Predictive Analytics for Software Testing

Keynote Paper*

Federica Sarro
Department of Computer Science, University College London
London, United Kingdom
f.sarro@ucl.ac.uk

ABSTRACT
This keynote discusses the use of Predictive Analytics for Software Engineering, and in particular for Software Defect Prediction and Software Testing, by presenting the latest results achieved in these fields leveraging Artificial Intelligence, Search-based and Machine Learning methods, and by giving some directions for future work.

CCS CONCEPTS
• Software and its engineering → Search-based software engineering;

KEYWORDS
Predictive Analytics, Search-based Predictive Modelling

ACM Reference Format:

1 INTRODUCTION
The ever increasing volume of data produced by realising and using software calls for a new generation of analytical techniques that can help software engineers better understand their software processes, products and customers. Devising effective Predictive Analytics for Software Engineering holds the promise to maximise product quality, users’ satisfaction and revenues.

This keynote illustrates how search-based heuristics can be used to build effective predictive models for software engineering [2, 7] and highlights some of the recent developments in the production of adaptive [6, 11, 12], human-competitive [13] and scalable solutions [3, 4, 10].

The keynote also introduces the mutually beneficial relationship between predictive modelling and software testing by illustrating new ways to strengthen current testing practice through the use of predictive models [14] and, vice versa, to construct better predictive models through the use of information available from software testing activities [1].

*This one page paper provides an outline of the keynote talk given by Federica Sarro at SBST’18, with pointers to the literature for details of the results covered.

The talk concludes discussing how text mining, analytics and search-based techniques can be combined to obtain more efficient test automation, for example, exploiting artefacts written in natural language to infer behavioural models of software applications and to discover issues earlier in the process [5, 8, 9, 15].

ACKNOWLEDGMENTS
This work is supported by the EPSRC Programme Grant EP/J017515.

REFERENCES