data iku

PUTTING DATA SCIENCE IN PRODUCTION

Finding the Common Ground in 9 Steps

WHITE PAPER

www.dataiku.com

SUMMARY

INTRO_ DUCTION

Managing a successful data science project requires time, effort, and a great deal of planning. Defining the problems to solve and planning the project's scope is just the tip of the iceberg, as team members need to fully understand all aspects of a project in order to effectively contribute.

A critical challenge of data science projects is getting everyone on the same page in terms of project challenges, responsibilities, and methodologies. More often than not, there is a disconnect between the worlds of development and production. Some teams may choose to re-code everything in an entirely different language while others may make changes to core elements, such as **testing procedures, backup plans, and programming languages.**

Transitioning a data product into production could become a nightmare as different opinions and methods vie for supremacy, resulting in projects that needlessly drag on for months beyond promised deadlines.

The goal of this guide is to explore grounds for commonality and introduce strategies & procedures designed to bridge the gap between development and production. The topics range from **Best Operating Procedures** (managing environmental consistency, data scalability, and consistent code & data packaging) to **Risk Management** for unforeseen situations (roll-back and failover strategies). We also discuss modelling (continuously re-train of models, A/B testing, and multivariate optimization) and implementing communication strategies (auditing and functional monitoring).

Successfully building a data product and then deploying it into production is not an easy task — it becomes twice as hard when teams are isolated and playing by their own rules.

Hopefully this guide will help your organization find the common ground needed to empower your Data Science and IT Teams to work together for the benefit of your data projects as a whole.

CONSISTENT PACKAGING AND RELEASE_

In a data science production environment, there are multiple workflows: some internal flows correspond to production while some external or referential flows relate to specific environments. Moreover, data science projects are comprised of not only code, but also data:

- Code for data transformation
- Configuration and schema for data
- Public referential data, such as postcodes / zip codes
- Internal referential data, such as product category descriptions.

That's why, to support the reliable transport of code and data from one environment to the next, they need to be packaged together.

REAL WORD USAGE_

According to our statistics, a typical data science project over the course of a 6-week timespan would include:

AN AVERAGE OF **700 LINES** OF SQL;

AN AVERAGE OF **2,500 LINES** OF PYTHON;

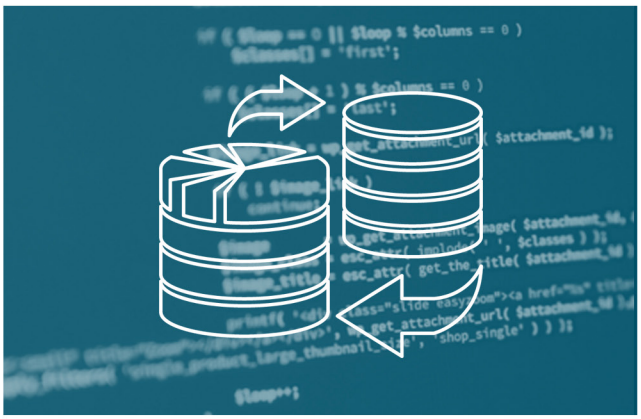
AN AVERAGE OF **200 LINES** OF JAVASCRIPT;

150 LINES OF CSV;

350 LINES OF CONFIGURATIONS (OR SCRIPTS) THAT COMBINE ALL OF THESE ELEMENTS;

WHY IS THIS IMPORTANT?_

If you do not support the proper packaging of code and data, the end result is inconsistent code and data during production.
These inconsistencies are particularly dangerous during training or when applying a predictive model.
The reason is because these inconsistencies, which are quite difficult to detect, can lead to a subtle degradation of the model's performance between development and production.
Without proper packaging, the deployment in production can pose significant challenges.



WORDS FROM THE TRENCHES_

"Where did Mike put the 10GB reference file he used to create the model? We only have the 10MB sample on Git!"

THE SOLUTION_

- [ ] The first step is to establish a versioning tool, such as Git, to manage all of the code versioning within your product.

- [ ] The next step is to package the code and data. Create packaging scripts that generate snapshots in the form of a ZIP file for both code and data inside the script; these should be consistent with the model (or model parameters) that you need to ship. Deploy that ZIP file to production.

- [ ] Lastly, remain aware of situations when the data file size is too large (e.g., > 1GB). In these scenarios, you need to snapshot and version the required data files in a storage.

CONTINUOUS RE-TRAIN OF MODELS_

It is critical to implement an efficient strategy for the re-training, validation, and deployment of models. The process needs to follow an internal workflow loop, be validated, and then passed on to production (with an API to log real outcomes).

In data science projects, predictive models need to be updated regularly because:

- in a competitive environment, models need to be continuously enhanced;
- the environment changes (e.g., new customers with new behaviors); and,
- the underlying data is constantly changing.

REAL WORD USAGE_

In our experience, there are 3 typical re-training scenarios:

People involved with online learning. Today, online learning systems are particularly popular for recommender systems and online advertising platforms;

People who automatically re-train and update their model (typically every 2.3 days); and,

People who re-train their model offline and then manually release it to production (typically updated every 24 days). This group includes the greatest number of participants who are unsure of how many models they have put into production within the last 6 months.

WHY IS THIS IMPORTANT?_

If your organization has not implemented a "re-train in production" methodology, then deploying a model becomes an actual "deploy to production" task… with the result requiring **significant manpower and a loss of agility.** If relying on a manual approach, then the model deployment could consume from 5 to 30 days of manpower.

In addition, the cost of not deploying a re-trained model could be huge. In a typical real-life situation, a scoring model AUC could degrade by 0.01 per week due to the natural drift of input data — remember, Internet user behavior changes along with its related data! This means that the 0.05 hard performance optimization that was painstakingly tuned during project setup could disappear within a few weeks.

WORDS FROM THE TRENCHES_

"I've re-played the cross validation score for the model and, after 4 weeks in production, we're back to square one. Should I tell my boss?"

THE SOLUTION_

The solution to the re-training challenge lies in the data science production workflow. This means that you need to implement a dedicated command for your workflow that does the following:

☐ → Re-trains the new predictive model candidate.

☐ → Re-scores and re-validates the model (this step produces the required metrics for your model).

☐ → Swaps the old predictive model with the new one.

With regards to implementation, the re-train / re-score / re-validate steps should be automated and executed every week. The final swap is then manually executed by a human operator that performs the final consistency check. This approach provides a good balance between automation and reduced re-train cost while maintaining the final consistency check.

AUTOMATED MODEL CHECKING_

Automated model checking is the testing and comparison of metrics on old & new models, with a new model put into production when its performance exceeds its predecessor. This comparison process is done post re-validation and should reflect current data (e.g., daily model updates). A more complex method, called Multivariate Testing, can be used in scenarios that require the continuous testing of multiple models.

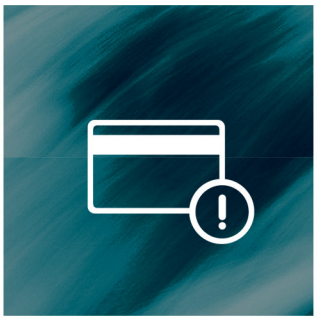
FROM A/B TESTING TO MULTIVARIATE OPTIMIZATION_

The purpose of A/B testing different models is to be able to evaluate multiple models in parallel and then comparing expected model performance to actual results.

WHY IS THIS IMPORTANT?_

Offline testing is not sufficient when validating the performance of a data product. Here are a few reasons why:

- In use cases such as credit scoring and fraud detection, only real world tests can provide the actual data output required. Offline tests are simply unable to convey real-time events, such as credit authorizations (e.g., is the credit offering aligned with the customer's repayment ability?);
- A real-world production setup may be different from your actual setup. As mentioned above, data consistency is a major issue that results in misaligned productions;
- If the underlying data and its behavior is evolving rapidly, then it will be difficult to validate the models fast enough to cope with the rate of change.



WORDS FROM THE TRENCHES_

"We put the model in production just after Christmas. The performance dropped. It took us two weeks to convince the business team that it was perfectly normal at this time of the season, and not due to the model's recent release."

THE SOLUTION_

There are three levels of A/B testing that can be used to test the validity of models:

- Simple A/B testing
- Multi-armed bandit testing; and
- Multi-variable armed bandit testing with optimization.

The first, simple A/B testing, is required for most companies engaged in digital activities while the latter is used primarily in advanced, competitive real-time use cases (e.g., real-time bidding/ advertising and trading).

SIMPLE A/B TESTING

Simple A/B testing is the maintenance of several versions of your product that serves different versions of your model; the goal is to compare the end performance of your product according to its associated model.

Implementation of simple A/B testing is facilitated by:

- ☐ Using A/B testing providers, such as Optimizely and AB Tasty;
- ☐ Implementing your own A/B testing in the front-end of your application. Check out open source frameworks to get a headstart, such as GitHub's Alephbet; and,
- ☐ implementing A/B testing in the backend of your application. Have a look at Proctor, a Java-based A/B testing framework and/or Split, a Rack-based A/B testing framework.

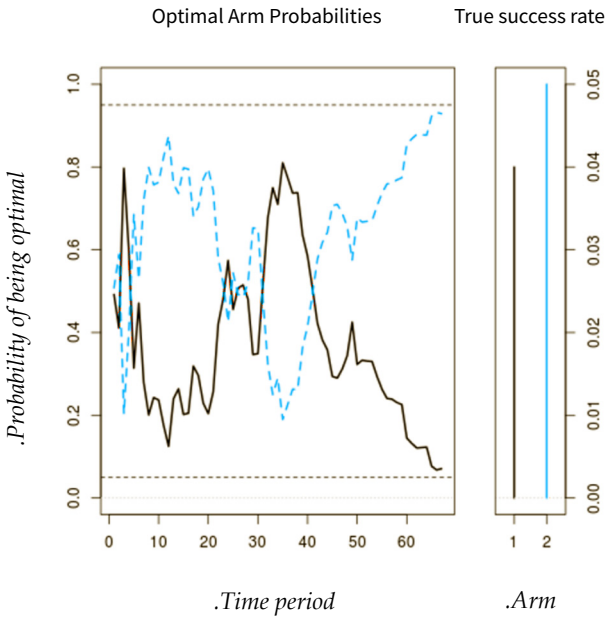
Note that the interpretation of an A/B test is not always obvious, particularly if one outcome is less frequent than the other — one case (e.g., actual transformation for model B) may not be frequent enough to draw conclusions. In these instances, the use of statistics is helpful.

MULTI-ARMED BANDIT

Although simple A/B testing is a satisfactory validation method, algorithms can be used to automatically select the best model. One algorithm in particular excels at this function: multi-armed bandit.

In the context of predictive analytics, multi-armed bandit works by continuously maintaining several models that are tested against each other. The algorithm then gathers live information about positive outcomes; if the model predicts the truth, then the end goal has been achieved. During the testing process, the ratio of each model is adjusted in order to provide more performance to the best performing algorithm. Other models are still tested, however, in order to address situations when a model's performance exceeds that of another.

Internally, the multi-armed bandit algorithm tries to estimate the probability of a given model becoming the "best one," with the caveat that "best one" status could be assigned based on luck (i.e., the data received so far enabled the model to outperform its competition).



More information about the multi-armed bandit algorithm can be found in the book ***Bandit Algorithms for Website Optimization.*** This book contains multiple examples of sample code that can assist you in the creation of your own algorithm; it is available online via GitHub.com. If you use Python, then Flask-MAB is a useful package for multi-armed bandits.

MULTI-VARIABLE ARMED BANDIT WITH OPTIMIZATION

In situations involving advanced real-time use cases, we recommend multi-variable armed bandit with optimization.
Full multi-variable testing involves continually running different models, with optimal variants being selected based on specific variables. For example, one model may be applicable for European customers while another model is used for Asian customers.

CONCLUSION_

Ideally, you should implement several levels of live testing with the following characteristics:

- Ability to compare your predictive model against a baseline;
- Ability to compare two live versions of your predictive models;
- Ability to continuously optimize and trade-off multiple versions of your predictive models based on their context.

To learn more about how multi-armed bandits work in terms of a Google Analytics content experiment, please visit this Google Support page: https://support.google.com/analytics/answer/2844870?hl=en

FUNCTIONAL MONITORING_

Functional monitoring is used to convey key functionality of the business model's performance to the business sponsors/owners. From a business perspective, functional monitoring is critical because it provides an opportunity to demonstrate the end-results of your predictive model and how it impacts the product. The kind of functional information that can be conveyed is variable and depends largely on the industry and use case. Examples of the kind of data displayed can include the number of contacts in a case, the number of broken flows in a system, and measurements of performance drifts.

REAL WORD USAGE_

Some applications of functional monitoring based on industry or functional need include:

Fraud: The number of predicted fraudulent events, the evolution of the prediction's likelihood, the number of false positive predictions, and rolling fraud figures;

Churn Reduction: The number of predicted churn events, key variables for churn prediction, and the efficiency of marketing strategies towards churners (e.g., opening rates of e-mails);

Pricing: Key variables of the pricing model, pricing drift, pricing variation over time, pricing variation across products, evolution of margin evaluations per day/year, and average transformation ratios.

BUSINESS SPONSOR INVOLVEMENT_

In most situations, business sponsors must have the capability to detect early signs of drift. This is possible by enabling sponsors to easily dig into the model's characteristics, view its history, determine the drift's validity, and then take appropriate action. For example, a marketing campaign with more customers from a certain age group (e.g., 20-30 years old) could result in an inaccurate transformation ratio prediction due to the relative inconsistency of that group's consumer behavior.

In addition, business sponsors must be provided with access to high-level technical errors. For example, if a pricing model is lacking data from a specific category, the business owner needs to be notified of the missing data so that they are aware of factors that impact their strategies.

WHY IS THIS IMPORTANT?_

Knowledge transparency must be constantly evangelized throughout an organization at every opportunity. A lapse in communication can compromise the importance of using machine learning technology within your organization.

WORDS FROM THE TRENCHES_

"One day, our CEO spotted a funny recommendation on the company website. We realised that part of the rebuild chain had been broken for 5 days without anyone noticing. Well, we decided to keep this to ourselves."

THE SOLUTION_

A successful communication strategy lies at the heart of any effective organization; such a strategy typically combines multiple channels:

☐ Channel for the quick and continuous communication of events — these are channels where events are seamlessly communicated to team members, such as:

- New model in production;
- Outliers in production;
- Drop or increase in model performance over the last 24 hours.

☐ E-mail based channel with a Daily Report. Such a report should be a succinct summary of key data, such as:

- A subject with core metrics;
- Top n customers matching specific model criteria;
- Three model metrics (e.g., a technical metric, high-level long-term metric, and a short-term business metric).

☐ A Web-based dashboard with drill-down capability; other channels should always include links to the dashboard in order to drive usage.

A real-time notification platform, such as Slack, is a popular option that provides flexible subscription options to stakeholders. If building a monitoring dashboard, visualization tools such as Tableau and Qlik are popular as well.

IT ENVIRONMENT CONSISTENCY_

The smooth flow of the modelling process relies heavily on the existence of a consistent IT environment during development and production. Modern data science commonly uses technologies such as Python, R, Spark, Scala, along with open source frameworks/libraries, such as H2O, scikit-learn, and MLlib.

In the past data scientists used technologies that were already available in the production environment, such as SQL databases, JAVA, and .NET.
In today's predictive technology environment, it is not practical to translate a data science project to older technologies like SQL, JAVA, and .NET — doing so incurs substantial re-write costs. Consequently, 80% of companies involved with predictive modelling use newer technologies such as Python and R.

80% companies

WHY IS THIS IMPORTANT?_

Putting Python or R into production poses its own set of unique challenges in terms of environment and package management. This is the case due to the large number of packages typically involved; data science projects rely on an average of 100 R packages, 40 Python packages, and several hundred Java / Scala packages (most of which are behind Hadoop dependencies). Another challenge is maintaining version control in the development environment; for example, scikit-learn receives a significant update about twice a year.

WORDS FROM THE TRENCHES_

"I'm never trusting them with R in production again. Last time we attempted to deploy a project with multiple R packages, it was literally a nightmare"

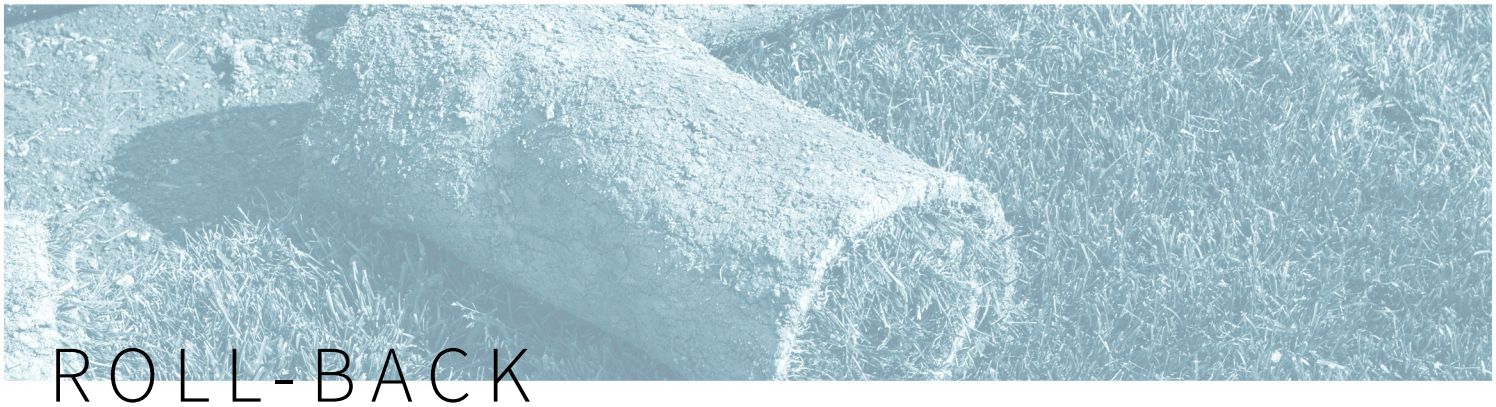
THE SOLUTION_

Fortunately, there are multiple options available when establishing a consistent IT environment, such as:

- ☐ Use the builtin mechanisms in open source distributions (e.g., virtualenv, pip for Python) or rely on 3rd party software (e.g., Anaconda™ for Python). Anaconda™ is becoming an increasingly popular choice amongst Python users, with one-third of our respondents indicating usage. For Spark, Scala, and R, a vast majority of the data science community is relying solely on open source options;

- ☐ «Use a build from» source system (e.g., pip for Python) or a binary mechanism (e.g., wheel). In the scientific community, binary systems are enjoying increased popularity. This is partly due to the difficulty involved in building an optimized library that leverages all of the capabilities of scientific computing packages, such as NumPy;

- ☐ Rely on a stable release and common package list (in all of your systems) or build a virtual environment for each project. In the former, IT would rather maintain a common list of "trusted" packages and then push those packages to software development. In the latter, each data project would have its own dedicated environment. Remember that the first significant migration or new product delivery may require you to maintain several environments in order to support the transition.



ROLL-BACK STRATEGY_

A roll-back strategy is required in order to return to a previous model version after the latest version has been deployed.

REAL WORD USAGE_

Since a subtle impact on your model production may be visible after a few days or weeks, a roll back is required. For example, a model may show a 1% or 2% drop in performance after a new release. In our studies, 82% of companies had to roll-back at least once after initially putting predictive modelling technologies into production.

82%
of companies roll-back

WHY IS THIS IMPORTANT?_

Without a functional roll-back plan, your team may face an existential crisis the first time something goes wrong with the model. A roll-back plan is like an insurance policy that provides a second chance in the production environment.

WORDS FROM THE TRENCHES_

"When our model started showing a 3% drop in performance, we all panicked. Getting back to a previous version of the model took us over 4 days!"

THE SOLUTION_

A successful roll-back strategy must include all aspects of the data project, such as:

- ☐ TRANSFORMATION CODE
- ☐ DATA
- ☐ SOFTWARE DEPENDENCIES
- ☐ DATA SCHEMAS

The roll-back will need to be executable by users who may not be trained in predictive technologies, so it must be established as an accessible and easy-to-use procedure that could be implemented by an IT Administrator. Roll-back strategies must be tested for usage in a test environment and be accessible in both development and production environments.

ROBUST DATA FLOW_

Preparing for the worst is part of intelligent strategizing; in the world of predictive analytics, this means having a robust failover strategy. After all, robust data science applications must rely on failover and validation procedures in order to maintain stability. A failover strategy's job is to integrate all of the events in the production system, monitor the system in case the cluster is doing poorly, and immediately alert IT if the job is not working. The question that needs to be asked is, how do you script a process that would re-run or recover in case of failure?

CHALLENGES IN STRATEGY FORMULATION_

WHAT IF...

Creating a failover strategy, and developing relevant scripting solutions, requires an awareness of multiple "What if...?" scenarios:

- What happens when some data is missing?
- What happens when there is a new column in the underlying data?
- What happens when there is a change in the data?
- What happens when the workflow relies on an external API (e.g., geocoding, social data), but the API cannot be reached?
- What happens if the script's computing time exceeds the expected limit? For example, the time for an overnight job might oscillate from one hour to several hours, effectively making it difficult to finish the job within an expected time range. There could be multiple underlying reasons behind the time variations, such as a physical error on a server or resources being pulled by another process.

ADOPTING ETL METHODS

In traditional business intelligence systems, ETL (extract - transform - load) provides some failover mechanisms, but the failover and validation strategies depend on the intrinsic knowledge of the application. In addition, ETL technologies do not connect well to Python, R, predictive models, and Hadoop. Some levels of access are possible, but it is unlikely that the level of detail required for reliable scripting exists. For example, when running on Hadoop, it is common to leverage the underlying MD5 and hash facilities in order to check file consistency and store/manage the workflow. This capability is typically not easy to do with ETL.

BIG DATA METHODS

Formulating a failover strategy in a big data workflow presents some unique challenges, mostly due to the sheer volume of the data involved. It's not feasible to take a "rebuild" approach, as there is just too much information to do this efficiently. Given this, a big data workflow must be "state-aware," meaning that it must make decisions based on a previously calculated state. As expected, ETL methods are typically not capable of encoding this kind of logic.

WHY IS THIS IMPORTANT?_

Without a proper failover strategy, your data workflow will fail. The result will be a loss of credibility for using a data science approach in your IT environment. It is important to dedicate time and attention to the creation of your failover strategy, as they are notoriously difficult to perfect the first time around.

WORDS FROM THE TRENCHES_

"Our whole rebuild was relying on a catalog file updated daily by a third party, from an FTP file. One day, the file was empty and everything went berserk. I wish I had implemented some tests and fallback scripts in anticipation."

THE SOLUTION_

The Ends: The end-points (start & finish) require particular attention, as they are frequently the weakest points of a big data workflow. In a real-world environment, a weak point can be an FTP server, an API, or even a daily e-mail with a CSV attachment. When working with end-points, your scripts should meticulously check for errors codes, file size, and so on;

Be Parallel: In our study, 90% of big data workflows have multiple branches, with an initial data input used in two ways and then merged together to train the predictive model. This means that some processing can be parallelized at the workflow level, as opposed to the cluster, map reduce, and Spark level. As the product evolves, it is likely that the number of branches will grow — using a parallel methodology helps to keep your system fast;

Intelligent Re-execution: This simply means that data is automatically re-updated after a temporary interruption in data input, such as a late update or temporarily missing data. For example, your big data workflow may retrieve daily pricing data via FTP; your workflow combines this data with existing browser and order data in order to formulate a pricing strategy. If this 3rd party data is not updated, the pricing strategy can still be created using existing up-to-date data… but, ideally, the data would be re-updated when the missing data becomes available.

User Interface: Graphically conveying workflow data enables users to more fully understand, and investigate, the overall progress of the workflow. At some point a textual interface, or raw logs, reach their limit in terms of being able to describe the big picture. When this happens, an easy-to-use Web-based user interface is the best option.

USING A WORKFLOW FRAMEWORK_

Some programming environments, such as Spark, provide a consistent workflow mechanism. This is a good option if you do not have requirements for other technologies and if your data science processes are created by software developers. For large integrated workflows, a programming framework such as Cascading may be a wise choice if you want to implement a single framework / language.

AUDITING_

Auditing, in a data science environment, is being able to know what version of each output corresponds to the code that was used to create it. In regulated domains, such as healthcare and financial services, organizations must be able to trace everything that is related to a data science workflow. In this context, organizations must be able to:

Trace any wrongdoing down to the specific person who modified a workflow for malicious purposes;

Prove that there is no illegal data usage, particularly personal data;

Trace the usage of sensitive data in order to avoid data leaks;

Demonstrate quality and the proper maintenance of the data flow.

Providing proof during an audit is best facilitated via logs that convey detailed information, such as when data was modified, which code modified the data, when it was modified (i.e., date & timestamp), and the user(s) involved in the modification.

WHY IS THIS IMPORTANT?_

Failure to comply with auditing requirements, particularly in highly-regulated sectors, can have a profound impact on smooth business continuity. Regulatory-sensitive organizations may run the risk of heavy fines and/or the loss of a highly-coveted compliance status.

Non-regulated companies still must meet auditing requirements in order to understand exactly what's going on with their data and workflows, especially if they are being compromised. The ramifications of not implementing an auditing strategy is typically felt the most when a data science practice moves from the arena of experimentation to actual real-world production and critical use cases.

WORDS FROM THE TRENCHES_

"Because we hadn't designed a proper auditing scenario, they stopped us from deploying a project we'd been working on for 8 months. To them and to their legal framework, the risk of data leaks was just too risky."

THE SOLUTION_

In order to implement a comprehensive auditing strategy, you need to be able to:

☐ Version all changes in terms of parameters, configuration, and code from your models using a configuration management system (e.g., Git);

☐ Maintain and gather log information from your database systems, including table creation, modification, and schema changes. The log should also include information about the person responsible for the data/coding along with the identifier used for the job/program that launched the process.

With regards to specific tools:

- Hadoop: HDFS AuditLogger is a good first step, but ultimately not sufficient because it does not provide complete audit logs for other applications (e.g. Hive, Hue);
- Cloudera: Cloudera Navigator Audit;
- Hortonworks: Ranger.

PERFORMANCE AND SCALABILITY_

Performance and scalability go hand-in-hand: as scalability limits are tested (more data / customers / processes), performance needs to meet or exceed those limits. Strategically, the challenges lies in being able to create an elastic architecture; the kind of environment that can handle significant transitions (e.g., from 10 calls per hour to 100k per hour) without disruption. As you push your data science workflow into production, you need to consider appropriate increases in your production capability.

Volume Scalability: What happens when the volume of data you manage grows from a few gigabytes to dozens of terabytes?;

Request Scalability: What happens when the number of customer requests is multiplied by 100?;

Complexity Scalability: What happens when you increase the number of workflows, or processes, from 1 to 20?;

Team Scalability: Can your team handle scalability-related changes? Can they cooperate, collaborate, and work concurrently?

WHY IS THIS IMPORTANT?_

Obviously, there is no silver bullet to solve all scalability problems at once. Some real-world samples, however, may help to illustrate the unique challenges of scalability and performance:

- **Overnight Data Overflow**: Multiple dependent batch jobs that last 1 or 2 hours tend to eventually break the expected timespan, effectively running throughout the night and into the next day. Without proper job management and careful monitoring, your resources could quickly be consumed by out-of-control processes;

- **Bottlenecks**: Data bottlenecks can pose a significant problem in any architecture, no matter how many computing resources are used. Regular testing can help to alleviate this issue;

- **Logs and Bins**: Data volume can grow quickly, but at the vanguard of data growth are logs and bins. This is particularly true when a Hadoop cluster or database is full — when searching for a culprit, always check the logs and bins first as they're typically full of garbage.

WORDS FROM THE TRENCHES_

As our team grew, so did the number of workflows we were running. The nightly build slowly shifted, finishing at 10am, then at 11am, and so on. We started to worry the day it took 26 hours.

THE SOLUTION_

In order to address common issues related to scalability and performance, consider the following solutions:

- ☐ Build a proactive strategy for dangling data detection. For example, a script could inspect the configuration of your data science workflows and list every file that is no longer of any use;
- ☐ Actively monitor job execution time;
- ☐ Actively Leverage large-scale technology, such as Spark, for large demanding jobs.

CONCLUSION

The ultimate success of a data science project comes down to contributions from individual team members working together towards a common goal. As can be seen from the topics discussed, “effective contribution” goes beyond specialization in an individual skill-set. Team members must be aware of the bigger picture and embrace project level requirements, from diligently packaging both code and data to creating Web-based dashboards for their project’s business owners. When all team members adopt a “big picture” approach, they are able to help each other complete tasks outside of their comfort zone.

Data science projects can be intimidating; after all, there are a lot of factors to consider. In today’s competitive environment, individual silos of knowledge will hinder your team’s effectiveness. Best practices, model management, communications, and risk management are all areas that need to be mastered when bringing a project to life. In order to do this, team members need to bring adaptability, a collaborative spirit, and flexibility to the table. With these ingredients, data science projects can successfully make the transition from the planning room to actual implementation in a business environment.

ABOUT DATAIKU

Dataiku strives to be the acknowledged advanced analytics leader and preferred software solution in helping organizations succeed in the world's rapidly evolving data-driven business ecosystem. Guided by the belief that true innovation comes from the effective combination of diversity of cultures, of mindsets, and of technologies, Dataiku's purpose is to enable all enterprises to imagine and deliver the data innovations of tomorrow.

ABOUT DATAIKU DSS

(DATA SCIENCE STUDIO)

Dataiku DSS is a collaborative data science software platform that enables teams to explore, prototype, build, and deliver their own data products more efficiently. It is an open platform designed to accommodate rapidly evolving programming languages, big data storage and management technologies and machine learning techniques, and is conceived to accommodate the needs and preferences of both beginning analysts and expert data scientists. It also uniquely support:

Collaboration

Collaboration features make it easy to work as a team on ambitious data projects, to share knowledge amongst team members and to onboard new users much faster. You can add documentation, information or comments on all DSS objects.

Reproducibility

Every action in the system is versioned and logged through an integrated Git repository. Follow each action from the timeline in the interface, with easy rollback to previous versions.

Production Deployment

DSS lets you package a whole workflow as a single deployable and reproducible package. Automate your deployments as part of a larger production strategy. Run all your data scenarios using our REST API.

Governance and Security

DSS helps you create clearly defined projects and make sure your data is organized. And with fine grained access rights, your data is available only to the right persons.

Try Dataiku DSS for free by visiting www.dataiku.com/try