sesa lab
SOFTWARE ENGINEERING SALERNO

Predicting The Energy Consumption Level of Java Classes in Android Apps

An Exploratory Analysis

Emanuele Iannone, Manuel De Stefano, Fabiano Pecorelli, Andrea De Lucia

eiannone@unisa.it

https://emaiannone.github.io

Context and Motivations

Using devices with a long-lasting battery is VERY pleasant!

Successful apps MUST keep track of how much they consume.

Measuring or estimating energy consumption is a key winning factor

Context and Motivations

battery

ack of

energy
ing factor

HARDWARE MEASUREMENT

- ✓ More realistic measurements
- ✗ Complex to set up and use

PowerScope:
A Tool for Profiling the Energy Usage of Mobile Applications

Jason Flinn and M. Satyanarayanan

Green Mining: Investigating Power Consumption across Versions

Abram Hindle
Department of Computing Science
University of Alberta,
Edmonton, Canada
abram.hindle@ualberta.ca

***Abstract*—Power consumption is increasingly becoming a concern for not only electrical engineers, but for software engineers as well, due to the increasing popularity of new power-limited contexts such as mobile-computing, smart-phones and cloud-computing. Software changes can alter software power consumption behaviour and can cause power performance regressions. By tracking software power consumption we can build models to provide suggestions to avoid power regressions. There is much research on software power consumption, but little focus on the relationship between software changes and power consumption. Most work measures the power consumption of a single software task; instead we seek to extend this work across the history (revisions) of a project. We develop a set of tests for a well established product and then run those tests across all versions of the product while recording the power usage of these tests. We provide and demonstrate a methodology that enables the analysis of power consumption performance for over 500 nightly builds of Firefox 3.6; we show that software change does induce changes in power consumption. This methodology and case study are a first step towards combining power measurement and mining software repositories research, thus enabling developers to avoid power regressions via power consumption awareness.**

***Keywords*-power; power consumption; mining software repositories; dynamic analysis; sustainable-software**

I. Introduction

Until recently, the effect of software, such as desktop applications and server software, on power consumption has been ignored under the tacit assumption that software engineers could use all available resources; but with the advent of contexts such as mobile and cloud computing it is clear that the design and implementation of software has a significant impact on power consumption and software engineers should be aware of it.

Within these contexts resources are limited: smart-phones/mobile devices have slower CPUs and limited battery life; cloud computing nodes are resource limited in terms of CPU speed, memory size, disk I/O [1], heat, and network bandwidth, all of which use power; software services consume power by simply being available. It is estimated [2], [3] that computer power usage alone costs many mid-sized businesses more than $100000 per year, producing over 1000 tonnes of CO_2! Thus resource conservation results in more availability, more battery-life, and smaller power bills.

Current power consumption research tends to focus on CPU usage and ignores the actual patterns of power consumption induced by software evolution and change [4]–[6]. Thus we propose *green mining*, a research agenda that aims to help developers understand power consumption issues related to their code, based on a corpus of software changes associated with power consumption. *Green mining* rests on two pillars, *mining software repositories* research, to find actual software changes, and *instrumentation* combined with *dynamic analysis*, in order to measure and correlate change to power usage. *Mining software repositories* (MSR) research is about the analysis of artifacts found within the huge corpus of software-repositories, such as the version control systems of open-source projects [7]. *Instrumentation* and *dynamic analysis* will be used to investigate the factors and resource utilization of changing software. We will look at each change in a version control system and dynamically measure its effect on power consumption. We plan to compile these patterns of software changes in order to answer questions about software power consumption. **Green mining leverages historical information extracted from the corpus of publicly available software to help provide software power consumption advice.**

Green Mining models how software maintenance impacts a system's power usage. It aims to help software engineers reduce the power consumption of their own software by estimating the impact of software modifications on power consumption. This is a novel extension of *mining software repositories research* into the realm of software-based power consumption as previous research does not address the effect of software change on power consumption. Our research questions include:

- Does Software Change relate to power consumption?
- What information needs to be gathered to estimate power consumption? Is CPU measurement enough?

This work has a potential environmental impact: deployed software will consume less resources and thus have a direct reduction on the CO_2 emissions [3]. This work is industrially relevant because vendors interested in GreenIT [3], such as Apple, Microsoft, Intel, and IBM, have already begun investing in power-management documentation and tools [8]–[12].

In this paper we will address the first steps towards *green mining*: measuring the power use of multiple versions of a

J. Flinn and M. Satyanarayanan, "PowerScope: a tool for profiling the energy usage of mobile applications," Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications, 1999, pp. 2-10.

A. Hindle, "Green mining: Investigating power consumption across versions," 2012 34th International Conference on Software Engineering (ICSE), 2012, pp. 1301-1304.

Context and Motivations

battery

ack of

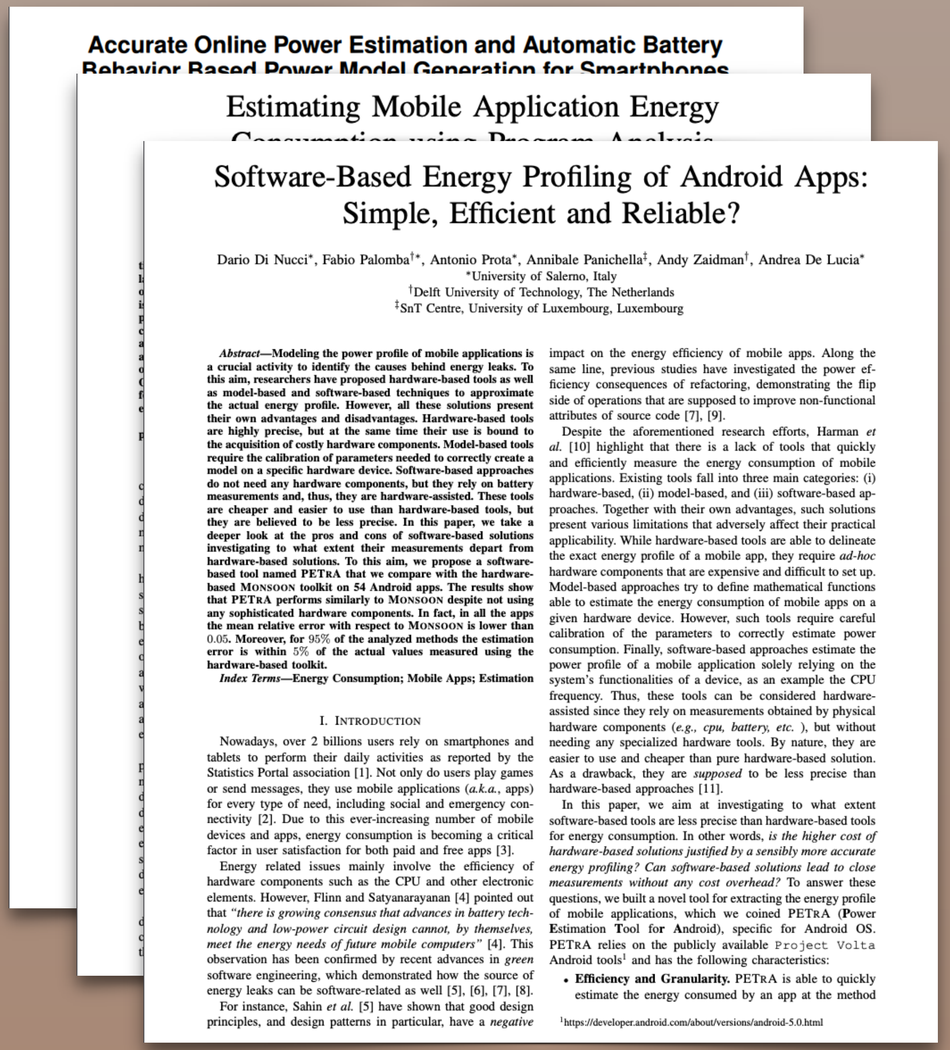
energy
ing factor

HARDWARE MEASUREMENT

- ✓ More realistic measurements
- ✗ Complex to set up and use

SENSORS ESTIMATION

- ✓ Estimates using sensor data
- ✗ Less precise



Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones

Estimating Mobile Application Energy Consumption using Program Analysis

Software-Based Energy Profiling of Android Apps: Simple, Efficient and Reliable?

Dario Di Nucci*, Fabio Palomba†*, Antonio Prota*, Annibale Panichella‡, Andy Zaidman†, Andrea De Lucia*
*University of Salerno, Italy
†Delft University of Technology, The Netherlands
‡SnT Centre, University of Luxembourg, Luxembourg

***Abstract*—Modeling the power profile of mobile applications is a crucial activity to identify the causes behind energy leaks. To this aim, researchers have proposed hardware-based tools as well as model-based and software-based techniques to approximate the actual energy profile. However, all these solutions present their own advantages and disadvantages. Hardware-based tools are highly precise, but at the same time their use is bound to the acquisition of costly hardware components. Model-based tools require the calibration of parameters needed to correctly create a model on a specific hardware device. Software-based approaches do not need any hardware components, but they rely on battery measurements and, thus, they are hardware-assisted. These tools are cheaper and easier to use than hardware-based tools, but they are believed to be less precise. In this paper, we take a deeper look at the pros and cons of software-based solutions investigating to what extent their measurements depart from hardware-based solutions. To this aim, we propose a software-based tool named PETRA that we compare with the hardware-based MONSOON toolkit on 54 Android apps. The results show that PETRA performs similarly to MONSOON despite not using any sophisticated hardware components. In fact, in all the apps the mean relative error with respect to MONSOON is lower than 0.05. Moreover, for 95% of the analyzed methods the estimation error is within 5% of the actual values measured using the hardware-based toolkit.**

***Index Terms*—Energy Consumption; Mobile Apps; Estimation**

I. INTRODUCTION

Nowadays, over 2 billions users rely on smartphones and tablets to perform their daily activities as reported by the Statistics Portal association [1]. Not only do users play games or send messages, they use mobile applications (*a.k.a.*, apps) for every type of need, including social and emergency connectivity [2]. Due to this ever-increasing number of mobile devices and apps, energy consumption is becoming a critical factor in user satisfaction for both paid and free apps [3].

Energy related issues mainly involve the efficiency of hardware components such as the CPU and other electronic elements. However, Flinn and Satyanarayanan [4] pointed out that *"there is growing consensus that advances in battery technology and low-power circuit design cannot, by themselves, meet the energy needs of future mobile computers"* [4]. This observation has been confirmed by recent advances in *green* software engineering, which demonstrated how the source of energy leaks can be software-related as well [5], [6], [7], [8].

For instance, Sahin *et al.* [5] have shown that good design principles, and design patterns in particular, have a *negative* impact on the energy efficiency of mobile apps. Along the same line, previous studies have investigated the power efficiency consequences of refactoring, demonstrating the flip side of operations that are supposed to improve non-functional attributes of source code [7], [9].

Despite the aforementioned research efforts, Harman *et al.* [10] highlight that there is a lack of tools that quickly and efficiently measure the energy consumption of mobile applications. Existing tools fall into three main categories: (i) hardware-based, (ii) model-based, and (iii) software-based approaches. Together with their own advantages, such solutions present various limitations that adversely affect their practical applicability. While hardware-based tools are able to delineate the exact energy profile of a mobile app, they require *ad-hoc* hardware components that are expensive and difficult to set up. Model-based approaches try to define mathematical functions able to estimate the energy consumption of mobile apps on a given hardware device. However, such tools require careful calibration of the parameters to correctly estimate power consumption. Finally, software-based approaches estimate the power profile of a mobile application solely relying on the system's functionalities of a device, as an example the CPU frequency. Thus, these tools can be considered hardware-assisted since they rely on measurements obtained by physical hardware components (*e.g., cpu, battery, etc.*), but without needing any specialized hardware tools. By nature, they are easier to use and cheaper than pure hardware-based solution. As a drawback, they are *supposed* to be less precise than hardware-based approaches [11].

In this paper, we aim at investigating to what extent software-based tools are less precise than hardware-based tools for energy consumption. In other words, *is the higher cost of hardware-based solutions justified by a sensibly more accurate energy profiling? Can software-based solutions lead to close measurements without any cost overhead?* To answer these questions, we built a novel tool for extracting the energy profile of mobile applications, which we coined PETRA (**P**ower **E**stimation **T**ool for **A**ndroid), specific for Android OS. PETRA relies on the publicly available `Project Volta` Android tools[1] and has the following characteristics:

- **Efficiency and Granularity.** PETRA is able to quickly estimate the energy consumed by an app at the method

[1]https://developer.android.com/about/versions/android-5.0.html

L. Zhang et al., "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," 2010 IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2010, pp. 105-114.

S. Hao et al., "Estimating mobile application energy consumption using program analysis," 2013 35th International Conference on Software Engineering (ICSE), 2013, pp. 92-101.

D. Di Nucci et al., "Software-based energy profiling of Android apps: Simple, efficient and reliable?," 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2017, pp. 103-114.

Context and Motivations

battery

ack of

energy
ing factor

HARDWARE MEASUREMENT

- ✓ More realistic measurements
- ✗ Complex to set up and use

SENSORS ESTIMATION

- ✓ Estimates using sensor data
- ✗ Less precise

STATIC ESTIMATION

- ✓ Does not execute the code
- ✗ Coarse-grained estimation

Mining Energy Traces to Aid in Software Development: An Empirical Case Study

Ashish Gupta[1], Thomas Zimmermann[2], Christian Bird[2], Nachiappan Nagappan[2], Thirumalesh Bhat[3], Syed Emran[3]

The Power of System Call Traces: Predicting the Software Energy Consumption Impact of Changes

Karan Aggarwal, Chenlei Zhang, Joshua Charles Campbell, Abram Hindle, and Eleni Stroulia

Department of Computing Science
University of Alberta
Edmonton, Canada
{kaggarwa, chenlei.zhang, joshua2, abram.hindle, stroulia}@ualberta.ca

Abstract

Battery is a critical resource for smartphones. Software developers as the builders and maintainers of applications, are responsible for updating and deploying energy efficient applications to end users. Unfortunately, the impact of software change on energy consumption is still unclear. Estimation based on software metrics has proved difficult. As energy consumption profiling requires special infrastructure, developers have difficulty assessing the impact of their actions on energy consumption. System calls are the interface between applications and the OS kernel and provide insight into how software utilizes hardware and software resources. As profiling system calls requires no specialized infrastructure, unlike energy consumption, it is much easier for the developers to track changes to system calls. Thus we relate software change to energy consumption by tracing the changes in an application's pattern of system call invocations. We find that significant changes to system call profiles often induce significant changes in energy consumption.

1 Introduction

Software tends to undergo constant development. Even during maintenance periods, bug fixes and new features are committed to a project. Developers understand that with change comes risk, especially the risk of performance regressions. One kind of performance that is difficult for developers to measure or predict is software energy consumption [1].

Due to this difficulty there has been much effort put into studying, explaining, and estimating software power use, especially on mobile platforms. Researchers have tried to model energy consumption on smartphones by creating energy consumption models based on hardware components [2], system run-time statistics [3], finite state machines [4], and byte code instruction usage [5]. However, none of the studies listed above have investigated the impact of software change on application energy consumption based on actual commit histories of software projects. Hindle *et. al.* [6] has made the first step toward revealing the relationship between software change and energy consumption. Much is to be learned by measuring energy consumption of multiple software versions and describing potential correlations between software metrics and energy consumption. Hindle *et. al.* [1] also created a dedicated test bed called *Green Miner*, and demonstrated that one requires multiple test runs to reliably calculate the energy consumption of an application on smartphones. This kind of testing methodology is useful but it requires specialized equipment and large amounts of time to obtain reliable data.

We want to help developers estimate if their source code changes cause changes in their soft-

A. Gupta et al., "Mining Energy Traces to Aid in Software Development: An Empirical Case Study," 2014 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2014, p. 1-8.

K. Aggarwal et al., "The power of system call traces: predicting the software energy consumption impact of changes", 24th Annual International Conference on Computer Science and Software Engineering (CASCON), 2014, p. 219–233.

ENT
ts

N
a

Estimating energy consumption using ONLY static predictors

LIGHTWEIGHT
No code instrumentation or execution

RAPID
Extract characteristics from the source code

OUR INVESTIGATION

Experiment with supervised learning to predict the likely consumption of Java classes in Android apps

Experimental Setup

How good is a machine-learning-based classification model in predicting the energy consumption level of Java classes in Android apps?

60 Android apps
(Palomba et al.)

620 Classes

5 Code Smells
14 Android-specific Smells

Method-level consumption estimates in Joule

Code Smells

Structural metrics

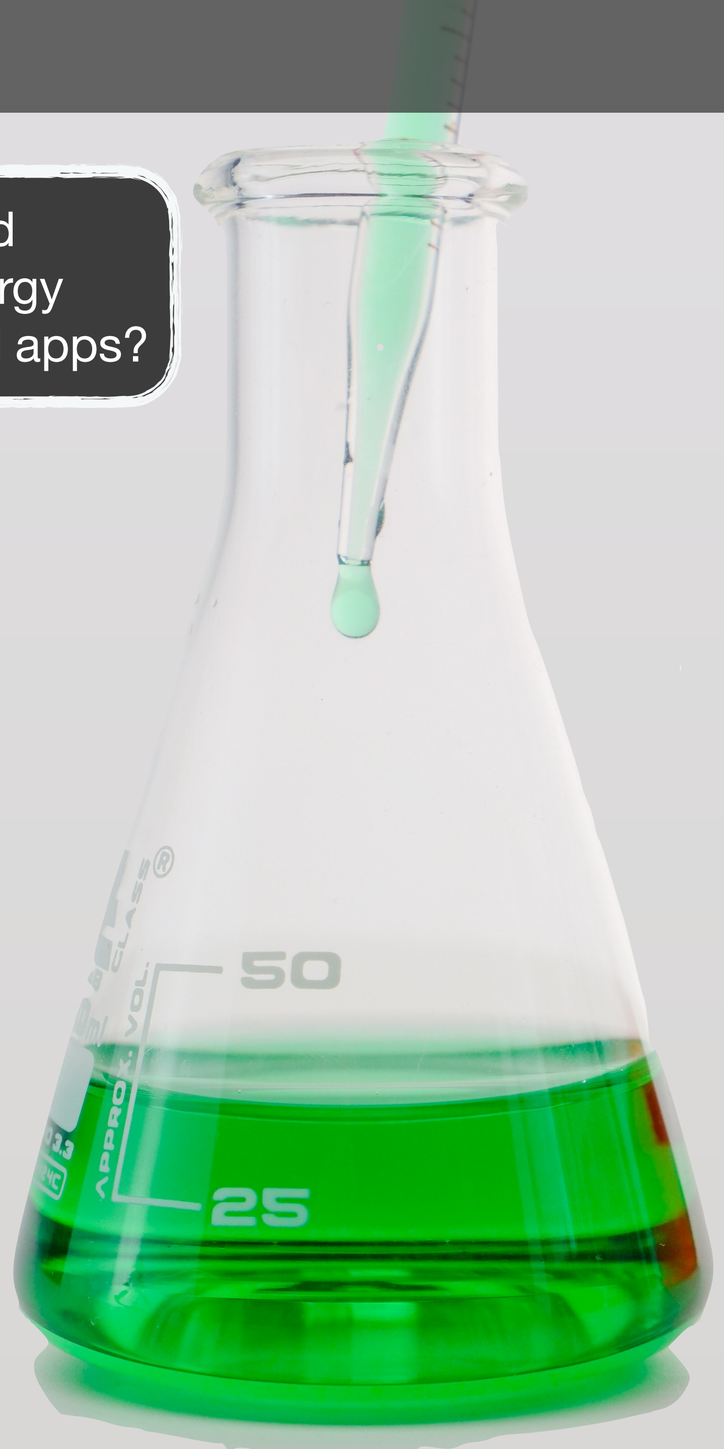
Class-level estimates

Log Normalization

Quartile Splitting

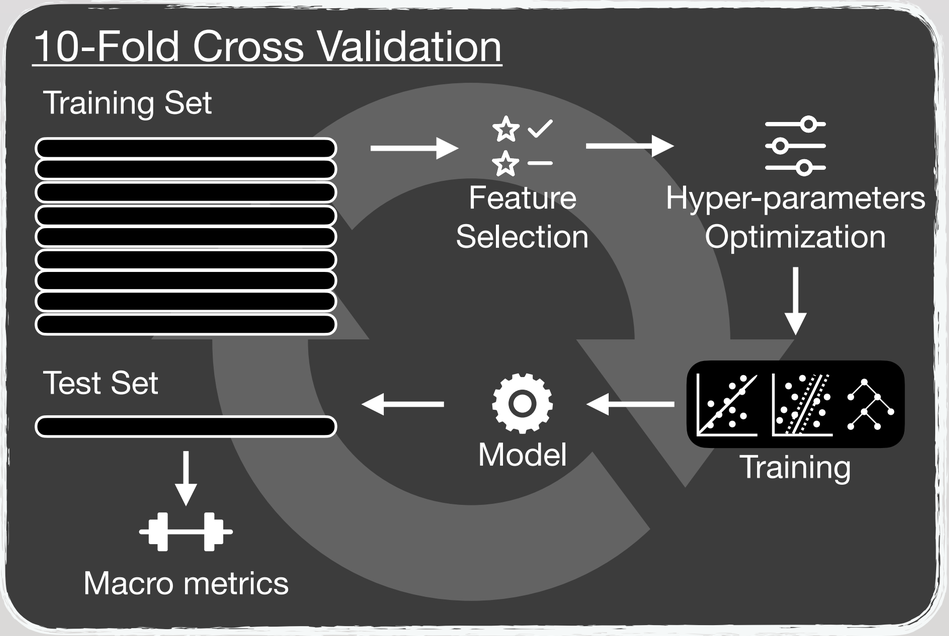
Independent Variables

Dependent Variable



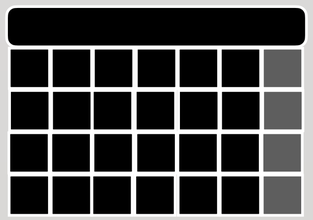
Experimental Setup

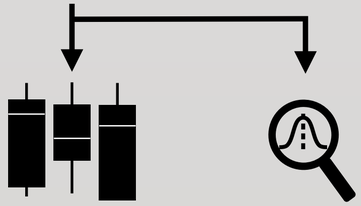
How good is a machine-learning-based classification model in predicting the energy consumption level of Java classes in Android apps?

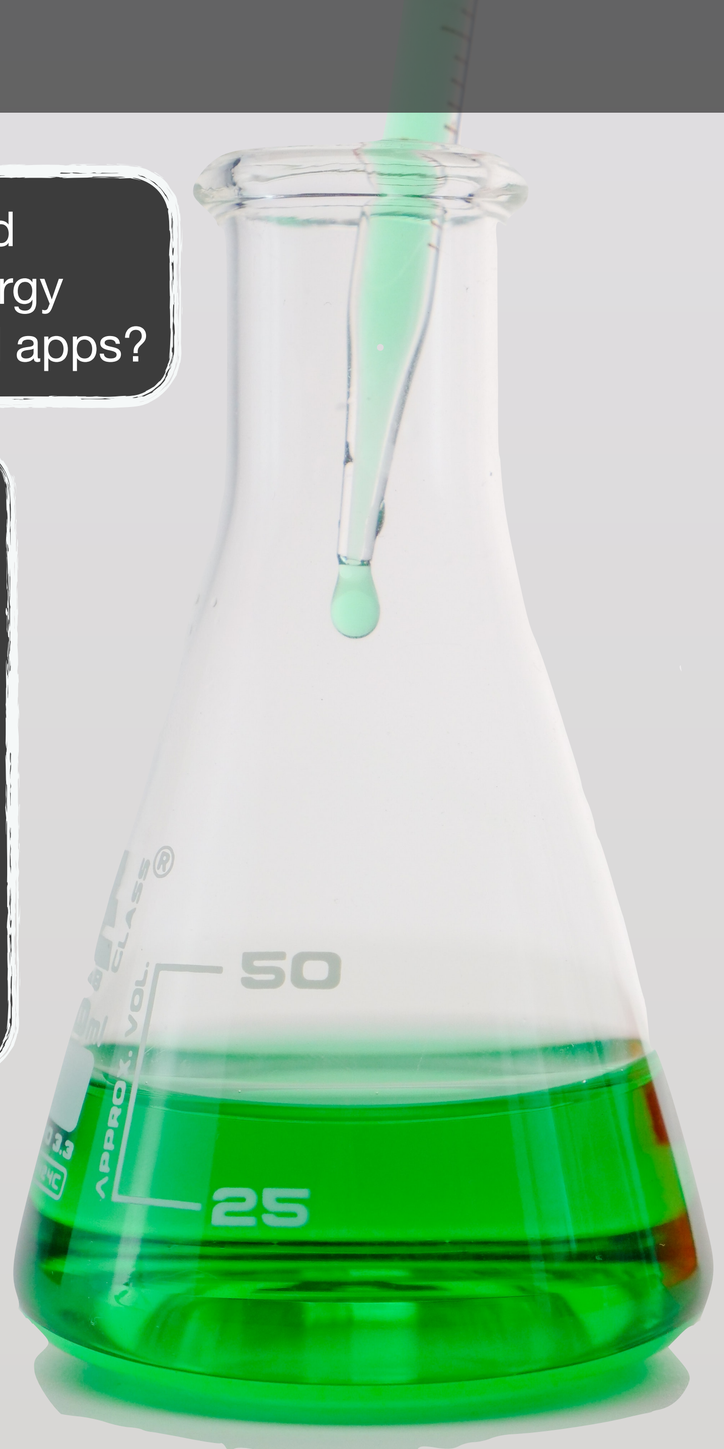


Independent Variables

Dependent Variable

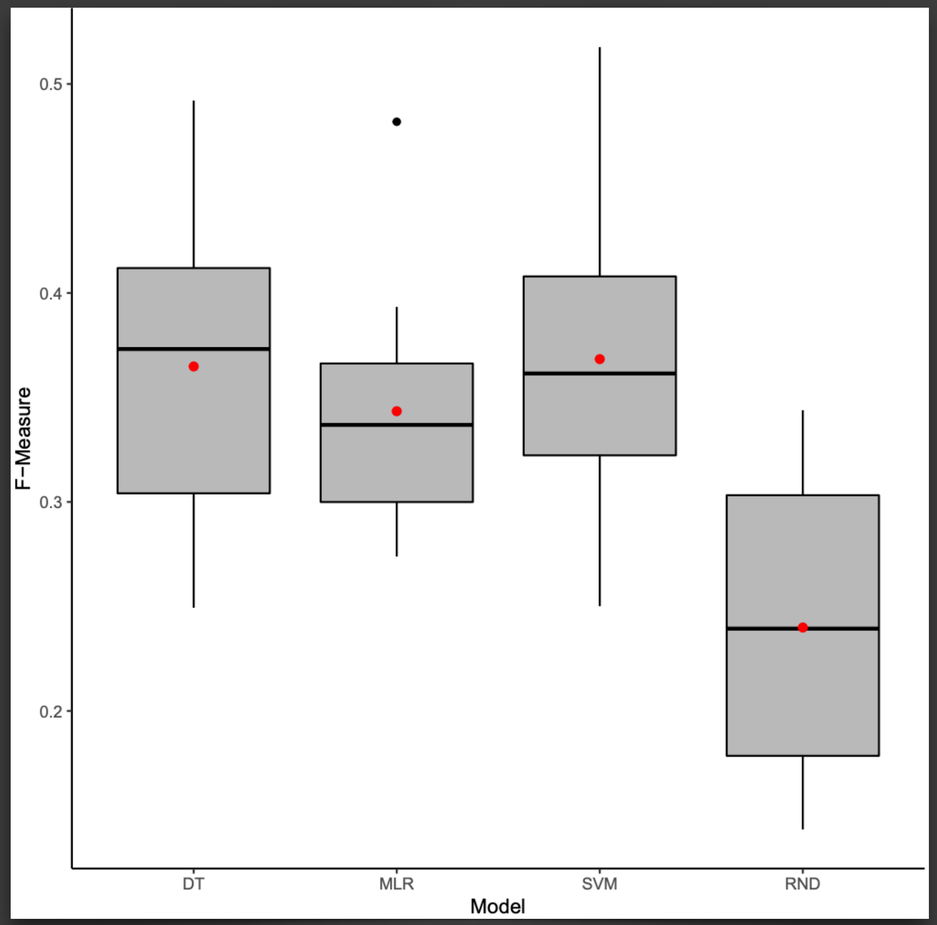






Results & Discussion

We discuss only the energy consumption values aggregated with the “sum”

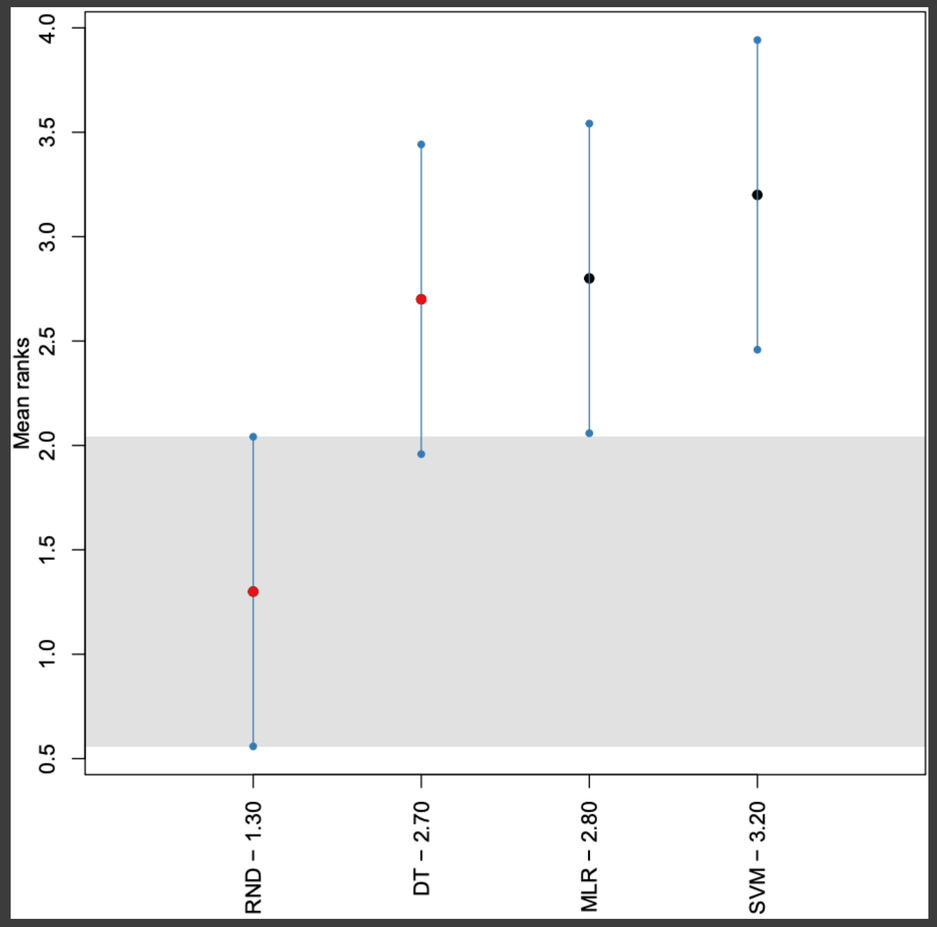


Better than Random!
Machine learning for energy estimation might be an idea!

Limited Performance
All models’ average F-measures did not go beyond 0.4. As expected, the results were not so good…

Results & Discussion

We discuss only the energy consumption values aggregated with the “sum”

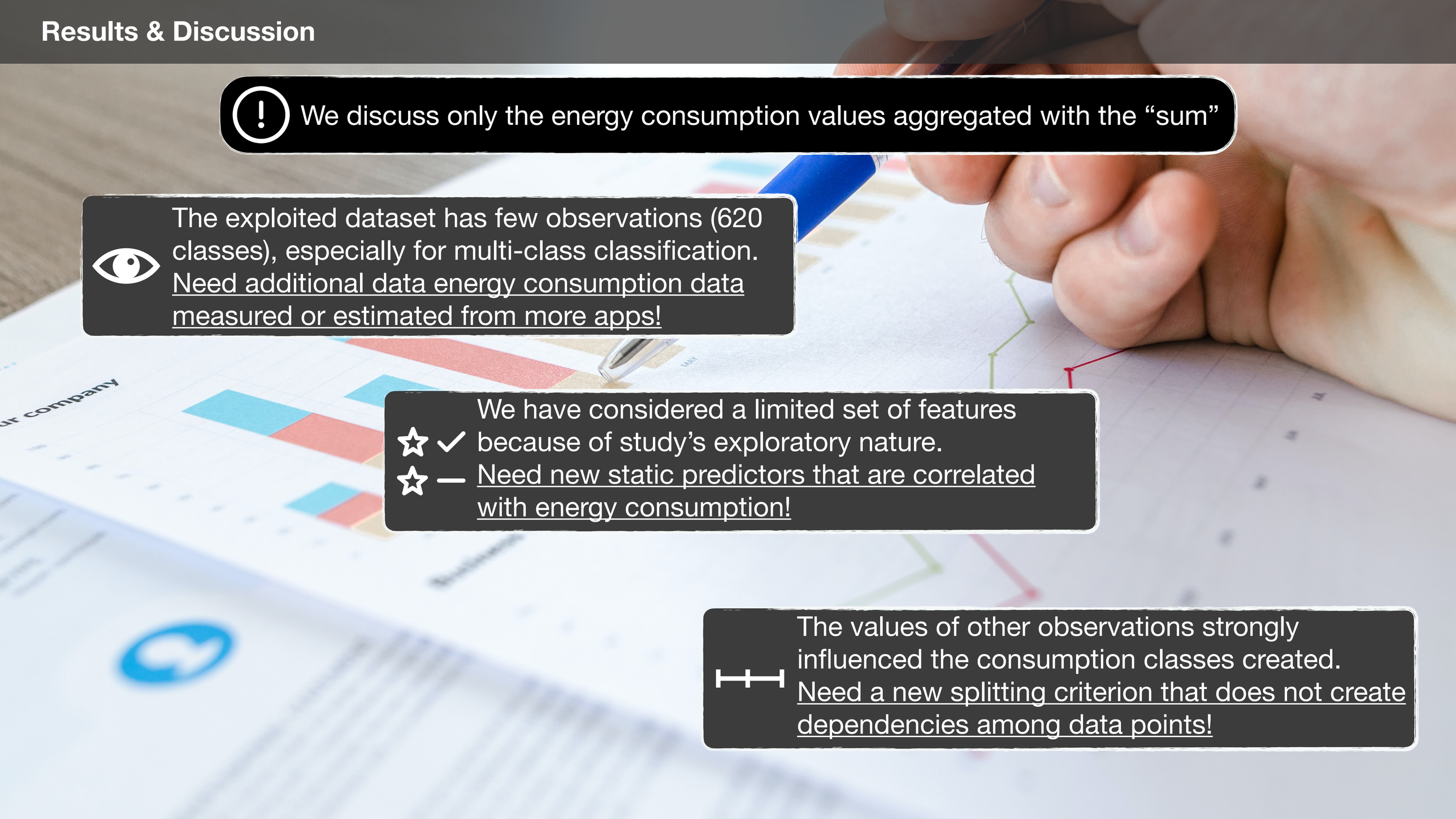


Better than Random!
Machine learning for energy estimation might be an idea!

Limited Performance
All models’ average F-measures did not go beyond 0.4. As expected, the results were not so good…

SVM wins, but…
SVM and MLR results significantly differ from the performance of DT and RND.

Results & Discussion



We discuss only the energy consumption values aggregated with the “sum”

The exploited dataset has few observations (620 classes), especially for multi-class classification. Need additional data energy consumption data measured or estimated from more apps!

We have considered a limited set of features because of study’s exploratory nature. Need new static predictors that are correlated with energy consumption!

The values of other observations strongly influenced the consumption classes created. Need a new splitting criterion that does not create dependencies among data points!

Conclusion

Experiment **within-project** scenario

Consider the effect of **confounding factors**

Interpret the models' predictions

Try new ways to label the **consumption levels**

Thank You!

sesa lab
SOFTWARE ENGINEERING
SALERNO

Predicting The Energy Consumption Level of Java Classes in Android Apps

An Exploratory Analysis

Emanuele Iannone, Manuel De Stefano, Fabiano Pecorelli, Andrea De Lucia

eiannone@unisa.it

https://emaiannone.github.io