|  |  |  |  |
|  |  |  |  |