IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 19 June 2024, accepted 10 July 2024, date of publication 16 July 2024, date of current version 29 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3429241

RESEARCH ARTICLE

Development of a Modularized Undergraduate Data Science and Big Data Curricular Using No-Code Software Development Tools

HARRY D. MAFUKIDZE[1], ACTION NECHIBVUTE[1], ABID YAHYA[2], (Senior Member, IEEE), IRFAN ANJUM BADRUDDIN[3], SARFARAZ KAMANGAR[3], AND MOHAMED HUSSIEN[4]

[1]Department of Applied Physics and Telecommunications, Midlands State University, Gweru, Zimbabwe
[2]Department of Electrical, Computer and Telecommunications Engineering, Botswana International University of Science and Technology, Palapye, Botswana
[3]Department of Mechanical Engineering, College of Engineering, King Khalid University, Abha 61421, Saudi Arabia
[4]Department of Chemistry, Faculty of Science, King Khalid University, Abha 61413, Saudi Arabia

Corresponding author: Harry D. Mafukidze (mafukidzehd@staff.msu.ac.zw)

The authors extend their appreciation to the Deanship of Research and Graduate Studies at King Khalid University for funding this work through Large Research Project under grant number RGP.2/127/45.

ABSTRACT Over the last decade, Data Science has emerged as one of the most important subjects that has had a major impact on industry. This is due to the continual development of scientific methods, algorithms, processes, and computational tools that help to extract knowledge from raw data efficiently and cost-effectively, compared with early-generation tools. Professional data scientists create code that processes, analyses and extracts actionable insights from high volumes of data. This process requires a deep understanding of mathematical principles, statistics, business knowledge, and computer science. But most importantly, the data science development chain requires knowledge of a high-level programming tool and its dependencies. This is a major problem in some aspects due to the steep learning curve. In this paper, we describe and present a modularized Data Science curriculum for undergraduate learners that relies on no-code software development tools as programming aids for non-computer science majors. No-code development tools have been added to the traditional teaching pedagogy to improve students' motivation and conceptual understanding of coding despite their limited programming skills. The study aims to assess the impacts of visual programming languages on the performance of non-computer science majors on programming problems. The study's sample consists of 50 fourth-year students from the Faculty of Science and Technology at the Midlands State University. A post-survey questionnaire and assessment items were administered to the control and experimental groups. Results show that the students drawn from the experimental group benefited from the use of a visual programming language. These results offer evidence-based recommendations for incorporating high-performance no-code software development tools in the formal curriculum to aid teaching and learning data science programming for students of diverse academic backgrounds.

INDEX TERMS Curriculum, data science, education, no-code tools, visual programming languages.

I. INTRODUCTION

The recent technological advances in computing, coupled with the increase in the demand to process high volumes of data have led to the development of computer algorithms that extract information and knowledge from different sources of data. Consequently, this has also pushed the demand for specialists, analysts, and engineers, who develop and maintain that code. An earlier report by McKinsey Global Institute (MGI) in 2011 estimates that hundreds and thousands of data-related jobs will be required in the next few years [1]. This means that there is an absolute need to train more data scientists to meet this ever-increasing demand. To democratize training in this field, especially in

The associate editor coordinating the review of this manuscript and approving it for publication was Martin Reisslein.

universities, a high-powered delegation comprising 25 undergraduate faculty members from a variety of institutions across the US met to develop a series of curriculum guidelines for an undergraduate data science course [2]. Drawn from the three major disciplines; mathematics, statistics and computer science, the guidelines stipulate that a graduating student majoring in data science must be proficient in subjects such as computational and statistical thinking, mathematics, model building, algorithms, data modelling and communication. Such guidelines define the skills that learners are supposed to have after completion of the course [3].

As the report by MGI states, data is now a key asset for companies, and data analytics can improve the company's key operations or help launch new business models to expand the markets. Considering this goal, there are two ways to achieve this: *(i)* engaging professional data scientists, or *(ii)* up-skilling existing professionals who are not data scientists with the necessary data-based skills required to meet the needs of industry. Producing data scientists is straightforward, students would graduate with a major specialization in data science and are then deployed in various industries. On the other hand, developing data literacy skills in graduate students who do not have the pre-requisite programming experience may be challenging due to the steep learning curve of text-based programming languages that have been traditionally used to teach or learn the practical aspect of data science. However, a different programming paradigm has been developed over the years. They rely on visual, drag-and-drop, no-code computer programming tools, where instructions are encapsulated in blocks, instead of text-based formal languages. Several blocks can then be connected sequentially to solve a programming problem. As we can expect, such no-code models offer several advantages, especially to new learners. Mainly, it enables the learner to focus on algorithm development, instead of struggling with the intricacies of the structure or style of the programming language.

This work is motivated by our experiences in teaching Data Science as a module in the Department of Applied Physics and Telecommunications under the Faculty of Science and Technology at Midlands State University in Zimbabwe. The Department offers a four-year Bachelor of Engineering Degree in Telecommunication Engineering, where Data Science is offered in the final year. In the module, students learn the fundamentals of data science as well as the applications of data science and big data in various domains. The Department also offers a four-year Bachelor of Science Degree In Industrial Physics and Instrumentation. Of late, data science has emerged as students' favourites, with the majority of them implementing data science concepts in their final year projects. However, we realized that although our students have a strong Mathematical background, the majority of them, especially those from BSc Industrial Physics and Instrumentation often struggle with developing practical computer code. As a result, this has affected the majority of students who intend to apply data science concepts in their final year projects. To assist our students, we came across VPLs, that have helped them develop data-related projects without actually worrying about the intricacies of coding.

The primary motive driving the present work is the need to reduce the data science learning curve for non-majors, especially on the practical side, that is mainly characterized by heavy programming. Currently, minimum effort has been made to formally merge data science education with this new visual programming tool, despite their promising advantages. As of now, the two fields have existed in parallel with each other, thereby failing to provide an adequate broad-based curriculum required to support professional development in data science. This paper presents a data curriculum initiative using No-Code tools (NCTs). The curriculum has been designed in such a way that the knowledge base, course structure and content are similar, in principle to the curriculum that is based on traditional programming languages. It should be noted, however, that the proposed curriculum has not added or removed content from the existing data science curriculum, rather, the present work proposes the integration of emerging and flexible programming environments in data science education initiatives. We define the appropriate teaching and evaluation methods that are suitable for this type of programming. The primary objectives of our work are to integrate data science education with NCTs and accelerate data science education using such tools to reduce the amount of time required to up-skill a non-data professional. We summarize the key contributions of our work as follows:

1) We describe key features and processes for visual programming approaches.
2) We demonstrate the feasibility of no-code paradigms as a potential aid for programming in data science education.
3) We evaluate the performance of visual programming environments, and demonstrate that they are comparable with text-based programming languages in terms of data science education.
4) We provide a survey on NCTs, as well as empirical evidence on the use of NCTs in education.
5) We demonstrate, through experiments, that NCTs are enablers of student success, especially non-computer science majors in data science programming.

This paper is organized as follows; Section II describes the data science curriculum initiative. The need for a supplementary data science programming tool and the current state of visual programming and data science education are discussed. Furthermore, empirical evidence on the use of VPLs in formalized educational environments is presented. In Section III, we discuss the data science topics that can be implemented using VPLs as well as develop assessment items for Python and visual programming languages. Chapter IV discusses the teaching methods suitable for visual programming languages and Chapter IV discusses the assessment

methods. Results of the experiments conducted are presented in Chapter VI. Finally, we conclude the paper in Chapter VII.

II. THE DATA SCIENCE CURRICULUM INITIATIVE

This Chapter addresses the major concerns raised in the literature about existing coding practices in data science education and discusses the goal of integrating visual programming languages in coding data science algorithms. Next, the section presents the basic architecture of VPLs, as well as key competencies students must have in Data Science. Lastly, the section concludes by offering empirical evidence on the use of NCTs/VPLs in Education.

A. THE NEED FOR A SUPPLEMENTARY DATA SCIENCE PROGRAMMING ENVIRONMENT

Data science is usually offered as an independent degree spanning a few years. The course features specializations in subjects such as data structures, algorithms, statistics, computer science, machine learning, and mathematics. There is a need, however, to present this course as a condensed module to students specializing in different fields such as Physics, Engineering, Telecommunications or any field of specialization to make data-driven decisions without having extensive computer programming skills. To support non-computer science majors in this field, special considerations must be made to teach these students to apply a range of data science technical skills in their areas of specialization. In short, this study contributes to the development of the objectives, subject content, teaching methods, and evaluation instruments, as well as effective data science learning outcomes for non-computer science audiences. Further, we explore activities and learning strategies that demonstrate the data science workflow using visual tools. This requires simplifying the domain problem, knowledge of data formats, and the appropriate analytical techniques. To achieve this, the following three major questions arise:

1) Do non-computer science learners and professionals find no-code development tools helpful?
2) Do visual programming tools provide the necessary tools to equip learners with the knowledge and skills to solve data science problems?
3) How should test items be structured to assess and evaluate data science education based on NCTs?

The goal of this project was to define a set of no-code data science curriculum structures and guidelines to be incorporated into current and future undergraduate modules to support effective skill acquisition and analytical thinking for non-computer science majors. The proposed work is intended to address key unknowns such as what to teach and how to teach data science using no-code development tools. We examined a set of work-flows on five key areas of data science summarized by Hastie et al. [4] as data acquisition and storage, data pre-processing and cleaning, exploratory data analysis (EDA), predictive modelling and machine learning, data visualization and communication

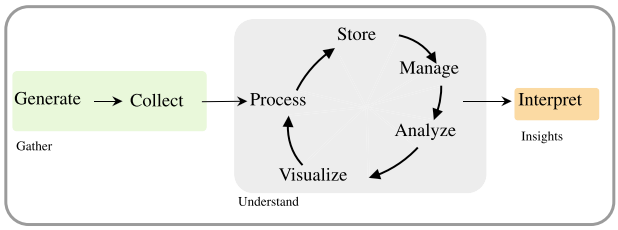

FIGURE 1. The data science work-flow [8], [9]. The data science process starts with the generation of data. This could be raw, real-world data or synthetic data that is artificially created using mathematical models or simulations to mimic real-world data. Next, according to Wing et al., not all data generated will be used [8], so the following process concerns systematically gathering of information from various sources. The collected data is then processed. This may include conducting preparatory steps on the raw data before analysis or modelling. Next, the data goes through several stages, such as data cleaning, storage, transformation, analysis, and visualization, all of which enable extraction of valuable insights from the data.

using no-code software platforms. Potential tools include (1) Orange developed by Demsar et al [5], (2) KNIME from KNIME AG company [6], and Neural Designer from Artelnics [7].

In this work, we present programming workflows implemented in Orange [5], as well as a scaffolded project that can be implemented using NCTs. This project highlights the data-science life cycle as described by Wing [8] and Zhang et al. [9], as shown in Figure 1, and it takes learners through a practical-based learning experience, to assist them in discovering patterns in the data, hence making valuable insights.

B. CURRENT STATE OF VISUAL PROGRAMMING AND DATA SCIENCE EDUCATION

1) VISUAL PROGRAMMING LANGUAGES

We intend to address the data science skills gap that exists, especially during the current era of the rapid growth of data-driven industries. We address this gap by outlining how an emerging model of no-code computer programming tools can support data science education, and help undergraduate learners extract information from data in their relevant field. Additionally, we hope that the uptake of such NCTs will help in up-skilling existing data science tutors and professionals alike in data-related roles in various industries. Determining the strengths, limitations and the future of using such NCTs will help us to achieve our objective. In this section, we survey the literature for visual programming languages since they are the same as NCTs, but different nomenclature.

A central feature of NCTs is that algorithms or functions are encapsulated into containers called *widgets*. These are the basic building blocks of the data analysis pipeline, and they can be connected to create a visual workflow, as shown in Figure 2. Although the appearance of a widget may vary depending on the platform, their structure typically consists of the following components, as shown in Table 1:

The classical definition of NCTs has been described by early pioneers in the context of Visual Programming.

TABLE 1. Components of a widget.

Component	Description
Title	Text or label that provides a name or describes the widget. This is usually placed at the top of the widget for identification.
Input Port	This receives input from data sources or other widgets for processing. The data transmitted through the input port must be in a form compatible with the widget.
Output Port	This transmits the data processed to other components in the workflow.
Settings	Settings of a widget enable the user to configure the parameters or options that control its functionality.
Visualization or Output Area	This typically refers to how the widget presents data to the user.

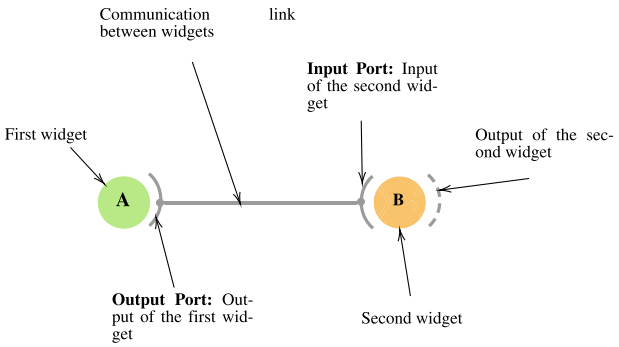

FIGURE 2. Basic layout and connection of widgets in NCTs. Two widgets are connected sequentially via a communication link that relays information between the two. In this workflow, widget *A* processes the data, and transmits it to the next widget connected to it through its output port. The input of widget *B* receives this data, processes it further, and transmits using its output port. Depending on the configuration, multiple widgets could be connected to one widget to reveal several instances of the data.

In 1986, Myers defined this as any system that could be programmed by the user in 2D or multiple dimensions [10]. This requires environments that use graphical techniques to aid the entire processes of programming and developing computer applications [10], [11]. Unlike conventional text-based languages, visual programming is motivated by the ideology that graphical techniques, particularly pictures, can convey more information concisely, compared with 1-dimensional linear programming languages [11]. In addition, pictures can break the language barrier, simplifying the process of programming for users regardless of the language they speak [11]. In summary, graphical tools provide two things; (i) a visual environment for programming, and (ii) a language interface for expressing visual information flow [12]. These are some of the predominant factors that have influenced the development of no-code programming languages.

To establish a common understanding, Kuhail et al. [13] combined Myers [10] and Burnett and Bakers [14] taxonomies to divide existing VPLs into four broad areas: form-based languages, diagram based, icon-based, and block-based. According to the authors, *form-based programming* allows end-user programmers to configure a graphical form, whereby parameters are added or changed using drop-down menus or windows. *Diagram-based languages* enable end-user developers to connect basic shapes such as rectangles, parallelograms, circles, etc by arrows, lines or connectors to represent programming constructs. On the other hand, *icon-based languages* rely on the use of icons to visualize the organization, design and flow of a program. Lastly, *block-based languages* enable end-user developers to drag and drop components of a program in the form of blocks. Several blocks can then be connected to define program flow. In general, using visual or graphical expressions as a way of writing computer programs greatly reduces syntax errors, and is easily comprehended by users of diverse backgrounds, since the human visual system and visual information processing is greatly optimized for multi-dimensional data [10].

Many VPLs have been developed in the literature. However, a large number of those do not possess the features of a true VPL. Although there is no consensus on what makes up a complete VPL [15], certain criteria must be met first to be considered a VPL. It is generally agreed that the language of a visual programming environment must be able to convey meaningful information for programming, rather than just cosmetic graphics [12], or rich graphical user interface features. To extend the criteria, Burnett and Baker develop a classification scheme for VPL research papers [14]. In their work, they highlight a set of important features of a VPL, which can then be used for comparison. A detailed criteria is presented by Kiper et al. [15]. They suggest that VPLs can be assessed based on visual nature, functionality, ease of comprehension, paradigm support, and scalability.

Although this field has received considerable interest in the past, work is ongoing to address inherent problems that have plagued early generations of visual programming tools. In the past, researchers had faced difficulty developing large programs or processing large datasets [12]. This problem has been solved by recent advances in computer graphics, abstraction and cloud computing. It is now possible to fit a lot of blocks, icons or diagrams on the same computer screen. Users are now able to navigate large programs through the use of multiple sheets. The wide uptake of cloud computing, and related services has played a crucial role in processing large datasets efficiently, and hence improves the functionality of visual programming tools. In other spheres, early researchers have cited a lack of functionality as another major drawback of early visual programming tools. Indeed, even some modern VPLs are hindered by this problem. Besides the lack of functionality, novice programmers may face limited or no room at all to develop and integrate customized program elements due to; proprietary software and the steep learning curve of the tool. Another aspect that characterized early generations of VPLs is *inefficiency*. This was a major challenge due to slow program execution [12], and large memory requirements. However, this is no longer a problem, nowadays, as most tools leverage web-based online environments to deliver programming tools with high

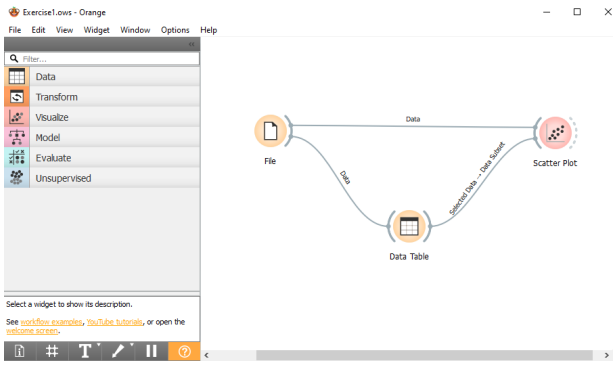

FIGURE 3. Screenshot of the Orange Development Environment showing widget repository and workspace. The workflow shown here presents a scatter plot of the dataset provided by the file widget.

computational performances. Most platforms are developed using high-level programming languages such as C/C++ which are widely known for managing memory capacity well.

a: PERFORMANCE OF VISUAL PROGRAMMING LANGUAGES

A key question that arises is the need to evaluate the performances of VPLs against set metrics. A deficiency in the literature, however, is an apparent lack of benchmarks for comparing VPLs against text-based programming languages. This is mainly due to the different architectures, and operating principles of the programming environments. As already discussed, VPLs can be assessed based on attributes such as (i) visual nature, (ii) functionality, (iii) ease of comprehension, (iv) paradigm support, and (v) scalability [15]. There is a need, however, to develop cross-platform evaluation metrics to assess how VPLs fair against well-established and well-supported high-level programming languages such as C/C++, Python, JAVA, MATLAB etc. On the other hand, it is possible to generate or export text-based code such as Python or C/C++ from VPLs and run on supported hardware or platforms just like any program written natively in that code. This also paves the way for students to transition from VPL-based to text-based programming.

These, and many more improvements, especially in graphical techniques and data management, have motivated the uptake of such tools in modern software development, and we believe that they will be the cornerstone of data science education for years to come.

2) ORANGE AS A VISUAL PROGRAMMING ENVIRONMENT FOR DATA SCIENCE EDUCATION

The developers describe the Orange software as fruitful and fun, offering a visual programming environment that facilitates the implementation of the entire data science programming chain with only a few steps. Aided by a vast library of functions encapsulated in graphical blocks called widgets, Orange allows users to drag and drop widgets from different categories such as Data, Transform, Visualize, Model, Evaluate, and Unsupervised, among others to build a data science workflow. Some of the widgets that can be used are shown in Table 2. Several VPLs exist, and to select the *"best"* VPL, Dobesova et *al.* conducts a comparative study to evaluate the performance of Orange software in teaching machine learning tasks in the Department of Geoinformatics at Palacký University Olomouc [16]. They show that the graphic representation of the program workflow, as well as the design procedures in the Orange application, is simple to use and its visual language is semantically transparent [16] compared with others. Multiple works in the literature corroborated their statement [17], [18], [19], [20], and this motivated our work to apply Orange as a programming aid to our final-year students. An example workflow is shown in Figure 3.

TABLE 2. Some of the widgets that can be used to build programming workflows in the Orange software.

Category	Widgets
Data.	File, CSV File Import, Datasets, SQL Table, Data Table, Paint Data, Data Info, Rank, Edit Domain, Color, Feature Statistics, Save Data.
Transform	Data Sampler, Select Columns, Select Rows, Transpose, Merge Data, Concatenate, Select by Data Index, Unique, Aggregate Columns, Group by, Pivot Table, Apply Domain, Preprocess, Impute, Continuize, Discretize, Randomize, Purge Domain, Melt, Formula, Create Class, Create Instance, Python Script.
Visualize	Tree Viewer, Box Plot, Violin Plot, Distributions, Scatter Plot, Line Plot, Bar Plot, Sieve Diagram, Mosaic Display, FreeViz, Linear Projection, Radviz, Heat Map, Venn Diagram, Silhouette Plot, Pythagorean Tree, Pythagorean Forest, CN2 Rule Viewer, Nomogram
Model	Constant, CN2 Rule Induction, Calibrated Learner, kNN, Tree, Random Forest, Gradient Boosting, SVM, Linear Regression, Logistic Regression, Naive Bayes, AdaBoost, PLS, Curve Fit, Neural Network, Stochastic Gradient Descent, Stacking, Save Model, Load Model.
Evaluate	Test and Score, Predictions, Confusion Matrix, ROC Analysis, Performance Curve, Calibration Plot, Permutation Plot
Unsupervised	Distance File, Distance Matrix, t-SNE, Correlations, Distance Map, Hierarchical Clustering, k-Means, Louvain Clustering, DBSCAN, Manifold Learning, Outliers, PCA, Neighbors, Correspondence Analysis, Distances, Distance Transformation, MDS, Save Distance Matrix, Self-Organizing Map.

3) DATA SCIENCE EDUCATION

The concept of data science is not entirely new, in fact, the evolution of data science can be attributed to the interdisciplinary integration of various subjects such as statistics, mathematics, computer science and domain knowledge [21]. To put into perspective, Conway [22] drafted what is to be known as the "data science Venn diagram", as shown in Figure 4. This is a diagrammatic representation illustrating the interdisciplinary nature of data science.

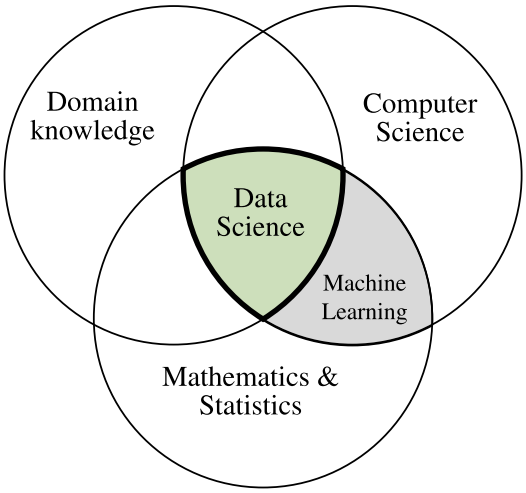

FIGURE 4. Data science Venn diagram. The field of data science lies at the intersection of three primary fields; domain knowledge, computer science, and mathematics & statistics [22].

From the Venn diagram in Figure 4, it is quite evident that the three fields existed independently before the introduction of the data science nomenclature. It is only after the efforts of several authors such as John Tukey [23], Chambers [24], Cleveland [25] among others, who called for the establishment of an interdisciplinary field that was based on the expansion of statistics and integration with computer science and domain knowledge [26], [27].

We have witnessed a rapid increase in the field of data science over the past few years. The explosion of data, advances in technology and the demand for data science professionals are some of the major drivers behind this rapid increase [28]. To bridge the gap between professional data scientists and the data deluge, there have been concerted efforts along two fronts to expedite data analytic training. First, traditional educational institutions such as colleges and universities throughout the world have introduced several specializations in data science at various levels of education. Apart from the mainstream formal data science education, there has also been a significant push through massive online open courses (MOOCs). The latter model offers several online data-related programs, often for free, in collaboration with tutors at major universities. This ensures customized and flexible learning modes, which greatly appeal to learners who are already working and might not have the time to commit to a full-time or part-time formal educational system. Despite the learning mode, every graduating data scientist must be proficient with certain skills and knowledge, what we shall refer to in this report as ''key competencies in data science''.

a: KEY COMPETENCIES IN DATA SCIENCE

Studies on the core competencies required for the graduating data scientist are scattered throughout the literature. However, the work by Donoho, in their report on the 50 years of data science articulates unambiguously the six data activities that can be studied or taught in a data science course: *(i)* data exploration and preparation, *(ii)* data representation and transformation, *(iii)* computing with data, *(iv)* data modelling, *(v)* data visualization and presentation, and *(vi)* science about data science [27].

The curriculum of data science courses offered by many educational institutions throughout the world often revolve around these activities. They may be different in wording or nomenclature, but the fundamentals do not differ. Further, Hazzan et al. suggested that such data activities promote computational thinking, statistical/mathematical thinking, and data thinking, which all have cognitive abilities [29]. In the report, ''A Guide to Teaching Data Science'', Hicks et al. present the five basic guiding principles for developing a data science curriculum [26]. They suggest that: *(i)* the course must be organized around a set of diverse case studies, *(ii)* computing must be integrated into every facet of the course, *(iii)* reliance on mathematical notation must be minimized, while promoting abstraction, *(iv)* the course activities must be structured to mimic a data scientist's experience, and lastly, *(v)* the importance of critical thinking must be demonstrated through examples [26].

These guiding principles and activities have formed the cornerstone of data science education for years. They have been able to develop data literacy among students to gain actionable insights and extract meaning from data. However, In our view, one aspect that is often neglected in this process is the challenge of developing computational thinking skills among students. The challenge recognized here is the need to simplify the development of computer code that supports the entire processes of data analysis, especially for non-computer science majors. Teaching programming has traditionally relied on text-based languages, which has been cited by many novice programmers as one of the barriers to entry in programming [30], and consequently data science. Here, we believe that using engaging learning environments, such as NCTs, for programming will make developing computer code manageable to a larger number of students, while still developing key competencies in data science as text-based programming languages.

4) EMPIRICAL EVIDENCE ON THE USE OF NCTS/VPLS IN EDUCATION

This section seeks to establish the usefulness of NCTs to early-stage learners and professionals. Then determines if VPLs provide the necessary tools to equip learners with the knowledge and skills required to solve programming problems. Finally, the section establishes the effectiveness of NCTs in facilitating the understanding of complex programming concepts. To address these questions, we conduct a short survey to review works that have implemented VPLs in any formal educational activities, from junior to tertiary education.

A significant effort to document the benefits of visual programming languages, compared with textual languages

is reported in [30]. In their work, Kelleher et al. survey a variety of graphics-oriented programming languages that can be used for different application areas. They cite syntax and program style as primary barriers to programming and demonstrate that simplifying the mechanics of programming, especially for novice programmers, greatly reduces the barrier to learning to program. Later in 2007, the authors considered using storytelling to motivate programming [31]. In their later work, they attribute a falling interest in Computer Science in the US to the uninspiring courses taught. To inspire middle-school girls' interest in programming, they use the Storytelling Alice programming environment to create custom 3D animated movies from in-built characters and environments. Their results show most of the participants were able to create a sequential program in Storytelling Alice within just two hours of programming, while 87% were able to create a program with multiple flow control mechanisms. Based on these results, the authors concluded that offering computer programming in the form of storytelling encouraged the target group to learn to develop computer programs.

To support self-directed learning among young learners in developing computer programs, Maloney et al. developed the Scratch programming language and environment [32]. Maloney's programming environment allows students, especially primary ages to create engaging projects such as animated stories, games and simulations [32]. A distinguishing feature of Scratch is that program flow is constructed sequentially by joining together building blocks that represent actions or flow control mechanisms. Its primary goal is to introduce programming education to learners who have little or no programming experience. This goal has motivated the worldwide use of the tool, and by 2010, the program had been offered in nearly fifty languages and had supported almost two million users. By January 2024, the number of subscribers has risen to over one hundred million registered users [33], signifying the importance of graphical-based teaching of computer programming. The program has been so popular that it has been incorporated into formal education streams, targeting early-stage programmers in different fields.

A pilot study by Friss et al. at ORT Uruguay University during the 2nd semester of 2007 experimented with Scratch in two scenarios. They incorporated Scratch in; (i) a university course and (ii) vocational studies environments to improve students' capabilities in computer science courses [34]. In their work entitled Scratch: Applications in Computer Science 1, the authors conduct formative and summative assessments on a group of students who were randomly selected from the class. They administered scratch, over three weeks with the control group solving the same programming tasks manually. For their results, 88% of students who had used the Scratch programming environment described their learning experiences as ''motivating'' or ''easy'', while 80% of the control group described their learning experiences as ''normal'' or ''difficult''.

In perhaps an interesting and related application area, Estevez et al. introduced Artificial Intelligence to high-school students using Scratch. Their work is motivated by a strong need to grasp the attention of young learners through the use of interactive graphical programming tools in computational sciences, which is usually characterized by a lack of appeal of the presentations [35]. In their work, they teach young learners two basic methods of Artificial Intelligence: data clustering, and artificial neural networks learning. They selected 37 students and followed a scaffolded teaching approach, where they provided a pre-built template for the students to fill the gaps among lines of code. Just like the authors in the previous work, Estevez's approach also conducts a formative and summative assessment of the target group. A quantitative analysis of the results reveals that the students acquired confidence to understand the fundamentals of Artificial Intelligence algorithms, and its holistic view.

A case study by Ase et al. to teach engineering modules using computer-aided animations was conducted at the University of Hertfordshire. The study focused on applying visualization and 3D animations in automotive engineering courses to help improve conventional teaching resources. They explore the benefits of automation, with particular emphasis on 3D computer-aided animation tools for automotive studies. This is an innovative paradigm shift from the conventional methods of delivery that are based on 1D flowcharts, schematics and static objects. In their results, they report that over 61% of the students reported a better understanding of automotive engineering modules taught using animations.

We have provided empirical evidence where VPL tools have been applied successfully in education. Results show that such programming environments are helpful to early-stage learners, they have the necessary tools to foster learning. Furthermore, it has been shown that such tools grasp the attention of learners, promote their motivation to learn, and improve their learning experiences, without focusing on the mechanics and intricacies of programming, which have been shown to be a barrier to programming. With these results, we are convinced that such tools can also be integrated with data science education to improve problem-solving, creativity, motivation, collaboration and data science communication at the tertiary level.

III. DATA SCIENCE PROGRAMMING USING NCTs

The following Chapter discusses the sections of the data science curriculum and develops section-specific assessment items for evaluating Python coding against NCT workflows as shown in Table 4 – Table 12.

It should be noted, however, that the chapters for this proposed curriculum have been adapted from the conventional data science curriculum, as found in modern textbooks such as [4] and [36], and teaching has been modified to support data science education using no-code programming environments. This was done to ensure learners would be exposed to the same content that is offered in a conventional

TABLE 3. Overview of the proposed curriculum: The proposed data science curriculum using NCTs, including the aims, knowledge area and learning objectives in nine chapters. The design follows a typical data science structure for majors, except that the practical component does not rely on textual programming.

Curiculum aims	
Students will be able to: 1. understand the foundations of data science concepts, principles, and methodologies using NCTs. 2. Develop proficiency in NCTs and platforms used in data science, such as Orange, Neural Designer, or KNIME. 3. Develop practical data analysis skills by working hands-on with real-world datasets 4. Develop key competencies in data exploration, data transformation, and predictive modelling using NCTs. 5. Develop critical thinking and problem-solving skills in data science to help identify patterns, draw insights, and make data-driven decisions 6. Present their findings and insights to data science and non-data science audiences through visualizations and reports	
Topic	**Learning Content**
1. Introduction to Practical Data Science and NCTs	Introduction to popular no-code data science tools and platforms
2. Data Collection and Preparation	Understanding data types, formats, and sources
3. Data Visualization	Creating interactive and informative visualizations
4. Unsupervised Learning	Clustering algorithms (K-means, hierarchical clustering, DBSCAN.)
5. Supervised Learning	- Introduction to supervised learning algorithms (classification and regression) - Training and evaluating models
6. Feature Engineering and Selection	- Techniques for creating and selecting relevant features - Feature transformation and scaling
7. Model Deployment	Deploying models
8. Time Series Analysis	Forecasting and trend analysis
9. Introduction to Machine Learning Automation	Using no-code AutoML platforms for rapid model development

data science curriculum while being taught in dynamic and interactive learning environments. This curriculum can also be used by tutors who want to study introductory data science using such tools.

A. CHAPTER 1: INTRODUCTION TO PRACTICAL DATA SCIENCE AND NCTs

As shown in Table 3, the proposed curriculum begins with an overview of practical data science approaches using NCTs. This introduction takes learners through the fundamental concepts of the application of data science and gives a detailed discussion of the working principles of NCTs. Additionally, this introductory part discusses the importance of data science in industry, the opportunities, challenges, future and the impacts of data science education using highly abstractive frameworks. The learning objective of the first module is to understand the most popular NCTs, in addition to basic knowledge of data science and its applications, especially in the student's application area. At the end of the first chapter, new learners must have developed a thorough understanding of data science, and its application areas, along with the knowledge of visual tools for developing data-centric frameworks.

Table 4 shows some of the assessment items for the introduction to practical data science and NCTs chapter.

TABLE 4. Assessment items for Chapter 1.

Python-based Assessment Items	VPL-based Assessment Items
– Discuss the applications of data science in different areas. – Download and install Python and the related packages for machine learning.	– Discuss the applications of data science in different areas. – Download and install Orange.

Here, the main emphasis is on introducing NCTs to the learners and installing the software locally on their machines.

B. CHAPTER 2: DATA COLLECTION AND PREPARATION

The following chapter is by far the most critical. In fact, some texts in the literature cite this phase as the most time-intensive [37]. The data collection and preparation phase requires a substantial investment of time, money, and resources due to the intricacies of data, planning and design processes, and ethical and legal considerations. It should be noted, however, that data collection is not platform-specific. It is the same across different platforms. On the other hand, data preparation is different. It requires extensive knowledge of the data structure and procedures to sort, and prepare it for subsequent stages.

TABLE 5. Assessment items for Chapter 2.

Python-based Assessment Items	VPL-based Assessment Items
– Write Python code that takes a dataset, and performs appropriate transformations to clean and prepare the dataset such as removing rows with missing values or adding missing values.	– Use suitable widgets to load the dataset into your workflow, clean the data and visualize the dataset.

TABLE 6. Assessment items for Chapter 3.

Python-based Assessment Items	VPL-based Assessment Items
– Write a Python script that uses *matplotlib* to create basic plots of the dataset such as scatterplots, bar graphs or histograms	– Drag and drop data visualization widgets such as scatter plots, bar plots or distributions into your workspace to visualize the data.

Therefore, the curriculum for data collection and preparation focuses on students understanding the various data types, formats, and pre-processing stages that are performed on datasets using NCTs, especially very large datasets.

The starting point of any data-related problem is the collection of usable, representable and unbiased data. This is a critical process that requires (1) a prior understanding of the problem at hand, (2) formulating research questions, and (3) a thorough comprehension of the subsequent objectives of the data analysis [38]. Several authors in the literature discuss various principles and procedures that must be followed to ensure data integrity, however, that is beyond the scope of this work. A comprehensive overview is provided by [39]. In this chapter, we will only focus on working with the data that has already been collected. However, the rule of thumb is to ask "What data?". Navigating this space will require identifying the relevant data sources and planning the data collection and processing methods. On the contrary, research shows that a lot of students and novice data scientists often struggle with this part [40], hence the use of NCTs to simplify the data collection and preparation process.

Table 5 shows some of the assessment items for the data collection and preparation chapter. The primary focus is to take learners through the processes of collecting data and preparing it for the subsequent steps.

The learning outcomes of this chapter have been developed as follows: (i) identify the sources of data for a particular project, (ii) evaluate the reliability of data sources and the data collection procedures, (iii) collect that data from different sources and (iv) clean and pre-process the data to detect and handle inconsistencies such as missing values, and outliers.

C. CHAPTER 3: DATA VISUALIZATION

Data visualization is considered to be one of the most important topics in data science [41]. As such, a lot of emphasis has been placed on the development of programming languages, visualization libraries and frameworks to enable data-driven decision-making. Recent efforts are in huge graph visualization with big data infrastructure [42].

The concept of data visualization can be traced back centuries to ancient Greek mathematicians who utilized latitude and longitude information to visualize geographic information [43]. Subsequent developments in coordinate systems and Cartesian graphs by scientists, mathematicians and philosophers in the 17^{th} Century are widely considered to have laid the foundations of modern data visualization techniques [44]. However, it is in the 20^{th} Century that data visualization rose to prominence due to major developments in computer graphics, technology, scientific visualization, personal computers, and software tools [43]. The continual developments in software tools into the 21^{st} Century, especially open-source tools, enabled users to create custom visualizations using simple, yet powerful data visualization libraries such as Pandas, Matplotlib and Seaborn [45], among others.

The objectives of the data visualization stage in the data processing pipeline vary depending on a lot of factors. However, any data-proficient student must be able to effectively communicate insights and findings, support informed decision-making, and identify patterns, trends, and correlations derived from the data analysis, irrespective of the platform. NCTs support data visualization through the use of widgets that create visual elements such as scatter plots, line plots, histograms, bar charts, and heat maps, among others. At the end of this chapter, learners must be able to confidently create interactive and informative visualizations that (1) facilitate the understanding of relatively straightforward or complex data and (2), provide a holistic view of the data, thus facilitating more informed knowledge discovery.

Table 6 shows some of the assessment items for the data visualization chapter. Here, the focus is on using relevant widgets to visualize relations in the data.

D. CHAPTER 4: UNSUPERVISED LEARNING

The subject of unsupervised learning rose from a strong need to detect anomalies or discover hidden structures or trends in unlabeled datasets. A central feature of these algorithms is that they do not require prior knowledge or output labels of the datasets. In other words, they do not require training datasets to learn data dependencies, instead, they learn features on their own from uncategorized data on the fly. This chapter aims to study a range of unsupervised machine learning algorithms for clustering such as K-means, hierarchical, and density-based spatial clustering of noisy datasets. The chapter proceeds with a discussion on dimensionality reduction techniques. These are a set of algorithms that reduce the number of variables or features, creating a lower dimensional representation of the dataset. Principal component analysis is widely used for this. Lastly, the chapter explores algorithms for anomaly detection. This is a very critical and broad area that has witnessed significant research over the years due to its capability of detecting

TABLE 7. Assessment items for Chapter 4.

Python-based Assessment Items	VPL-based Assessment Items
– Write a Python script that uses the k-means clustering algorithm from the *scikit-learn* library to identify distinct clusters in a dataset.	– Load an example dataset using the file widget and connect it to the k-means clustering widget to identify distinct clusters in the dataset. Adjust the settings of the k-means clustering widget and use the scatter plot widget to display the clustering results.

unusual or inconsistent data points in various applications. These algorithms can be applied in areas such as network security, fraud detection, and quality control, among others to distinguish between normal and unexpected behaviour. Finally, at this stage, we are only interested in three learning outcomes for this chapter. Students must; (i) understand unsupervised learning algorithms, (ii) apply unsupervised learning models on real-world data and (iii) evaluate the performance of unsupervised learning algorithms.

Table 7 shows some of the assessment items for the unsupervised learning chapter. Learners can work with different widgets such as k-means, DBSCAN, and PCA, among others, to identify distinct clusters in the dataset.

E. CHAPTER 5: SUPERVISED LEARNING

Unlike unsupervised learning algorithms that do not require prior knowledge of the data, there exists another learning paradigm that requires knowledge of the class labels to make predictions or decisions. Supervised learning algorithms require a lot of training data that consists of input features and their corresponding output labels. The goal is to put related objects with identical features in the same class or label. The algorithms then learn these features or study the patterns or relationships between members of the same class. Training is realized through an iterative process of adjusting the model's parameters based on minimizing the training error. Several supervised learning algorithms have been developed in the past, and some of the most popular include neural networks, linear regression, support vector machines, and decision trees, among others. After model training and evaluation, the model is now ready to make predictions on new or unseen datasets. It is a common goal for a well-trained model to make accurate predictions on new datasets. The overarching aim of this chapter is to take students through the entire process of training, testing/validating, making predictions/inferences and deploying supervised learning algorithms using NCTs.

Table 8 shows some of the assessment items for the supervised learning chapter. These questions test the student's ability to create models that learn from the data.

F. CHAPTER 6: FEATURE ENGINEERING AND SCALING

The next chapter in the curriculum covers feature engineering and selection using no-code tools. This is another important practice that involves creating or selecting better representations or informative features in the dataset for machine learning algorithms. A major advantage of selecting features that carry more information is that the overall performance of the machine-learning model is improved. In addition to that, the overall computational cost and memory requirements are reduced due to an overall reduction in the number of variables. This is another research area that has received considerable attention, and progress has led to the development of algorithms that use computational methods for automatic feature engineering and selection. On another front, deep neural networks learn the relevant features automatically, and this has wide applications in image, video or speech processing. Although automated feature engineering has shown considerable success in the literature, there is a need for data science learners to grasp the background knowledge of such an important aspect. As such, the goal of this chapter is to equip students with the skills of creating and selecting valuable features from the data, along with feature transformation and scaling using NCTs.

Table 9 shows some of the assessment items for the feature engineering and scaling chapter. The questions ask learners to create new features from the dataset.

TABLE 8. Assessment items for Chapter 5.

Python-based Assessment Items	VPL-based Assessment Items
– Write a Python script that uses the *scikit-learn* library to create a linear regression model and fit it into a given dataset. Apply the model to new data to make predictions.	– Create a workflow that uses the Linear Regression widget to build a model that learns a linear function from the dataset. The settings of the linear regression widget can be changed to include or exclude regularization parameters. You can use the Data Table widget to view the calculated linear regression coefficients.
– Write a Python script that implements the SVM algorithm from the scikit-learn library to build a supervised machine learning model.	– Create a workflow that uses the SVM widget to train the model on a sample data set. Use the file widget to import the data and the scatter plot widget for visualization. The settings of the SVM widget can be changed to select a suitable kernel and optimization parameters.
– Implement the confusion matrix from the *sklearn* library to evaluate the quality of classification algorithms on a sample data set.	– Experiment with different widgets such as Test and Score, Predictions, and Confusion Matrix, among others to observe the performance of learning algorithms on a sample data set.

G. CHAPTER 7: MODEL EVALUATION AND DEPLOYMENT

After model development, a natural question that typically follows is "How good is that model?" Although this is a trivial question, there is a strong need to know if the model can make accurate predictions on unseen data. If poorly trained, machine learning models tend to memorize the training data

TABLE 9. Assessment items for Chapter 6.

Python-based Assessment Items	VPL-based Assessment Items
– Write a Python script that uses *numpy* and *pandas* to construct new features from existing data.	– Use various transformation and combination widgets to create new features for learning algorithms.

TABLE 10. Assessment items for Chapter 7.

Python-based Assessment Items	VPL-based Assessment Items
– Deploy your model as a web application using Python and *Streamlit*.	– Use the Save Model widget to export your model in Python *pickle* format. The model can be loaded directly into Python for classification and subsequent deployment.

very well, leading to poor predictions on new samples. This is a huge problem, especially in safety-critical applications, or the development of new algorithms. Answering this question requires a scientific methodology that addresses a specific question: "How do we evaluate the generalization performance of machine learning models?

The next chapter in the curriculum discusses concepts of model evaluation and deployment using NCTs. In simplest terms, model evaluation refers to the process of determining the performance of models using certain metrics or other approaches. Typical evaluation metrics include but are not limited to accuracy, precision, recall, and F1 score [46]. At the end of this chapter, students must be able to assess machine learning models using various metrics and techniques, along with deployment in real-world scenarios.

Table 10 shows some of the assessment items for the model evaluation and deployment chapter.

H. CHAPTER 8: TIME SERIES ANALYSIS

The content so far has only focused on modelling the spatial relationships in the data. There is a certain area, however, that studies the temporal dependencies in the data. For example, problems such as power output forecasting, predicting stock prices, weather prediction, energy usage, fuel consumption monitoring, and many more, require historical data at pre-determined intervals over a longer period to predict future values. Time Series Analysis gives insights into the behaviour or trends of time-varying observations. This involves studying the temporal sequence of features using statistical techniques, recurrent neural networks with memory blocks, or 3-dimensional deep learning models. Several architectures with different mechanisms for time series data exist, and their performance has been widely documented in multiple texts in the literature [47], [48]. Since this is a large area with so many application scenarios, this curriculum hopes to equip students with the knowledge of forecasting and trend analysis of sequential data.

TABLE 11. Assessment items for Chapter 8.

Python-based Assessment Items	VPL-based Assessment Items
– Use the *pandas* package to read a time series dataset into a dataframe, and use the *matplotlib* package to visualise the series. Identify and analyse long-term trends in the data.	– Use the Time Series widgets to identify and analyse long-term trends in the data, and use the Line Chart widget to visualize the time series sequence.
– Write Python code that implements a Recurrent Neural Network (RNN) model from tensorflow to forecast future values based on historical data.	–Use the Time Series widgets to forecast future values from historical data.

TABLE 12. Assessment items for Chapter 9.

Python-based Assessment Items	VPL-based Assessment Items
– Write Python code that uses the *h2o* library to automatically build and optimize machine learning models.	– Use no-code tools to automatically build and optimize machine learning models.

Table 11 shows some of the assessment items for the time series analysis chapter. The questions determine the student's ability to extract temporal information from the data.

I. CHAPTER 9: INTRODUCTION TO MACHINE LEARNING AUTOMATION

The last component in this curriculum deals with machine learning automation. This is a particularly interesting subject that uses computer algorithms and tools to perform tasks such as data preparation, feature engineering, best model selection, training, and hyperparameter tuning, among others. This came from an urgent need to address a pertinent question in the literature: "How can users design more efficient and effective machine learning models?" Although this is a broad question, an answer to this has led to the removal of "human-in-the-loop", since some aspects of the machine learning workflow rely on a trial and error basis, for example, determining the best learning rate for the training dataset. The aim of automating machine learning workflows is to speed up the development and subsequent deployment of machine learning models. This is helpful, especially to novice data science professionals, however, human expertise is still crucial in the process. At the end of this chapter, students must be able to leverage the power of NCTs to develop and implement automated machine learning workflows and use the knowledge they have gained to interpret the results.

In this section, we have presented a nine-chapter syllabus that takes students through the entire data analytic workflow using NCTs. The data science curriculum based on NCTs is not in any way different from the conventional curriculum. Students still learn the basics of data science, but using a different tool for programming. The structure follows traditional

content that consists of an introduction to data science, data collection, data visualization, unsupervised learning, supervised learning, feature engineering, model evaluation and deployment, time series analysis, and machine learning automation. In that regard, we believe that more emphasis should be placed on developing teaching, assessment and evaluation methods suitable for NCT-based education.

Table 12 shows some of the assessment items for the machine learning automation chapter. The emphasis is on testing student's abilities to automate the entire process of machine learning.

IV. TEACHING METHODS

A key pillar in the curriculum is teaching. In this Chapter, we review the teaching methods that are suitable for NCTs and provide relevant examples of their implementation. We discuss lecture-based, instructor-led problem-based and project-based learning approaches about their strengths, and weaknesses and further give an example of implementing project-based learning on a sample fall detection project.

There are several approaches to learning, and each approach produces different outcomes [49]. Since the new data science curriculum is composed mainly of theoretical and practical aspects, we believe that a combination of multiple learning strategies is more suitable to teach programming for data science using NCTs. We suggest a combination of lecture-based, instructor-led problem-based and project-based learning approaches. In the past, lecture-based learning has been the primary mode of instruction at most tertiary institutions. This instructional approach traditionally involves a classroom set-up, where the lecturer leads the class and presents information while the students listen and take notes [50]. This model is relevant in this curriculum since the students need organized and structured background information and literature from the lecturer to understand the basics. Although this mode offers efficient delivery of information, it continues to be the subject of criticism in various disciplines [51], [52]. Zhao and Potter argue that lecture-based learning is mainly lecturer-centred, and promotes the superficial acquisition of knowledge [52]. We believe that this criticism is valid, to a larger extent, if the entire curriculum is based solely on this delivery method.

To reduce the gap between lecture material and practice, problem-based learning led by an instructor is envisioned to complement deficiencies in an all lecture-based learning model. Problem-based pedagogy is an instructional method that has origins in the medical field and has been reported to foster active learning by creating a need to solve an authentic problem [53]. This is a mainly student-centred pedagogical approach where the starting point is to solve a context-specific problem that the students can handle [53]. Problems can then be solved by the students in small collaborative groups, often led by a lecturer [54]. This area has had a long history and has received a lot of attention in the literature. Different authors have proposed distinctive guidelines for implementing and assessing problem-based learning in schools. In this curriculum, we adopt the seven-step method described by Graaff et al. to help students analyze the problem [55]. These are *(i)* clarify the concepts, *(ii)* define the problem, *(iii)* analyze the problem, *(iv)* find the explanation, *(v)* formulate the learning objective, *(vi)* search for further information and *(vii)* report and test new information [55]. This is envisaged to intrinsically motivate students, improve communication and effective collaborative skills, and provide a more enjoyable learning experience [56].

TABLE 13. Methods of creating project-based learning workflow.

Reference	Methods
Grant [60]	Presents an anatomy of project-based learning. 1) Introduction (The introduction presents the subject area and provides background to the project.) 2) Task 3) Resources 4) Process 5) Guidance and scaffolding 6) Collaborative learning 7) Reflection
Krajcik [61]	Project-based learning has the following features: 1) Start with a driving question 2) Students explore the driving question 3) Collaborative engagement between students and teachers to find solutions to the question 4) Students are scaffolded with learning technologies that help them participate 5) Students create tangible products that address the driving question

Project-based learning is another student-centred approach to learning with particular emphasis on investigating real-world systems and collaboration to gain knowledge [57]. Although it shares similar characteristics with other instructional approaches such as Problem-Based learning, here the focus is on the use of projects to promote learning [58]. Condliffe argues that solving projects stimulates learning [59]. This is aided by working on and implementing the design principles that are being taught. Table 13 outlines the methods of creating project-based learning workflows, as described by two prominent authors, Grant [60] and Krajcik [61]. Figure 5 presents an anatomy of a project-based learning approach inspired by Grant [60]. This discusses several steps that can be implemented using NCTs such as Orange. We believe Grant offers a stronger approach to project-based learning due to important and relevant stages. We also provide an example of implementing project-based learning using NCTs on a project to develop a real-time fall detection and monitoring system for the elderly.

All these three instructional approaches will be adopted in this curriculum to help new learners develop skills in data science. In this curriculum, each chapter will begin with conducting lectures, following the conventional lecture-based method. As students understand the concepts, problem-based learning is gradually introduced, and it covers the rest of the chapter. Towards the end of the module, students are given

a project to solve individually, or in groups. An example project, along with the implementation layout is shown in Figure 5.

V. ASSESSMENT METHODS

This Chapter addresses five important questions: *(i)* how do you assess learners who use graphical tools for programming?, *(ii)* who has reported on the use of visual programming languages, especially in higher and tertiary education scenarios?, *(iii)* has it been successful? *(iv)* what were their recommendations?, and *(v)* how can this impact our curriculum design?

The question of student assessment, especially in general Andragogy, has been addressed thoroughly throughout the literature [63], [64], [65]. This is mainly conducted to evaluate how well students have performed against a set of learning outcomes at various stages of learning. This provides quantifiable evidence that can then be used by both students and lecturers to evaluate the knowledge and skills gained through learning [66]. In [67], Llamas-Nistal et al. discussed the two main categories of assessment as continuous, and summative assessment. Continuous assessment is usually carried out during the instructional process to gather and analyze information on student's performance [68]. On the other hand, summative assessment is carried out towards or at the end of the learning process to evaluate cumulative knowledge and skills gained [68].

According to the computing curriculum developed in 2013 by the Association for Computing Machinery – IEEE-Computer Society, the generic learning outcome of any programming course is to design, implement, test, and debug a program that incorporates some basic programming constructs [69]. This is a guideline that has been used by many authors in the literature to assess students in programming and has been used as a basis to judge the coding abilities of first-year computer science students [70]. Assessing students who use graphical tools for programming is not in any way different from assessing students who use conventional programming tools. With VPL tools, tutors can also test students' main learning outcomes such as designing, implementing, testing, and debugging a program that incorporates basic to advanced programming constructs. Here, students will connect various widgets of a VPL to demonstrate program sequence, selection functions, and iteration loops. Afterwards, the tutor will run their programs to evaluate these concepts.

A. VPL-BASED CURRICULUM ASSESSMENT AND EVALUATION

1) PARTICIPANTS

Participants were students enrolled in the Department of Applied Physics and Telecommunications and the Department of Computer Science under the Faculty of Science and Technology at Midlands State University. Students were enrolled in one of the 3 Degree programs: BEng Honors

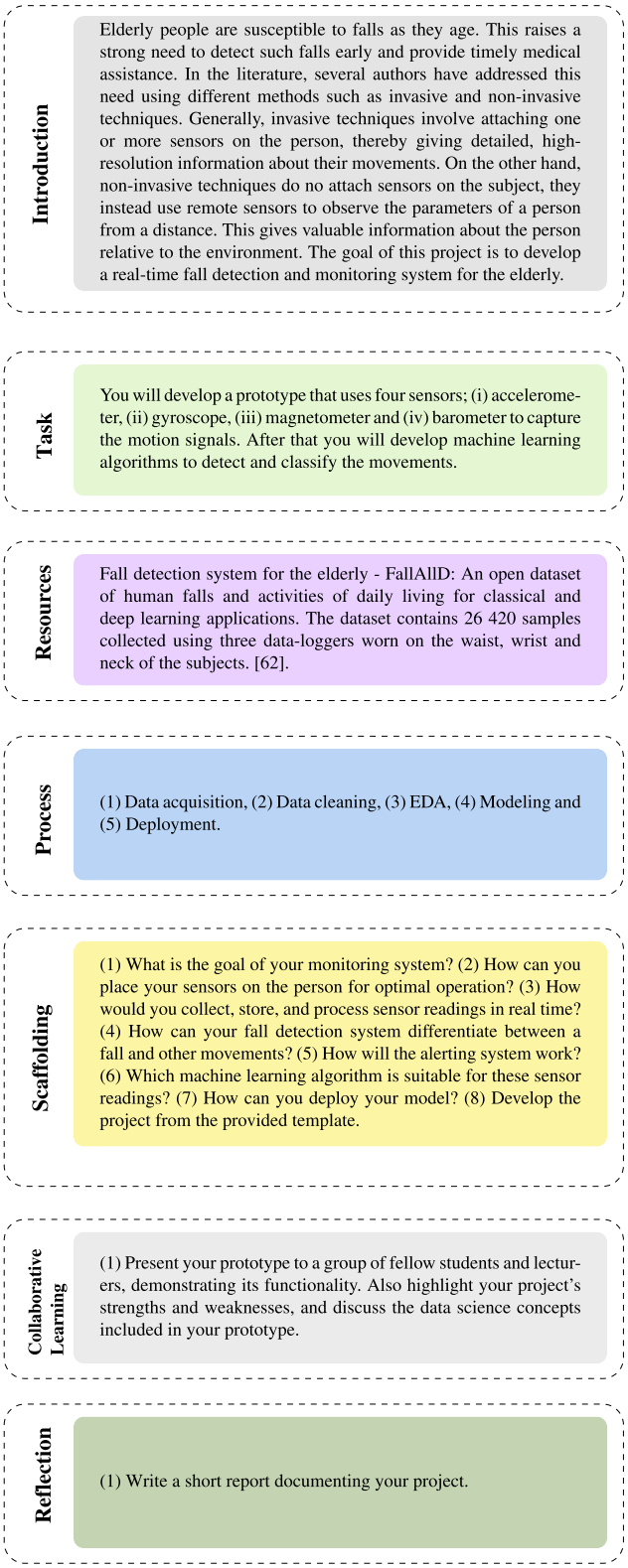

FIGURE 5. Project-based learning implementation on a sample project. The capstone project was developed as an example to assist students through the workflow. The starting point is an introduction that presents the subject area and provides the background of the project. Next, the task presents several activities to be accomplished. The resources outline the materials required to do the project. For this example, we used the fall dataset for the elderly [62]. This is followed by processes, scaffolding, collaborative learning and finally, reflection.

in Telecommunication Engineering Degree, BSc Honors Degree in Industrial Physics and Instrumentation and BSc Honors Degree in Computer Systems Engineering. The study targeted final-year students who were working on completing their dissertation projects. Three classes from six different semesters over three years were studied. Class sizes ranged from 10 to 25 students. Most participants were male. All participants completed a first-year introduction to computer programming module. Participants from the Telecommunications Engineering and Industrial Physics Degrees had studied at least two computer science-related modules, and are thus considered non-majors. Participants from the Computer Science Degree have vast computer programming experience, after having studied at least 20 computer science modules, and are thus considered majors.

2) METHODS

The research process focused on implementing and assessing the effectiveness of three pedagogical methods, a lecture-based model [50], [51], [52], instructor-led problem-based model [53], [54], [56], and the project-based model [57], [58], [59], [60], [61] using VPLs on data science problems. The research was implemented in two interventions. The first intervention motivated this research, and it took place over six semesters covering three years from March 2021 to January 2024. Here, the study focused on qualitatively observing students from BEng Honors in Telecommunication Engineering class and BSc Honors in Industrial Physics and Instrumentation class on their ability to implement final-year dissertation projects using VPLs (project-based learning). Post-interviews were conducted to determine the usefulness of VPLs in solving data science and related programming problems.

The second intervention centred on two academic semesters from August 2023 to January 2024 of pedagogical practice in data science education using the lecture-based model and instructor-led problem-based model. Here, the study focused on quantitatively observing students from the three classes on the VPL assessment items mentioned previously in Chapter III and post-study questionnaires.

VI. RESULTS

This Chapter presents the results of qualitative and quantitative data analysis of the first intervention performed on the effectiveness of VPLs in teaching programming aspects of data science. Specifically, the first research question, "Do non-computer science learners and professionals find no-code development tools helpful?", is addressed by qualitatively assessing the performance of students in successfully implementing a data science-related project using VPLs. The second research question, "Do visual programming tools provide the necessary tools to equip learners with the knowledge and skills to solve data science problems?" is answered through questionnaires on students' experiences with the VPL. Finally, the third research question, "How should test items be structured to assess and evaluate data science education based on NCTs?" is answered by quantitatively assessing the performance of students on sample assessment items.

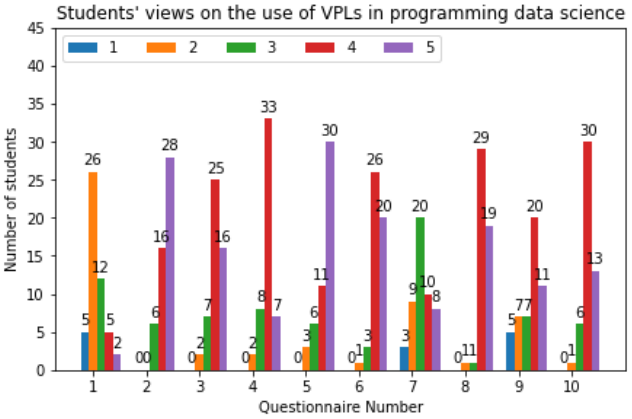

FIGURE 6. Students' summaries of the responses to the questionnaire.

A. STUDENT'S PERFORMANCE ON VPL-BASED PROJECT-BASED LEARNING

The first research question is centred on the application of qualitative analysis, observing the performance of non-computer science students in implementing a final-year dissertation project based on VPLs. Students were observed throughout the development of the project, which allowed us to evaluate the usefulness of VPLs. There were two groups of students, the control group using Python code to develop and implement a dissertation project, and the experimental group using VPLs to develop and implement a slightly different dissertation project. Results from observations revealed that students who used Python code to build a project took longer to complete, seeking assistance and asking a lot of questions from lecturers, fellow students, online forums and tutorials. On the other hand, students who used VPLs completed their projects with minimum help. From the results, it can be stated that NCTs or VPLs are useful, especially for students who do not have vast programming experience.

B. VPLS' ABILITY TO SOLVE DATA SCIENCE PROBLEMS

The second research question is focused on establishing students' views and experiences on the use of VPLs in solving data science programming problems. To address this question, quantitative data was collected from post-study questionnaires. The questionnaires consisted of 10 questions, as shown in Table 14 measuring students' opinions, experiences and motivation toward the use of VPLs in data science education. A $[1-5]$ numerical rating scale [71] "maximum" (5), "minimum" (1) was used. 50 students completed the post-study questionnaire.

From the responses to the questions in Table 14, 77% of the experimental group students admitted to struggling with writing their code using Python or C/C++ before the study, while only 4% demonstrated proficiency in developing code. 88% of the students found the VPL tool to be easy to use,

TABLE 14. Questions regarding students' views and experiences on the use of VPLs in their data science projects.

Number	Question
1	How proficient were you with writing your own code in Python or C/C++?
2	As a student with minimum programming experience, how would you rate the ease of use of VPLs for programming compared to Python?
3	Did the VPL improve your understanding of data science concepts?
4	Did the VPL facilitate the implementation of data science algorithms in your project?
5	In your opinion, how important are VPLs as programming tools for data science?
6	How satisfied are you with the widgets of the VPL?
7	How likely are you to continue using VPLs for programming?
8	Would you recommend using VPLs for programming data science concepts, especially for non-computer science majors?
9	Did the VPL provide a wide range of widgets and flexibility for your project?
10	Finally, how satisfied are you with using VPLs for coding in your data science project?

and 80% said VPLs facilitated the implementation of data science algorithms in their projects. An impressive 96% of the students from the experimental group highly recommended using VPLs for programming to their fellow students who did not have vast programming experience. The responses are summarised in Figure 6.

C. ASSESSMENT AND EVALUATION OF DATA SCIENCE PROGRAMMING USING PYTHON AND NCTs

The third research question uses the second intervention which centres on quantitatively assessing and evaluating the performance of students from the three classes; BEng Honors in Telecommunication Engineering class, BSc Honors in Industrial Physics and Instrumentation class and the BSc Honors in Computer Systems Engineering class in Python and VPL assessment items. The control group is drawn from the BSc Honors in Computer Systems Engineering class while the experimental group is drawn from the non-majors (BEng Honors in Telecommunication Engineering and the BSc Honors in Industrial Physics and Instrumentation classes).

Code snippets of the assessment items of the two groups were analyzed to assess their performance in developing code using Python against VPLs. All sessions were supervised, and they were conducted in a typical computer laboratory setup. An analysis of the results is given in Table 15

The benefits of using visual programming languages to develop code were also confirmed in how students performed in VPL-based assessment items against standard Python-based assessment items. The performance of the students from the two groups was tested by a Two-Sample t-Test and found significant differences between the control group and the experimental group. The results obtained a t-value of 12.57, at a 0.05 significance level and a probability '$p < 0.0001$'. The results are shown in Table 15.

TABLE 15. t-Test – Statistical analysis of the performance of non-computer science majors (Physics and Telecommunications students) on Python-based and VPL-based assessment items.

	Experimental group score	*Control group score*
Mean	71.76	46.28
Variance	39.37	165.79
Observations	50	50
Hypothesized Mean Difference	0	
df	71	
t Stat	12.57	
$P(T <= t)$ one-tail	$p < 0.0001$	
t Critical one-tail	1.67	
$P(T <= t)$ two-tail	$p < 0.0001$	
t Critical two-tail	1.99	

This provides sufficient evidence to suggest that the use of visual programming languages increased students' abilities to develop code for solving data science challenges.

VII. CONCLUSION

As the data industry grows exponentially, so does the need to train new professionals or upskill existing professionals with data-related skills and new knowledge. This paper has structured a modularized undergraduate curriculum for data science education using no-code programming tools. Modular in the sense that it makes only one component of a course and the learning content is spread across one university semester. Guided by well-established educational philosophies, this curriculum adopts effective teaching and learning methods that are interactive and student-centred. Students benefit mainly from the advantages of graphical-based tools for learning programming and also the theoretical concepts of data science, which are delivered using the conventional lecture-based approach. The main teaching methods identified for the new method of learning data science are discussed in detail. Moreover, this curriculum adheres to the data science guidelines that define the key competencies in data science for undergraduate learners.

This work demonstrates the benefits of using visual programming languages to develop code for implementing data science concepts. Overall, the student's engagement in learning data science increased, and their assessment marks greatly improved. Notably, the control group obtained a mean of 46.28% on Python questions, while the experimental group obtained a mean of 71.76% on VPL-based questions. For the sake of comparison, the computer science class obtained a mean of 64.28% and 75.32% on Python and VPL questions respectively. There was a small difference of 3.56% between the means of non-computer science majors and computer science majors who had used VPLs to solve data science programming questions. This means that VPLs can assist

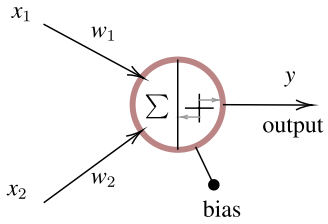

FIGURE 7. Basic representation of an artificial neuron. The artificial neuron is analogous in operation to the biological neuron. It has a bias parameter (constant), and accepts two or more inputs, which are multiplied by their respective coefficients (w_i). These weighed inputs are summed together to produce an output $z = \sum_{i=0}^{n} w_i x_i$ which is then passed through an activation function to produce $y = f(z)$ [72].

non-computer science majors to achieve higher marks that are comparable to the marks of computer science majors.

In this work, VPLs have shown great potential as a pedagogical aid to data science students who do not have a strong programming background. Research on assisting learners to transition from no-code tools to text-based programming languages is a work of further research.

In conclusion, the overarching contribution of this work will support students, tutors, educational institutions and data industries by (i) reducing the time and resources required to learn data science programming, (ii) offering an alternative approach to text-based programming in data science education, (iii) providing detailed procedures for developing teaching, learning, evidence-based assessment and evaluation methods using interactive learning environments.

APPENDIX A
OVERVIEW OF ARTIFICIAL NEURAL NETWORKS AND DATA SCIENCE

The human brain consists of several interconnected cells that transmit information encapsulated in electrical and chemical signals from various parts of the brain. These cells or neurons receive sensory inputs, process the signals and relay the output to other neurons. Neurons can work together to learn the solution to a problem by creating a neural pathway. This pathway becomes more accurate through trial and error by identifying neurons that regularly communicate. Through regular practice, the brain learns to solve a problem.

This simple, yet complex operation is the fundamental principle that has influenced the development of the artificial neuron as shown in Figure 7. Each neuron can be seen as an individual computing node that accepts one or more inputs x_i, and produces an output y based on an activation function $f(z)$. Non-linear functions Sigmoid functions are typically used, however, several activation functions are sufficient for this purpose. A summary of the most common activation functions for neural networks is presented by et al. in [72].

A collection of these neurons form an artificial neural network that learns from data by adjusting their parameters and finding the correct solutions on their own, thereby mimicking human intelligence. In what follows, we describe the technical principles that govern the operation of artificial neural networks.

REFERENCES

[1] M. Analytics, "The age of analytics: Competing in a data-driven world," in *McKinsey Global Institute Research*. McKinsey & Company, 2016. [Online]. Available: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-age-of-analytics-competing-in-a-data-driven-world

[2] R. D. De Veaux et al., "Curriculum guidelines for undergraduate programs in data science," *Annu. Rev. Statist. Appl.*, vol. 4, pp. 15–30, Aug. 2017.

[3] M. J. Ramzan, S. U. R. Khan, Inayat-Ur-Rehman, T. A. Khan, A. Akhunzada, and C. Naseeb, "A conceptual model to support the transmuters in acquiring the desired knowledge of a data scientist," *IEEE Access*, vol. 9, pp. 115335–115347, 2021.

[4] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, vol. 2. Springer, 2009.

[5] J. Demsar, T. Curk, A. Erjavec, C. Gorup, T. Hočevar, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starič, M. Štajdohar, L. Umek, L. Žagar, J. Žbontar, M. Žitnik, and B. Župan, "Orange: Data mining toolbox in Python," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 2349–2353, 2013.

[6] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel, "KNIME–the Konstanz information miner: Version 2.0 and beyond," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 26–31, Nov. 2009.

[7] A. Santos. (2023). *Home*. [Online]. Available: https://www.neuraldesigner.com/

[8] J. M. Wing, "The data life cycle," *Harvard Data Sci. Rev.*, vol. 1, no. 1, p. 6, 2019.

[9] A. X. Zhang, M. M´uller, and D. Wang, "How do data science workers collaborate? Roles, workflows, and tools," *Proc. ACM Hum.-Comput. Interact.*, vol. 4, no. CSCW1, pp. 1–23, May 2020.

[10] B. A. Myers, "Visual programming, programming by example, and program visualization: A taxonomy," *ACM SIGCHI Bull.*, vol. 17, no. 4, pp. 59–66, Apr. 1986.

[11] N. C. Shu, "Visual programming languages: A perspective and a dimensional analysis," in *Visual Languages*. Cham, Switzerland: Springer, 1986, pp. 11–34.

[12] N. C. Shu, "Visual programming: Perspectives and approaches," *IBM Syst. J.*, vol. 38, no. 2, pp. 199–221, 1999.

[13] M. A. Kuhail, S. Farooq, R. Hammad, and M. Bahja, "Characterizing visual programming approaches for end-user developers: A systematic review," *IEEE Access*, vol. 9, pp. 14181–14202, 2021.

[14] M. M. Burnett and M. J. Baker, "A classification system for visual programming languages," *J. Vis. Lang. Comput.*, vol. 5, no. 3, pp. 287–300, Sep. 1994.

[15] J. D. Kiper, E. Howard, and C. Ames, "Criteria for evaluation of visual programming languages," *J. Vis. Lang. Comput.*, vol. 8, no. 2, pp. 175–192, Apr. 1997.

[16] Z. Dobesova, "Evaluation of orange data mining software and examples for lecturing machine learning tasks in geoinformatics," in *Computer Applications in Engineering Education*. Hoboken, NJ, USA: Wiley, 2024.

[17] U. Thange, V. K. Shukla, R. Punhani, and W. Grobbelaar, "Analyzing COVID-19 dataset through data mining tool 'Orang,'" in *Proc. 2nd Int. Conf. Comput., Autom. Knowl. Manage. (ICCAKM)*, Jan. 2021, pp. 198–203.

[18] A. Abdelmagid and A. Qahmash, "Utilizing the educational data mining techniques," *Inf. Sci. Lett.*, vol. 12, no. 3, pp. 1415–1431, 2023.

[19] I. Popchev and D. Orozova, "Algorithms for machine learning with orange system," *Int. J. Online Biomed. Eng.*, vol. 19, no. 4, pp. 109–123, Apr. 2023.

[20] J. Demsar and B. Zupan, "From experimental machine learning to interactive data mining," in *Proc. Knowl. Discovery Databases*, 2005, pp. 537–539.

[21] B. Cukic, D. Hague, and M. Lou Maher, "An innovative interdisciplinary undergraduate data science program: Pathways and experience," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, Oct. 2020, pp. 1–5.

[22] D. Conway. (2010). *The Data Science Venn Diagram*. [Online]. Available: http://www.dataists.com/2010/09/the-data-science-venn-diagram

[23] J. W. Tukey, "The future of data analysis," *Ann. Math. Statist.*, vol. 33, no. 1, pp. 1–67, 1962.

[24] J. M. Chambers, "Greater or lesser statistics: A choice for future research," *Statist. Comput.*, vol. 3, no. 4, pp. 182–184, Dec. 1993.

[25] W. S. Cleveland, "Data science: An action plan for expanding the technical areas of the field of statistics," *Stat. Anal. Data Mining: ASA Data Sci. J.*, vol. 7, no. 6, pp. 414–417, Dec. 2014.
[26] S. C. Hicks and R. A. Irizarry, "A guide to teaching data science," *Amer. Statistician*, vol. 72, no. 4, pp. 382–391, 2018.
[27] D. Donoho, "50 years of data science," *J. Comput. Graph. Statist.*, vol. 26, no. 4, pp. 745–766, Oct. 2017.
[28] N. Corte-Real, P. Ruivo, T. Oliveira, and A. Popovic, "Unlocking the drivers of big data analytics value in firms," *J. Bus. Res.*, vol. 97, pp. 160–173, Apr. 2019.
[29] O. Hazzan and K. Mike, *Guide to Teaching Data Science: An Interdisciplinary Approach*. Springer, 2023.
[30] C. Kelleher and R. Pausch, "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers," *ACM Comput. Surveys*, vol. 37, no. 2, pp. 83–137, 2005.
[31] C. Kelleher and R. Pausch, "Using storytelling to motivate programming," *Commun. ACM*, vol. 50, no. 7, pp. 58–64, Jul. 2007.
[32] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The scratch programming language and environment," *ACM Trans. Comput. Educ.*, vol. 10, no. 4, pp. 1–15, Nov. 2010.
[33] *Scratch Statistics*. Accessed: Feb. 7, 2024. [Online]. Available: https://scratch.mit.edu/statistics/
[34] I. F. de Kereki, "Scratch: Applications in computer science 1," in *Proc. 38th Annu. Frontiers Educ. Conf.*, Oct. 2008, pp. 1–7.
[35] J. Estevez, G. Garate, and M. Graña, "Gentle introduction to artificial intelligence for high-school students using scratch," *IEEE Access*, vol. 7, pp. 179027–179036, 2019.
[36] R. J. Brunner and E. J. Kim, "Teaching data science," *Proc. Comput. Sci.*, vol. 80, pp. 1947–1956, Dec. 2016.
[37] H. Habibzadeh, K. Dinesh, O. Rajabi Shishvan, A. Boggio-Dandry, G. Sharma, and T. Soyata, "A survey of healthcare Internet of Things (HIoT): A clinical perspective," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 53–71, Jan. 2020.
[38] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE Access*, vol. 2, pp. 652–687, 2014.
[39] A. K. Pandey, A. I. Khan, Y. B. Abushark, Md. M. Alam, A. Agrawal, R. Kumar, and R. A. Khan, "Key issues in healthcare data integrity: Analysis and recommendations," *IEEE Access*, vol. 8, pp. 40612–40628, 2020.
[40] B. K. Daniel, "Big data and data science: A critical review of issues for educational research," *Brit. J. Educ. Technol.*, vol. 50, no. 1, pp. 101–113, Jan. 2019.
[41] X. Qin, Y. Luo, N. Tang, and G. Li, "DeepEye: An automatic big data visualization framework," *Big Data Mining Analytics*, vol. 1, no. 1, pp. 75–82, Mar. 2018.
[42] A. Perrot and D. Auber, "Cornac: Tackling huge graph visualization with big data infrastructure," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 80–92, Mar. 2020.
[43] M. Aparicio and C. J. Costa, "Data visualization," *Commun. Design Quart.*, vol. 3, no. 1, pp. 7–11, Jan. 2015, doi: 10.1145/2721882.2721883.
[44] R. Descartes, *The Philosophical Works of Descartes.[2 Vols.]*. Dover, 1955.
[45] E. Bisong, "Matplotlib and seaborn," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Cham, Switzerland: Springer, 2019, pp. 151–165.
[46] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 233–240.
[47] A. N. Shewalkar, "Comparison of RNN, LSTM and GRU on speech recognition data," *Comput. Sci. Masters Papers*, 2018.
[48] Y. Bai, J. Xie, C. Liu, Y. Tao, B. Zeng, and C. Li, "Regression modeling for enterprise electricity consumption: A comparison of recurrent neural network and its variants," *Int. J. Electr. Power Energy Syst.*, vol. 126, Mar. 2021, Art. no. 106612.
[49] K. H. Lycke, P. Grøttum, and H. I. Strømsø, "Student learning strategies, mental models and learning outcomes in problem-based and traditional curricula in medicine," *Med. Teacher*, vol. 28, no. 8, pp. 717–722, Jan. 2006.
[50] M. Khatiban, S. N. Falahan, R. Amini, A. Farahanchi, and A. Soltanian, "Lecture-based versus problem-based learning in ethics education among nursing students," *Nursing Ethics*, vol. 26, no. 6, pp. 1753–1764, Sep. 2019.
[51] L. D. Kantar and S. Sailian, "The effect of instruction on learning: Case based versus lecture based," *Teaching Learn. Nursing*, vol. 13, no. 4, pp. 207–211, Oct. 2018.
[52] B. Zhao and D. D. Potter, "Comparison of lecture-based learning vs discussion-based learning in undergraduate medical students," *J. Surgical Educ.*, vol. 73, no. 2, pp. 250–257, Mar. 2016.
[53] W. Hung, D. H. Jonassen, and R. Liu, "Problem-based learning," *Handbook Res. Educ. Commun. Technol.*, vol. 3, no. 1, pp. 485–506, 2008.
[54] M. A. Albanese and L. C. Dast, "Problem-based learning," in *Understanding Medical Education: Evidence, Theory and Practice*. Wiley, 2013, pp. 61–79.
[55] E. de Graaff and A. Kolmos, "Characteristics of problem-based learning," *Int. J. Eng. Educ.*, vol. 19, pp. 657–662, Jan. 2003.
[56] C. Onyon, "Problem-based learning: A review of the educational and psychological theory," *Clin. Teacher*, vol. 9, no. 1, pp. 22–26, Feb. 2012.
[57] D. Kokotsaki, V. Menzies, and A. Wiggins, "Project-based learning: A review of the literature," *Improving Schools*, vol. 19, no. 3, pp. 267–277, Nov. 2016.
[58] N. Hosseinzadeh and M. R. Hesamzadeh, "Application of project-based learning (PBL) to the teaching of electrical power systems engineering," *IEEE Trans. Educ.*, vol. 55, no. 4, pp. 495–501, Nov. 2012.
[59] B. Condliffe, "Project-based learning: A literature review. working paper," in *Proc. MDRC*, 2017, pp. 1–11.
[60] M. M. Grant, "Getting a grip on project-based learning: Theory, cases and recommendations," *Meridian, A Middle School Comput. Technol. J.*, vol. 5, no. 1, p. 83, 2002.
[61] J. S. Krajcik and P. C. Blumenfeld, *Project-Based Learning*. Cambridge Univ. Press, 2006.
[62] M. Saleh, M. Abbas, and R. B. Le Jeannès, "FallAllD: An open dataset of human falls and activities of daily living for classical and deep learning applications," *IEEE Sensors J.*, vol. 21, no. 2, pp. 1849–1858, Jan. 2021.
[63] S. C. dos Santos, "PBL-SEE: An authentic assessment model for PBL-based software engineering education," *IEEE Trans. Educ.*, vol. 60, no. 2, pp. 120–126, May 2017.
[64] P. Abichandani, V. Sivakumar, D. Lobo, C. Iaboni, and P. Shekhar, "Internet-of-Things curriculum, pedagogy, and assessment for STEM education: A review of literature," *IEEE Access*, vol. 10, pp. 38351–38369, 2022.
[65] G. V. Helden, V. Van Der Werf, G. N. Saunders-Smits, and M. M. Specht, "The use of digital peer assessment in higher education—An umbrella review of literature," *IEEE Access*, vol. 11, pp. 22948–22960, 2023.
[66] H.-P. Yueh, T.-L. Chen, L.-A. Chiu, S.-L. Lee, and A.-B. Wang, "Student evaluation of teaching effectiveness of a nationwide innovative education program on image display technology," *IEEE Trans. Educ.*, vol. 55, no. 3, pp. 365–369, Aug. 2012.
[67] M. Llamas-Nistal, F. A. Mikic-Fonte, M. Caeiro-Rodríguez, and M. Liz-Domínguez, "Supporting intensive continuous assessment with BeA in a flipped classroom experience," *IEEE Access*, vol. 7, pp. 150022–150036, 2019.
[68] J. Moreno and A. F. Pineda, "A framework for automated formative assessment in mathematics courses," *IEEE Access*, vol. 8, pp. 30152–30159, 2020.
[69] S. Draft, *Computer Science Curricula*. New York, NY, USA: ACM, 2013.
[70] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz, "A multi-national, multi-institutional study of assessment of programming skills of first-year cs students," in *Working Group Reports From ITiCSE on Innovation and Technology in Computer Science Education*. Association for Computing Machinery, 2001, pp. 125–180.
[71] A. Joshi, S. Kale, S. Chandel, and D. Pal, "Likert scale: Explored and explained," *Brit. J. Appl. Sci. Technol.*, vol. 7, no. 4, pp. 396–403, Jan. 2015.
[72] R. Parhi and R. D. Nowak, "The role of neural network activation functions," *IEEE Signal Process. Lett.*, vol. 27, pp. 1779–1783, 2020.

HARRY D. MAFUKIDZE received the B.Sc. degree (Hons.) in physics from Midlands State University, Gweru, Zimbabwe, in 2009, and the M.Eng. degree in electronic engineering from the University of Stellenbosch, Stellenbosch, South Africa, in 2014. He is currently with the Department of Applied Physics and Telecommunications, Midlands State University. His research interests include radar signal processing, data science, machine learning, and deep learning and their applications.

ACTION NECHIBVUTE received the B.Sc. degree in physics from Midlands State University, in 2001, the B.Sc. degree in mathematics from the University of Zimbabwe, in 2001, the M.Sc. degree in physics from the University of Botswana, in 2008, and the Ph.D. degree in physics in the area of energy harvesting for wireless sensor devices from Midlands State University, in 2015. He is currently an Academic Researcher with Midlands State University.

ABID YAHYA (Senior Member, IEEE) received the bachelor's degree in electrical and electronic engineering, major in telecommunication from the University of Engineering and Technology Peshawar, Peshawar, Pakistan, and the M.Sc. and Ph.D. degrees in wireless and mobile systems from Universiti Sains Malaysia. He began the career on an engineering path, which is rare among other researcher executives. He is currently with Botswana International University of Science and Technology. He is also a Professional Engineer certified by Botswana Engineers Registration Board (ERB). He has many research publications in numerous reputable journals, conference articles, and book chapters. He received several awards and grants from various funding agencies and supervised several master's and Ph.D. candidates. His recent four books *Emerging Technologies in Agriculture, Livestock, and Climate* (Springer, 2020), *Mobile WiMAX Systems: Performance Analysis of Fractional Frequency Reuse* (CRC Press|Taylor & Francis, 2019), *Steganography Techniques for Digital Images*, *LTE-A Cellular Networks: Multi-hop Relay for Coverage, Capacity and Performance Enhancement* (Springer International Publishing, July 2018 and January 2017), and are being followed in national and international universities.

IRFAN ANJUM BADRUDDIN received the Graduate degree in mechanical engineering, in 1998, the Master of Technology degree, in 2001, and the Ph.D. degree in heat transfer from Universiti Sains Malaysia, in 2007. He is currently a Professor with the Department of Mechanical Engineering, King Khalid University, Saudi Arabia. He works in the interdisciplinary fields. He has more than 300 articles to his credit.

SARFARAZ KAMANGAR received the Ph.D. degree in mechanical engineering. He is currently an Assistant Professor with the Department of Mechanical Engineering, King Khalid University, Saudi Arabia. He has more than 13 years of research and teaching experience at well-known universities. He has published more than 100 articles at international journals and conferences.

MOHAMED HUSSIEN was born in Menofia, Egypt, in 1981. He received the B.Sc. degree in chemistry and the Ph.D. degree in organic chemistry from Menofia University, in 2002 and 2023, respectively. He is currently a Lecturer in organic chemistry with the Department of Chemistry, Faculty of Science, King Khalid University, Abha, Saudi Arabia. He has many research articles in heterocyclic chemistry against pests and using as anticancer agents published in international journals. His research interests include synthesis and chemical reactivity of phosphorus compounds contain bioactive heterocyclic systems.