Reproducibility Report for ACM SIGMOD 2021 Paper: "Conformance Constraint Discovery: Measuring Trust in Data-Driven Systems"

MARIA DALTAYANNI, University of San Francisco, USA

This report shows that all the paper contributions can be verified experimentally, and all results can be reproduced successfully. There are seven figures produced in the paper that illustrate how the suggested conformance constraint discovery method, CCSyth, is a good indicator of how accurate a machine learning model will be when run on a test set and how it outperforms the state-of-the-art approaches for drift detection. The reproducibility process mainly consisted of obtaining the sources, code, and data through the repository provided by the authors, then running a single script to install the required libraries and reproduce all of the paper results, including the paper. The results were run on a WSL distribution of Ubuntu and took about 2.5 hours to reproduce. The experimental results for the evaluation of CCSynth were all easy to reproduce with minimal communication with the authors, making the paper an **ideal reproducibility submission**.

1 INTRODUCTION

This is the reproducibility report for the paper "Conformance Constraint Discovery: Measuring Trust in Data-Driven Systems" [1], authored by Anna Fariha (University of Massachusetts Amherst) (afariha@cs.umass.edu), Ashish Tiwari (Microsoft) (ashish.tiwari@microsoft.com), Arjun Radhakrishna (Microsoft) (arradha@microsoft.com), Sumit Gulwani (Microsoft Research) (sumitg@microsoft.com), Alexandra Meliou (University of Massachusetts Amherst) (ameli@cs.umass.edu).

This paper satisfies all five steps of an **ideal reproducibility submission**, as all steps of installing required systems, obtaining data, rerunning experiments, generating graphical results, and producing the final paper file are successfully accomplished through one master script.

2 SUBMISSION

In their reproducibility submission, the authors provide the following:

- github repository with code, scripts and paper at: github.com/afariha/ConformanceConstraintsReproducibility
- detailed Readme file at: https://github.com/afariha/ConformanceConstraintsReproducibility/blob/main/README.md
- data sources at: https://github.com/afariha/ConformanceConstraintsReproducibility/blob/main/data.zip

This is an **ideal reproducibility submission**, as it satisfies all five reproducibility criteria. In particular, the authors provide a master script which:

(1) downloads and installs all systems needed (OS installation and update packages, Python and machine learning libraries, specific libraries for the models to be run and visualized)
(2) fetches all needed input data (downloaded from git, then unpacked)
(3) reruns all experiments (a virtual environment is created where all dependencies are set for the experiments to run successfully) and generates all results (7 scripts are run in sequence, generating plots that are saved in an output folder)
(4) generates all graphs and plots (the scripts being run, execute the algorithms and perform the comparisons that are required in each experiment and produce all of the figures described in the paper)
(5) recompiles the sources of the paper and produces a new PDF for the paper that contains the new graphs.

A complete set of scripts that install the system, produce the data, run experiments and produce the resulting graphs is provided not only through a single script that runs the above in sequence, but also the authors provide itemized scripts that run each step of the experiments individually. Finally, the authors provide a detailed and clear Readme file that describes the process of running the experiments step by step so it can be easily reproduced.

3 HARDWARE AND SOFTWARE ENVIRONMENT

The environment used in the paper and in the reproducibility is summarized in Table 1.

Table 1. Hardware & Software environment

	Paper	Repro Review
CPU	Apple M1	Intel i7-1165G7
cores	8	4
GHz	3.20GHz	2.80GHz
RAM	16GB	32GB
Storage	SSD	SSD
OS	macOS Monterey, Version 12.0.1	(WSL) Ubuntu 20.04.3 LTS

4 REPRODUCIBILITY EVALUATION

4.1 Process

During the reproducibility process, the following tasks were a simple and straightforward procedure: using Windows Subsystem for Linux (Ubuntu), running machine learning library installation packages, downloading the data from git and unpacking, setting up the virtual environment to run the 7 scripts that produce Figures 4, 5, 6a, 6b, 6c, 7 and 8 of the experiments, running the actual experiments and producing the final plots described in the paper.

For Figure 7, the authors were contacted and asked to update the version of Jinja2, a templating engine for Python, from 2.10.1 to 2.11 or newer. This was a very easy thing to fix.

For Figure 8, the results could not be produced in the first run of the code, since the sourcing of the virtual environment failed. Running the master script for a second time worked with no faults. Also, running the steps manually, also produced the same output without complications.

The authors were also contacted to include the paper production code in their master script. This was a very quick and minor update on their overall work.

4.2 Results

All major findings of the paper were reproduced. The following results were produced during the reproducibility process:

(1) Figure 4 showcases how the conformance constraints produced by the author's suggested algorithm, CCSynth, can be used as an indicator for low accuracy performance when a linear regression (LR) model is run on test data (as opposed to training data). The experiments were run on a deterministic sample dataset, and thus the figure did not change with different runs.
(2) Figure 5 compares the conformance constraints violation against the mean average error (MAE) of the LR model prediction on the test data. A high violation occurs with a high error in the model prediction. Again, deterministic sampling was used, and the reproduced results did not change with different runs.

(3) Figures 6a and 6b explore how constraint violation and a classifier's model accuracy drop behave similarly, as more data are added to the input, with and without the presence of noise. The experiments were run ten times, and the average performance was plotted.
(4) Figure 6c compares constraint violation against a Principal Components Analysis (PCA) method and illustrates how PCA fails to follow the increase in data drift, whereas CCSynth detects it.
(5) Figure 7 shows a representation of persons using constraints produced from their activity data, using a human activity recognition dataset. Representations are then used to compare drift between pairs of persons. The comparisons show similarities between people who otherwise seem to have similar features such as BMI and fitness. This experiment used Jinja2, and it exported results in an HTML format.
(6) Figure 8 shows how CCSynth's drift detection outperforms almost all the state-of-the-art approaches run on the Extreme Verification Latency benchmark. In total, 16 plots were derived, each of which compares CCSynth with another three approaches.

5 SUMMARY

This paper was easy to read and reproduce. The requirements in software and hardware were reasonable, and the experiments were reproduced within 2.5 hours on average.

REFERENCES

[1] Anna Fariha, Ashish Tiwari, Arjun Radhakrishna, Sumit Gulwani, and Alexandra Meliou. 2021. *Conformance Constraint Discovery: Measuring Trust in Data-Driven Systems.* Association for Computing Machinery, New York, NY, USA, 499–512. https://doi.org/10.1145/3448016.3452795