A HOW-TO GUIDE ON SCALING DATA SCIENCE FOR ENTERPRISE

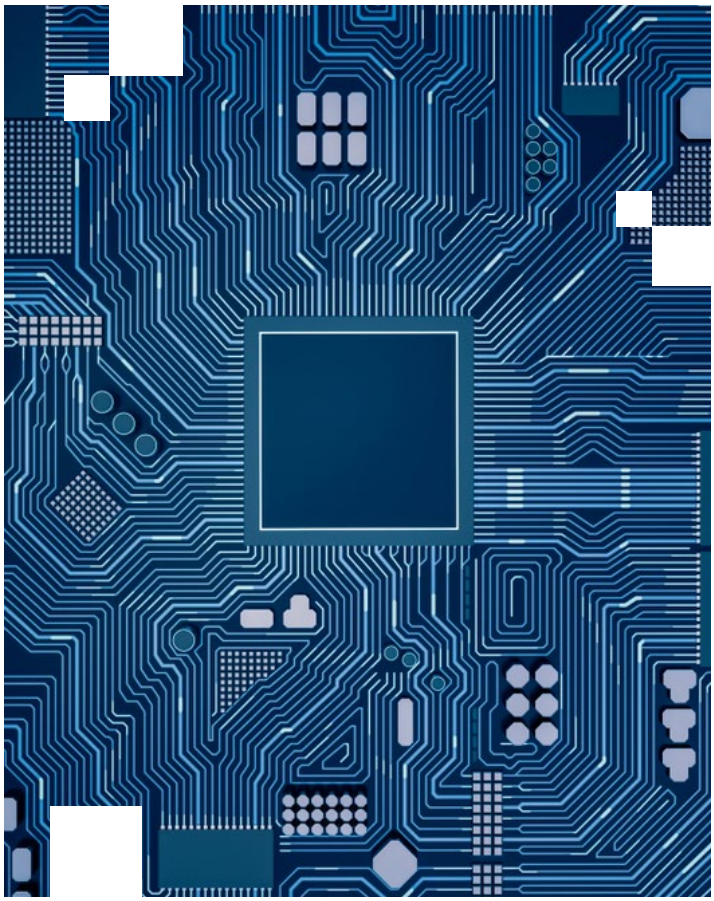
GIRI TATAVARTY

June 2018

A HOW-TO GUIDE ON SCALING DATA SCIENCE FOR ENTERPRISE

INTRODUCTION

Traditionally, retailers have been using data and analytics to improve logistics, pricing, assortment, promotion and marketing efforts. However, the deluge of big data with e-commerce and availability of open source machine learning tools create both opportunities and challenges for scalability to retailers. Doing Data Science at scale involves a radically different approach towards picking the right processes and tools in data preparation, data management, analysis, visualization, automation and execution of models. There are many problems where addition of large datasets to model training leads in impressive results and better models (such as spell check, image recognition, recommender systems); however, there may be cases where any amount of additional data only results in insignificant improvement. It is important to understand the nature of the problem and then decide what is the correct scale and data that is needed to solve the problem in a meaningful and efficient manner. This white paper will mainly cover three types of scalability – amount of data, number of models and finally organizational scale with respect to number of data scientists and problems that are solved. There are also some real-life case studies presented which serve as a template to scale the data science solutions.

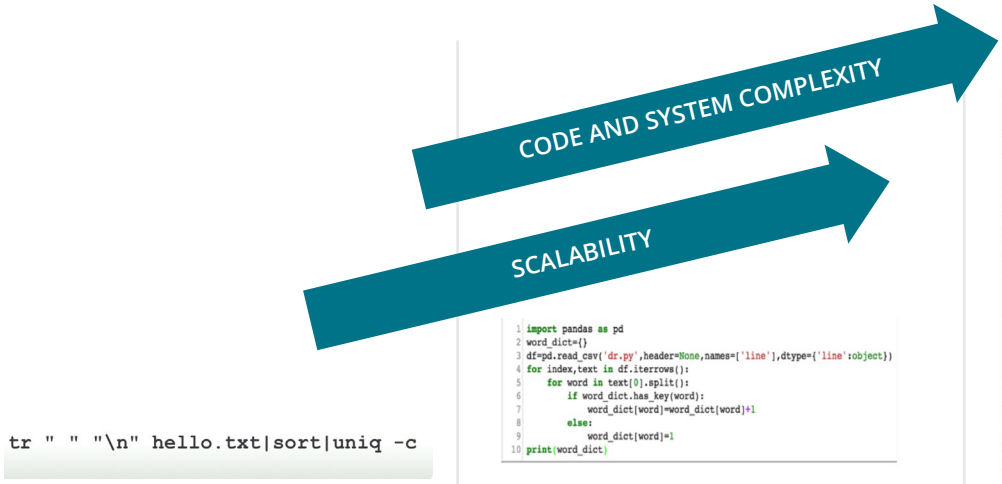


SCALE

The scale of data and operations has grown exponentially over the last few years. As an example: The top e-commerce retailer in USA has approximately 600 million products listed. Building recommender systems at this scale is a challenge. Also, if we are trying to better understand customer sentiment or trends, then there are around 200 billion tweets per year and billions more forms of user-generated content on other social media platforms. To build data science products or tools at this scale requires us to put scale at a foundational level. Discussions about scale must be done very early in the design process and will determine the choice of data storage platforms, analysis platforms and tools and finally the choice of visualization tools. It would be naïve and at worst a lost opportunity to proceed with solving a data science problem first and then later thinking about scalability.

Scale also comes with a price - complexity of the solution along with the specialized talent and tools to build scalable data science solutions. As an example, let's look at a simple program which counts the number of words in a document. Simply, with a given a document, the program finds the number of occurrences of each unique word. The following figure shows three different solutions developed by three different developers – an enthusiast, data scientists or graduates and a specialized data scientist. The results are same with one line of code vs 167 lines of code, however the code written by data engineer can scale up to file(s) sizes of Terabytes or even larger.

SCALE COMES AT THE COST OF COMPLEXITY



SINGLE LINE OF BASH SCRIPT
Written for exploration

10 LINES OF CODE
Written by Data Scientists, Grads

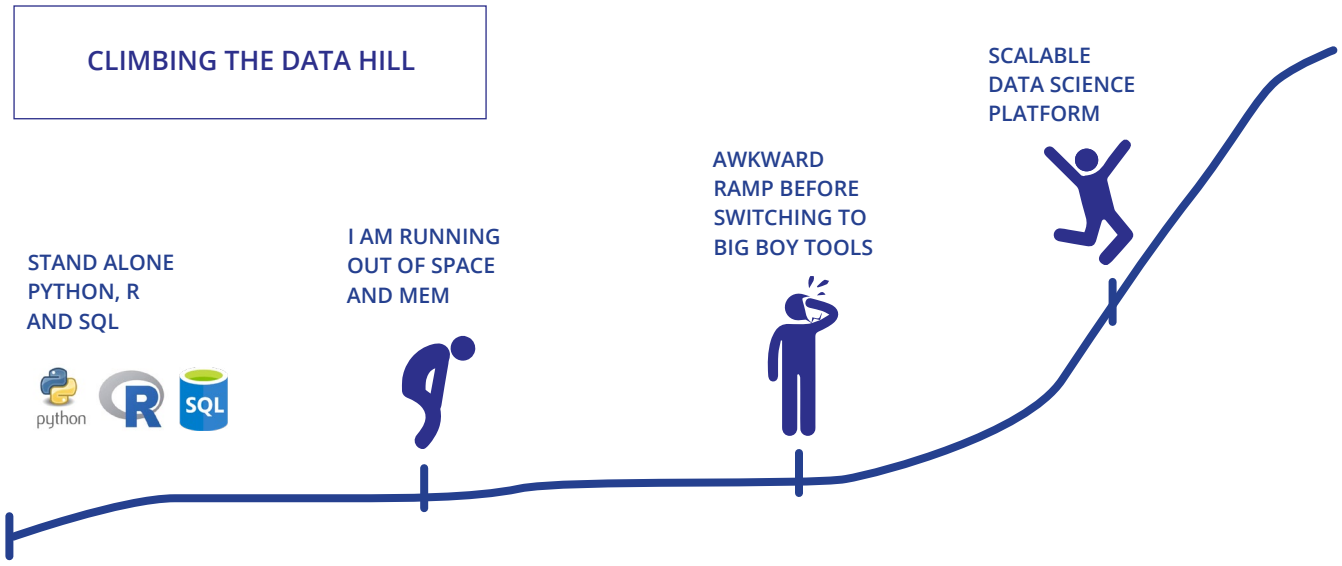
167 LINES OF APACHE BEAM CODE
Written by Data Engineer

SCALE DILLEMA

Many times, organizations start with the systems on hand such as open source languages like R and Python, and soon find that either they continuously need to update their code and systems to cope with the increased data demands or be faced with a redesign and implementation in a scalable data science platform. The earlier this decision is made, the smoother the journey will be. Also, the skills required for doing scalable data science are much specialized and come from following areas:

- **Databases and Distributed Computing:** Ability to write optimized SQL, knowledge of map-reduce programming methodology and executing DAG.
- **Algorithms and Statistics:** Ability to design approximate and faster/streaming versions of traditional algorithms, sampling techniques and knowledge of approximate statistical methods.
- **Systems Programming:** Ability to optimize code with lower level programming languages like C/C++, Java and CUDA/GPU programming knowledge.

IF YOU CANNOT FIT IT ON YOUR LAPTOP AND ANALYZE IT - IT IS **BIG DATA** AND IT IS TIME TO SCALE



TERABYTE DATASETS

The key to scaling is to understand which problems truly need terabytes of data and which don't. There are many problems where time tested statistical techniques such as sampling work very well alongside models that are built on a sample of data, which also generalize well and are representative of an entire population. In such cases, adding more data to models would only add complexity without giving benefits of additional data.

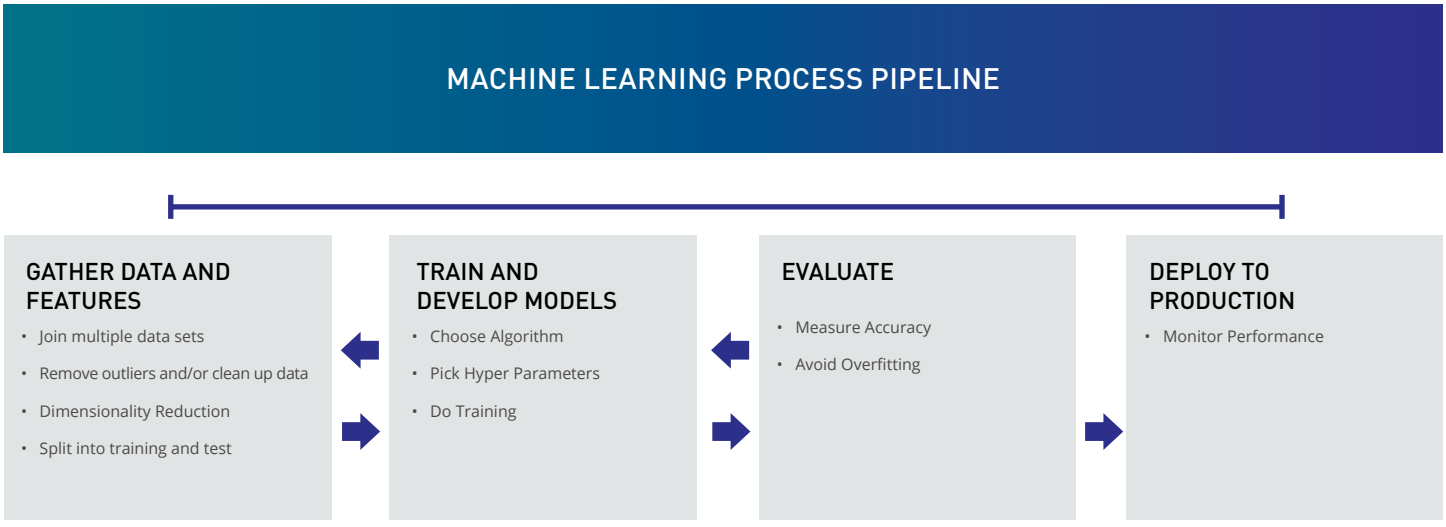
CASE STUDY 1:

PREDICT CUSTOMER VISITS AND SPEND

Let's consider a problem of predicting if a customer is going to visit a store next week and then predict which categories the customer is going to spend in. This is a classical supervised learning problem where we can train a classification machine learning model with customer information from past purchases and visits. Here is some of the data that can be used be for such a problem:

- Customer level transactions for period of two years. Billions of rows.
- Customer visits
- Customer spend on different departments and categories
- Weather
- Holidays

Typical steps in the solution would be described as a machine learning processing pipeline with steps such as gathering data, developing models, and evaluating and deploying the models.



SCALING THE ML PIPELINE FOR SUPERVISED LEARNING

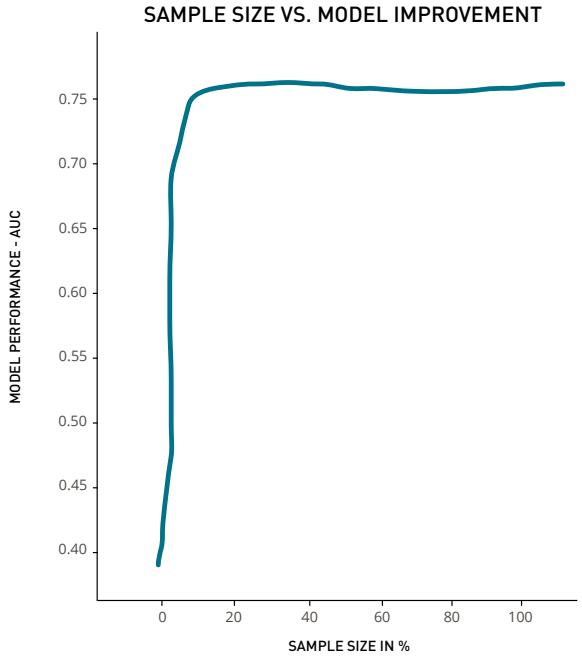
The key steps to scaling such ML models is utilizing the data store to do much of the aggregations and number crunching. Also, it helps to organize the data store by partitioning by customers split in an equal number of partitions (e.g. 100). If data is partitioned well, we can repeat the analysis for 1% or 10% or any percent easily. It also pays back to reduce the dimensionality of the problem by reducing the key features to be less than 100. Fast PCA, auto encoders are some examples of scalable dimensionality techniques.

While picking the actual ML tools and platforms, there are only a few choices such as H2O, SPARK ML, VOWPAL wabbit, and TensorFlow which are scalable with large amounts of data. Also, start with a simpler model such as logistic regression or linear models. Add complex models only when the benefits are justified in increased performance. While performing the evaluation, it is important to record all the experiments and results in a structured database. Sacred and MIT Model DB are a couple examples of experimentation frameworks.

The key questions that need to be answered after model building are:

1. How much data is enough to develop the model? At what point does adding more sample data not provide an increased benefit in terms of model performance?
2. What other sources of data and features can improve the model?

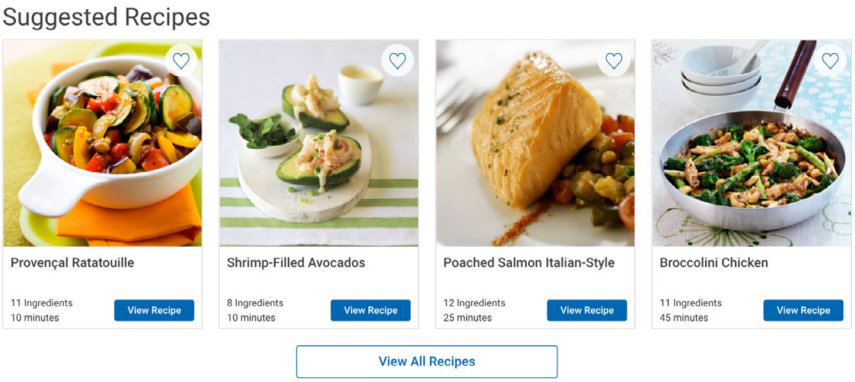
Once we can answer Question 1, we can stop adding more of the same data to the model and focus on adding different sources of data, which can improve model performance. In this case, it is very apparent that adding more of the same data to the model does not improve accuracy. The model built with 4-5% data performs just as well as the model built with 100% data. It is indeed preferable to have a simpler generalizable model than including the model with 100% of the data.



CASE STUDY 2:

BUILDING A RECOMMENDER SYSTEM

Let's consider an unsupervised learning of building a recommender system, which would recommend recipes and products based on past purchases.



The data required for this problem could include:

- Past purchase history
- Context – what is already in the cart?
- Seasonality
- Weather & geo-location

Collaborative Filtering and the Nearest Neighbor Algorithm are two common methods used in recommender systems. These algorithms match products based on attributes or behavioral data, and then recommend products bought by similar customers. These products can include those that a specific customer has not bought so far.

SCALING THE ML PIPELINE FOR SUPERVISED LEARNING

The biggest challenge with these types of problems is handling the data itself. As an example, if a retailer has 10 million UPCs and 10 million customers, the customer-product matrix has 10^14 cells, which cannot be possibly stored on any computer system currently. However, not every product is bought by every customer and such a matrix would be highly sparse. In such cases, we would like to store and process data in a sparse matrix only.

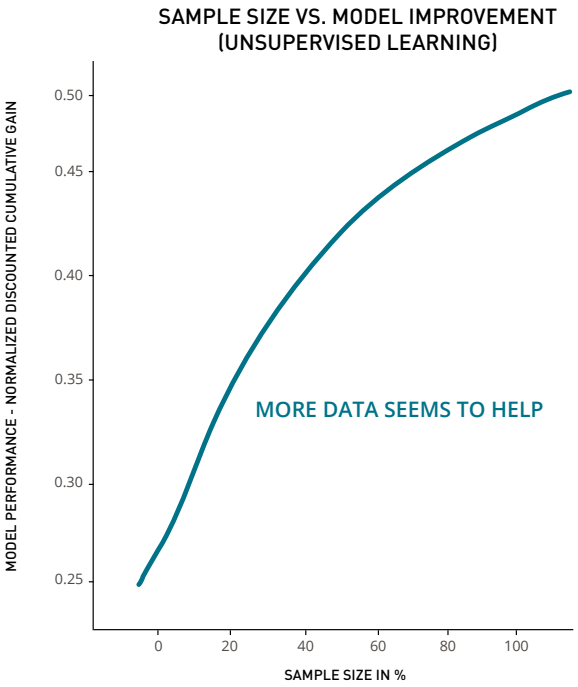
Customer ID	Product ID
Customer 1	Product 1
Customer 1	Product 3
Customer 2	Product 1
Customer 2	Product 2
Customer 2	Product 10^7
Customer 10^7+1	Product 1
Customer 10^7+1	Product 2
Customer 10^7+1	Product 10^7

10^14 CELLS OF DATA. MOSTLY ZERO

	Product 1	Product 2	Product 3	...	Product 10^7
Customer 1	x		x		
Customer 2	x	x			x
:					
Customer 10^7+1	x	x			x

There are many Graph databases which can store such data efficiently like neo4j, grafana, and graphX. It may also be beneficial to further develop the models at a department level. For example, customers who are shopping for clothing may not be interested in suggestions for groceries or produce.

Algorithmic improvements might include using approximate similarity instead of exact similarity for scalable computation. Techniques like locality sensitive hashing, approximate a k-nn scale well compared to their exact counterpart on large datasets. In these type of problems, large data will trump over sophisticated algorithms or specially engineered features. Below is the trend of performance of the recommender system with different sample sizes of data. We can observe that adding more data to the recommender system seems to improve the quality of recommendations. Other classes of problems where adding more data is beneficial are spell check systems, language translation, natural language processing systems, image classification, and word embeddings.



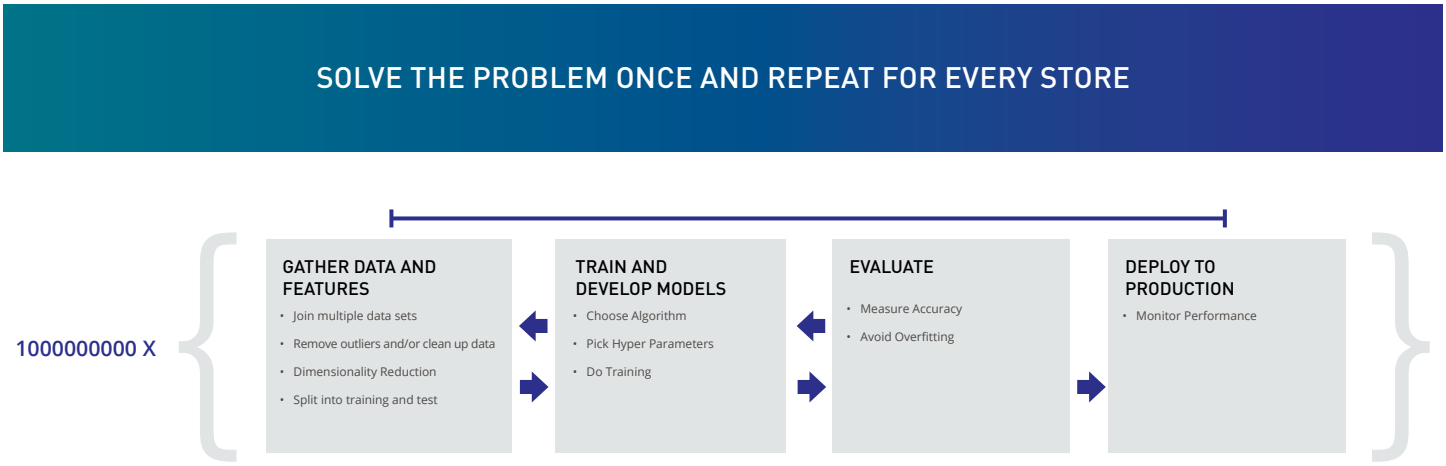
MILLIONS OF MODELS

Sometimes, scale does not come from the amount of data used in the model but the number of models that are needed to be built and maintained. An example of such problem is forecasting demand for inventory management. A retailer might have hundreds of stores each carrying thousands of products, which need to be ordered and replenished regularly. The ordering system uses the current inventory at hand and the forecast for the upcoming days to determine the quantity to be ordered and replenished. The complexity of the system evolves from the fact that each UPC is unique and the forecast depends on many variables such as seasonality, day of week, holidays, price, promotion activities and competitor pricing and campaigns. It is indeed a technical challenge to build millions of forecasting models for every UPC level for every UPC in every individual store.

CASE STUDY 3:

ENTERPRISE FORECASTING SYSTEM

A typical enterprise forecasting system predicts demand for every UPC in every store at least every day. The prediction model can use historical sales data, weather and holiday information, and pricing and promotion data for the enterprise and competitors. The models are not very complex and use linear regression models or time series based models; however, the number of models make it a challenge. The process is like case study 1, with the only difference that it needs to be repeated few million times on a much smaller amount of data.



The best process here is to first come up with a good naïve baseline model, then build and maintain models only if complex models are better than naïve baseline model. A simple naïve baseline model in this case can be something like forecasting the observed demand of the last week's same day data from observed sales data (or even use moving average). Linear regression based models are very fast to compute and score, so they are a natural first choice. Also, always use a fast linear algebra library for implementation. Some examples include ATLAS, MKL, Scipy, Armadillo, GSL and Eigen. Specialized CUDA/GPU packages can also be utilized to build these models quickly and in parallel.

It is also important understand the variance in the forecasts and have the model fallback to simpler or baseline models when high variance is observed. Bayesian models have an advantage of being able to cope with incomplete data or for new products or stores where there is not enough past sales data. Alternatively, Hierarchical Bayesian models utilize the similarity between sub-categories and categories of products, and then pool multiple models together to reduce variance. Finally, store the model coefficients only and use a distributed computing platform like Spark to efficiently score and build these models in parallel.

THOUSANDS OF PROBLEMS AND HUNDREDS OF DATA SCIENTISTS

Scaling the organization to solve large number of problems and support hundreds of data scientists requires a commitment to use the best-in-class tools and practices alongside a data-driven culture. It would require organizational investment in data science platforms, collaboration platforms and mechanisms that emphasize repeatability and openness to sharing code and data. Here are some of the best practices which are captured as Do's and Don'ts.

DON'T DO	DO
Doing data science from laptops alone	Invest in a data science platform - Dominos, Yhat, Dataiku, Microsoft, IBM, JupyterLab, R Studio Server
Code with folder names _latest , _v1 , _20171231 ,_1	Source control – Git
Sharing results and collaboration through EMAIL	Share through Notebooks and using a collaboration platform. One that has all the data and replicable code
Skype, Jabber chat	Use Persistent Chat – Slack, Mattermost
Installing all packages in the same environment	Use Virtualenv, containers
Direct installs on servers	Using package management software (e.g., Conda) and internal repositories. Use change tracking and change management software
Every project is started from scratch	Rich set of custom productivity modules, functions and templates. Examples and starter projects available as a starting point
Results established on a one-off data set that nobody else has access to	Results, code, datasets and dependent packages all are available to download and results can be reproduced with minimal effort
Share only successful results	Share positive and negative results

DON'T DO	DO
No comparable results or benchmarks given with results	Always report the results in CONTEXT of what is already there
Using datasets which are ad-hoc created	Enterprise GOLD datasets and benchmark datasets
Search emails and ask friends for data	Enterprise Data Catalogue
Rushed meetings and PowerPoint presentations	Open peer reviews and self-contained white paper published
Only one version of science is in production	Use an A/B testing framework which can run multiple versions of science simultaneously
Results for only of the latest examples are present	All experiments are recorded in a permanent database – Hyper parameters, data lineage, package versions and run times

CONCLUSION

Scale is foundational. Doing data science at scale involves ensuring that data management, data processing and data science platforms are indeed scalable. It also requires hiring the specialized talent who have expertise in the technology and science at hand. Each problem needs to be thoroughly studied before throwing all the available data at it. There are problems which seem to require large datasets; however, with solid statistical methods such as sampling, we can solve these problems efficiently with a lot less data. There are also data science problems such as unsupervised learning methods, where adding more data is beneficial by continuing to improve the solution. Production prediction systems require specialized data libraries and data stores to efficiently build and score at scale. Finally, organizational scaling in the number of problems tackled and the number of data scientists on staff will require commitment to openness of code and data while using the best-in-class tools, data science processes and adherence to the scientific method.