**A Tool to Extract Accumulated Technical Debts in Data Science Software**

1. **Background**

Coined by Ward Cunningham, “Technical Debt” (TD) is a metaphor used in modern software engineering practices to describe the concept of cost when developers implement rapid solutions to expedite the time to market of their product. Accumulated TD is not always apparent on a functionality level, instead, TD affects the improvability and maintainability of software during its lifecycle. Although multiple researchers have attempted to create a universal definition and unit of measurement for TDs, there is no widely accepted approach for TD measurement or TD forecasting [3]. However, a well-known approach to search for TD according to a project organizer’s defined rules on the individual component level of a software system is the SQALE model [2].

Data Science (DS) is the field that has grown in the intersection of Computer Science and Statistics and contains the increasingly popular field of Machine Learning (ML). ML is the research area that mathematically allows a computer to learn and improve without being explicitly programmed. Python has become the ideal programming language for creating ML models because of its many libraries that supply this functionality (pandas, numpy, sklearn, tensorflow, …).

DS is a young field, with many uncertainties regarding how DS dependent software systems will age in the coming years and how developers of the future will maintain or extend the ML models that developers today will train. A more formulated question on the matter is then “how is the DS community to anticipate the same problems being solved by TD researchers?”. In what ways do DS programs accumulate TDs that traditional software does, and more importantly what forms of TD are native *only* to DS software? Sculley *et al* presents a wide assortment of antipatterns involving ML models and their surrounding software architecture that they propose could be forms of ML independent TD due to the restraints they place on developers who want to maintain their source code [1].

1. **Project Goals**

The goal of this project will be to accent an ongoing study involving the mining of ML open-source projects to gather empirical support for the existence of DS specific TD. This concurrent study utilizes a foundational topic modelling technique [5] along with text mining techniques [4] on a dataset consisting of over a million commit logs from 2,000 ML GitHub repositories.

The results from this study will be used to support the objective of this project which is to incorporate the SQALE method mentioned previously to identify where a DS program or codebase encounters the discovered TDs. It is important to note that SQALE depends entirely on the remediation table created by each individual project because there is no universally accepted approach to measure TD.

Because of this, the proposed features of this tool are as follows:

* The tool will support user-defined Python code to indicate if an area is technically indebted or not. These are called “remediation functions” and will be used later.
* The tool will map remediation functions to appropriate costs, these are called “remediation tables” and will be used later.
* The tool will supply APIs that can help DS developers when they are creating remediation functions. (Example: “isHighLevel” would be an API that determines if a ML program contains an import of a high-level library like sklearn or a low-level library such as tensorflow)
* The tool can run any number of remediation tables on all Python files in a codebase and will return how technically indebted each individual Python file is according to all remediation tables.
* To supply an example for future developers, and to demonstrate the researched forms of DS native TDs, the tool will provide sample remediations tables to detect the proposed forms of DS TD.

1. **Cited References**

**[1]** D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in Machine learning systems. *In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15). MIT Press, Cambridge, MA, USA, 2503–2511.*

**[2]** Letouzey, Jean-Louis. (2012). “The SQALE method for evaluating Technical Debt”. 10.1109/MTD.2012.6225997.

**[3]** D. Tsoukalas, M. Siavvas, M. Jankovic, D. Kehagias, A. Chatzigeorgiou and D. Tzovaras, "Methods and Tools for TD Estimation and Forecasting: A State-of-the-art Survey," 2018 International Conference on Intelligent Systems (IS), Funchal - Madeira, Portugal, 2018, pp. 698-705, doi: 10.1109/IS.2018.8710521.

**[4]** Q. Huang, E. Shihab, X. Xia, D. Lo, and S. Li. Identifying self-admitted technical debt in open source projects using text mining. *Empirical Software Engineering*, 23(1):418–451, 2018.

**[5]** David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. *Latent dirichlet allocation.* J. Mach. Learn. Res. 3, null (3/1/2003), 993–1022.