information

MDPI

Article

Software Development of Business Intelligence Dashboards: Empirical Study of COSMIC ISO/IEC 19761 Size-Based Effort Estimation Using Machine Learning Techniques

Ammar Abdallah [1,*], Alain Abran [2], Munthir Qasaimeh [3] and Donatien Koulla Moulla [4]

[1] King Talal School of Business Technology, Princess Sumaya University for Technology, Al-Jubeiha, P.O. Box 1438, Amman 11941, Jordan
[2] Department of Software Engineering and Information Technology, École de Technologie Superieure (ÉTS), Montreal, QC H3C 1K3, Canada
[3] Department of Computer Engineering and Software Engineering, Polytechnique Montreal, Universite de Montreal, Montreal, QC H3T 0A3, Canada
[4] Centre for Augmented Intelligence and Data Science, School of Computing, University of South Africa, Johannesburg 1709, South Africa
* Correspondence: a.qasaimeh@psut.edu.jo

Abstract

Measuring the functional size of business intelligence (BI) and digital marketing analytics dashboard development can provide quantitative input for estimating the required time and cost of software development efforts. However, digital analytics practitioners currently lack practical guidance for planning dashboard development using measurement-based inputs that support reliable estimation models. Therefore, this study proposes the first Common Software Measurement International Consortium (COSMIC) ISO/IEC 19761 framework to measure the functional size of software user requirements of Power BI, Data Studio, and Tableau. An empirical study was conducted to measure the functional sizes of two datasets based on COSMIC. These functional sizes were considered inputs to several machine learning (ML) models to predict the effort required for BI dashboard development. The outcomes of these ML models showed that projects with consistent, low-variability effort should be prioritized when collecting or selecting data for training effort-estimation models and recommended the use of Stacking Regressor, Linear Regression, and Voting Regressor models for similar datasets because these models recorded the highest standardized accuracy for predicting BI development efforts with high generalization performance. These findings demonstrate the importance of applying COSMIC as a software engineering standard to BI and digital marketing analytics projects by offering measurements that allow for decisions informed by data rather than intuition or guesswork.

Keywords: business intelligence (BI); common software measurement international consortium (COSMIC); software effort estimation

Check for updates

Academic Editors: Prabhat Mahanti and Kalpdrum Passi

Received: 29 June 2026
Revised: 5 September 2026
Accepted: 9 September 2026
Published: 11 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Business intelligence (BI) allows organizations to collect, transform, and visualize data through dashboards that provide insights into key performance indicators (KPIs) [1–5]. BI is increasingly important for organizations to enable data-driven decision-making in many industries, such as digital marketing, manufacturing, retail, education, and health. BI dashboards play this role by synthesizing complex data and offering easier and more accessible decision-making dashboards [6]. Dashboard development must integrate business and user

requirements, including custom dashboards tailored to specific user contexts and business requirements, such as custom selections and drilldowns [6,7]. Their development requires proper planning, including collecting requirements (e.g., a digital marketing dashboard), designing, and integration with backend processes, such as data extraction, transformation, loading (ETL), and online analytical processing [8].

To effectively plan BI and digital marketing analytics dashboard software development, it is important to implement some of the best practices in software engineering, such as measuring the functional size of the software requirements and using such software size as the key quantitative input to estimate the required development effort and the related software development cost. Functional size-based effort software estimation is important in software project management to ensure that software is delivered as planned, considering time and budget [9].

Functional size measurement (FSM) standards based on functional user requirements lead to measurement results that are consistent across teams and domains, easy to understand, and correlated with the effort required to design, build, test, and maintain software [10,11]. Of the five current FSM ISO standards (IFPUG (ISO/IEC 20926), MKII (ISO/IEC 20698), NESMA (ISO/IEC 24570), and FISMA (ISO/IEC 29881) and Common Software Measurement International Consortium (COSMIC)—ISO/IEC 19761, COSMIC is the only one considered for the second-generation FSM. It is based on software engineering principles and is applicable to various domains, such as mobile applications, web applications, assembly programs, the Internet of Things (IoT), and real-time applications [11].

None of the studies in the literature have directly investigated the feasibility and relevance of FSM in BI and digital marketing analytics dashboard development and effort estimation. However, some studies have been conducted on sizing data warehouses using the COSMIC FSM. The studies in [12–14] proposed the use of the COSMIC FSM method to size data warehouse systems. They illustrated that COSMIC can be applied to layers in data warehouse systems, including various components, such as data staging areas and data marts. In addition, the COSMIC group published a public data warehouse guideline to demonstrate how the COSMIC FSM is applied in data warehouse system measurements [15]. However, these studies are limited to data warehouse environments and do not extend to other BI architectural layers (such as data presentation, processing, and data sources) and do not apply COSMIC FSM to BI dashboard development and effort estimation. Therefore, there is a gap in addressing the functional size measurements of all BI architectural layers in the context of dashboard development.

Consequently, BI dashboard and digital analytics practitioners lack guidance on how to plan dashboard development based on measurements that can be used as key inputs for estimating time and cost. Therefore, this study makes the following contributions:

1. It offers the first theoretical COSMIC application for BI dashboard development. This is a novel theoretical basis that presents BI dashboard setups as measurable software components.
2. It has the potential to be replicated as a guideline in BI dashboard development.
3. An empirical study was conducted with 150 BI dashboard development for digital marketing projects and machine learning techniques were used to estimate the software development efforts.
4. It offers a theoretical alignment between COSMIC and Vercellis' methodology [16] for implementing BI platforms and explains how COSMIC can be used to assist in the project management of BI phases.

The remainder of this paper is organized as follows: Section 2 presents the related work on sizing software using COSMIC-ISO 19761. Section 3 presents the sizing of three-layer BI dashboard development projects with COSMIC software functionality. Section 4 presents

the empirical research methodology approach to investigate the size–effort relationship in a set of 150 BI dashboard projects developed with Data Studio. Section 5 presents a data analysis of the size–effort relationship using machine learning (ML) techniques. Section 6 presents a discussion of the results, and Section 7 concludes the paper.

2. Related Work on Sizing Software with COSMIC—ISO 19761

A large body of literature exists on sizing software that uses COSMIC. For example, the study in [17], on a COSMIC automated measurement prototype, includes a case study on two X86 assembly programs from the embedded systems industry. These measurement results can be useful for effort and cost estimations in embedded software projects and for benchmarking. The study in [18] mapped the COSMIC concepts to the ARM assembly program and presented a COSMIC FSM procedure to measure, reporting that applying the functional size in COSMIC function points (CFPs) on an ARM program can benefit the embedded systems industry, such as IoT and portable devices, and will help the compiler industry, as the CFP count can provide insights into the difference between the sizes of the two compilers.

The study in [19] presented a COSMIC-based approach to address the functional size measurement of IoT devices by presenting mapping guidelines and best practices for using COSMIC to quantify the size of IoT devices developed by the ESP8266 System-on-Chip (SoC) using the Arduino platform. The study reported that measuring the functional size of an IoT device is important for maintenance. For real-time embedded software (RTES) projects, studies [19,20] recommend using COSMIC FSM as a standard method for the FSM of IoT devices.

Other studies have proposed innovative approaches for applying COSMIC to software codebases. For example, the study in [21] proposed a measurement for the functional size of any software written in the C programming language and recommended that COSMIC FSM be used for benchmarking and estimation purposes for projects in the C programming language. The study in [22] developed a COSMIC FSM programming language compiler to automatically calculate the COSMIC functional size based on the source code, which can track the functional size during the software development life cycle and workflow.

Other studies [11,23] proposed the application of COSMIC FSM in the emerging field of quantum computing. The study in [11] proposed Q-COSMIC as a standardized method for measuring the functional size of quantum computing by extending the COSMIC (ISO/IEC 19761) functional size measurement method. Applying COSMIC to quantum computing will allow organizations to manage quantum projects with a level of capability similar to that in classical software development.

The literature on estimating mobile development efforts, such as [24], posits that applying COSMIC to mobile application size estimation is effective in managing development projects. Moreover, Ref. [25] mapped COSMIC and the Unified Modeling Language (UML) to estimate the size of mobile applications. In addition, the study in [26] applied COSMIC to 18 mobile applications for postnatal care and reported that COSMIC is a reliable method for sizing software in mobile applications and healthcare. In the study in [27], a regression model was presented to predict the test effort for mobile applications using COSMIC, which can be used at the start of the mobile app testing life cycle. The results showed that it is suitable and provides good predictions using multiple linear regression for the employed dataset of mobile applications.

The study in [28] reported that COSMIC is widely used in various industrial applications, such as the following:

1. The automotive industry, where Renault used COSMIC to estimate project effort and cost for embedded control units (ECUs) in vehicles.

2. In agile software development, organizations have replaced story points with COSMIC function points (CFPs) for effort estimation. This has led to more accurate estimates that improved predictability in scrum-based projects and was later confirmed in the context of agile software projects [29].
3. COSMIC has been integrated into tools such as MATLAB Simulink, UML models, and CASE tools, enabling automated FSM [28].

Other COSMIC applications include the following:

1. Web applications, such as in [30], with a measurement procedure designated as an object-oriented hypermedia COSMIC function point (OO-HCFP) for automatically sizing web applications developed with a model-driven web development method.
2. Effort estimation based on machine learning, such as in [29], which compared COSMIC FSM with story points using a support vector regressor (SVR) model for effort prediction. The results revealed that the COSMIC FSM is more accurate than story points.
3. A COSMIC-based measurement framework for Google Analytics in [31] and its application in estimating effort (e.g., hours).
4. In addition to other examples, such as using COSMIC to measure enterprise architecture [32], further guidelines and practical applications are documented on the COSMIC official website [33].

3. Sizing the Software Functionality of a Three-Layer BI Dashboard Development Project

3.1. Overview of the COSMIC (ISO 19761) FSM Method

According to [34], COSMIC sizing is divided into the following phases:

1. Strategy phase: this phase includes the identification of the measurement scope, purpose, software boundaries, layers, and services provided by the software (known as functional user requirements, FURs).
2. The mapping phase includes the identification of functional processes and data movements and maps the functional user requirements to the COSMIC Generic Software Model.
3. The measurement phase included calculating the size by counting the data movements for each function. COSMIC function points (CFPs) were used as the measurement units.

One CFP is assigned to each data movement of a single data group. Data movement is a core concept in COSMIC and corresponds to the interaction between functional users and software applications. COSMIC defines four (4) types of data movements as follows:

- Entry (E): the data group moved from a functional user to the software;
- Exit (X): a data group moves from software to a functional user;
- Write (W): a data group moved from software to persistent storage;
- Read (R): a data group is transferred from persistent storage to software.

Figure 1 illustrates an example of data movement for a given software application. The functional size of such software is calculated by summing all data movements across all functional processes, as shown in Equation (1).

$$\text{Size(functional processes)} = \Sigma\ \text{size(Entries)} + \Sigma\ \text{size(Exits)} + \Sigma\text{size(Reads)} + \Sigma\ \text{size(Writes)} \quad (1)$$

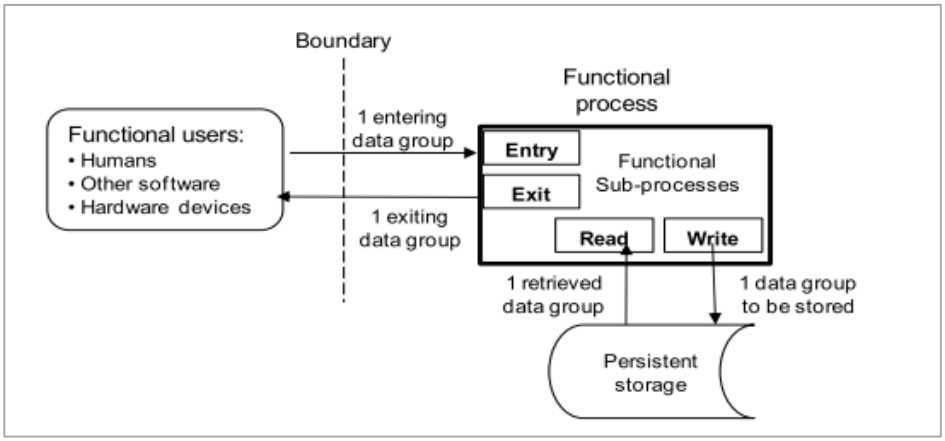


Figure 1. Data movements for a given software application [34].

3.2. Sizing Strategy for Sizing a Dashboard Using Three Distinct BI Tools

The sizing strategy phase consists of determining the purpose and scope of the measurement, followed by the identification of the functional requirements, software layers, software functional users, and related software boundaries.

3.2.1. Purpose of Sizing

The purpose of measurement refers to "to a statement that defines why a measurement is being made, and what the result will be used for" [34]. The purpose of measuring the functional size is to develop a dashboard using three distinct BI Tools (Data Studio, Power BI, and Tableau), which include various measurement purposes and can be determined based on the parties involved.

- To plan the resources required for designing and developing dashboards using the specified BI tools based on the estimates;
- Measure the change in functional requirements to manage scope as the project progresses;
- Once a project is completed, the functional size is measured to calculate the productivity of the development project.

3.2.2. Scope of Measurements

The scope of measurement is referred to as a "set of functional user requirements to be included in a specific functional size measurement instance" [34]. The scope of measuring functional size is limited to the Software Being Measured and is based on the purpose of the measurement. For example:

- Scope for the whole application: "If the objective is to identify the productivity of a software project across one piece of software and is on one technical platform";
- Scope for each piece of software in each layer: "If the objective is to measure a software across multiple architectural layers, the work output of each layer should be measured independently under a separate measurement scope, and the results then aggregated to reflect the full project effort";
- Scope for each application: "If the objective is to measure the total size of an application portfolio, a separate measurement scope should be defined for each application and should be measured separately".

Therefore, based on the three-layer architecture illustrated in Figure 2, the scope of measuring the functional size resulting from the development of a dashboard is defined for each layer (e.g., data presentation, data processing, and data source layers) separately. Functional users, as determined by the measurement's purpose and scope, are not limited to human operational users of the delivered dashboard; they may include other software

components and external systems with which the measured software exchanges data. The data-source layer is measured specifically where genuine development effort was required to address it—for example, where connectors or data movements to an external database, API, or cloud platform had to be developed or modified as part of the project. Functionality already supplied by an external system, requiring no development work, is not attributed to the measured software's size.

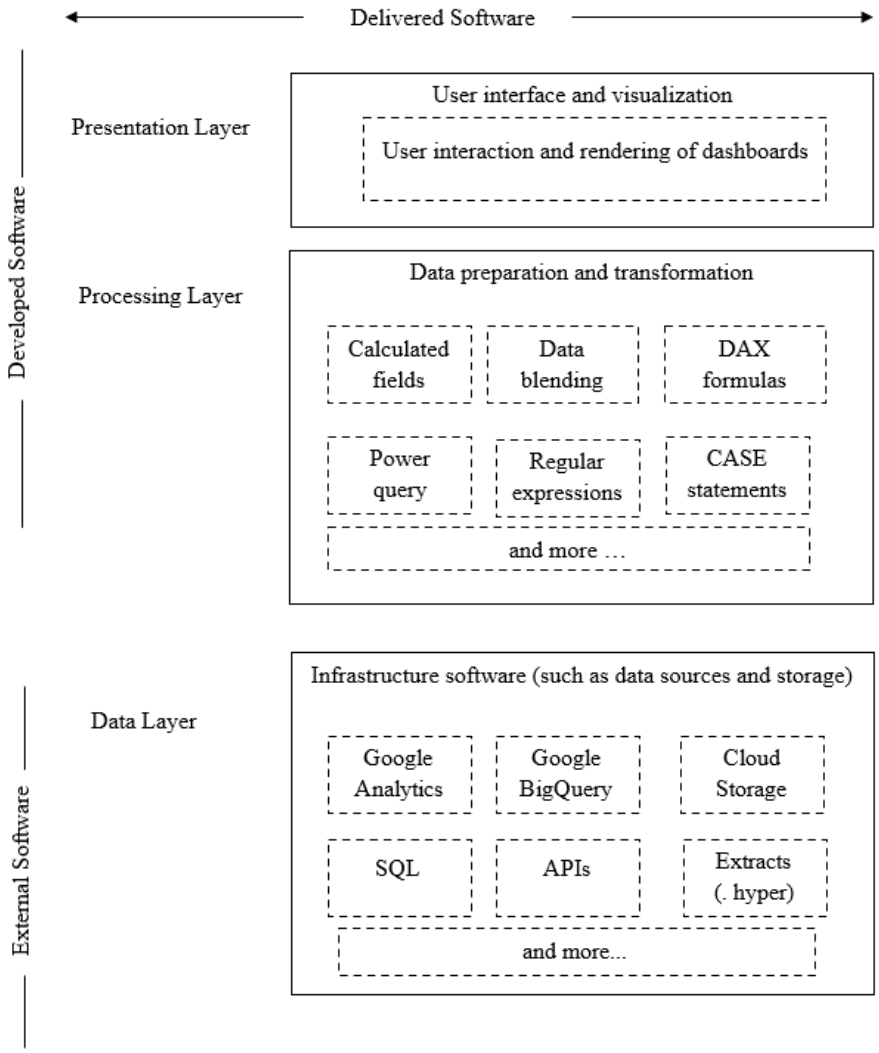


Figure 2. The proposed three-layer architecture layer of a dashboard development.

3.2.3. Identification of the Functional User Requirements (FURs)

According to [34], COSMIC defines an FUR as a "sub-set of the user requirements describing what the software shall do, in terms of tasks and services". Accordingly, an FUR is derived by the measurer from software engineering artifacts (e.g., use cases, user stories, architecture, and E/R diagrams) that can be produced before the software implementation. Consequently, the functional size of the dashboard can be measured prior to its deployment.

Table 1 illustrates the most common FUR for developing a dashboard based on BI tool documentation.

- Data Studio documentation [35];
- Power BI documentation [36];
- Tableau documentation [37].

Table 1. Most common functional user requirements (FURs) for developing a dashboard using three distinct BI tools.

Functional User Requirements (FURs)	BI Tools Environment Description for Power BI, Data Studio and Tableau
Connect to data sources	The system connects to SQL, Excel, API, and Cloud in Power BI and Tableau and to Google Analytics, BigQuery, Excel, and Google Sheets in Data Studio.
Data transformation	The system uses Power Query (M language) in Power BI, Custom fields, and blended data sources in Data Studio and data joins and data blending in Tableau.
Calculated fields	The system uses Data Analysis Expressions (DAX) formulae in Power BI and CASE statements in Data Studio and Calculated fields in Tableau.
Data storage	The system uses dataflows and import mode in Power BI; live connections and extracts in BigQuery and in Data Studio; and Tableau extracts (.hyper) and external storage in Tableau.
Aggregation	The system uses SUM, AVG, COUNT, and ROLLUP in Data Analysis Expressions (DAX) in Power BI; Aggregation in calculated fields and charts in Data Studio; and Level of Detail expressions (LOD) expressions in Tableau.
Filters	The system creates filters in the data (e.g., Regular Expressions in Data Studio) and creating drill downs.
Data blending	The system blends multiple data sources.
Visualization	The system creates an interactive dashboard that includes items such as tables, charts, scorecards, maps, etc.
Scheduled refresh	The system runs a scheduled query for data extract and refresh.
Data export	The system exports data to Excel or any other platform or format.

3.2.4. Identification of the Dashboard Software Architecture Layers

COSMIC specifies that the measurement must be performed for a single software layer at a time, where a layer is defined as a partition resulting from the functional division of a software system [34]. In the context of software architecture, the measurer must size the layers of the software architecture identified within the FSM scope(s). COSMIC notes that the layers can be as follows:

- Hierarchical (referring to software in layer A that can use services from software in layer B, but not vice versa);
- Bidirectional (referring to software in layer A that can use services from software in layer B and vice versa);
- Software components in a three-layer architecture (referring to software architecture, including user interface, business rules, and data service components) and other software architectures can be explored in COSMIC documentation.

In this study, we considered the development of a dashboard as a three-layer architecture layer, as illustrated in Figure 2. Each layer has software components and delivers distinct functions and services as follows.

- Data Presentation Layer: this layer handles user interaction with the dashboard and reports;
- Data Processing Layer: this layer handles the connection to the data sources, data calculations and transformations, and sends the data to the presentation layer;
- Data Source Layer: this layer is considered the infrastructure software that provides data to the processing and presentation layers of the system.

3.2.5. Identification of the Functional Users

Functional users can be human users or systems (including other software) that interact with the measured software [34]. For example, a sensor that detects the temperature is considered a functional user, a barcode reader at a supermarket is a functional user, and software 'A' can be a functional user of software 'B' given that software 'B' is the measured software.

The functional users of the proposed three-layer architecture of the dashboard development are listed in Table 2, which illustrates the architecture layer, its functional users, and a description.

Table 2. Identification of the functional users in the three-layer architecture of a dashboard development.

Architecture Layer	Functional Users	Description
Data presentation layer	Processing layer	The processing layer sends updated data to the charts in the presentation layer.
	Human user	The human user reads reports, apply filters, control access, and share reports.
	Other systems such as embedded systems and web applications	These systems (such as an API) can read the data and display it in another system (such as a website) as a dashboard.
Data processing layer	Presentation layer	The presentation layer fetches the transformed data to the dashboard and reports.
	Data layer	The components in the data layer send data to the processing layer.
	Human user	The human user can directly develop a regular expression, a calculated field, and other components such as scripts.
Data source layer	Processing layer	The processing layer requests data from the data sources and storage.

3.2.6. Establishing the Software Boundary

A boundary is a conceptual border (also known as an interface) that splits the software measured from its functional user [34]. Functional users, the software, and persistent storage all communicate across this boundary. Figure 3 illustrates a context diagram that includes functional users, software boundaries, and persistent storage.

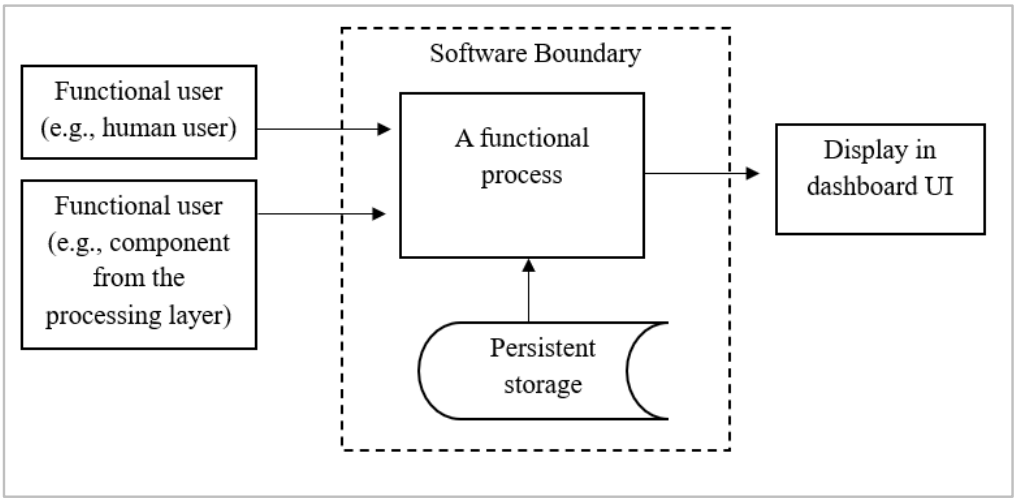


Figure 3. Proposed example of a dashboard software context diagram.

3.3. The COSMIC Mapping and Measurement Phases for a Three-Layer BI Dashboard Development Project

The COSMIC mapping and measurement phases for this three-layer BI dashboard development project consist of the identification of the functional processes, their objects of interest, and data groups, followed by the identification of their corresponding data movements. In this section, we present several examples of this.

3.3.1. Identification of Functional Processes

A functional process (FP) is the smallest unit of functionality triggered by a functional user and should meet the functional user requirements. It performs meaningful tasks that contribute to the system's purpose. It is defined independently of other processes (e.g., it should not rely on multiple other processes to function) and resides in the same layer as the measured software. Tables 3–5 illustrate some functional processes in each layer for dashboard development.

Table 3. Functional process in the data presentation layer.

Functional Process	Example
User interaction with the dashboard	User applies a filter User interacts with a chart User exports a dashboard

Table 4. Functional processes in the data processing layer.

Functional Process	Example
The system aggregates data	The system receives the data, aggregates it based on an attribute (e.g., age) and returns the results ready for display.
The system executes regular expressions	The system executes regular expressions after receiving the user's request to filter the data.
The system executes (DAX, LOD, SQL) calculations	The execution is triggered by a user's request to process calculations on the data.
The system blends data from multiple data sources	The system fetches raw data from various sources and dynamically blends data for a report.
The system executes scheduled data refresh and extract	The system autonomously executes a process based on a triggering event (e.g., scheduled refresh).

Table 5. Functional processes in the data source layer.

Functional Process	Example
Retrieve data from SQL	The system retrieves data from SQL database
Retrieve from Google Cloud	The system retrieves data from Google Cloud
Retrieve from Google Analytics	The system retrieves data from Google Analytics
Retrieve from Data Warehouse	The system retrieves data from Data Warehouse
Retrieve data from API	The system retrieves data from API

3.3.2. Determining Data Groups and Objects of Interest

The objects of interest are the entities processed and exchanged by the system [34]. For instance, software can receive objects of interest, such as a 'message' including a data group: message ID, sender ID, and recipient ID. For the three-layer dashboard architecture discussed earlier, the object of interest is the entity that the system processes, stores, or moves between the layers. These objects can represent physical entities (users, reports, and databases) or logical components (key performance indicators (KPIs), filters, and transactions).

Each functional process involves transferring data groups between objects of interest, such as a 'sales report', including attributes (e.g., date, region, and revenue) sent back to the

dashboard for display. For the three-layer dashboard architecture, Tables 6 and 7 illustrate the objects of interest and data groups, respectively.

For the three-layer dashboard architecture discussed earlier, the object of interest is the entity that the system processes, stores, or moves between the layers. These objects can represent physical entities (users, reports, and databases) or logical components (key performance indicators (KPIs), filters, and transactions). Each functional process involves transferring data groups between objects of interest, such as a 'sales report' including attributes (e.g., date, region, and revenue) sent back to the dashboard for display. For the three-layer dashboard architecture, Tables 6 and 7 illustrate the objects of interest and data groups, respectively.

Table 6. Objects of interest for the three-layer dashboard architecture.

Architecture Layer	Objects of Interest	Details
Data presentation layer	Dashboard user interface (UI) User actions	Charts, reports, tables, KPIs displaying processed data. Filters, slicers, drilldowns, selections made by users
Data processing layer	Filter component Aggregation component Blending component Other calculation and transformation components	Filtered data Computed data summaries (e.g., revenue over time) Blended data Any aggregation and formula calculations
Data source layer	Raw data sources	SQL databases, API sources, CSV files

Table 7. Data groups for the three-layer dashboard architecture.

Architecture Layer	Data Group	Attributes	Example
Data presentation layer	User Inputs (Filters, Slicers)	Filter Type, Selected Value, User ID	Region = 'North America'
	Visualization Output	Chart Type, Data Points, Aggregation Level	Bar chart showing revenue by region
Data processing layer	Calculated Metrics	KPI Name, Formula Applied, Computed Value	Profit Margin = Revenue − Cost
	Aggregated Results	Grouped Data	Total Sales = $500,000 for Q1
Data layer	Data Retrieval	Table Name, Columns, Rows	Retrieve transactions from SQL, Retrieve traffic from Google Analytics for Q1

3.3.3. Identification of Data Movements

The next step was to identify all data movements within each functional process. for the three-layer dashboard architecture Tables 8–10 illustrate the functional processes, together with their data movements and a related description.

- In Table 8, an Entry data movement is counted if the data crosses the boundary from the user passing into the software. Exit data movement, crossing the boundary from the software to the user, is counted if the software dashboard shows, for example, a chart to the user.
- In Table 9, 'Request data from data layer' is counted as an Exit, since it sends data across this layer's boundary to the data source layer; the data layer's response is counted as a new Entry in the corresponding functional process. 'Apply business logic' is an internal Entry/Write, since it neither originates from nor crosses to another layer.
- In Table 10, Reads from external databases, APIs, or files (e.g., SQL, data warehouse, GA4, Google Cloud, and a linked API) are counted as boundary-crossing, since each

external source is a functional user of the software under measurement (COSMIC ISO/IEC 19761); each retrieval is recorded as an Entry (request), Read (data retrieval), and Exit (delivery of results) sequence.

Table 8. Examples of data movements in the data presentation layer.

Functional Process	Data Movement	Description
Capture user interaction	E, W	User provides a filter or input; UI captures (E) and records it (W).
Transmit request to processing	E, X	UI receives validated input (E) and sends it to the processing layer (X).
Receive processed data	E, W	UI receives results from processing (E) and stores them for rendering (W).
Render dashboard visuals	E, X	UI receives data for display (E) and outputs charts/tables (X).

Table 9. Examples of data movements in the data processing layer.

Functional Process	Data Movement	Description
Register query request	E, W	Processing receives the user request (E) and records/logs it (W).
Request data from data layer	E, X	Processing receives query instructions (E) and sends them to the data layer (X).
Apply business logic	E, W	Processing receives raw data (E), performs calculations (SUM, AVG, COUNT), and stores computed results (W).
Deliver processed results to UI	E, X	Processing receives final computed results (E) and sends them to the UI (X).

Table 10. Examples of data movements in the data source layer.

Functional Process	Data Movement	Description
Retrieve data from SQL	E, R, X	Data layer receives request (E), reads data from SQL (R), and delivers results (X).
Retrieve data from data warehouse	E, R, X	Data layer receives request (E), reads data from data warehouse (R), and delivers results (X).
Retrieve data from Google Analytics (GA4)	E, R, X	Data layer receives request (E), reads data from GA4 (R), and delivers results (X).
Retrieve data from Google Cloud	E, R, X	Data layer receives request (E), reads data from Google Cloud (R), and delivers results (X).
Retrieve data from API	E, R, X	Data layer receives request (E), reads data from a linked API (R), and delivers results (X).

3.3.4. Example with the Data Studio Dashboard

In the following example, the objective is to measure the functional size of the Data Studio dashboard using four functional processes:

1. User selects 'Create Report' and chooses analytics as the data source;
2. User adds a Region-Level Summary Chart (e.g., Sales by Region);
3. User enables drill-down interaction;
4. User configures time series chart.

Tables 11–14 illustrate each functional process with their Entry, Exit, Write, and Read data movements.

Table 15 demonstrates the COSMIC measurement phase and functional size by applying Equation (1) to Entry, Exit, Read, and Write data movements.

Table 11. COSMIC size of the functional process: user selects 'Create Report' and chooses Google Analytics as the data source.

Data Movement Type	Number of Data Movements	Description of Data Movements
Entry (E)	1	The user initiates the connection by selecting 'Add data', choosing Google Analytics, logging in via OAuth, and selecting the desired GA4 property. This is a functional user input.
Read (R)	2	Read 1: Data Studio reads the list of available GA4 properties after authentication. Read 2: It reads the schema (dimensions, metrics, metadata) for the selected GA4 property.
Write (W)	1	Data Studio writes the data source configuration (property ID, selected fields, connection settings) to its internal storage or project structure.
Exit (X)	1	After the connection is established, Data Studio displays the data schema (dimensions and metrics) in the user interface, confirming a successful setup. This is a data movement from software to user.

Table 12. COSMIC size of the functional process: user adds a Region-Level Summary Chart (e.g., Sales by Region).

Data Movement Type	Number of Data Movements	Description of Data Movements
Entry (E)	2	Entry 1: The user selects chart type. A Functional input specifying that a new chart should be created. Entry 2: The user configures the chart. Selects 'Region' as dimension, and 'Sales' as metric Input of chart configuration (metrics, dimensions).
Read (R)	2	Read 1: Data Studio fetches available schema to populate dropdowns Read 2: Data Studio retrieves actual sales data grouped by region.
Write (W)	1	Data Studio saves the chart settings internally.
Exit (X)	1	Displaying the visual chart (bar, map, pie, etc.) to the user.

Table 13. COSMIC size of the functional process: user enables drill-down interaction.

Data Movement Type	Number of Data Movements	Description of Data Movements
Entry (E)	2	Entry 1: User selects a chart and enables drill-down option. Entry 2: User configures drill-down levels (e.g., region, country, and city).
Read (R)	2	Read 1: Data Studio reads the schema to validate dimension hierarchy Read 2: Data Studio fetches sub-level data dynamically (e.g., countries in region).
Write (W)	1	Data Studio saves drill-down configuration to report.
Exit (X)	1	Data are displayed interactively in the chart.

Table 14. COSMIC size of the functional process: user configures time series chart.

Data Movement Type	Number of Data Movements	Description of Data Movements
Entry (E)	2	Entry 1: User adds a chart component. Entry 2: User configures metric (Revenue) and time dimension.
Read (R)	2	Read 1: Data reads available schema (to validate dimensions). Read 2: Data retrieves actual time series data from GA4.
Write (W)	1	Data Studio saves the chart to report.
Exit (X)	1	Data Studio displays the chart.

Table 15. COSMIC size of the Data Studio dashboard example.

Functional Process	E (Entry)	R (Read)	W (Write)	X (Exit)	Total Size in CFPs
User selects 'Create Report' and chooses Google Analytics as the data source	1	2	1	1	5 CFP
User adds a Region-Level Summary Chart (e.g., Sales by Region)	2	2	1	1	6 CFPs
User enables drill-down interaction	2	2	1	1	6 CFPs
User configures time series chart	2	2	1	1	6 CFPs
Total size in CFPs for the Data Studio Dashboard					23 CFPs

4. Empirical Research Study: Methodology Approach to Investigate the Size–Effort Relationship in BI Dashboard Development Projects

This section presents the empirical research approach selected to investigate the size–effort relationship using machine learning techniques. Section 4.1 presents the size data collection process for 150 BI dashboard projects, Section 4.2 presents the machine learning techniques to be used with the data collected, and Section 4.3 presents the criteria to be used to investigate the performance of the ML techniques on the datasets available.

4.1. Data Collection

A total of 150 BI dashboard projects for digital marketing developed with Data Studio were used for this study and divided into two datasets as follows:

- Dataset 1 contains information on 50 Data Studio projects for digital marketing, including user-triggered functional processes (based on user actions) and system-triggered functional processes (based on system actions, such as automated refresh);
- Dataset 2 contains information on 100 Data Studio projects for digital marketing that included only user-triggered functional processes (based on user actions) and excluded system-triggered functional processes (based on system actions, such as automated refresh).

The 150 projects analyzed in this study are illustrative of small-to-moderate BI dashboard development tasks and are not presented as representative of the full range of three-layer BI development project sizes. For each project included in these datasets, the following size attributes are documented, in addition to the recorded effort in hours: the count of data movements, and COSMIC functional size measured in COSMIC function points (CFPs). Tables A1 and A2 in Appendix A provide information on the datasets used in the empirical study.

These datasets were collected from practitioners who received training on COSMIC, applied COSMIC to Data Studio projects, and reported the actual hours required to develop the dashboard. In addition, practitioners went through the factual pre-check check (FMC) method provided in [38] to assess whether COSMIC was correctly applied.

4.2. Machine Learning Techniques

In the empirical study, eleven regression-based machine learning models were employed in this study, spanning linear, tree-based, instance-based, and ensemble approaches, including Linear Regression [39], Extra Trees [40], AdaBoost [41], Random Forest [42], Decision Tree [43], K-Nearest Neighbors [44], Gradient Boosting [45], Bagging Regressor [46], Support Vector Regressor, Voting Regressor, and Stacking Regressor [47]. A comprehensive description of these models is provided in our earlier work [31]. In the present study, these models were applied within a COSMIC-based effort estimation framework, with continuous numerical outputs predicted via regression [48]. Hyperparameters were set to fixed, moderate values—in several cases matching scikit-learn's own defaults (e.g., KNN n_neighbors = 5 and SVR epsilon = 0.1) and in others adjusted conservatively for the small sample sizes involved (e.g., constraining tree depth, increasing ensemble size beyond library defaults for more stable averaging), rather than tuned via grid or random search, to avoid any risk of hyperparameter selection leaking information from the cross-validation test folds. Full model configurations, including the base learners and final estimator used in the Voting and Stacking Regressors, are reported in Table A3 in the Appendix A. All 11 models were trained using Entry, Read, Write, Exit, and CFP together as the predictor set (five features). For K-Nearest Neighbors and Support Vector Regressor, feature scaling (StandardScaler) was fitted exclusively on each fold's training data and then applied to transform the corresponding held-out fold, preventing information leakage from the test data into the scaling parameters.

4.3. Empirical Setup and Performance Evaluation

In order to produce stable results, the setup included a repeated 10-fold cross-validation based on five randomly selected seeds as follows: 0, 7, 42, 123, and 2024. For each selected seed, the datasets were shuffled and split into 10 folds. Therefore, each ML model was trained in nine folds and tested on the remaining fold. The resulting out-of-fold predictions from all 10 folds were pooled before computing that seed's performance metrics (RMSE, R^2, MAE, MBRE, MIBRE, LSD, MMRE, MdMRE, PRED(25), and SA). This entire procedure was then repeated independently for each of the five seeds, yielding five seed-level metric values per model per dataset. The mean and 95% confidence interval across these five values (t-distribution, df = 4) are reported as the final result for each model. The implementation is available on our public GitHub (check the Supplementary Materials).

Several performance criteria in Equations (2)–(11) were applied to assess the performance of the ML models. These criteria are supported by the findings of a systematic literature review [49]. This empirical study builds on established evaluation practices in software development effort estimation research [50,51] by introducing a distinct set of evaluation criteria that together provide a thorough evaluation of model performance as follows:

- To evaluate the accuracy of the model, the root-mean-square error (RMSE) was used because it considers the size of the prediction errors. The coefficient of determination (R^2) was used to explain the variance in the target variable;
- Unlike the MMRE, which is known to favor underestimation, the mean balanced relative error (MBRE, Equation (5)) and mean inverted balanced relative error (MIBRE, Equation (6)) offer more balanced assessments by being less susceptible to systematic bias [51–55]. Furthermore, the standardized accuracy (SA, Equation (8)), established in [56], was applied to assess model stability across different time periods. SA is useful for determining whether an estimation method performs better than a random-guessing baseline (P0), where MAE_p0 is defined as the mean absolute error of predicting each project's effort by randomly sampling another project's actual value;

- The logarithmic standard deviation (LSD, Equation (7)), measures the spread of the log-transformed residuals, $\lambda_i = \ln(y_i) - \ln(\hat{y}_i)$, adjusted by a bias-correction term $s^2/2$ that accounts for the systematic bias introduced when converting log-scale predictions back to normal-scale effort values [51];
- The mean magnitude of relative error (MMRE, Equation (9)) and its median-based counterpart (MdMRE, Equation (10)) were also reported for comparability with prior effort-estimation studies, despite MMRE's known bias toward underestimation;
- PRED(25) (Equation (11)) reports the percentage of predictions falling within 25% of the actual value.

$$\text{Root Mean Square Error Formula (RMSE)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

$$\text{R-Squared } (R^2) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3}$$

$$\text{Mean Absolute Error (MAE)} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4}$$

$$\text{Mean Bias Relative Error (MBRE)}\frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{\min(y_i, \hat{y}_i)} \tag{5}$$

$$\text{Mean Impact Bias Relative Error (MIBRE)} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{\max(y_i, \hat{y}_i)} \tag{6}$$

$$\text{Logarithmic Standard Deviation (LSD)} = \sqrt{\frac{\sum_{i=1}^{N}(\lambda_i + s^2/2)^2}{N-1}} \tag{7}$$

$$\text{Standardized Accuracy (SA)} = 1 - \frac{MAE_{pi}}{MAE_{p0}} \quad where\ MAE_{p0} = \frac{\sum_{i \neq j}|y_i - y_j|}{N(N-1)} \tag{8}$$

$$\text{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \tag{9}$$

$$\text{MdMRE} = median\frac{|y_i - \hat{y}_i|}{y_i} \tag{10}$$

$$\text{PRED}(25) = \frac{1}{n}\sum_{i=1}^{n}1\left[\frac{|y_i - \hat{y}_i|}{y_i} \leq 0.25\right] \times 100\% \tag{11}$$

5. Data Analysis of the Size–Effort Relationship with ML Techniques

5.1. Descriptive Statistics of the Datasets Available

An overview of the descriptive statistics of these two datasets is presented in Table 16 in terms of the minimum, maximum, average, and median of the effort, data movements, and size in COSMIC CFP units.

Table 16. Descriptive statistics of the datasets—calculated from Appendix A, with effort std dev and effort coefficient of variation (CV) = std dev/avg.

Dataset Id	No. Projects	Effort (h)						Size (CFPs)			
		Min	Max	Avg.	Median	Std Dev	CV	Min	Max	Avg.	Median
Dataset 1	50	3.2	17.2	8.946	8.5	3.484	0.389	14	43	28.66	30.0
Dataset 2	100	1.2	3.6	2.458	2.5	0.529	0.215	5	20	12.55	13.0

5.2. Evaluation of ML Models in Software Effort Estimation

This section presents the evaluation of the ML models on Datasets 1 and 2 in terms of the following criteria:

- Prediction accuracy and generalization: R^2, RMSE, and MAE;
- MBRE, MIBRE, LSD, SA, and MdMRE.

5.2.1. Generalization and Prediction Accuracy Comparison

Table 17 presents the empirical results for the experiment of Cross-Validated (CV) RMSE, R^2, and MAE on Datasets 1 and 2.

Table 17. Cross-validated (CV) results of dataset 1 (*N* = 50) and dataset 2 (N = 100) for RMSE, R^2, and MAE.

Model	Dataset	RMSE (95% CI)	R^2 (95% CI)	MAE (95% CI)
Linear Regression	Dataset 1 (N = 50)	1.860 (1.818, 1.901)	0.709 (0.696, 0.722)	1.496 (1.473, 1.519)
Decision Tree	Dataset 1 (N = 50)	2.085 (1.823, 2.347)	0.632 (0.540, 0.723)	1.629 (1.434, 1.825)
Support Vector Regressor	Dataset 1 (N = 50)	2.257 (2.132, 2.383)	0.571 (0.523, 0.618)	1.748 (1.677, 1.820)
AdaBoost Regressor	Dataset 1 (N = 50)	2.247 (2.072, 2.422)	0.574 (0.508, 0.641)	1.742 (1.605, 1.880)
Bagging Regressor	Dataset 1 (N = 50)	2.275 (2.182, 2.368)	0.565 (0.529, 0.600)	1.785 (1.714, 1.856)
Gradient Boosting Regressor	Dataset 1 (N = 50)	2.538 (2.455, 2.621)	0.458 (0.423, 0.494)	1.909 (1.838, 1.979)
Random Forest Regressor	Dataset 1 (N = 50)	2.259 (2.182, 2.337)	0.571 (0.541, 0.600)	1.772 (1.697, 1.847)
Extra Trees Regressor	Dataset 1 (N = 50)	2.491 (2.395, 2.586)	0.478 (0.438, 0.518)	1.911 (1.835, 1.987)
K-Nearest Neighbors	Dataset 1 (N = 50)	2.066 (2.017, 2.114)	0.641 (0.624, 0.658)	1.604 (1.560, 1.649)
Voting Regressor	Dataset 1 (N = 50)	1.941 (1.830, 2.051)	0.683 (0.647, 0.719)	1.546 (1.464, 1.628)
Stacking Regressor	Dataset 1 (N = 50)	1.911 (1.876, 1.947)	0.693 (0.681, 0.704)	1.524 (1.471, 1.576)
Linear Regression	Dataset 2 (N = 100)	0.125 (0.125, 0.126)	0.943 (0.943, 0.944)	0.108 (0.107, 0.109)
Decision Tree	Dataset 2 (N = 100)	0.159 (0.151, 0.167)	0.909 (0.899, 0.918)	0.129 (0.124, 0.135)
Support Vector Regressor	Dataset 2 (N = 100)	0.167 (0.156, 0.178)	0.899 (0.886, 0.913)	0.135 (0.125, 0.145)
AdaBoost Regressor	Dataset 2 (N = 100)	0.136 (0.135, 0.138)	0.933 (0.931, 0.934)	0.113 (0.112, 0.115)
Bagging Regressor	Dataset 2 (N = 100)	0.147 (0.144, 0.151)	0.922 (0.918, 0.926)	0.125 (0.122, 0.128)
Gradient Boosting Regressor	Dataset 2 (N = 100)	0.145 (0.143, 0.148)	0.924 (0.921, 0.927)	0.121 (0.114, 0.127)
Random Forest Regressor	Dataset 2 (N = 100)	0.145 (0.141, 0.149)	0.924 (0.920, 0.928)	0.123 (0.120, 0.126)
Extra Trees Regressor	Dataset 2 (N = 100)	0.138 (0.136, 0.141)	0.931 (0.929, 0.933)	0.118 (0.115, 0.121)
K-Nearest Neighbors	Dataset 2 (N = 100)	0.171 (0.161, 0.180)	0.895 (0.883, 0.906)	0.141 (0.132, 0.150)
Voting Regressor	Dataset 2 (N = 100)	0.132 (0.128, 0.135)	0.937 (0.934, 0.940)	0.112 (0.109, 0.115)
Stacking Regressor	Dataset 2 (N = 100)	0.125 (0.121, 0.128)	0.944 (0.940, 0.947)	0.106 (0.103, 0.110)

The results of the empirical study in Table 17 provide guidance to BI and digital marketing analytics practitioners to better select ML models and datasets for predicting dashboard development efforts as follows:

As illustrated in Table 17, the results of the repeated 10-fold cross-validation (five seeds) related to Dataset 1 show limited predictive performance and generalization as follows:

- Linear Regression achieved the highest R^2 = 0.709 and the lowest RMSE = 1.860 h among all ML models, confirming a relatively strong linear relationship between functional size and effort;
- Stacking Regressor and Voting Regressor achieved an R^2 = 0.693 and 0.683, respectively. Indicating that ensemble models are the next strongest ML models, however, remains relatively weaker than Liner Regression;
- The Decision Tree Regressor and KNN achieved an R^2 = 0.632 and 0.641, respectively. Followed by moderate performing ML models, such as AdaBoost, SVR, Random Forest, and Bagging. All have an R^2 between 0.565 and 0.574, while the MAE falls between 1.74 and 1.79 h;
- Extra Trees and Gradient Boosting Regressors reported the weakest models, with an R^2 = 0.478 and 0.458, respectively. Both regressors reported the highest MAE and RMSE.

As illustrated in Table 17, the results of the repeated 10-fold cross-validation (five seeds) related to Dataset 2 showed better predictive performance and generalization as follows:

- Linear Regression and Stacking Regressor achieved the best predictive accuracy, with an R^2 = 0.943 and 0.944, respectively, and an RMSE around 0.125 h. This confirms a strong linear relationship between functional size and effort;
- Extra Trees, AdaBoost, and Voting Regressors presented similar predictive accuracies, with R^2 values between 0.931 and 0.937 and RMSEs between 0.132 and 0.138 h;
- The ensemble models (Gradient Bosting, Bagging Regressor, and Random Forest) presented a strong predictive accuracy with R^2 values between 0.922 and 0.924 and MAEs less than 0.125 h. This confirms that ensemble models preserve a strong generalization performance;
- KNN, Support Vector Regressor, and Decision Tree reported the weakest predictive accuracies, with R^2 values between 0.895 and 0.909. However, this is still stronger than the performance of all models in Dataset 1.

Among the 11 ML models, the results show the higher predictive performance on Dataset 2 compared to Dataset 1. On Dataset 2, R^2 remained high with a moderate change during the repeated 10-fold cross-validation and five random cross-validation seeds. This indicates that the models reported a stable pattern rather than a single 80/20 training/testing data split.

In addition, since Dataset 1 and Dataset 2 differ, for instance, in sample size, a controlled resampling check of Dataset 2 was conducted to test whether the higher performance of Dataset 2 is due to the larger sample size. For all 11 ML models, Dataset 2 was subsampled to exactly match the size of Dataset 1 (N = 50). Ten independent subsamples drawn without replacement, each evaluated under the same repeated 10-fold cross-validation. The results show that R^2 remained unchanged, for example:

- In Linear Regression: R^2 = 0.939 (Dataset 2, N = 50) vs. 0.943 (Dataset 2, N = 100);
- In Stacking Regressor: R^2 = 0.934 (Dataset 2, N = 50) vs. 0.944 (Dataset 2, N = 100);
- While other ML models remain above the corresponding Dataset 1 result (see Section 6.1 for the full comparison).

5.2.2. Evaluation Using Error Metrics (MBRE, MIBRE, MdMRE)

Table 18 presents empirical results for the MBRE, MIBRE, LSD, SA, and MdMRE experiments.

In Table 18, the results related to Dataset 1 show that most of the ML models reported high error rates and limited predictive accuracy as follows:

- MBRE ranged from 18.9% to 25.5%, while MIBRE ranged between 14.9% and 18.3%, showing substantial estimation bias;
- MdMRE ranged from 14.4% to 18.7% for approximately all the models. These high errors suggest that the models suffered from data irregularities and noise, with a higher effort variability observed in Dataset 1, as presented in Table 16 (coefficient of variation 0.389 vs. 0.215 for Dataset 2);
- Models such as Linear Regression, Stacking Regressor, and Voting Regressors reported narrow confidence intervals and consistent error rates during the five cross-validation seeds. However, tree-based models, such as Decision Tree, reported an SA between 0.370 and 0.505 and varied depending on the cross-validation seed. This indicates that these models are less stable across repeated seed in each fold.

The results for Dataset 2 show a reduction in the estimation errors across all models, as follows:

- MBRE dropped between 4.7% and 6.6% compared to Dataset 1. While MIBRE ranged between 4.4% and 5.9%;

- MdMRE also confirmed consistent improvements, while most models maintain values below 5.5%. These improvements indicate that the models trained on the Dataset 2 achieved much stronger predictive accuracy;
- The reduced errors are consistent with Dataset 2's lower effort variability (coefficient of variation 0.215 vs. 0.389 for Dataset 1), though this study did not independently isolate effort variability from other uncontrolled factors, such as measurement scope, feature distributions, and project population differences (see Section 6.2).

Table 18. Results of Dataset 1 (N = 50) and Dataset 2 (*N* = 100) for MBRE, MIBRE, MMRE, MdMRE, PRED(25), and SA.

Model	Dataset	MBRE (95% CI)	MIBRE (95% CI)	LSD (95% CI)	MMRE (95% CI)	MdMRE (95% CI)	PRED(25) (95% CI)	SA (95% CI)
Linear Regression	Dataset 1 (N = 50)	0.189 (0.186, 0.192)	0.149 (0.147, 0.151)	0.200 (0.197, 0.204)	0.175 (0.172, 0.178)	0.147 (0.138, 0.156)	82.4% (78.7%, 86.1%)	0.626 (0.620, 0.631)
Decision Tree	Dataset 1 (N = 50)	0.216 (0.194, 0.238)	0.162 (0.149, 0.176)	0.232 (0.212, 0.252)	0.195 (0.176, 0.214)	0.144 (0.117, 0.172)	68.0% (62.4%, 73.6%)	0.592 (0.543, 0.641)
Support Vector Regressor	Dataset 1 (N = 50)	0.225 (0.219, 0.232)	0.168 (0.164, 0.172)	0.237 (0.232, 0.243)	0.210 (0.205, 0.215)	0.158 (0.140, 0.177)	69.6% (67.5%, 71.7%)	0.563 (0.545, 0.581)
AdaBoost Regressor	Dataset 1 (N = 50)	0.225 (0.208, 0.242)	0.169 (0.159, 0.179)	0.238 (0.220, 0.255)	0.203 (0.191, 0.215)	0.164 (0.149, 0.178)	68.4% (64.3%, 72.5%)	0.564 (0.530, 0.598)
Bagging Regressor	Dataset 1 (N = 50)	0.235 (0.229, 0.241)	0.175 (0.171, 0.178)	0.246 (0.240, 0.253)	0.213 (0.209, 0.217)	0.180 (0.169, 0.191)	69.6% (68.5%, 70.7%)	0.553 (0.536, 0.571)
Gradient Boosting Regressor	Dataset 1 (N = 50)	0.255 (0.250, 0.261)	0.183 (0.179, 0.187)	0.269 (0.265, 0.273)	0.227 (0.222, 0.233)	0.180 (0.176, 0.185)	64.0% (61.0%, 67.0%)	0.523 (0.505, 0.540)
Random Forest Regressor	Dataset 1 (N = 50)	0.233 (0.225, 0.240)	0.173 (0.168, 0.178)	0.244 (0.238, 0.250)	0.211 (0.206, 0.217)	0.177 (0.166, 0.188)	70.0% (68.2%, 71.8%)	0.557 (0.538, 0.576)
Extra Trees Regressor	Dataset 1 (N = 50)	0.253 (0.246, 0.260)	0.183 (0.179, 0.187)	0.265 (0.259, 0.270)	0.229 (0.223, 0.236)	0.187 (0.185, 0.189)	63.6% (61.5%, 65.7%)	0.522 (0.503, 0.541)
K-Nearest Neighbors	Dataset 1 (N = 50)	0.206 (0.204, 0.209)	0.156 (0.153, 0.159)	0.224 (0.220, 0.227)	0.191 (0.189, 0.193)	0.151 (0.139, 0.164)	70.8% (69.4%, 72.2%)	0.599 (0.587, 0.610)
Voting Regressor	Dataset 1 (N = 50)	0.198 (0.190, 0.206)	0.153 (0.148, 0.159)	0.211 (0.204, 0.218)	0.183 (0.177, 0.190)	0.153 (0.142, 0.164)	77.6% (75.5%, 79.7%)	0.613 (0.593, 0.634)
Stacking Regressor	Dataset 1 (N = 50)	0.193 (0.185, 0.200)	0.151 (0.144, 0.157)	0.206 (0.201, 0.211)	0.177 (0.171, 0.184)	0.147 (0.137, 0.157)	77.2% (72.7%, 81.7%)	0.619 (0.606, 0.632)
Linear Regression	Dataset 2 (N = 100)	0.047 (0.047, 0.048)	0.044 (0.044, 0.045)	0.054 (0.053, 0.054)	0.046 (0.046, 0.046)	0.042 (0.041, 0.043)	100.0% (100.0%, 100.0%)	0.821 (0.819, 0.822)
Decision Tree	Dataset 2 (N = 100)	0.061 (0.057, 0.065)	0.055 (0.052, 0.058)	0.076 (0.071, 0.082)	0.058 (0.055, 0.061)	0.047 (0.042, 0.051)	99.0% (98.1%, 99.9%)	0.786 (0.776, 0.795)
Support Vector Regressor	Dataset 2 (N = 100)	0.062 (0.058, 0.067)	0.056 (0.053, 0.060)	0.078 (0.072, 0.083)	0.060 (0.056, 0.065)	0.048 (0.044, 0.052)	98.8% (97.2%, 100.0%)	0.775 (0.759, 0.792)
AdaBoost Regressor	Dataset 2 (N = 100)	0.051 (0.050, 0.053)	0.047 (0.046, 0.049)	0.062 (0.059, 0.065)	0.050 (0.048, 0.051)	0.044 (0.042, 0.046)	99.8% (99.2%, 100.0%)	0.812 (0.809, 0.814)
Bagging Regressor	Dataset 2 (N = 100)	0.056 (0.054, 0.059)	0.052 (0.050, 0.054)	0.066 (0.062, 0.069)	0.054 (0.052, 0.056)	0.045 (0.043, 0.048)	99.8% (99.2%, 100.0%)	0.792 (0.787, 0.798)
Gradient Boosting Regressor	Dataset 2 (N = 100)	0.053 (0.050, 0.056)	0.049 (0.047, 0.052)	0.063 (0.062, 0.064)	0.051 (0.048, 0.054)	0.045 (0.039, 0.050)	100.0% (100.0%, 100.0%)	0.800 (0.789, 0.810)
Random Forest Regressor	Dataset 2 (N = 100)	0.055 (0.053, 0.058)	0.051 (0.049, 0.053)	0.065 (0.061, 0.069)	0.053 (0.051, 0.055)	0.046 (0.043, 0.048)	99.8% (99.2%, 100.0%)	0.795 (0.790, 0.800)
Extra Trees Regressor	Dataset 2 (N = 100)	0.052 (0.051, 0.054)	0.049 (0.047, 0.050)	0.060 (0.058, 0.063)	0.051 (0.049, 0.052)	0.047 (0.046, 0.049)	100.0% (100.0%, 100.0%)	0.804 (0.800, 0.808)
K-Nearest Neighbors	Dataset 2 (N = 100)	0.066 (0.061, 0.070)	0.059 (0.055, 0.063)	0.082 (0.077, 0.086)	0.064 (0.060, 0.068)	0.055 (0.053, 0.056)	97.8% (97.2%, 98.4%)	0.766 (0.751, 0.781)
Voting Regressor	Dataset 2 (N = 100)	0.051 (0.050, 0.053)	0.048 (0.046, 0.049)	0.061 (0.059, 0.064)	0.050 (0.048, 0.051)	0.042 (0.040, 0.043)	100.0% (100.0%, 100.0%)	0.814 (0.809, 0.819)
Stacking Regressor	Dataset 2 (N = 100)	0.047 (0.045, 0.048)	0.044 (0.043, 0.045)	0.054 (0.053, 0.056)	0.045 (0.044, 0.047)	0.040 (0.036, 0.044)	100.0% (100.0%, 100.0%)	0.824 (0.818, 0.829)

5.2.3. Standardized Accuracy (SA) Comparison

The authors of [51] recommended that standardized accuracy (SA) is a primary criterion for evaluating estimation models on software effort. SA offers a more robust evaluation

than typical error criteria, such as the mean relative error (MRE) and mean magnitude of relative error (MMRE), which have been criticized for systematic bias and favor underestimation. Therefore, the following key comparisons presented in Table 19 highlight the performance of the ML models in predicting the Data Studio effort based on COSMIC as follows:

- Voting Regressor, Stacking Regressor, and Linear Regression reported the highest rank based on SA and a consistent highest performance among Datasets 1 and 2;
- Across every ML model, SA reported higher performance on Dataset 2. This is consistent with the previous findings established based on R^2 and RMSE and error metrics in Tables 17 and 18, and with the lower effort variability of Dataset 2 reported in Table 16;
- Extra Trees Regressor and Gradient Boosting Regressor were the weakest models on Dataset 1; K-Nearest Neighbors and Support Vector Regressor were weakest on Dataset 2.

Table 19. SA comparison across the 11 ML models with confidence intervals from the repeated CV.

Model (Ranked by Dataset 1 SA)	SA (95% CI) Dataset 1	R^2 Dataset 1	SA (95% CI) Dataset 2	R^2 Dataset 2
Linear Regression	0.626 (0.620, 0.631)	0.709	0.821 (0.819, 0.822)	0.943
Stacking Regressor	0.619 (0.606, 0.632)	0.693	0.824 (0.818, 0.829)	0.944
Voting Regressor	0.613 (0.593, 0.634)	0.683	0.814 (0.809, 0.819)	0.937
K-Nearest Neighbors	0.599 (0.587, 0.610)	0.641	0.766 (0.751, 0.781)	0.895
Decision Tree	0.592 (0.543, 0.641)	0.632	0.786 (0.776, 0.795)	0.909
AdaBoost Regressor	0.564 (0.530, 0.598)	0.574	0.812 (0.809, 0.814)	0.933
Support Vector Regressor	0.563 (0.545, 0.581)	0.571	0.775 (0.759, 0.792)	0.899
Random Forest Regressor	0.557 (0.538, 0.576)	0.571	0.795 (0.790, 0.800)	0.924
Bagging Regressor	0.553 (0.536, 0.571)	0.565	0.792 (0.787, 0.798)	0.922
Gradient Boosting Regressor	0.523 (0.505, 0.540)	0.458	0.800 (0.789, 0.810)	0.924
Extra Trees Regressor	0.522 (0.503, 0.541)	0.478	0.804 (0.800, 0.808)	0.931

6. Discussion

Based on the empirical results, this study offers some key insights and recommendations.

6.1. Analysis of the Empirical Cumulative Distribution Function (ECDF)

Figure 4 illustrates the empirical cumulative distribution function (ECDF), which provides a visual representation of the proportion of observations for both the testing accuracy and standardized accuracy (SA) across the two datasets. In Figure 4, the curve corresponding to Dataset 1 consistently lies behind that for Dataset 2. This positioning indicates that, at any given threshold of accuracy or SA, the proportion of models achieving that threshold was lower in Dataset 1 than in Dataset 2. The ECDF analysis clearly demonstrated that the ML models applied to Dataset 1 exhibited lower performance than those applied to Dataset 2. The models trained and evaluated on Dataset 2 achieved higher testing accuracy and SA, confirming their superior predictive capability and generalization performance compared to those assessed using Dataset 1.

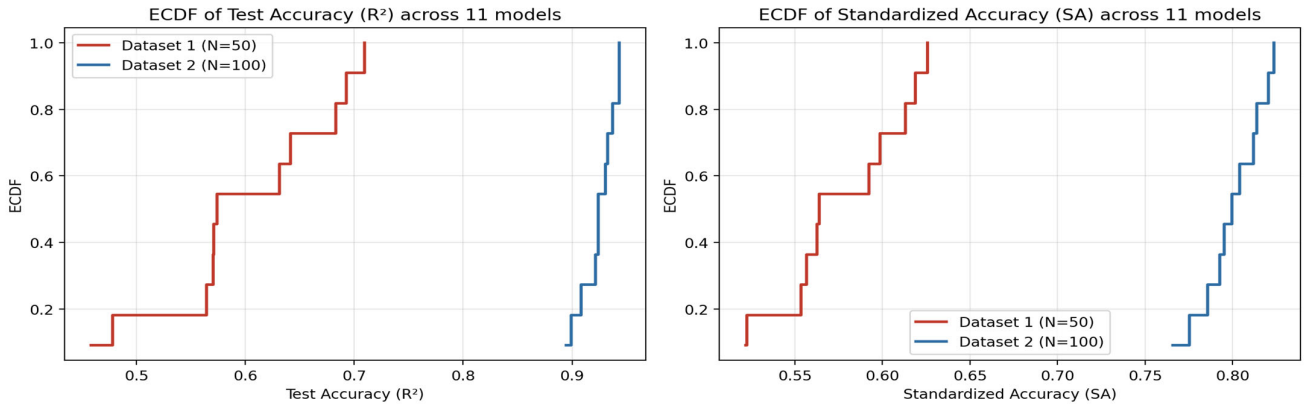


Figure 4. Empirical cumulative distribution function (ECDF) of test accuracy (R^2) and standardized accuracy (SA) for the two datasets, 11 ML models, repeated 10-fold cross-validation.

6.2. Resampling Check: Resampling the Size of Dataset 2 to the Size of Dataset 1

A resampling check was performed to investigate whether the higher performance in Dataset 2 was attributed to its larger sample size. Therefore, the resampling was performed on Dataset 2 including ten independent subsamples matching the size of Dataset 1 (50 projects), without replacement, across the 11 ML models, and in the same repeated 10-fold cross-validation. The results in Table 20 illustrate that accuracy continued to be high on the resampled subsets. For example, the Linear Regression reported an R^2 = 0.939, while the spread of R^2 values across the ten resamples ranged between 0.926 and 0.959. The Stacking Regressor reported an R^2 = 0.934, while the spread of R^2 values across the ten resamples ranged between 0.917 and 0.957. Compared to the performance at the full sample size (N = 100), these results are closely corresponding and matching. This indicates that the sample size may not be the only leading factor that explains the difference between Dataset 1 and Dataset 2.

Table 20. Results of evaluating the subsample with 11 ML models under the same repeated 10-fold CV protocol (5 seeds).

Model	Dataset 1 Full R^2 (N = 50)	Dataset 2 Full R^2 (N = 100)	Dataset 2 Subsampled to N = 50: Mean (Range, SD)
Stacking Regressor	0.693	0.944	0.934 (0.917–0.957, SD = 0.013)
Linear Regression	0.709	0.943	0.939 (0.926–0.959, SD = 0.012)
Voting Regressor	0.683	0.937	0.916 (0.894–0.943, SD = 0.017)
AdaBoost Regressor	0.574	0.933	0.892 (0.872–0.924, SD = 0.018)
Extra Trees Regressor	0.478	0.931	0.904 (0.872–0.934, SD = 0.023)
Random Forest Regressor	0.571	0.924	0.888 (0.859–0.923, SD = 0.022)
Gradient Boosting Regressor	0.458	0.924	0.882 (0.847–0.916, SD = 0.023)
Bagging Regressor	0.565	0.922	0.888 (0.858–0.923, SD = 0.022)
Decision Tree	0.632	0.909	0.852 (0.807–0.891, SD = 0.031)
Support Vector Regressor	0.571	0.899	0.852 (0.807–0.913, SD = 0.038)
K-Nearest Neighbors	0.641	0.895	0.837 (0.794–0.877, SD = 0.025)

Further consideration is based on the descriptive statistics of these two datasets presented in Table 16. The resampling result rules out sample size as an explanation but does not on its own establish what does explain the accuracy difference. Dataset 2 exhibited a substantially lower coefficient of variation in effort (0.215) than Dataset 1 (0.389). This lower effort variability is consistent with, but does not on its own establish, Dataset 2's higher predictive accuracy. Measurement scope, feature distributions, and underlying project population differences were not independently controlled in this study and remain plausible contributing factors that were not isolated from effort variability.

Disentangling these effects is left for future work using datasets that vary along only one of these dimensions at a time.

Therefore, based on the analysis of the empirical cumulative distribution function (ECDF) and the resampling check, we propose the following recommendation:

- When collecting or selecting data for training effort-estimation models, in BI dashboard software development using COSMIC, prioritize projects with consistent, low-variability effort.

6.3. Analysis of ML Models Performance

Figure 5 illustrates the performance of all ML models, demonstrating the high- and low-performing ML models using Dataset 2, including the cross-validated SA, accuracy, RMSE, and MAE in Tables 17 and 18. The results show the high performance of the Stacking Regressor and Linear Regression (high SA, accuracy and RMSE and MAE) and the low performance of Support Vector Regressor and K-Nearest Neighbors models (low SA and accuracy and high RMSE and MAE).

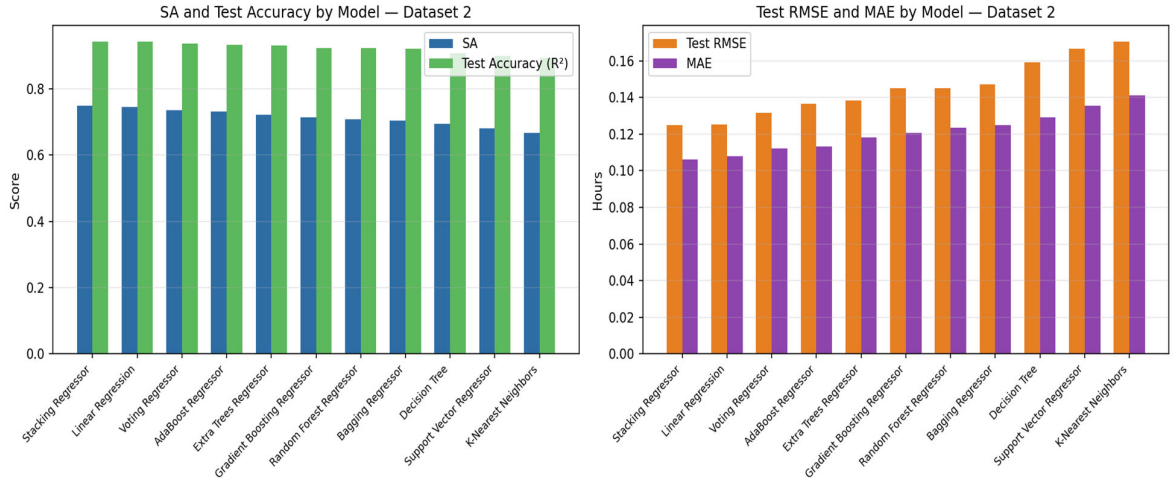


Figure 5. SA, Accuracy, RMSE, and MAE by ML Model using dataset 2, ranked by SA (repeated 10-fold CV).

Figures 6 and 7 display the actual (real) effort shown in green alongside the predictions shown in blue and orange. Stacking Regressor and Linear Regression models showed high predictive performance in predicting effort values, and the Support Vector Regressor and K-Nearest Neighbors models showed lower than other models in predicting effort values. Therefore, we propose the following recommendations.

- It is recommended to use Stacking Regressor and Linear Regression models for similar datasets because these models recorded the highest standardized accuracy for predicting BI development efforts with high generalization performance;
- It is recommended to avoid using the Support Vector Regressor and K-Nearest Neighbors models for similar datasets because they performed the worst.

As an additional robustness check on the predictor set itself, the cross-validation pipeline was re-run with CFP excluded, since CFP = Entry + Read + Write + Exit. The recommended models were unaffected or only marginally affected by this change: Linear Regression's R^2 was unchanged on both datasets (0.709 and 0.943), and the Stacking Regressor's accuracy remained essentially the same. Decision Tree, by contrast, showed a substantial decline on Dataset 2 (R^2 falling from 0.909 to 0.704) without CFP, since shallow decision trees do not reliably reconstruct an informative linear combination of features through a limited number of single-feature splits. This further supports the

recommendation above: the models recommended for similar datasets (Stacking Regressor and Linear Regression) are robust to this redundancy, whereas models not recommended here would be more sensitive to predictor-set choices of this kind.

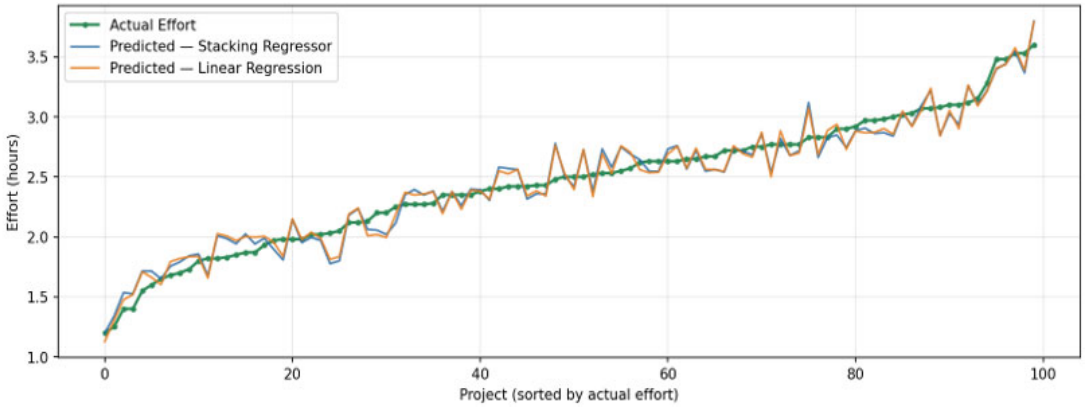


Figure 6. Predicted vs. actual values using dataset 2—Stacking Regressor and Linear Regression (top-performing models).

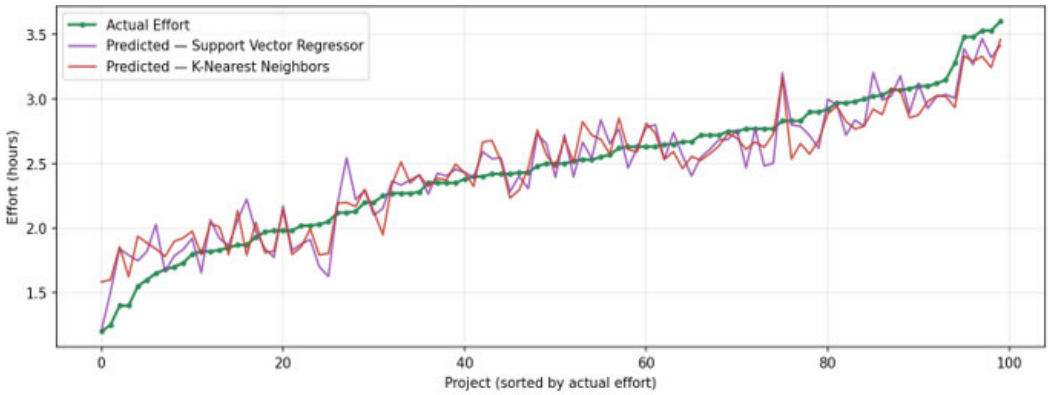


Figure 7. Predicted vs. actual values using dataset 2—Support Vector Regressor and K-Nearest Neighbors (weakest-performing models).

6.4. A Theoretical Foundation Between COSMIC and Vercellis' Methodology for Implementing BI Platforms

Although COSMIC has been widely applied in a large variety of software projects, its application in BI dashboard development, such as in Data Studio, has not been previously explored. This study contributes to researchers and practitioners by offering interdisciplinary research that theoretically maps software measurements and BI dashboard development.

The proposed BI-oriented COSMIC measurement framework in this study suggests that the application of COSMIC to BI dashboard development is feasible and effective. An empirical experiment with 150 BI development projects illustrated that practitioners who used Data Studio (or other BI platforms) could size them with COSMIC-ISO/IEC 19761 and then use such software functional size to build effort estimation models. Therefore, with decisions informed by data rather than intuition or guesswork, the proposed BI-oriented COSMIC measurement framework supports stakeholders in making informed decisions regarding the allocation of suitable resources to BI dashboard development projects.

We posited that the proposal of this BI-oriented COSMIC measurement framework is beneficial to BI practitioners in the following ways:

1. It offers the first theoretical COSMIC application for BI dashboard development. This is a novel theoretical basis that presents the BI dashboard as a measurable software component.
2. It has the potential to be published as a guideline in COSMIC-ISO/IEC 19761 for BI dashboard development. This will extend the BI-oriented COSMIC measurement framework as an applicable reference for other BI dashboard projects in the future.
3. It offers a theoretical alignment between COSMIC and Vercellis' methodology [16], as presented in Figure 8 and Table 21, for implementing BI platforms and explains how COSMIC can be used to assist in the project management of BI phases.

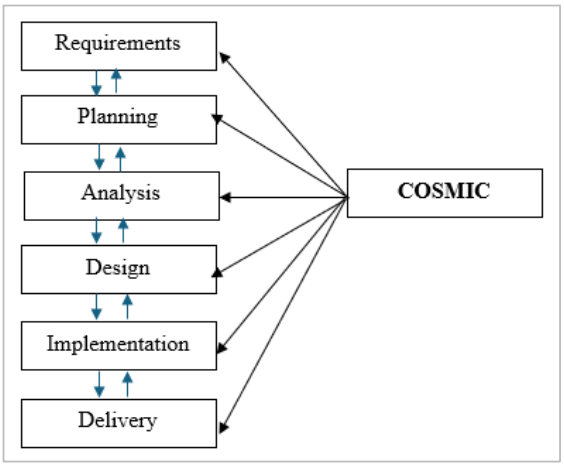


Figure 8. Incorporating COSMIC with [16] methodology for implementing the BI platform.

Table 21. Vercellis' BI phases and corresponding COSMIC tasks.

BI Phase	Description	COSMIC Tasks
Requirements	What the dashboard must achieve.	Used when functional user requirements are gathered.
Planning	Timeline, scope, cost, technologies, and comparing with historical data.	Used for early estimation, benchmarking and comparing with previous COSMIC-sized projects.
Analysis	Detailed specifications of functional processes, and data sources.	Used when functional processes, and data movements (E, X, R, W) are defined.
Designing	Build the architecture of the system and deliver a prototype or mock-up to validate before development.	Used when a prototype or mock-up of the BI dashboard is available prior to development. This can confirm the functional size.
Implementation	Practitioners start building the dashboard based on previous phases.	Used for project monitoring that development is progressing according to plan (e.g., with respect to functional size and effort estimates).
Delivery	The dashboard is produced and documented.	Used for benchmarking and productivity analysis along with estimation models.

6.5. Practical Implications of Applying COSMIC to BI Dashboard and How the Proposed Machine Learning Techniques Can Support Practitioners in Real-World Applications

In addition to the proposed theoretical contribution of COSMIC and business intelligence (BI), this section provides a summary of the related practical implications as follows:

- Software development effort estimation: Since COSMIC can be applied to software development before development starts, BI and digital marketing analytics practitioners can build estimation models (e.g., using Stacking Regressor or Linear Regression model) at the requirements and design phases supporting early resource planning

and budget approval. This will reduce the time needed for manual estimation, which supports faster project planning and decision-making;

- A benchmark across other BI and digital marketing analytics projects: practitioners can use the functional sizes to compare productivity, time, and budget across teams and projects;
- ML model selection to build estimation models: This study offers a guideline for BI and digital marketing analytics practitioners building their own estimates. The high performance of Stacking Regressor, Linear Regression, and Voting Regressor provides an initial direction on recommended ML models;
- Data collection and planning: While dataset size is important, this study stresses the importance of data quality. The results indicate that organizations should focus on reliable datasets to build reliable estimation ML models. In addition, this study offers two datasets at the proposal of researchers and practitioners for future work directions.

7. Conclusions

Business intelligence (BI) dashboards are important tools for visualizing organizational performance by presenting key performance indicators (KPIs) through interactive data visualization. Because dashboard development goes through phases, such as requirements, planning, analysis, design, implementation, and delivery, BI dashboard development requires effort and cost.

To ensure that software functionality is delivered on time and within budget, previous studies on estimating the effort of software development posit that estimating software development plays a crucial role in effective software project management. COSMIC functional size measurement is considered the standard method. It provides a standardized software functional size that is independent of technology and programming languages. To the best of our knowledge, no theoretical foundation that maps COSMIC to BI dashboard development has been established. Consequently, BI and digital marketing analytics practitioners are unable to gather reliable measures, such as COSMIC, to estimate future development projects. Therefore, this study offers a theoretical foundation that maps COSMIC concepts and rules with BI to measure the functional size of BI dashboard development and explains it with examples.

Next, the COSMIC FSM method was applied to two datasets of Data Studio projects. The resulting functional sizes were used as inputs to the ML models to predict the development effort in hours. The outcomes of these ML models showed that projects with consistent, low-variability effort should be prioritized when collecting or selecting data for training effort-estimation models, and recommended the use of Stacking Regressor, Linear Regression, and Voting Regressor models for similar datasets because these models recorded the highest standardized accuracy for predicting BI development efforts with high generalization performance.

In addition, this study offers a theoretical foundation between the COSMIC and BI dashboard concepts, which provides novel theoretical mapping that presents BI dashboard setups as measurable software components. It also presented how BI and digital marketing analytics practitioners can incorporate COSMIC with Vercelli's methodology for implementing dashboards and how to use the functional size measurement of various BI technologies, such as Tableau, Power BI, and Data Studio, for benchmarking, estimation, and planning.

To our knowledge, this study is the first on BI and digital marketing analytics dashboard development and COSMIC, and the results of the empirical study demonstrate the feasibility of applying functional size for Data Studio and for using functional size in predicting BI dashboard development efforts.

Future work will aim to extend this BI-oriented COSMIC measurement framework to address the following limitations:

- This empirical study focused on Data Studio projects. In future work, we aim to address other BI platforms, such as Power BI and Tableau;
- The proposed COSMIC measurement framework for BI dashboards should be investigated through large-scale empirical studies involving diverse BI projects across different industries and organizational contexts to assess its generalizability and measurement consistency;
- Future studies could investigate the relationship between COSMIC functional size and actual development effort in BI projects using industry datasets, which would provide practitioners with quantitative baselines for project planning and control;
- Future work should isolate the individual contributions of measurement scope, feature distributions, and project population characteristics to the accuracy differences observed among datasets in this study, using controlled datasets that vary along only one of these dimensions at a time, since the present study's resampling analysis could rule out sample size as an explanation but could not independently isolate these remaining factors from effort variability;
- Future work could also examine low-code and workflow-based analytics platforms, where the functional boundary between data transformation pipelines and reporting output challenges conventional COSMIC measurement assumptions, and new mapping rules may be needed to account for their node-based modular development paradigm;
- Future work should also examine dataset heterogeneity using additional project-level covariates such as practitioner experience, data-source count, transformation complexity, dashboard reuse, number of visuals, security requirements, refresh logic, testing effort, and project type, applying stratified, grouped, or hierarchical analyses where such data becomes available;
- This study has several limitations related to data provenance. Formal inter-rater measurement reliability was not assessed, as this was not part of the original measurement design and the original measurers are not available for re-measurement. Practitioner and organizational clustering of projects and detailed effort-recording procedures are not reported beyond what is available in Appendix A;
- Future work could explore alternative feature-engineering approaches that preserve CFP as the standard COSMIC size measure while yielding uniquely identifiable, interpretable estimates of each data movement type's contribution to effort.

Supplementary Materials: Executable analysis code reproducing all empirical tables and figures in this manuscript is available at: https://github.com/muntherqasaimeh/EmpricalWork.git (accessed on 8 September 2026).

Author Contributions: Conceptualization, A.A. (Ammar Abdallah), A.A. (Alain Abran) and D.K.M.; Methodology, A.A. (Ammar Abdallah) and A.A. (Alain Abran); Software, A.A. (Ammar Abdallah) and M.Q.; Validation, A.A. (Alain Abran); Formal analysis, A.A. (Ammar Abdallah), A.A. (Alain Abran), M.Q. and D.K.M.; Resources, A.A. (Ammar Abdallah) and M.Q.; Data curation, M.Q.; Writing—original draft, A.A. (Ammar Abdallah); Writing—review & editing, A.A. (Ammar Abdallah), A.A. (Alain Abran) and D.K.M.; Project administration, A.A. (Ammar Abdallah). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data extracted and analyzed are contained within the article. Additional information may be obtained from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Table A1. Dataset 1 of the 50 Data Studio projects—Experiment 1.

Project	ENTRY Movements	READ Movements	WRITE Movements	EXIT Movements	COSMIC Functional Size (CFPs)	Actual Effort (Hours)
P1	14	13	8	8	43	14.4
P2	12	9	7	7	35	8.6
P3	12	10	7	7	36	11.4
P4	4	4	3	3	14	3.8
P5	8	7	4	4	23	6.3
P6	9	8	5	5	27	8.3
P7	12	9	6	6	33	11.3
P8	6	5	3	3	17	6.7
P9	7	4	4	4	19	6.2
P10	9	8	5	5	27	9.2
P11	15	12	8	8	43	11.9
P12	14	12	8	8	42	15.2
P13	12	10	6	6	34	11.3
P14	11	9	6	6	32	11.2
P15	13	10	7	7	37	14.8
P16	7	5	4	4	20	7.4
P17	6	6	4	4	20	6.4
P18	10	8	6	6	30	10.1
P19	6	4	3	3	16	5.7
P20	8	6	5	5	24	4.8
P21	14	11	8	8	41	10.6
P22	14	12	8	8	42	16.1
P23	13	10	7	7	37	7.8
P24	13	9	7	7	36	12
P25	12	10	7	7	36	12.6
P26	15	11	8	8	42	9.4
P27	7	7	4	4	22	6.8
P28	13	11	7	7	38	14.1
P29	6	4	3	3	16	5.3
P30	7	7	4	4	22	6.5
P31	11	8	6	6	31	8.4
P32	10	8	6	6	30	10.4
P33	6	4	3	3	16	4.9
P34	6	5	4	4	19	4.7
P35	6	3	3	3	15	3.2
P36	11	11	7	7	36	8.7
P37	9	9	6	6	30	11.6
P38	10	8	6	6	30	7.8
P39	6	4	3	3	16	3.4
P40	9	7	5	5	26	10
P41	12	10	7	7	36	7.7
P42	7	6	4	4	21	6.6
P43	5	4	3	3	15	5
P44	13	11	7	7	38	10.6
P45	15	12	8	8	43	17.2
P46	6	4	3	3	16	4.8
P47	8	6	4	4	22	5.7
P48	11	8	6	6	31	10.3
P49	12	10	7	7	36	12.5
P50	8	6	4	4	22	7.6

Table A2. Dataset 2 of the 100 Data Studio projects—Experiment 2.

Project	ENTRY Movements	READ Movements	WRITE Movements	EXIT Movements	COSMIC Functional Size (CFPs)	Actual Effort (Hours)
P1	5	5	5	2	17	3.28
P2	1	5	3	4	13	2.5
P3	2	5	5	5	17	3.12
P4	5	3	2	2	12	2.42
P5	5	4	2	5	16	3.1
P6	4	3	1	5	13	2.53
P7	1	2	4	1	8	1.55
P8	2	1	1	1	5	1.2
P9	5	4	3	2	14	2.83
P10	2	1	1	3	7	1.4
P11	3	5	1	1	10	1.85
P12	4	3	1	4	12	2.27
P13	4	1	4	3	12	2.5
P14	3	5	2	5	15	2.75
P15	3	3	4	2	12	2.27
P16	3	5	1	5	14	2.75
P17	5	5	4	5	19	3.53
P18	4	2	3	5	14	2.65
P19	2	4	5	4	15	3.1
P20	1	3	3	4	11	2.35
P21	5	3	1	1	10	1.87
P22	2	4	1	2	9	1.68
P23	2	5	1	2	10	1.98
P24	5	4	3	3	15	2.92
P25	5	1	1	5	12	2.35
P26	4	5	1	1	11	1.98
P27	1	5	1	1	8	1.65
P28	3	4	5	1	13	2.4
P29	5	5	3	5	18	3.53
P30	4	4	1	5	14	2.72
P31	5	2	2	1	10	2.2
P32	4	1	4	5	14	2.48
P33	2	4	2	5	13	2.42
P34	4	4	5	2	15	2.77
P35	4	2	3	1	10	2.2
P36	4	4	5	2	15	2.83
P37	2	4	5	5	16	3.15
P38	3	1	1	3	8	1.82
P39	2	2	5	2	11	2.12
P40	5	4	5	4	18	3.48
P41	1	3	1	2	7	1.4
P42	5	2	3	3	13	2.42
P43	4	5	3	3	15	3.08
P44	3	2	4	5	14	2.63
P45	1	3	5	1	10	2.02
P46	5	4	1	5	15	3
P47	1	3	4	5	13	2.62
P48	5	1	3	4	13	2.65
P49	4	2	1	2	9	1.73
P50	3	2	2	2	9	1.8
P51	4	2	2	4	12	2.28
P52	4	4	4	1	13	2.63
P53	2	1	2	1	6	1.25
P54	1	5	3	1	10	1.87
P55	3	3	5	2	13	2.67
P56	3	4	4	1	12	2.43
P57	1	5	2	5	13	2.77
P58	5	2	3	2	12	2.43
P59	5	4	1	4	14	2.63

Table A2. *Cont.*

Project	ENTRY Movements	READ Movements	WRITE Movements	EXIT Movements	COSMIC Functional Size (CFPs)	Actual Effort (Hours)
P60	4	4	5	5	18	3.48
P61	2	4	2	1	9	1.7
P62	5	4	2	4	15	2.97
P63	4	1	3	4	12	2.35
P64	5	5	5	5	20	3.6
P65	4	1	4	2	11	2.35
P66	4	3	3	2	12	2.27
P67	4	2	1	4	11	2.12
P68	3	5	4	4	16	3.02
P69	2	5	2	1	10	1.97
P70	5	4	2	1	12	2.4
P71	5	2	5	1	13	2.72
P72	3	2	4	5	14	2.9
P73	3	5	5	1	14	2.77
P74	2	2	4	1	9	1.98
P75	4	4	3	5	16	3.07
P76	2	5	4	3	14	2.77
P77	2	2	3	2	9	2.05
P78	4	3	4	4	15	2.98
P79	2	3	3	2	10	1.82
P80	5	5	5	2	17	3.07
P81	5	3	4	2	14	2.57
P82	4	2	1	3	10	1.93
P83	1	2	2	5	10	2.13
P84	5	4	5	1	15	2.97
P85	5	1	4	4	14	2.72
P86	5	1	4	5	15	2.9
P87	3	3	3	2	11	2.25
P88	1	4	3	1	9	2.03
P89	1	3	3	1	8	1.6
P90	3	3	2	2	10	1.83
P91	3	3	2	2	10	2.02
P92	3	5	3	3	14	2.53
P93	2	3	3	5	13	2.63
P94	4	4	3	1	12	2.52
P95	4	2	3	4	13	2.67
P96	3	4	5	2	14	2.5
P97	2	3	5	5	15	3.03
P98	4	2	3	3	12	2.38
P99	4	2	5	3	14	2.55
P100	3	4	4	5	16	2.83

Table A3. Model configurations and hyperparameters used in the repeated cross-validation experiment—all models were implemented using scikit-learn version 1.8.0.

Model	Library/Class	Hyperparameters	Feature Scaling
Linear Regression	sklearn.linear_model.LinearRegression	Ordinary least squares; no regularization (default fit_intercept = True)	No
Decision Tree	sklearn.tree.DecisionTreeRegressor	max_depth = 4, min_samples_leaf = 3, random_state = seed	No
Support Vector Regressor	sklearn.svm.SVR	kernel = ‘rbf’, C = 10, epsilon = 0.1	Yes (StandardScaler)
K-Nearest Neighbors	sklearn.neighbors.KNeighborsRegressor	n_neighbors = 5	Yes (StandardScaler)
AdaBoost Regressor	sklearn.ensemble.AdaBoostRegressor	n_estimators = 100, random_state = seed (base estimator: DecisionTreeRegressor(max_depth = 3), scikit-learn’s resolved default)	No

Table A3. *Cont.*

Model	Library/Class	Hyperparameters	Feature Scaling
Bagging Regressor	sklearn.ensemble.BaggingRegressor	n_estimators = 100, random_state = seed (base estimator: DecisionTreeRegressor() with unrestricted depth, scikit-learn's resolved default)	No
Gradient Boosting Regressor	sklearn.ensemble.GradientBoostingRegressor	n_estimators = 100, max_depth = 3, random_state = seed	No
Random Forest Regressor	sklearn.ensemble.RandomForestRegressor	n_estimators = 200, max_depth = 6, random_state = seed	No
Extra Trees Regressor	sklearn.ensemble.ExtraTreesRegressor	n_estimators = 200, max_depth = 6, random_state = seed	No
Voting Regressor	sklearn.ensemble.VotingRegressor	Unweighted average of: Linear Regression (defaults) + Decision Tree (max_depth = 4, min_samples_leaf = 3) + K-Nearest Neighbors (n_neighbors = 5, scaled)	Yes, within KNN base learner only
Stacking Regressor	sklearn.ensemble.StackingRegressor	Base learners: Linear Regression (defaults) + Decision Tree (max_depth = 4, min_samples_leaf = 3) + K-Nearest Neighbors (n_neighbors = 5, scaled); final_estimator = LinearRegression(); internal cv = 5	Yes, within KNN base learner only

References

1. Widjaja, S.; Mauritsius, T. The Development of Performance Dashboard Visualization with Power BI as Platform. *Int. J. Mech. Eng. Technol.* **2019**, *10*, 235–249.
2. Oey, E.; Harno, S.S.S.; Zain, C. Developing Integrated Performance Dashboards with Power BI: A Case Study in a Medium-Size Manufacturer. In Proceedings of the 2021 International Conference on Information Management and Technology (ICIMTech 2021), Jakarta, Indonesia, 19–20 August 2021; pp. 265–270. [CrossRef]
3. Akbar, R.; Wahyudi; Rahmadoni, J.; Lubis, H.A. The Implementation of Business Intelligence on Visualisation of Transaction Data Analysis Using Dashboard System Case Study: XYZ Convenience Store. In Proceedings of the 2022 International Symposium on Information Technology and Digital Innovation (ISITDI 2022), Padang, Indonesia, 27–28 July 2022; pp. 42–49. [CrossRef]
4. Hmoud, H.; Al-Adwan, A.S.; Horani, O.; Yaseen, H.; Zoubi, J.Z. Al Factors Influencing Business Intelligence Adoption by Higher Education Institutions. *J. Open Innov. Technol. Mark. Complex.* **2023**, *9*, 100111. [CrossRef]
5. Kongthanasuwan, T.; Sriwiboon, N.; Horbanluekit, B.; Laesanklang, W.; Krityakierne, T. Market Analysis with Business Intelligence System for Marketing Planning. *Information* **2023**, *14*, 116. [CrossRef]
6. Lavalle, A.; Maté, A.; Trujillo, J.; Rizzi, S. Visualization Requirements for Business Intelligence Analytics: A Goal-Based, Iterative Framework. In Proceedings of the 2019 IEEE 27th International Requirements Engineering Conference (RE 2019), Jeju Island, Republic of Korea, 23–27 September 2019; pp. 109–119. [CrossRef]
7. Orlovskyi, D.; Kopp, A. A Business Intelligence Dashboard Design Approach to Improve Data Analytics and Decision Making. In Proceedings of the 7th International Conference Information Technology and Interactions (IT&I-2020), Kyiv, Ukraine, 2–3 December 2020; Volume 2833, pp. 48–59.
8. Gonçalves, C.T.; Gonçalves, M.J.A.; Campante, M.I. Developing Integrated Performance Dashboards Visualisations Using Power BI as a Platform. *Information* **2023**, *14*, 614. [CrossRef]
9. Abran, A. *Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers*; Wiley-IEEE Computer Society Press: Hoboken, NJ, USA, 2015.
10. Common Software Measurement International Consortium (COSMIC). Knowledge Base: COSMIC Software Sizing. Available online: https://cosmic-sizing.org/cosmic-publications/overview/ (accessed on 23 June 2026).
11. Valdes-Souto, F.; Perez-Gonzalez, H.G.; Perez-Delgado, C.A. Q-COSMIC: Quantum Software Metrics Based on COSMIC (ISO/IEC 19761). *arXiv* **2024**, arXiv:2402.08505. [CrossRef]
12. Samuel, A.; Pandey, A.K.; Sharma, V.K. Estimation of Functional Size of a Data Warehouse System Using COSMIC FSM Method. In Proceedings of the International Conference on Advances in Computer Science and Application, Bareilly, India, 21–22 June 2013; Volume 2, pp. 25–30.
13. van Heeringen, H. Measuring the Functional Size of a Data Warehouse Application Using COSMIC-FFP. In Proceedings of the Software Measurement European Forum (SMEF 2006), Rome, Italy, 10–12 May 2006; pp. 2–13.

14. Santillo, L. Size and Estimation of Data Warehouse Systems. In Proceedings of the 4th European Conference on Software Measurement and ICT Control (FESMA-DASMA), Heidelberg, Germany, 8–11 May 2001; pp. 173–184.
15. Common Software Measurement International Consortium (COSMIC). *Guideline for Sizing Data Warehouse Application Software*; COSMIC: Montréal, QC, Canada, 2018. Available online: https://cosmic-sizing.org/wp-content/uploads/2015/07/COSMIC-Method-v4.0.2-DWH-Guideline-v1.pdf (accessed on 23 June 2026).
16. Vercellis, C. *Business Intelligence: Data Mining and Optimization for Decision Making*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
17. Moulla, D.K.; Kitikil, A.A.; Mnkandla, E.; Soubra, H.; Abran, A. Functional Size Measurement for x86 Assembly Programs. In Proceedings of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-Mensura 2023), Rome, Italy, 14–15 September 2023; Volume 3455.
18. Darwish, A.; Soubra, H. COSMIC Functional Size of ARM Assembly Programs. In Proceedings of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-Mensura 2020), Mexico City, Mexico, 29–30 October 2020; Volume 2725, pp. 1–14.
19. Rajapaksha, P.; Wijayasiriwardhane, T.K. Functional Size Measurement of IoT Devices Using COSMIC Function Point: A Novel Set of Guidelines. In *Proceedings of the 2024 IEEE International Conference on Smart Computing and Systems Engineering (SCSE 2024)*; IEEE: Piscataway, NJ, USA, 2024; pp. 1–7. [CrossRef]
20. Soubra, H.; Abran, A. *Functional Size Measurement for the Internet of Things (IoT): An Example Using COSMIC and the Arduino Open-Source Platform*; Part F1319; Association for Computing Machinery: New York, NY, USA, 2017; pp. 122–128. [CrossRef]
21. Moulla, D.K.; Oumate; Mnkandla, E.; Soubra, H.; Abran, A. Automated COSMIC Function Points Measurement for C Program Using Regular Expressions. In Proceedings of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-Mensura 2022), Izmir, Turkey, 28–30 September 2022; Volume 3272.
22. Attallah, Y.; Soubra, H. Towards a COSMIC FSM Programming Language Compiler. In Proceedings of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-Mensura 2022), Izmir, Turkey, 28–30 September 2022; Volume 3272.
23. Khattab, K.; Elsayed, H.; Soubra, H. Functional Size Measurement Of Quantum Computers Software. In Proceedings of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-Mensura 2022), Izmir, Turkey, 28–30 September 2022; Volume 3272.
24. Mushtaq, Z.; Wahid, A. Cost Estimation for Mobile Application Development: Review. *IOSR J. Eng.* **2018**, *8*, 20–26.
25. Mushtaq, Z.; Wahid, A. Revised Approach for the Prediction of Functional Size of Mobile Application. *Appl. Comput. Inform.* **2024**, *20*, 181–193. [CrossRef]
26. Moumane, K.; Sardi, L.; Idri, A.; Abran, A. Functional Size Measurement of Postnatal Care Apps: Morocco Case Study. *Inform. Health Soc. Care* **2024**, *49*, 116–135. [CrossRef]
27. Kaur, A.; Kaur, K. A COSMIC Function Points Based Test Effort Estimation Model for Mobile Applications. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 946–963. [CrossRef]
28. Symons, C.; Abran, A.; Ebert, C.; Vogelezang, F. Measurement of Software Size: Advances Made by the COSMIC Community. In *Proceedings of the Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-Mensura)*; IEEE: Piscataway, NJ, USA, 2017; pp. 75–86. [CrossRef]
29. Sakhrawi, Z.; Sellami, A.; Bouassida, N. Support Vector Regression for Enhancement Effort Prediction of Scrum Projects from COSMIC Functional Size. *Innov. Syst. Softw. Eng.* **2022**, *18*, 137–153. [CrossRef]
30. Abrahão, S.; De Marco, L.; Ferrucci, F.; Gomez, J.; Gravino, C.; Sarro, F. Definition and Evaluation of a COSMIC Measurement Procedure for Sizing Web Applications in a Model-Driven Development Environment. *Inf. Softw. Technol.* **2018**, *104*, 144–161. [CrossRef]
31. Abdallah, A.; Abran, A.; Qasaimeh, M.; Qasaimeh, M.; Abdallah, B. Measurement of the Functional Size of Web Analytics Implementation: A COSMIC-Based Case Study Using Machine Learning. *Future Internet* **2025**, *17*, 280. [CrossRef]
32. Abdallah, A.; Abran, A. Towards the Adoption of International Standards in Enterprise Architecture Measurement. In Proceedings of the 8th International Conference on Data Science, Technology and Applications (DATA 2019), Prague, Czech Republic, 26–28 July 2019; pp. 1–8. [CrossRef]
33. Common Software Measurement International Consortium (COSMIC). Available online: https://cosmic-sizing.org/ (accessed on 23 June 2026).
34. Common Software Measurement International Consortium (COSMIC). *COSMIC Measurement Manual for ISO 19761: Part 1—Principles, Definitions and Rules, Version 5.0*; COSMIC: Montréal, QC, Canada, 2021. Available online: https://cosmic-sizing.org/publications/measurement-manual-v5-0-may-2020-part-1-principles-definitions-rules/ (accessed on 23 June 2026).
35. Google. Data Studio Documentation. Available online: https://docs.cloud.google.com/data-studio (accessed on 23 June 2026).
36. Microsoft. Power BI Documentation. Available online: https://learn.microsoft.com/en-us/power-bi/ (accessed on 23 June 2026).

37. Tableau Software, LLC. Tableau Knowledge Base. Available online: https://www.tableau.com/support/knowledgebase (accessed on 23 June 2026).
38. Kane, J.V.; Barabas, J. No Harm in Checking: Using Factual Manipulation Checks to Assess Attentiveness in Experiments. *Am. J. Political Sci.* **2019**, *63*, 234–249. [CrossRef]
39. Jørgensen, M. Regression Models of Software Development Effort Estimation Accuracy and Bias. *Empir. Softw. Eng.* **2004**, *9*, 297–314. [CrossRef]
40. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely Randomized Trees. *Mach. Learn.* **2006**, *63*, 3–42. [CrossRef]
41. Duffy, N.; Helmbold, D. Boosting Methods for Regression. *Mach. Learn.* **2002**, *47*, 153–200. [CrossRef]
42. Jin, Z.; Shang, J.; Zhu, Q.; Ling, C.; Xie, W.; Qiang, B. RFRSF: Employee Turnover Prediction Based on Random Forests and Survival Analysis. In *Web Information Systems Engineering–WISE 2020*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2020; Volume 12343, pp. 503–515. [CrossRef]
43. Podgorelec, V.; Zorman, M. Decision Tree Learning. In *Encyclopedia of Complexity and Systems Science*; Meyers, R.A., Ed.; Springer: New York, NY, USA, 2014; pp. 1–28. [CrossRef]
44. Kramer, O. K-Nearest Neighbors. In *Dimensionality Reduction with Unsupervised Nearest Neighbors*; Intelligent Systems Reference Library; Springer: Berlin/Heidelberg, Germany, 2013; Volume 51, pp. 13–23. [CrossRef]
45. Schapire, R.E. The Boosting Approach to Machine Learning: An Overview. In *Nonlinear Estimation and Classification*; Denison, D.D., Hansen, M.H., Holmes, C.C., Mallick, B., Yu, B., Eds.; Springer: New York, NY, USA, 2003; pp. 149–171. [CrossRef]
46. Richman, R.; Wüthrich, M.V. Nagging Predictors. *Risks* **2020**, *8*, 83. [CrossRef]
47. Liapis, C.M.; Karanikola, A.; Kotsiantis, S. A Multivariate Ensemble Learning Method for Medium-Term Energy Forecasting. *Neural Comput. Appl.* **2023**, *35*, 21479–21497. [CrossRef]
48. Wei, P.; Beer, M. Regression Models for Machine Learning. In *Machine Learning in Modeling and Simulation: Methods and Applications*; Springer: Cham, Switzerland, 2023; pp. 341–371. [CrossRef]
49. Wen, J.; Li, S.; Lin, Z.; Hu, Y.; Huang, C. Systematic Literature Review of Machine Learning-Based Software Development Effort Estimation Models. *Inf. Softw. Technol.* **2012**, *54*, 41–59. [CrossRef]
50. Mahmood, Y.; Kama, N.; Azmi, A.; Khan, A.S.; Ali, M. Software Effort Estimation Accuracy Prediction of Machine Learning Techniques: A Systematic Performance Evaluation. *Softw. Pract. Exp.* **2022**, *52*, 39–65. [CrossRef]
51. Cabral, J.T.H.; Oliveira, A.L.I.; da Silva, F.Q.B. Ensemble Effort Estimation: An Updated and Extended Systematic Literature Review. *J. Syst. Softw.* **2023**, *195*, 111542. [CrossRef]
52. Miyazaki, Y.; Takanou, A.; Nozaki, H.; Nakagawa, N.; Okada, K. Method to Estimate Parameter Values in Software Prediction Models. *Inf. Softw. Technol.* **1991**, *33*, 239–243. [CrossRef]
53. Foss, T.; Stensrud, E.; Kitchenham, B.; Myrtveit, I. A Simulation Study of the Model Evaluation Criterion MMRE. *IEEE Trans. Softw. Eng.* **2003**, *29*, 985–995. [CrossRef]
54. Myrtveit, I.; Stensrud, E.; Shepperd, M. Reliability and Validity in Comparative Studies of Software Prediction Models. *IEEE Trans. Softw. Eng.* **2005**, *31*, 380–391. [CrossRef]
55. Jørgensen, M.; Halkjelsvik, T.; Liestøl, K. When Should We (Not) Use the Mean Magnitude of Relative Error (MMRE) as an Error Measure in Software Development Effort Estimation? *Inf. Softw. Technol.* **2022**, *143*, 106784. [CrossRef]
56. Shepperd, M.; MacDonell, S. Evaluating Prediction Systems in Software Project Estimation. *Inf. Softw. Technol.* **2012**, *54*, 820–827. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.