An Open Source Covariate based Software Reliability Assessment Tool A Guide for Contributors

Author1 Name[1], Author2 Name[1], and Lance Fiondella[1]
[1]Electrical and Computer Engineering, University of Massachusetts Dartmouth, MA, USA
Email: {lfiondella}@umassd.edu

***Abstract*—This paper presents the application architecture of xxx, a free and open source application to promote the**

I. Introduction

Do not worry about this for now

II. Software Architecture

This section presents...

The covariate tool graphical user interface is written in Python using PyQt5, which provides Python bindings for the Qt libraries. The tool is available to download from xxx.

Need to have PyQt5 libraries installed

The tool accepts failure data input in Excel (.xls, .xlsx) or comma separated value (.csv) formats. The first column of the table must contain failure times and the second column must contain the number of failures at that time. The third column must contain covariate data, with any additional columns containing data for additional covariates.

The provided data can be analyzing using any subset of the covariates, including zero. Analysis can be performed using different hazard functions, of which the tool includes three: Geometric, Negative Binomial (Order 2), and Discrete Weibull (Order 2). After analysis is performed, several goodness of fit measures are calculated and displayed in a table. These include AIC, BIC, ... , and xxx.

[example data sets, user guide links?]

Functions of tool: fitting models to data, comparing goodness of fit, prediction

A. xxx

Shows list of loaded models to choose

View imported data as graph or table

Select covariates to perform measurements on once data is loaded. Reads covariate names from data file. If no header, generic names (Metric1, Metric2, etc..) given to covariates.

Figure

III. Model Specifications

Generic model class contained in model.py in the core directory. Most definitions and methods are in model.py. Specific models are stored in the models directory, where one each model has its own file. This file can have any title, although for ease of use it is best to title the file the name of the model. The specific model must contain: an import of the generic Model class (from core.model import Model), the class must inherit from Model, a string containing the name of the model assigned to variable name, the initialization method (always exactly the same), and a calcHazard function. The calcHazard function takes one argument (b), and returns a list of values of the function evaluated for each discrete time of the imported failure data.

Code

IV. Model Integration and Testing

xxx

V. Conclusion and Future Research

This paper presents an open source...

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant Number (#1749635). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References