A Multi-Criteria Model for Prioritizing Product Backlog Items in AI and Data Science Projects

Dimas Cassimiro Nascimento
Universidade Federal do Agreste de Pernambuco (UFAPE)
Garanhuns, Brazil
dimas.cassimiro@ufape.edu.br

Daliton da Silva
Universidade Federal do Agreste de Pernambuco (UFAPE)
Garanhuns, Brazil
daliton.silva@ufape.edu.br

Igor Medeiros Vanderlei
Universidade Federal do Agreste de Pernambuco (UFAPE)
Garanhuns, Brazil
igor.vanderlei@ufape.edu.br

Rian Gabriel Santos Pinheiro
Universidade Federal de Alagoas (UFAL)
Maceió, Brazil
rian@ic.ufal.br

Rodrigo Cardoso Andrade
Universidade Federal do Agreste de Pernambuco (UFAPE)
Garanhuns, Brazil
rodrigo.andrade@ufape.edu.br

Abstract

Agile AI and Data Science projects involve backlog prioritization decisions under uncertainty, metric-driven experimentation, resource constraints, and technical risks. This paper proposes a multi-criteria model to support the prioritization of Product Backlog Items during sprint planning. Each item is represented by criteria such as criticality, dependency blocking power, distance to target metrics, expected metric gain, effort efficiency, resource fit, urgency, uncertainty reduction, risk mitigation, and historical prioritization balance. The proposed model can be combined with Pareto-based prioritization, weighted aggregation, and resource-aware selection, offering a structured and adaptable mechanism to support decision-making in agile AI and Data Science projects.

Keywords

Agile Software Development, Product Backlog Prioritization, AI Projects, Data Science Projects, Multi-Criteria Decision Analysis, Sprint Planning, Pareto Dominance

1 Introduction

Agile software development is widely used to organize software projects through short feedback cycles, incremental delivery, and continuous adaptation. In this context, the Product Backlog connects stakeholder expectations, technical work, and sprint-level decisions. However, deciding which Product Backlog Items (PBIs) should be selected for the next sprint is not a trivial task. Requirements prioritization has long been treated as a complex decision problem involving value, cost, risk, urgency, dependencies, and stakeholder preferences [1]. In AI and Data Science projects, this problem becomes even more challenging because backlog items may involve not only software features, but also data preparation, model training, validation, monitoring, fairness assessment, and explainability analysis.

AI-based systems introduce specific engineering challenges because their behavior depends on data, models, experiments, and operational conditions. Amershi et al. [2] describe machine learning development as a workflow that includes data collection, feature engineering, model training, evaluation, deployment, and monitoring. In this type of project, some PBIs may not produce immediate user-visible functionality, even when they are essential for project success. For example, Sambasivan et al. [4] show how poor data practices can generate data cascades in high-stakes AI systems, while Sculley et al. [5] discuss how machine learning systems can accumulate hidden technical debt through data dependencies, configuration issues, and feedback loops.

Backlog prioritization in AI and Data Science projects is also strongly influenced by quantitative targets and resource constraints. A team may need to improve recall, precision, F1-score, calibration, fairness, latency, robustness, or other domain-specific indicators. Wan et al. [7] show that machine learning changes several software development practices, including requirements, design, testing, and process. At the same time, sprint planning must consider the availability of data scientists, data engineers, machine learning engineers, software developers, and domain specialists. Serban et al. [6] emphasize the importance of engineering practices for software with machine learning components, and Kreuzberger et al. [3] highlight how MLOps adds concerns related to automation, deployment, monitoring, and operational reliability.

This paper proposes a multi-criteria model for prioritizing Product Backlog Items in AI and Data Science projects. The model represents each PBI through normalized criteria related to criticality, dependency blocking power, distance to target metrics, expected metric gain, effort efficiency, resource fit, urgency, uncertainty reduction, risk mitigation, and historical prioritization balance. Based on this representation, the paper introduces two complementary strategies: a Pareto-based approach, which identifies non-dominated PBIs without reducing all criteria to a single score, and a weighted aggregation approach, which provides a transparent and adjustable ranking mechanism. By combining multi-criteria reasoning, resource-aware selection, and sprint planning constraints, the proposed model offers a structured perspective for supporting prioritization decisions in agile AI and Data Science projects.

2 Related Work

Backlog prioritization is closely related to the broader problem of requirements prioritization in software engineering. Achimugu et al. [1] show that requirements prioritization has been studied through several techniques, including ranking, weighting, pairwise comparison, and value-oriented approaches. These methods are useful for

supporting release planning and implementation decisions, but AI and Data Science projects introduce additional concerns that are not fully captured by traditional prioritization criteria, such as data quality, model uncertainty, experimental results, and metric-driven improvement.

Software engineering for machine learning has also highlighted the need to rethink conventional development practices. Amershi et al. [2] describe how teams developing AI-based applications deal with a workflow that includes data collection, feature engineering, training, evaluation, deployment, and monitoring. Sculley et al. [5] argue that machine learning systems may accumulate hidden technical debt due to data dependencies, configuration problems, feedback loops, and system entanglement. These works reinforce the idea that backlog items in AI projects may represent invisible but essential tasks, such as improving data pipelines, validating models, reducing operational risks, or preparing the system for deployment.

Recent work on MLOps further emphasizes the operational nature of AI systems. Kreuzberger et al. [3] discuss principles, roles, components, and architectures needed to bring machine learning products into production. This perspective is important for backlog prioritization because sprint planning must consider not only feature delivery, but also automation, monitoring, reliability, and resource availability. In contrast to these studies, this paper focuses specifically on a formal multi-criteria model for prioritizing Product Backlog Items in AI and Data Science projects, combining Pareto dominance, weighted aggregation, and resource-aware selection.

3 A Multi-Criteria Prioritization Model for AI/Data Science Product Backlog Items

In Scrum terminology, the basic unit of work considered in this model is referred to as a *Product Backlog Item* (PBI). In the context of AI and Data Science projects, a PBI may represent a user-facing feature, a data engineering task, an exploratory data analysis activity, a model training experiment, a model validation procedure, a fairness or explainability assessment, a deployment-related task, or any other work item that contributes to the evolution of the product.

Let $\mathcal{I} = \{1, 2, \ldots, n\}$ be the set of candidate PBIs available in the Product Backlog at the beginning of a given Sprint Planning event. The goal is to select a subset $\mathcal{S} \subseteq \mathcal{I}$ of PBIs to be prioritized for the next sprint, considering multiple criteria related to business value, technical relevance, project urgency, dependency structure, expected impact on quantitative performance metrics, historical prioritization, and resource availability. For each PBI $i \in \mathcal{I}$, we define the following attributes.

Criticality. Let $c_i \in \{1, 2, 3\}$ denote the criticality level of PBI i, where 1, 2, and 3 indicate low, medium, and high criticality, respectively. The normalized criticality score is given by $C_i = (c_i - 1)/2$, such that $C_i \in [0, 1]$. Therefore, PBIs considered more critical by stakeholders receive larger values in this criterion.

Dependency Blocking Power. Let $G = (\mathcal{I}, \mathcal{E})$ be a directed dependency graph, where an edge $(i, j) \in \mathcal{E}$ indicates that PBI j depends on the completion of PBI i. The number of PBIs blocked by i is defined as $b_i = |\{j \in \mathcal{I} : (i, j) \in \mathcal{E}\}|$. The normalized blocking score is defined as follows:

$$B_i = \begin{cases} \dfrac{b_i}{\max_{j \in \mathcal{I}} b_j}, & \text{if } \max_{j \in \mathcal{I}} b_j > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Distance to Target Metric. Let m_i be the current value of a quantitative metric associated with PBI i, such as recall, precision, F1-score, AUC, calibration error, fairness score, latency, or data quality index. Let $m_i^\star$ be the target value for this metric. For benefit-like metrics normalized in $[0, 1]$, the distance-to-target score is $D_i = \max(0, m_i^\star - m_i)$. Thus, PBIs associated with larger gaps between the current metric and the target metric receive higher priority. For cost-like metrics, where lower values are better, such as latency, error rate, or memory usage, the score may instead be defined as $D_i = \max(0, m_i - m_i^\star)$.

Expected Metric Gain. Let $\Delta_i \in [0, 1]$ denote the expected improvement in the target metric if PBI i is completed. For instance, Δ_i may represent the expected increase in recall, F1-score, data quality, model robustness, fairness, or system availability. The normalized expected gain is defined as $M_i = \Delta_i$. This criterion favors PBIs whose completion is expected to produce measurable improvement in the product or model.

Historical Prioritization Penalty. Let $h_i \in \mathbb{N}$ denote the number of times PBI i has been prioritized in previous sprint planning sessions but has not been completed. This variable captures repeated postponement, planning instability, or excessive re-prioritization. The penalty is modeled as $H_i = h_i/(h_i + \eta)$, where $\eta > 0$ is a smoothing parameter. This formulation avoids dependence on the maximum value observed in the current backlog and makes the marginal effect of each additional unsuccessful prioritization decrease over time. A larger H_i reduces priority, unless the project manager explicitly interprets repeated prioritization as evidence of persistent importance.

Effort Cost. Let $e_i > 0$ denote the estimated effort required to complete PBI i, measured in story points, person-days, or person-sprints. The normalized effort cost is given by $E_i = e_i/\max_{j \in \mathcal{I}} e_j$. Since lower effort is preferable when other criteria are similar, we define the effort efficiency score as $Q_i = 1 - E_i$. This allows lower-effort items to be favored when they offer comparable value.

Resource Fit. Let $\mathcal{R}$ be the set of available human resources, such as data scientists, data engineers, machine learning engineers, domain specialists, software developers, and project managers. Let a_r denote the available capacity of resource r during the next sprint, and let q_{ir} denote the amount of resource r required by PBI i. The resource feasibility ratio is $\rho_i = \min_{r \in \mathcal{R}: q_{ir} > 0} a_r/q_{ir}$, and the normalized resource fit score is $R_i = \min(1, \rho_i)$. Thus, $R_i = 1$ indicates that the currently available capacity is sufficient, while $R_i < 1$ indicates some degree of resource bottleneck.

Time Pressure. Let $\delta_i \in \mathbb{N}$ denote the number of remaining sprints until PBI i reaches a relevant time boundary, such as a contractual deadline, delivery milestone, stakeholder review date, data availability window, or latest useful completion point. The time pressure score is defined as $U_i = 1/(1 + \delta_i)$. Thus, PBIs with closer deadlines or narrower useful completion windows receive higher urgency scores.

Uncertainty Reduction Value. AI and Data Science projects are often characterized by epistemic uncertainty regarding data quality,

model performance, feasibility, explainability, fairness, deployment constraints, and stakeholder expectations. Let $u_i \in [0,1]$ denote the expected uncertainty reduction obtained by completing PBI i. For instance, a small experiment that clarifies whether a modeling strategy is feasible may have high uncertainty reduction value. We define $L_i = u_i$.

Risk Mitigation Value. Let $r_i \in [0,1]$ denote the degree to which PBI i mitigates relevant project risks, such as data leakage, poor generalization, model drift, ethical risk, regulatory risk, integration risk, or deployment risk. We define $K_i = r_i$. This criterion favors PBIs that reduce threats to the technical, ethical, operational, or regulatory success of the project.

3.1 Multi-Criteria Representation

Each PBI $i \in \mathcal{I}$ is represented by a multi-criteria vector $\mathbf{x}_i = (C_i, B_i, D_i, M_i, 1 - H_i, Q_i, R_i, U_i, L_i, K_i) \in [0,1]^{10}$. The criteria are interpreted as follows: C_i represents the criticality score, B_i represents the dependency blocking score, D_i represents the distance-to-target score, M_i represents the expected metric gain, Q_i represents the effort efficiency score, R_i represents the resource fit score, U_i represents the time-sensitive urgency score, L_i represents the uncertainty reduction value, K_i represents the risk mitigation value, and $1 - H_i$ represents the historical prioritization balance score. All components are maximization criteria, meaning that larger values are preferred.

3.2 Pareto Dominance

Given two PBIs $i, j \in \mathcal{I}$, we say that PBI i Pareto-dominates PBI j, denoted by $i \succ j$, if and only if $x_{i\ell} \geq x_{j\ell}$ for all $\ell \in \{1, \ldots, 10\}$, and there exists at least one criterion $\ell' \in \{1, \ldots, 10\}$ such that $x_{i\ell'} > x_{j\ell'}$. In other words, PBI i dominates PBI j if i is at least as good as j in all criteria and strictly better in at least one criterion.

The Pareto frontier is defined as the set of non-dominated PBIs, that is, $\mathcal{F}_1 = \{i \in \mathcal{I} : \nexists j \in \mathcal{I} \text{ such that } j \succ i\}$. If the number of PBIs in $\mathcal{F}_1$ is larger than the sprint capacity, additional frontiers may be generated iteratively. Let $\mathcal{I}^{(1)} = \mathcal{I}$. For $p = 1, 2, \ldots$, the p-th Pareto frontier is defined as $\mathcal{F}_p = \{i \in \mathcal{I}^{(p)} : \nexists j \in \mathcal{I}^{(p)} \text{ such that } j \succ i\}$, with $\mathcal{I}^{(p+1)} = \mathcal{I}^{(p)} \setminus \mathcal{F}_p$. This produces a Pareto-based ranking of PBIs, where items in $\mathcal{F}_1$ have the highest priority, followed by items in $\mathcal{F}_2$, and so on.

3.3 Weighted Aggregation Score

As an alternative to Pareto-based prioritization, the criteria may be aggregated into a single prioritization score. Let $\mathbf{w} = (w_C, w_B, w_D, w_M, w_Q, w_R, w_U, w_L, w_K, w_H)$ be a vector of non-negative weights such that $\sum_{\ell=1}^{10} w_\ell = 1$. The aggregated priority score of PBI i is defined as $S_i = w_C C_i + w_B B_i + w_D D_i + w_M M_i + w_Q Q_i + w_R R_i + w_U U_i + w_L L_i + w_K K_i + w_H (1 - H_i)$. A higher value of S_i indicates a higher priority.

3.4 Sprint Capacity and Selection Size

Let $\mathcal{R}$ be the set of available human resources in the next sprint. For each resource $r \in \mathcal{R}$, let a_r denote the available capacity. For each PBI i, let q_{ir} denote the required amount of resource r. The total available capacity can be represented as $A = \sum_{r \in \mathcal{R}} a_r$.

If each PBI has a scalar effort estimate e_i, the approximate number of PBIs that can be selected for the next sprint may be estimated as $k = \max(1, \lfloor A/\bar{e} \rfloor)$, where $\bar{e} = (1/|\mathcal{I}|) \sum_{i \in \mathcal{I}} e_i$ is the average estimated effort of candidate PBIs. However, a more precise selection should consider resource-specific constraints. In this case, the selected set $\mathcal{S}$ must satisfy $\sum_{i \in \mathcal{S}} q_{ir} \leq a_r$ for all $r \in \mathcal{R}$.

3.5 Prioritization Algorithms

In this section, we propose two algorithms to tackle the investigated problem.

3.5.1 Pareto-Based Prioritization Algorithm. The Pareto-based approach (Algorithm 1) prioritizes PBIs according to successive Pareto frontiers, giving precedence to non-dominated items. Within each frontier, PBIs are ordered by the aggregated score (S_i), while resource feasibility determines whether an item can be included in the sprint.

3.5.2 Aggregation-Based Prioritization Algorithm. In turn, Algorithm 2 uses an aggregation-based approach that computes a single score for each PBI and selects the highest-ranked PBIs while respecting team capacity.

Algorithm 1 Pareto-Based Prioritization of Product Backlog Items

Require: Set of PBIs $\mathcal{I}$, dependency graph G, remaining sprints τ, resource capacities $\{a_r\}_{r \in \mathcal{R}}$, effort estimates $\{e_i\}_{i \in \mathcal{I}}$, weights $\mathbf{w}$
Ensure: Prioritized set $\mathcal{S}$
Compute normalized criteria $C_i, B_i, D_i, M_i, Q_i, R_i, U_i, L_i, K_i, 1 - H_i$ for each $i \in \mathcal{I}$
Construct the multi-criteria vector $\mathbf{x}_i$ for each $i \in \mathcal{I}$
Compute the aggregated score S_i for each $i \in \mathcal{I}$
Initialize $\mathcal{S} \leftarrow \emptyset$
Initialize the remaining set $\mathcal{I}' \leftarrow \mathcal{I}$
while $\mathcal{I}' \neq \emptyset$ **do**
    Compute the current Pareto frontier:
    $$\mathcal{F} = \{i \in \mathcal{I}' : \nexists j \in \mathcal{I}' \text{ such that } j \succ i\}$$
    Sort items in $\mathcal{F}$ in descending order of S_i (see Section 3.3)
    for each $i \in \mathcal{F}$ **do**
        if adding i does not violate resource constraints **then**
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{i\}$
        end if
    end for
    $\mathcal{I}' \leftarrow \mathcal{I}' \setminus \mathcal{F}$
    if no additional PBI can be added without violating resource constraints **then**
        break
    end if
end while
return $\mathcal{S}$

4 Practical Considerations

The proposed model is intended to support, rather than replace, the judgment of Product Owners, Scrum Teams, technical leaders, and domain specialists. In practical settings, backlog prioritization is not

Algorithm 2 Aggregation-Based Prioritization of Product Backlog Items

Require: Set of PBIs $\mathcal{I}$, remaining sprints τ, resource capacities $\{a_r\}_{r\in\mathcal{R}}$, effort estimates $\{e_i\}_{i\in\mathcal{I}}$, weights $\mathbf{w}$
Ensure: Prioritized set $\mathcal{S}$
1: Compute normalized criteria $C_i, B_i, D_i, M_i, Q_i, R_i, U_i, L_i, K_i, 1 - H_i$ for each $i \in \mathcal{I}$
2: Compute the aggregated score S_i (see Section 3.3)
3: Sort PBIs in descending order of S_i
4: Initialize $\mathcal{S} \leftarrow \emptyset$
5: **for** each PBI i in the sorted list **do**
6: **if** adding i does not violate resource constraints **then**
7: $\mathcal{S} \leftarrow \mathcal{S} \cup \{i\}$
8: **end if**
9: **end for**
10: **return** $\mathcal{S}$

only a mathematical problem, but also a socio-technical decision process. Therefore, the criteria, weights, and constraints used by the model should be discussed with the team before being applied in sprint planning. This helps ensure that the prioritization process reflects the current goals of the project, the maturity of the team, and the expectations of stakeholders.

A first practical recommendation is to keep the criteria simple and understandable. Although the model allows the representation of multiple dimensions, teams should avoid introducing criteria that cannot be consistently estimated. For example, criticality, effort, resource fit, expected metric gain, and risk mitigation should be defined using clear scales and shared interpretation guidelines. When possible, these values should be supported by evidence from previous sprints, model evaluation reports, issue trackers, system logs, or stakeholder feedback. This reduces subjectivity and improves the consistency of prioritization decisions over time.

Another important consideration concerns the estimation of expected metric gains and distance to target metrics. In AI and Data Science projects, these values may be uncertain, especially when a PBI involves exploratory analysis, model experimentation, data cleaning, or feature engineering. In such cases, teams should avoid treating estimates as precise predictions. Instead, they may use approximate ranges, conservative values, or confidence levels. PBIs with high uncertainty reduction value can also be prioritized when the team needs to learn whether a technical path is feasible before committing to more expensive development activities.

The Pareto-based strategy is particularly useful when the team wants to identify strong candidates without imposing a single scoring function too early. It can reveal PBIs that are relevant from different perspectives, such as items that remove important dependencies, mitigate risks, or address large metric gaps. However, Pareto frontiers may contain more items than the team can execute in a sprint. In these cases, the weighted aggregation score can be used as a tie-breaker, together with qualitative discussion among team members. This combination preserves the benefits of multi-criteria reasoning while still producing an actionable sprint plan. Also note that, as the number of prioritization criteria increases, Pareto dominance may become less discriminative, since the probability that one PBI dominates another across all dimensions tends to decrease. Therefore, in practice, the proposed prioritization model may be applied using a reduced set of criteria.

In turn, the aggregation-based strategy is easier to operationalize when the team needs a complete ranking of PBIs. However, the choice of weights should be treated with care. In early stages of an AI project, uncertainty reduction, data quality, and risk mitigation may deserve higher weights. In later stages, deployment readiness, resource fit, latency, robustness, and monitoring-related items may become more important. Resource feasibility should also be validated during sprint planning, since a high-priority PBI may still be infeasible when the required expertise or capacity is unavailable.

5 Conclusions and Future Work

This paper presented a multi-criteria model for prioritizing Product Backlog Items in AI and Data Science projects. The model considers criteria such as criticality, dependency blocking power, distance to target metrics, expected metric gain, effort efficiency, resource fit, urgency, uncertainty reduction, risk mitigation, and historical prioritization balance. It also combines Pareto-based prioritization, weighted aggregation, and resource-aware selection to support sprint planning decisions. The proposed approach makes backlog prioritization more explicit, traceable, and adaptable to the specific characteristics of AI projects. Instead of treating all items through a single notion of value, the model considers the trade-offs among experimentation, data quality, model performance, risks, dependencies, and team capacity.

Future work includes evaluating the model in real AI and Data Science projects, comparing different prioritization strategies, and investigating how sprint outcomes can be used to refine weights, effort estimates, and expected metric gains.

Artifact Availability

No artifact is currently available for this study.

References

[1] Philip Achimugu, Ali Selamat, Roliana Ibrahim, and Mohd Naz'ri Mahrin. 2014. A Systematic Literature Review of Software Requirements Prioritization Research. *Information and Software Technology* 56, 6 (2014), 568–585. doi:10.1016/j.infsof.2014.02.001

[2] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software Engineering for Machine Learning: A Case Study. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*. IEEE Press, 291–300. doi:10.1109/ICSE-SEIP.2019.00042

[3] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. 2023. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access* 11 (2023), 31866–31879. doi:10.1109/ACCESS.2023.3262138

[4] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M. Aroyo. 2021. "Everyone Wants to Do the Model Work, Not the Data Work": Data Cascades in High-Stakes AI. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, Article 39, 15 pages. doi:10.1145/3411764.3445518

[5] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. 2015. Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems*, Vol. 28. 2503–2511.

[6] Alex Serban, Koen van der Blom, Holger H. Hoos, and Joost Visser. 2024. Software Engineering Practices for Machine Learning: Adoption, Effects, and Team Assessment. *Information and Software Technology* 169 (2024), 107421. doi:10.1016/j.infsof.2023.107421

[7] Zhiyuan Wan, Xin Xia, David Lo, and Gail C. Murphy. 2021. How Does Machine Learning Change Software Development Practices? *IEEE Transactions on Software Engineering* 47, 9 (2021), 1857–1871. doi:10.1109/TSE.2019.2937083