An Agile Approach to Data Science Project Management

Ben Ziomek*

Abstract— **There is a lot of discussion, but few frameworks, on effective AI Project Management. Industry-standard frameworks for data analysis projects, like CRISP-DM, exist but most are very high-level, and none are effective for managing the development of AI products from deployment to production, especially given the unique challenges of maintaining models in production. Even fewer are useful for understanding the skills needed by development process participants beyond core data science development skills. The result is that many data science teams are focused on outputting one-off analytical projects, rather than building long-term, maintainable products that directly drive business processes and goals. This paper proposes an effective project management framework for building production data science systems based on our experience constructing such systems at large tech companies, late-stage start-ups, and early-stage ventures. Specifically, this paper proposes that a blend of CRISP-DM and the Extreme Programming agile development methodology can serve as a starting point for building organization-specific data science product development processes.**

I. Introduction

Searching arXiv and Preprints for "AI Project Management" in March 2020 resulted in fewer than 20 papers published since 2008. The only relevant ones are about the application of AI models to general project management, not the other way around. There is a lot of discussion, but few frameworks, on effective AI Project Management. Industry-standard frameworks for data analysis projects, like CRISP-DM, exist but most are very high-level, and none are effective for managing the development of AI products from deployment to production, especially given the unique challenges of maintaining models in production. Even fewer are useful for understanding the skills needed by development process participants beyond core data science development skills. The result is that many data science teams are focused on outputting one-off analytical *projects*, rather than building long-term, maintainable *products* that directly drive business processes and goals.

This paper proposes an effective project management framework for building production data science systems based on our experience constructing such systems at large tech companies, late-stage start-ups, and early-stage ventures. Specifically, this paper proposes that a blend of CRISP-DM and the Extreme Programming agile development methodology can serve as a starting point for building organization-specific data science product development processes. This paper will step through the key insights in both CRISP-DM and Extreme Programming, and then propose a structure that combines the best of both frameworks, which were selected due to the lightweight and flexible natures. Our approach to managing data science product development focuses on the empirical testing, analysis, and the constant evaluation of algorithmic models. Because the evaluation metrics for data science systems are somewhat different than those for software development, this framework emphasizes the significance of testing for model accuracy and performance beyond what is normally captured in agile development processes.

* B. Ziomek is the CTO and Chief Product Officer at Actuate, a computer vision company focused on building software to turn any camera into a threat-detecting smart camera

II. CRISP-DM Meets Extreme Programming

CRISP-DM (Cross-Industry Standard Process for Data Mining) is an industry-proven methodology that offers a structured approach to planning a data mining project. The framework excels in aligning data mining tasks with specific business needs and objectives. While CRISP-DM works well for data mining projects, its structure is too rigid for an agile organization, and it completely omits the focus on testing and iteration that is necessary when building data science or machine learning systems for production rather than one-off data analysis projects. As it says in the name, CRISP-DM is a data mining framework, not a data science product development framework, and thus requires modification to effectively frame development processes for production models.

A. CRISP-DM Overview

CRISP-DM is broken up into six phases intended to give structure to any data mining project, from strategic planning to final implementation and evaluation:

1) *Business Understanding*: The first stage of the CRISP-DM process is to clearly identify the business objectives of a project and translate them into data science goals. Assessment of the company's current situation in terms of resource availability and any other constraints and possible contingencies must be addressed so trade-offs can be evaluated accurately. This phase should end with an initial project plan that includes step-by-step plans for the remainder of the project.
2) *Data Understanding*: All the data identified as project resources are collected and analyzed for relevance and quality.
3) *Data Preparation*: Data is selected, normalized, and clean to form a final analysis dataset from the initial raw data evaluated previously.
4) *Data Modeling*: Many different modeling techniques are tried and models are trained.

5) *Evaluation*: Model results are evaluated with respect to the business goals outlined in Phase 1. The key question is if the data mining results properly achieved all intended objectives or if an important task/factor was been overlooked.
6) *Deployment*: Data mining results are compiled and a plan for their use is assembled. These results and insights should then be presented and documented in a viable manner for stakeholders and management.

B. Limitations of CRISP-DM

Though the CRISP-DM methodology offers a strong functional blueprint for conducting analytics-oriented data mining projects, it lacks the focus on testing and productization that is necessary for the deployment of robust data science systems. Its waterfall-style "try everything" approach results in unproductive data science teams and neglects to utilize the domain knowledge and heuristics that all strong data scientists possess. At the same time it decouples development from testing, resulting in long feedback cycles that limit flexibility. Lastly, "deployment" in CRISP-DM basically means producing a document or deck. Anyone who has worked in an organization knows that decks are where strong analysis goes to die.

While it works for data mining projects, CRISP-DM is too rigid for data science products, which require a combination of deductive and empirical processes. CRISP-DM also offers little to no guidance for how model testing should be performed for maximum ROI (Return on Investment). This is where agile development comes in.

C. Lessons of Agile Software Development

While the existing data science literature is almost silent on the subject of production systems, luckily the software engineering world has spilled gallons of ink trying to find efficient project management frameworks beyond traditional Waterfall Development.

Over the last 20 years the tech industry has realized waterfall project management's inherent unsuitability for cloud software development and data science project management. Because waterfall methodologies follow a linear sequence of processes that require the completion of the previous phase before moving on to the next, they are far too inflexible for projects that require iterative feedback and testing.

Agile Development is a key innovation that serves as our starting point: The paradigm is flexible enough to be used for rocket science, and so should have no problems being applied to data science. However, modifications are still required. Despite the similarity in tools, software development and data science have key differences that limit the applicability of some of the more baroque aspects of development processes, such as scrum. We can't wholesale import many of these ideas into data science and machine learning projects without closely examining their value. As such, we start by investigating one of the earlier and more lightweight versions of agile development, Extreme Programming for lessons that can be applied to data science.

Agile development's focus on communication, people, the product, and flexibility make it much more suitable for the changing demands of the consumer market, and the fast-moving cloud and data science fields, allowing organizations to better hit the quickly-moving target of marketplace success. The Extreme Programming model, first proposed in the eponymous book, is one of the earliest, most portable, and most flexible of the Agile frameworks. We focus on adapting it to data science because it is unburdened by the over-optimization of more recent frameworks, allowing us to more easily adapt its processes towards data science systems rather than software development.

D. Extreme Programming (XP) Core Principles

As a form of agile development, Extreme Programming presents a set of principles that address quickly dynamically changing software requirements and project risk mitigation. The framework for Extreme Programming is based on five core principles with respect to the five values of Communication, Simplicity, Feedback, Courage, and Respect:

1) *Rapid Feedback*: Software development is inherently a team sport, and direct-to-face and direct-to-consumer communication are critical. Unlike CRISP-DM, where the roles of strategy and engineering are separate, XP requires the programmer to understand the business objectives in order to build the technical solution.
2) *Assumed Simplicity*: Code should be written in the simplest possible form that solves the immediate problem while enabling future refactoring as needed.
3) *Assumed Simplicity*: Code should be written in the simplest possible form that solves the immediate problem while enabling future refactoring as needed.
4) *Incremental Changes*: Rather than trying to define a product up front, make small, incremental changes in response to constant consumer feedback. This allows the customer to have more control over the development process, resulting in better business outcomes.
5) *Embracing Change*: Accept and tackle changing consumer requests with open arms. If a client makes a proposal for changes, the team ought to support this decision and incorporate them into future work as quickly as possible.
6) *Quality Work*: Work collaboratively as a team with the common goal of producing a quality product.

E. Gaps in Extreme Programming

The main drawback of the Extreme Programming approach is that the methodology focuses extensively on deductive reasoning, direct customer feedback, and the human-driven aspects of coding, rather than empirical testing. This is in part why we've selected it as a starting point for process development: CRISP-DM's failure to utilize heuristics is one of its biggest gaps, and Extreme Programming effectively plugs it. That said, empirical testing is key to building data science systems, and we need to add it back in. Building different models is often less costly in data science than

engineering, and that needs to be taken into consideration in our approach. Our proposed framework offers a hybrid approach to both methodologies to better meet the needs of deploying data science systems for production.

III. The Core Process

Rather than reinventing the wheel, our proposed process combines the structured, empirical, and sequential nature of CRISP-DM with the best practices and heuristics-based approach of Extreme Programming. This marriage of frameworks acknowledges how data science product development is moving and more towards software development while also recognizing its unique character:

1) *Define Business Goals*: Identifying key business goals early in the project is critical to ensure subsequent development is focused on business impact. "Business" is a key word: While novel analytical approaches can be valuable for model performance and team morale, they should always take a back seat to building a useful product. To ensure that development remains on track, this stage should involve a very initial exploration of the data and compute resources that are available to the development process and align business goals with what is possible at a high level given available resources.
2) *Brainstorm Approaches*: Once goals have been determined, the team must determine which frameworks or methodologies are most suitable for this project. This stage leans heavily on Extreme Programming: Ideally, this stage of the process will move quickly, with individual team members proposing best practices based on their experience, recent research, and the data available. In the interests of speed, the best ideas are simple, easy to implement, and easy to build on top of. Usually this stage will result in more than one interesting approach. All of them should be pursued at least through the next stage to maximize output. Once ideas have been generated, they should be aligned with the business goals and available resources to determine an initial, quantified set of metrics that will determine how different analytical approaches compare to one another.
3) *MVPs*: MVPs, or minimum viable products, are a hallmark of the modern tech industry, not just in software development but also in business ideas. The core concept is that building a small, minimally functional prototype of a system is the most efficient way to test the core functionality of a system before allocating substantial resources to an approach. As such, once a set of approaches has been selected, the team should move as quickly as possible to build initial versions of the selected models. The key to building successful MVPs is to architect them so they can be used as an initial performance benchmark and be further developed to improve initial performance, allowing maximum reusability once an analytical approach is determined.
4) *Initial Evaluation*: Once MVP models are functional, they should be tested using realistic data across the business metrics that have been tested. The business element again is critical: While standard data science metrics such as mAP and F1 are important, they should be mapped to the key business outcomes in the same way that high frequency traders evaluate their models on profit rather than accuracy. That said, business metrics are not the only elements that should be considered: Beyond them, the team should have a wide-ranging discussion covering the scalability and maintainability of each model, including the compute resources and environment necessary to run them and the ability to re-train the model given new data. The goal is to come to a consensus of which MVP would best balance the costs and benefits to the business if it is built into a full-scale product.
5) *Train*: Self-explanatory: We then train the full-scale system so that it can best tested against a full set of business metrics.
6) *Testing and Continuous Learning*: The final phase of this process is testing. We end here because much like motion pictures, a data science product is never finished, only abandoned. This will make more sense as we break the testing process down into its three key sub-categories: data testing, synthetic testing, and production testing:

    a) **Data Testing**: Testing a proposed product on a full suite of business-focused metrics can be very time-consuming. As such, it's important to select a subset of data that can directionally reveal a model's performance. This is similar to a validation set in traditional data mining approaches, and it's likely the same dataset you used to evaluate MVPs. The data test should be scoped to provide as much information as possible about a system's performance on business metrics while being very easy to run.
    b) **Synthetic Testing**: This is where things get seriously detailed-oriented and metric-focused. Once a model passes an initial sniff test, you stress test it with edge cases. A strong synthetic test will include every scenario that the team has previously seen and all those it can imagine. As such, synthetic tests may require generating data, including with GANs or similar analytical approaches. The results from a synthetic test should give a nearly complete view of a model's retroactive performance on everything the organization can imagine appearing as input variables.
    c) **Production Testing**: Of course, reality is stranger than fiction, and real-life model deployments will inevitably encounter input variables beyond anything the team could have imagined when generating a test dataset. While ideally model performance at this stage should closely align with earlier testing, the real world has a way of throwing curve balls at any system you put into production. This is also when you can get a good view of how much compute and

storage a model takes: For many applications this type of computational performance is as important as a data science product's analytical performance. This is why testing a model on live data is always the last, and most important, phase of building a data science product. Early on in product development this may just mean that you put a model into full production while closely monitoring its performance. When you have a strong model in production, you should run candidate replacement models in parallel for significant amounts of time before switching them over. Never underestimate the real world's capability of surprising you.

A. Key Differences

The key difference for our framework when compared to other industry-proven paradigms is that it is very production-focused. Models like CRISP-DM fail to address key points that directly affect production, such as early evaluations of computation-inference performance trade-offs. Testing in multiple tiers is also critical for production; Data-only tests are often sufficient for data exploration and analysis projects, but don't come close to being sufficient for production systems. Similarly, putting models into production can be time-consuming and costly, even with efficient deployment systems. This means that testing must be structured to maximize learning while minimizing costs.

B. The Omission of Deployment

While this was covered in part in step 6, Test, it's worth reiterating that deployment was omitted as a specific step because its inclusion implies a type of finality that doesn't exist in data science development. All models should always been under constant evaluation, especially because input paradigms can shift at any time and models may need adjustment to continue to perform well.

IV. Skills Required

Effective and efficient development and deployment of data science systems calls for a high level of both business and engineering skills. Models like CRISP-DM are very focused on the model-building stage, with business stakeholders mainly involved in the initial planning and final evaluation stages. When running a data mining project, the main skills required are raw modeling, with some amount of business insight in the exploratory phase. When models go into production, especially in an AI context with significant computational requirements, a team that over-indexes on raw modeling talent inevitably runs into challenges:

1) Lack of business context understanding can cause development to drift away from core business outcomes and objectives
2) Lack of business operations and/or strategy understanding can limit the teams' ability to prioritize technical approaches. Rather than using heuristics and estimates to prioritize analytical approaches, teams may build all models for comparison, slowing development and increasing costs.

s noted above, testing needs to be as realistic as possible within the constraints of computational costs, production stability, and effort to put into development environments. This means data scientists either need more software engineering skills or need to forge close partnerships with the infrastructure engineering team. It is very unlikely that an individual will have all of these skills; Business-focused data scientists, and engineers with statistical training definitely exist, but come at a premium, and even such people won't be able to deliver in isolation.

As such, the best approach is to construct a "pod" consisting of at least one data science specialist, and either business- and technically focused team members or representatives from a strategy, operations, and/or engineering team.

V. Implementation

This framework is high-level, and charts out steps, not how to manage teams to deliver them. This leaves things like detailed project management and progress tracking up to individual team. That said, through discussions with data scientists experienced in enterprises, growth-stage companies, and early-stage enterprises, there are a few clear best practices.

The key is that most data science projects doesn't require sprawling teams for successful delivery. While exceptions exist, particularly the core engines underlying search projects and pricing engines, most projects work best when scoped for a team sized in the single digits. As such, while formal, two-week agile sprints can work well, they are often too heavyweight for small teams. We have found that one-week cycles that map each week's priorities to the overarching project plan makes the most sense. A single Project Manager, Product Manager, or Scrum Master should manage meetings, allowing each contributor to set goals so long as they fit into the overall structure laid out above.

One-week cycles generally work well at least as feedback and accountability periods, if not formal project management units, because many of these steps are difficult to complete individually, except perhaps steps 1 and 6. Traditional two-week sprints often lead to different analytical approaches drifting dangerously far apart. Kanban is also an effective method of managing teams to deliver data science in an agile framework, if teams prefer continuous flow to discrete sprints. Whatever specific timing and planning system is used, getting the entire pod together 1-3 times a week is critical for efficient task allocation and the accomplishment of things like prioritizing different analytical approaches, as different teammates often have very different perspectives.

VI. Conclusion

Data science, especially in the deep learning subfield, has moved extremely quickly over the last few years. Analytical performance in computer vision, natural language processing, and other fields is leaps and bounds ahead of

even the most optimistic projections at the beginning of the millennium. At the same time, management frameworks have not kept up: Too many data science teams are overly project-focused and move from problem to problem without getting clear feedback of if their work is having any type of business impact at all. This results in business owners not having a clear idea of how data science teams can impact their organization, and leaves data science sidelined. This is a problem for almost all organizations, and for society at large: It's a missed opportunity when a technology that has the potential to increase the efficiency of almost every workflow is not used to its fullest. This framework aims to combine best practices from lightweight data mining and agile software development frameworks into a new model that can enable data science teams to move from delivering one-off projects to valuable, maintainable projects that can deliver ongoing business impact.

REFERENCES

[1] Beck, Kent, and Cynthia Andres. Extreme Programming Explained. Addison-Wesley, 2006.

[2] Dam, Hoa Khanh et al. "Towards effective AI-powered agile project management."2018. arXiv 1812.10578.

[3] Elmousalami, Haytham H. "Comparison of Artificial Intelligence Techniques for Project Conceptual Cost Prediction." 2019. arXiv 1909.11637.

[4] Greene, Jennifer and Andrew Stellman. Learning Agile: Understanding Scrum, XP, Lean, and Kanban. O'Reilly Media, 2013.

[5] IBM Knowledge Center. CRISP-DM Help Overview. 2012.

[6] Schwalbe, Kathy. Information Technology Project Management. Cengage Learning, 2015.

[7] Sutherland, Jeff. Scrum: The Art of Doing Twice the Work in Half the Time. Random House, 2014.