applied sciences

MDPI

Article

Performance Analysis of Selected Programming Languages in the Context of Supporting Decision-Making Processes for Industry 4.0

Paweł Dymora * and Andrzej Paszkiewicz

Department of Complex Systems, The Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, 35-959 Rzeszów, Poland; andrzejp@prz.edu.pl

* Correspondence: Pawel.Dymora@prz.edu.pl

Received: 12 October 2020; Accepted: 26 November 2020; Published: 28 November 2020

Abstract: This study analyzes the possibility of using Go (Golang) in the context of Java and Python in decision-making processes, with particular emphasis on their use in industry-specific solutions for Industry 4.0. The authors intentionally compared Go with Java and Python, which have been widely used for many years for data analysis in many areas. The research work was based on decision trees data mining algorithms, and especially on classification trees, in which the measure of entropy as a heuristics to choose an attribute was taken into account. The tests were carried out on various parameters describing calculation time, RAM usage, and CPU usage. The source data, which were the basis for the computing of the decision tree algorithm implemented using these three languages, were obtained from a commercial remote prototyping system and were related to the target customers' choice of methods and means of the full design-creation process.

Keywords: go programming language; Industry 4.0; rapid prototyping; decision support; data mining; decision tree; entropy; information gain

1. Introduction

Present-day economic development is inevitably connected with the widespread implementation of the Industry 4.0 concept [1–3]. Automation and informatization of manufacturing processes have been progressing for many years [4–6], but there is still much to be done in this domain. The processes of planning, designing, optimization, manufacturing, quality control, and distribution are constantly changing, which requires constant analysis and adjustment to the current needs and requirements set by both customers and suppliers, as well as legal and technological conditions. Digitization of these processes connected with their integration allows for real implementation of the Industry 4.0 concept [7–10]. Such activities cover many areas of business operation and are usually carried out gradually. Only the establishment of entirely new factories or their full modernization creates sufficient conditions for such changes [11,12]. Therefore, the digitalization processes of factories in terms of Industry 4.0 can be realized in a controlled or uncontrolled way. In the first case, we are dealing with a planned and well-thought-out, and consistently implemented plan for automation and computerization of all technological and commercial processes, human resources, and financial processes. In the second case, however, there is an accidental, inconsistent, and in many cases, chaotic set of actions aimed at making individual changes. Therefore, as already mentioned, Industry 4.0 requires activities related to technological changes (including IoT (Internet of Things)) [13], thinking (approach to management) [14], processing (also in terms of IoE—Internet of Everything) [15], and even in the future legal changes [16]. All these areas are combined by two main aspects, namely the acquisition, processing, and analysis of large-scale data sets (often of a Big Data nature) [17–19] and the process approach [20].

In order to make the concept of Industry 4.0 a reality, it is necessary to use adequate programming tools and languages. Their purpose and role are very different, as the areas of their application are different. The structure of the framework and smart factories' architecture includes the physical layer, the communication layer, the service layer, as well as the application and analysis layer [21]. Each of these layers is associated with certain limitations and requirements, e.g., obtaining and controlling hardware resources. Therefore, we cannot be limited to only one tool or one programming language [22]. The physical layer that includes the regulation, inspection, and monitoring of manufacturing equipment, but also detectors and sensors, often requires the use of low-level languages such as Assembler [23], but C and Java [24,25] are also often used. Both Assembler and C languages have, for many years, been chosen as the primary programming languages closest to the hardware, mostly device drivers, low-level embedded systems, and real-time systems, as well as microcontrollers commonly used in the industry [26,27]. In the communication layer, C, C++, as well as Python and Java are used, among others. It should be noted that these languages refer to the creation of network services provided in different layers of the ISO/OSI model. The functionality at the second and third layer level for specialized SDN (software-defined networks) [28,29] can also be programmed using dedicated programming languages such as Frenetic, Procera, Pyretic, Kinetic, and NetCore [30–33]. Another area of implementation of programming languages in Industry 4.0 is the service layer, including, e.g., creation of computing clouds [34,35], dedicated applications for simulation and modeling [36], production process control [37], etc. In this area, popular programming languages such as Java, JavaScript, PHP, C, C++, C#, or Python are used. The last functional area, which is currently gaining in importance and increasing the efficiency of design, manufacturing, control, and management processes, includes the use of mechanisms and solutions in Big Data, machine learning, artificial intelligence, and expert systems. The whole area requires the implementation of effective algorithms supporting decision-making processes. In this area, modeling and programming languages such as UML (Unified Modeling Language) [38,39], various variations of the C language [40,41], and, for some time now, more and more often Java and Python [42,43] were used. Therefore, in this article, the authors have decided to refer to the possibility of using the relatively new Go language in the decision-making process in relation to Java and Python.

Of course, this study is not an overview article. It does not include an analysis of the extensive range of performance of programming languages in the context of constantly evolving software engineering, as well as the needs of different sectors of the economy and the needs of potential customers. It should also be noted that in the past few decades of IT development, very many programming languages have been developed. However, few of them have gained satisfactory popularity, expressed by the number of people using them and applications created on their basis. Analyzing current trends, it can be concluded that in the area of data analysis, machine learning, and artificial intelligence, Java and Python have great use and interest among users. Therefore, the authors have attempted to compare them with the relatively new and increasingly commonly used Go (Golang) language in decision-making processes in the context of solutions used in Industry 4.0.

Analyzing the needs of the area related to Industry 4.0, the authors pointed out the need to create programs characterized by high operation speed and low load on hardware resources. This state of the art results from the fact that many IT and automation systems supporting design and production processes have the character of closed systems with little possibility of expansion and modernization. Of course, considering the extensive range of application areas of this type of scenario in the industry, the focus was on a selected aspect. Namely, systems supporting decision-making processes in remote rapid prototyping systems based on decision trees take into account the measure of entropy as a heuristics to choose an attribute. The authors are aware of the fact that there are many methods and means, as well as mathematical tools and their implementation, supporting decision making processes. Since the authors take part in research and development work related to remote prototyping systems (the system is called iS Rapid, developed by InfoSoftware Polska Ltd., Rzeszów, Poland), they have relied on data from such a system. However, it should be noted that the results concerning the performance of individual programming languages are more universal and can potentially be used

in the future to implement solutions. Mostly solutions supporting decision-making processes in the industrial infrastructure regarding the selection of tools for individual manufacturing tasks, the type, composition of manufacturing materials, and the choice of control and verification mechanisms for the design and production process. Indeed, there are many well-described and repeatedly verified algorithms for analyzing and exploring data used in the decision making process. There are also many publications about their effectiveness and possibilities [44,45]. Therefore, in this publication, we focus on tools, mostly programming languages used for algorithmic and application implementation of these algorithms. Individual languages' properties may also improve the speed and, particularly, the data analysis process's computational complexity. In this respect, the programming solutions used have an essential role to play. Their efficiency and reliability determine the future design, manufacturing, and business systems' effectiveness in many cases. Hence, research is continuously being conducted on IT systems' efficiency [22,23] and their industrial applications [21,23,25]. This area is extensive, but the authors of this publication focused on selected programming languages such as Go, Java, and Python, which have great potential for their application, among other areas, in data analysis. Therefore, this paper aims to analyze the languages mentioned above in the context of data analysis efficiency based on decision trees: for example, application in industry, especially for the remote realization of whole design and production process such as rapid prototyping. The results obtained can guide the future use of the analyzed programming languages, including Golang in the area of Industry 4.0 and IoT elements.

The rest of the paper is organized as follows. Section 2 is a reference to works on the performance of programming and the methods and means of data analysis. Section 3 refers to implementing the decision tree mechanism for the study of the customer's selection and further details of the design and production process on the example of a commercial system supporting rapid remote prototyping and measures used in decision trees that are the basis of the presented algorithm. These assumptions and the data obtained in this way formed the foundation for research work on the validation of capabilities, and the effectiveness of the selected programming languages based on the implementation of the decision tree algorithm in their use in decision-making processes dedicated to use in Industry 4.0. The work ends with a summary of the obtained research results and conclusions.

2. Previous Works

The use of computer systems and, more widely, information systems in all areas of social, economic, and cultural life has also resulted in developing the programming area as a whole. This involves developing new programming languages and developing methods and means of software development (including various methodologies). Designing and then creating specialized software such as applications dedicated to the industry requires considering many factors and conditions related to its final purpose. These include functional aspects, but also technological, safety, efficiency, and reliability [46,47]. There are different classifications of programming languages [48]. One of them is the division into the machine, symbolic and problem-oriented. The second example is a classification covering first, second, third, fourth, and fifth-generation languages. Of course, there are also other classifications, but this work focuses on three selected languages and their analysis in the context of potential applications for industry 4.0. As part of the research, three languages were chosen for study: Java, Python, and Go. The choice of languages compared with Go was not accidental. These are languages with high application potential in areas such as data analysis, machine learning and artificial intelligence, networking, and industrial applications [49–56].

The importance of the performance of programming languages has a long history in software engineering. Individual languages are compared based on sets of basic instructions and benchmark programs [43,57–60]. One of the basic approaches used to compare and evaluate programming languages is to determine the overall performance indicator, which is calculated by averaging individual performance indicators obtained from individual tests [61]. Another way to compare programming languages is to refer to their characteristics in the form of code density, the processing

efficiency of selected data structures, e.g., tables, recursion efficiency, speed of creating dynamic structures, etc. [62–66]. The overall comparison of more than ten of the most popular programming languages from the point of view of design goals, syntax, semantics, and features was conducted as early as 1995 [66]. In the literature, we can also find examples of publications related to comparing specific groups of programming languages, such as object-oriented languages [67].

An interesting example is also the comparison of existing software metric tools [68]. In this publication, the authors present how different tools interpret and implement the definitions of object-oriented software metrics in different ways, directly impacting the results of analyses based on these metrics. Due to the development of the IoT (Internet of Things), it has been necessary to create also appropriate middleware for it to enable communication between devices using different standards and technologies. The paper [69] presents a typical case of using middleware in three other programming languages, followed by a comparative evaluation and analysis of the results. As a result of these studies, it was confirmed that, e.g., JavaScript presents better performance for small and medium applications, while worldwide Java is proving to be the best choice for IoT middleware. Considering the challenges of Industry 4.0 today, the challenges of the development of methods and measures supporting data acquisition processes, searching and analysis, then supporting the processes of reasoning and decision-making [70], an important aspect is the analysis of programming languages in terms of their effectiveness in these areas.

The analysis of data and the extraction of knowledge from it require adequate methods and means. The area that is responsible for this is data mining [71,72]. It meets the need for effective and rational use of the knowledge accumulated in these data to achieve specific goals. Thus, it deals with analyzing and discovering relationships, rules, and patterns in databases and knowledge bases. Very often, it concerns nontrivial, unknown relationships, similarities, or trends. It includes a set of different methods that support these processes. These include classification/regression, clustering, sequence discovery, characterization discovery, time waveform analysis, association discovery, change and deviation detection, web exploration, and text exploration [73]. Within these methods, we can distinguish various techniques such as statistics, neural networks, machine learning, evolutionary algorithms, fuzzy logic, and approximate sets [74–76]. These also include decision trees [77]. When we collect large sets of data in different forms, categorizing them and then formulating conclusions from them is a big challenge in many cases. Therefore, to solve, among others, problems of classification, information selection, or finding rules (learning), the decision trees are used [78]. In the simplest possible scenario, one can refer to the process of choosing between possible events based on the probability of their occurrence, their consequences, profits, and risks generated by a given event. From the structural point of view, the decision trees are an example of a multi-level decision-making process. In this process, instead of using a broad set of arguments describing a given phenomenon's characteristics, the decision is divided into many levels. Only specific subsets of these characteristics are considered.

Therefore, decision trees are widely used classifiers in data mining to predict a given input variable's target variable. They make it possible to indicate both a category variable (classification) and a continuous variable (regression) [79]. In regression trees, the division point is selected. There is the highest RSS (Residual Sum of Sequence) reduction, or the function's standard deviation reduction for training data is calculated. In the case of classification trees, entropy, and the information gain factor or Gini coefficient [80,81], is used. By using entropy, the group's uncertainty is determined and, consequently, the entropy reduction is achieved by selecting the appropriate attributes. In decision trees, the decision process starts in the root and goes towards the leaves based on the decisions made in the individual nodes. The decision process ends when the stop criterion is met.

Therefore, decision trees are the main tool to support decision processes that use the tree structure, as well as the possible consequences of individual decisions. Such an approach is already applied in many areas of industry, from marketing analysis [82], related to commercial research [83], to production and management processes [84]. However, in this paper, the authors have focused on analyzing the potential use of programming languages in decision tree-based decision-making processes.

3. Implementation of the Decision Tree Algorithm

The Industry 4.0 concept's evolution inevitably involves implementing new solutions based on IT systems for the entire design and production process. These systems often enable cooperation between different remote units with dispersed resources such as scientific units, companies, or production clusters. An essential role in implementing this concept is played by rapid prototyping, which allows us to create and develop entirely new ideas and modernize the existing ones. This form of production, based on incremental technologies, can be used both in the area of multi-series design, as well as in short-run production, or single elements, components, and prototypes. Considering various limitations resulting from the dispersion of design and production resources and crises, e.g., epidemics, the process of remote design and production can play a significant role.

In cooperation with the Rzeszów University of Technology's scientific community, such a system was created in InfoSoftware Polska Ltd., Rzeszów, Poland. This system is called iS Rapid and enables the remote realization of the whole design and production process. This process consists of three essential areas: design and modeling, manufacturing, and quality control. Individual works within these areas can be realized in different ways. The CMMN model (case management model and notation) of rapid prototyping process scenarios is presented in Figure 1. In the design and modeling process within the iS Rapid system, one can distinguish the main activities of the main process: independent preparation of the model for 3D printing by the customer at his premises, development of such a model in a virtual environment available within the iS Rapid platform, as well as commissioning such a design task to specialized staff within the iS Rapid system. In the next stage of the design and production process (related to the module *E. Production method specification* in the CMMN diagram, actually being a choice of 3D printing method), it is possible to create the target component within various incremental technologies: FFF (Fused Filament Fabrication), DLP (Digital Light Processing), SLA (Stereolithography), SLS (Selective Laser Sintering), PolyJet (Polymer Jetting), and CJP (ColorJet Printing). Of course, the choice of individual technologies is also connected with the possibility of choosing the right materials: ABS, PLA, PCABS, PET, TPU, Nylon, HIPS, PVA, PMM, light-curable resins, finely powdered metal alloy, polymer, ceramic powders, etc. On the other hand, in the case of the quality control stage (associated with the *H. Quality Control* in the CMMN diagram), the customer has the possibility of remote surveillance using high-resolution cameras (standalone), use of a specialist engineer to assess the entire physical manufacturing process, or to assess the condition of an already manufactured component (specialist). There is a possibility of giving up such an option in the design-creation process (none).

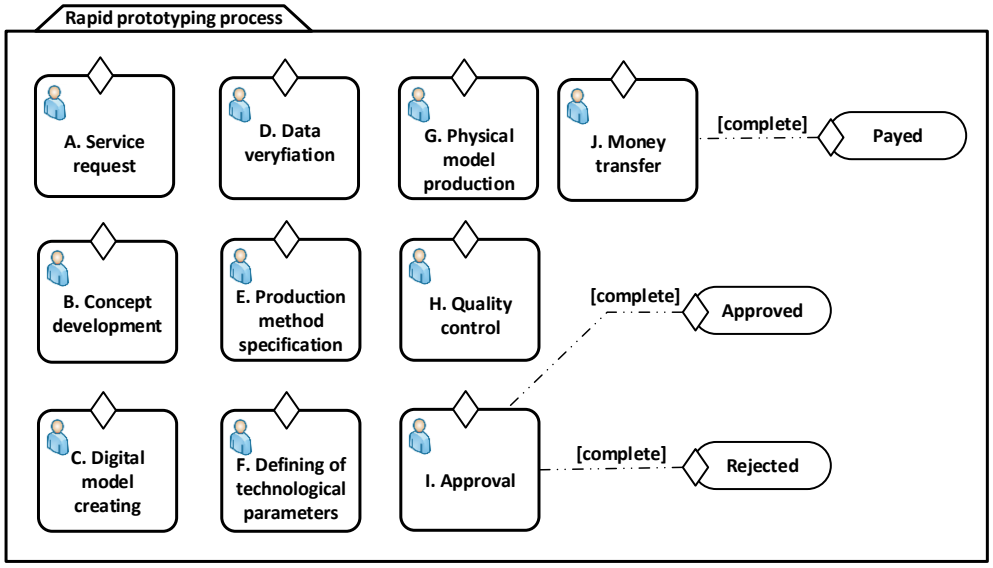

Figure 1. Case management model and notation (CMMN) model of the rapid prototyping process.

This article examines how customers choose particular options at different stages of the design and production process and the consequence of the final decision related to purchasing such a remote prototyping service. Of course, such variables as design variant, manufacturing technology, and quality control variant are exploratory variables, and the purchase is a target variable. Table 1 shows sample cases—orders registered in the system. One can notice that case no. 1 was realized in its entirety. The customer ordered all service elements, including quality control, from a specialist available in the company. In the second case, however, we can see that the service was purchased (fully paid), but the service does not include, as one can see, quality control.

Table 1. Sample Event Log registered for two cases.

Case ID	Event ID	dd-MM-yyyy-HH:mm	Activity	Resource	Costs
1	736286	03.01.2020-09:15	A. Service request	Consultant	50
1	736288	13.01.2020-13:45	B. Concept development	Consultant	50
1	736289	14.01.2020-8:33	C. Digital model creating	Engineer	100
1	736290	15.01.2020-14:12	D. Data verification	Analyst	200
1	736291	15.01.2020-15:21	E. Production method specification	Analyst	100
1	736292	16.01.2020-11:37	F. Defining technological parameters	Analyst	100
1	736293	16.01.2020-8:17	G. Physical model production	Engineer	200
1	736295	20.01.2020-10:27	H. Quality control	Specialist	200
1	736296	21.01.2020-14:41	I. Approval	Manager	150
1	736297	24.01.2020-15:13	J. Money transfer	Manager	150
2	736311	06.02.2020-8:35	A. Service request	Consultant	50
2	736312	06.02.2020-10:12	B. Concept development	Consultant	50
2	736313	11.02.2020-10:25	C. Digital model creating	Engineer	100
2	736314	13.02.2020-11:01	D. Data verification	Analyst	200
2	736315	14.02.2020-13:47	E. Production method specification	Analyst	100
2	736316	17.02.2020-8:42	F. Defining technological parameters	Analyst	100
2	736317	17.02.2020-10:28	G. Physical model production	Engineer	200
2	736318	18.02.2020-8:32	I. Approval	Manager	150
2	736319	19.02.2020-13:46	J. Money transfer	Manager	150

After a detailed analysis of process data and event log (part of it presented in Table 1), records can be extracted to determine individual customers' preferences in relation to individual choices at the conceptual stage, software processing stage, and rapid prototyping processing stage. A fragment of the consolidated data concerning the customers' preferences is presented in Table 2. In particular, one can make a detailed specification for individual cases for 3D printing method, Quality control type, and order status: bought or not (paid or not). It is worth emphasizing that the number of possible combinations of all decision-making attributes may be significant within the service.

Table 2. Sample customer preferences.

Case ID	Designing Method (E)	3D Printing Method (E)	Quality Control (H)	Money Transfer (J)
Case 1	Client	FFF (ABS)	Camera	No
Case 2	Virtual resources	FFF (ABS)	Camera	No
Case 3	Virtual resources	FFF (ABS)	Specialist	Yes
Case 4	Virtual resources	FFF (PLA)	Specialists	No
Case 5	Virtual resources	FFF (ABS)	Specialist	Yes
Case 6	System	FFF (PLA)	Camera	Yes
Case 7	Client	FFF (PLA)	Camera	No
Case 8	System	FFF (ABS)	Camera	Yes
Case 9	Virtual resources	FFF (PLA)	Camera	Yes
Case 10	Client	FFF (ABS)	Specialist	Yes

The database used to perform predictions and calculations was a database containing the preferences of individual customers. It was a database consisting of 10 thousand, 100 thousand, and half a million records, respectively. Based on these preferences, the predictions were carried out, and the results were saved to a CSV file. This database was a source of information for the decision tree algorithm, the scheme of which was presented on Algorithm 1.

Algorithm 1. Decision Tree Algorithm.

Input: An attribute-valued dataset D
Output: Decision tree T
$T = \{\}$
Start at the root node
Calculate the entropy of the whole data set
For each attribute $a \in D$:
- divide input data D into subsets
- calculate the entropy for each subset
- select these attributes for which the entropy variation is the highest. This entropy variation is measured by **IG** (Information Gain)
- group data along selected branches
- find attributes to dividing the data set in the nodes of the next level
- set the decision in the node
End For
If a stopping criterion is reached **Then**
Return decision tree T
Otherwise, terminate.

Measures Used in Decision Trees

Algorithms to construct the decision tree, when selecting the division for each node, solve the problem of maximization. It is based on searching all possible attribute values in order to find the best division, i.e., the highest value of the diversity measure. Usually, the criteria for diversity measures are based on entropy and information gain or Gini coefficient, among others.

In 1948, the American mathematician Claude E. Shannon introduced the entropy concept related to thermodynamics' second principle into communication systems [85]. It determines the direction of spontaneous (self-responsive) processes in an isolated thermodynamic system. Entropy is a measure of the degree of system disorder and energy dispersion [86]. In terms of communication systems, it has been applied in the area of analysis of transmitted information. Thus, from this perspective, entropy means the average amount of information per single message transmitted from an information source. In order to measure the degree of randomness of a given set of events, the concept of statistical entropy was introduced. Assuming that Z is a set of separable events $\left\{A_{1,}, \ldots, A_n\right\}$ comprehensively covering the elementary event space, where the probability $P(A_1, \ldots, A_n)$ is 1, the equation can determine the entropy of the Z set:

$$H(Z) = -\sum_{i=1}^{n} P(A_i) log_2 P(A_i). \quad (1)$$

In general, entropy can also be used to measure the degree of randomness of the X variable. Then entropy of a random X variable of a discrete type is described by the formula:

$$H(X) = -\sum_{i=1}^{n} P_i log_2 P_i, \quad (2)$$

where $P_i = P(X = x_i)$ denotes the probability that a random variable X takes the value x_i, and n is the number of values of a random variable X. Shannon's entropy, therefore, determines the average amount of information related to the probability distribution of a random variable X. The measure of entropy defined by C.E. Shannon has been applied in many fields of science, such as statistics and computer science.

Classification trees use the measure of entropy as heuristics to select an attribute. Given the assumption that entropy determines uncertainty, even chaos, a higher value of entropy in a given set means a higher disturbance. Figure 2 shows that entropy (uncertainty) is greatest when the probability is 0.5. This means that entropy is maximum when there is the same number of objects from different class attributes. On the other hand, entropy is lowest when the probability is 0 or 1 (there is no uncertainty or high probability), i.e., entropy is minimal if the node is blank, i.e., it is homogeneous

from the point of view of diversity for a given attribute. Ultimately, a minimum entropy for the tree is desirable, i.e., the nodes' classes are blank or uniform.

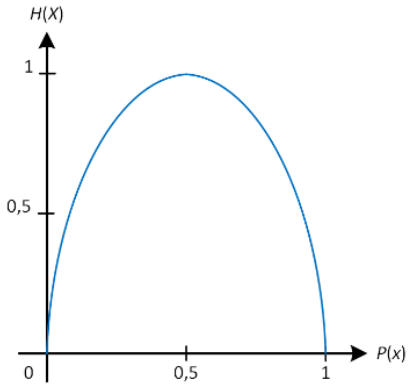

Figure 2. The value of entropy for a given set.

Therefore, by dividing the tree, the attributes that achieve the most significant reduction in entropy are selected in the classification tree. This reduction (or change) is measured by the information increment [78]:

$$IG(S, A) = H(S) - \sum_v \frac{|S_v|}{|S|} H(S_v), \quad (3)$$

where A determines the attribute with a given value, H is the entropy of the data set, S is the data set, and S_v is the data set reduced by vectors with a different value of the A attribute.

4. Results of the Research Work

Systems used in Industry 4.0 usually have the character of embedded systems, with limited interference in their structure, including in the area of memory expansion and processor exchange. However, IT systems are always based on the use of the same resources: processor (CPU), memory (RAM), input/output devices (I/O) e.g., HDD (Hard Drive), Internet network. Therefore, the first point that should always be monitored regardless of the project's subject matter is the resources. Thus, optimization of the code, e.g., in terms of required memory, and processor load, is a significant issue. Of course, server systems, especially cloud computing, offer great possibilities of scalability of necessary resources, but such solutions in the area of industry can be used only to a limited extent. Therefore, among other things, no tests have been conducted in a high-performance server environment where powerful processors are available, often with dedicated multi-level memory chips designed to increase processing efficiency. On the other hand, solutions dedicated to industrial and IoT applications usually use widely available memory chips and universal processors. Considering dedicated software, it can be said that these systems very often have a non-deterministic or temporary deterministic character. Thus, there are solutions in which the system's hardware part accelerates the selected range of activities performed by the system (for example, in terms of switching tasks and preserving their context). However, in such cases, it would be necessary to carry out tests focused only on specific hardware and software platforms without the possibility of a broader generalization of the obtained results. Therefore, the choice of classical computer hardware seemed to be the closest to the adopted assumptions. Taking into account the above assumptions, and after the settlements with InfoSoftware Polska Ltd. regarding the use of a hardware platform that meets the requirements of a dedicated module within the iS Rapid system, an environment with the following configuration was selected for testing: Intel® Core™ i7-7700 CPU @3.60 GHZ, 16 GB RAM, and 64 bit operating system, Windows Server 2016. Taking into account the modular architecture of the system, its individual modules can be processed both in a centralized and distributed environment. This ensures high scalability and the ability to adapt hardware platforms to current needs.

To verify the capabilities of selected programming languages based on the decision tree algorithm implementation's effectiveness, data received from InfoSoftware Poland Ltd. were used and multiplied to obtain 10 thousand, 100 thousand, and 500 thousand records. Simultaneously, the proportional distribution of individual values was maintained according to the iS Rapid system's actual data.

The computation time required to predict the target variable is taken into account; in the case of a database containing 10 thousand records, the best results were obtained for Java. A big surprise was receiving much longer computation time for the Python language. The Golang language was, in this case, in place 2. It should be noted that entirely different results were obtained for more massive datasets. Along with the increase in the number of data, the Python language results improved significantly concerning competing languages. In a situation where the database contained half a million records, Python did best, then Golang, and Java did worst. In the case of 100 records, the differences between the individual languages were the smallest. To verify the results obtained, we made several series of computations. Due to the fact that the results of these series are similar or even identical, their average values are given. The obtained results, presented in Table 3 and Figures 3–5, confirm a certain level of results obtained for particular programming languages and data set sizes. The obtained results demonstrate that in the case of using software for the current analysis of large data sets collected, e.g., in production processes, it is recommended to use Python and then the Golang language. In the case of small data sets, possible differences may be less significant. In addition, Java is optimized to maximize application throughput (e.g., the number of processed user requests sent to the webserver, amount of processed data, etc.) High throughput results in more increased delays. Golang, for example, is optimized for low latency from the beginning. While the bandwidth would be lower, user requests would be processed faster. This makes Golang the ideal solution for production systems that often require quick decisions and reactions to changing working conditions.

Table 3. Comparison of average values for the decision tree algorithm implementation.

	Go	Java	Python	Go	Java	Python	Go	Java	Python
Measures		**10, 000**			**100, 000**			**500, 000**	
Computation time [s]	1.15	0.567	2.6	29.47	20.037	21.27	328.35	436.29	114.08
RAM memory usage [MB]	18.54	261.02	142	75.38	857.46	168.7	411.38	1463.86	314.48
CPU usage [%]	7.8	16.84	24.36	25.74	23.54	25.47	49.98	40.6	27.5
Normalized computation time [%]	0.009	0.01	0.07	0.79	0.49	0.57	17.20	18.57	3.29
Normalized CPU usage [%]	9.33	20.79	31.5	30.82	29.07	32.94	59.84	50.13	35.56

In the case of RAM requirements, the worst results in the whole ranking were achieved by Java. For more massive datasets, both Golang and Python scored well. One can say that both languages are comparable in this aspect. On the other hand, a significant difference in favor of Golang was obtained for the smallest number of records (10 thousand). The results presented in Table 3 and Figure 4 show that the Golang language can be used in embedded IoT systems also used in Industry 4.0, where the effective use of RAM should be taken into account (e.g., when it is not possible to expand these systems in this aspect). Very often, IoT systems have generally limited resources, including power, memory, CPU power, etc. Usually, these are not systems that can be easily scalable, and in many cases, this is even impossible. Therefore, individual programming languages' ability to make more efficient use of, e.g., RAM can contribute to their application potential. The good results obtained for the Golang language result from the GC (garbage collector) implemented in it. This mechanism is designed to remove useless elements and thus free up memory. This process takes place simultaneously in the case of the Go language. Therefore the released memory is ready for reuse. Of course, such actions also have their costs, e.g., in the form of additional CPU load. Another feature of Golang related to memory management is memory allocation using different memory structures with varying cache levels for objects of various sizes. Splitting a single block of consecutive addresses received from the operating system into a multi-level cache improves memory allocation efficiency by reducing blockages. Then,

allocating the memory allocation according to a specific size reduces memory fragmentation and facilitates a faster GC after memory release.

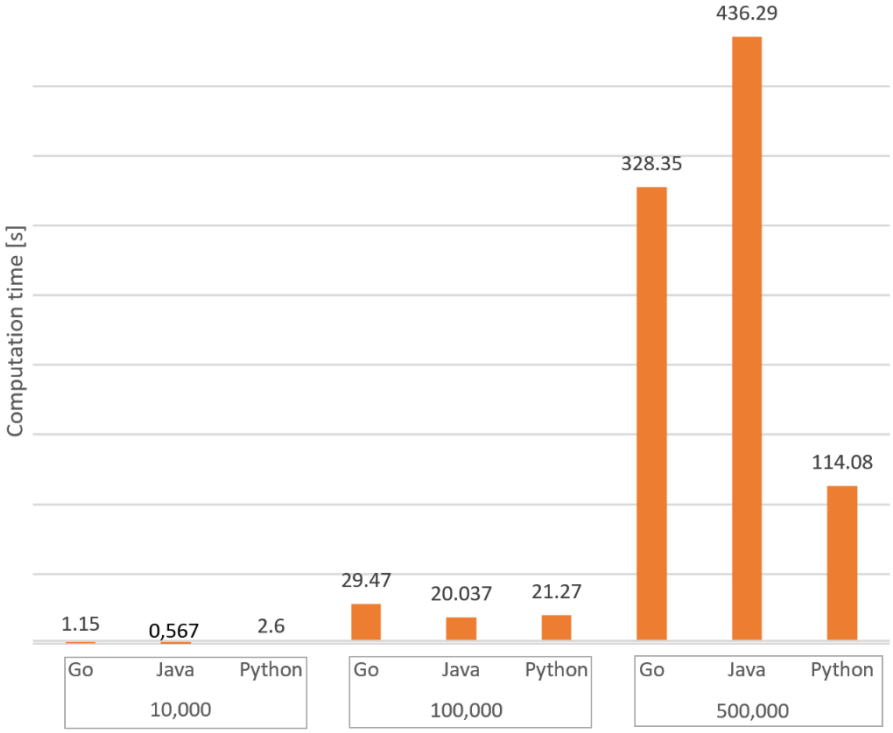

Figure 3. Comparison of computation time(s) for the decision tree algorithm implementation.

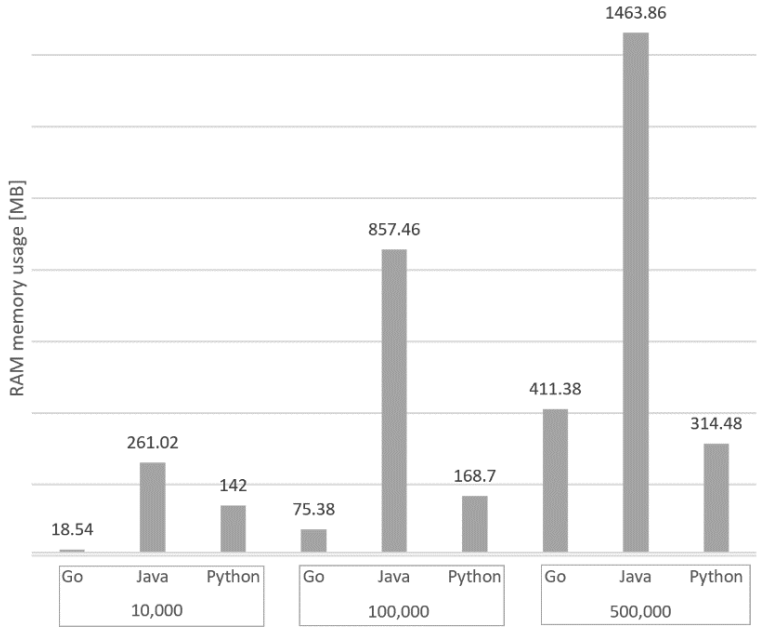

Figure 4. Comparison of RAM (MB) memory usage for the decision tree algorithm implementation.

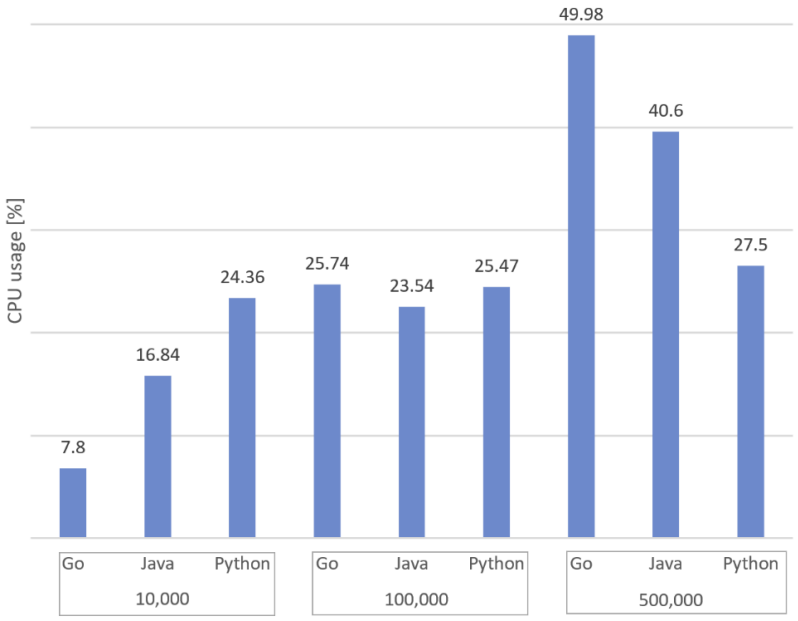

Figure 5. Comparison of CPU usage (%) for the decision tree algorithm implementation.

In Table 3 and Figure 5, the use of the CPU is presented. As one can see, the results are very different. In the case of a smaller number of data, the Golang language was the best in this list. On the other hand, when the number of records reached 500 thousand, the same language was the most loaded on the CPU. The vital information is that in the Python language, regardless of the number of data, the CPU usage was similar.

Obtained results depend on the properties of individual programming languages. Go language manages the memory via the garbage collector. This mechanism eliminates useless elements from memory and is based on giving priority to consistently low periodical delays at the expense of higher CPU usage. Such mechanisms can be of great importance, especially for solutions dedicated to Industry 4.0 and IoT. In the indicated areas, dedicated solutions are very often implemented, which have limited memory resources. Their expansion is difficult and, in many cases, impossile because it would involve replacing whole systems and sometimes even entire devices. Thus, the Go language garbage collector supports programmers by automatically releasing their programs' memory when it is no longer needed. However, tracking and cleaning the memory requires additional resources such as CPU time. The effect of this can be seen in Figure 5. Of course, the scope of optimizing the garbage collector to increase the efficiency of available resources is not the subject of this article.

As one can see in Table 3, there are also provided standardized measures of the two most essential parameters, CPU usage and computation time. Computer systems may have different processing speeds due to various hardware components. CPU standardization allows one to take into account the actual CPU speed instead of relying on the CPU clock frequency, which can be deceptive for different manufacturers and models. Calculation of the normalized values of the active jobs' computation time starts by obtaining the percent of the CPU time consumed during the most recent processing (the actual time used by the job) to summarize all CPU time used by all jobs. The obtained value is normalized and multiplied by the actual job CPU usage value. The normalized CPU usage by active jobs is obtained as the ratio of the CPU usage consumed during the most recent job calculation to the summary of all CPU usage used by the given language computations jobs. The obtained results are presented in Figures 6 and 7 in the form of trends. In Figure 6, due to the large differences in values, the results are presented on a logarithmic scale, while the trends are in a power form. On this basis, it can be observed that in the case of Python, the increase in normalized calculation time is much slower despite a significant increase in the number of processed data. On the other hand, Go and Java languages are characterized by similar dynamics of calculation time increase in relation to the number of data set.

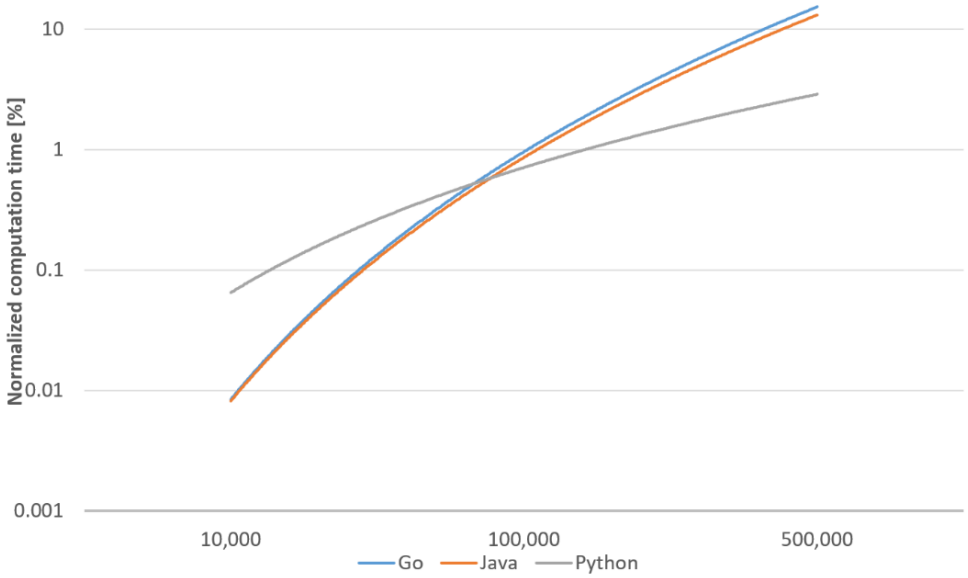

Figure 6. Comparison of the normalized values of the CPU computation time (%) for the decision tree algorithm implementation presented in trends.

As shown in Figure 7 in the context of standardized values, the Python language is also characterized by the lowest rate of CPU usage growth, while the highest rate refers to Go. In this comparison, the Java language is in the middle, which means that even with a small set of records, it has a higher CPU usage than Go and lower than Python. In this respect, the language holds the second position compared to 100 thousand and 500 thousand records. Therefore, from this perspective, Python proved to be the most stable language. Thanks to this aspect, it shows great scalability of calculations.

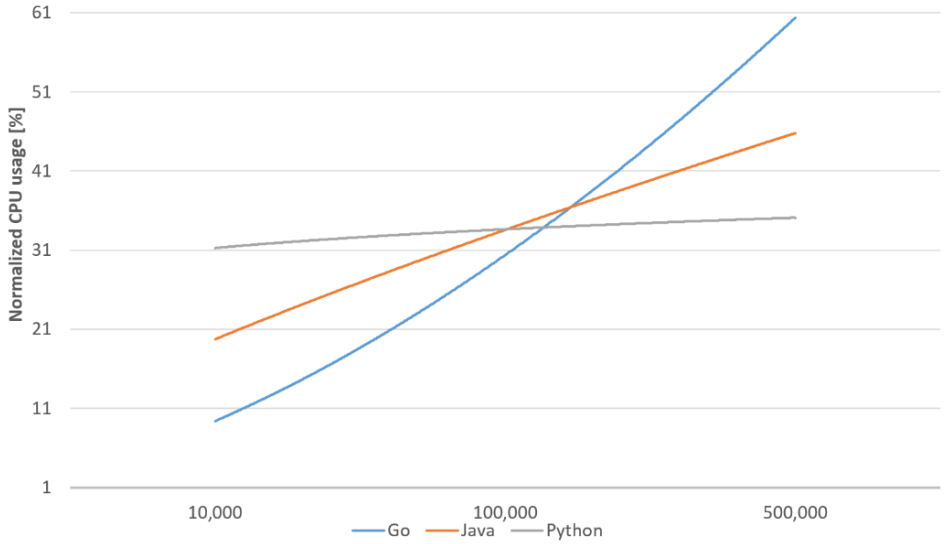

Figure 7. Comparison of the normalized values of the CPU usage (%) for the decision tree algorithm implementation presented in trends.

Nevertheless, its optimization in relation to specialized solutions dedicated to the industry could be based on a separate article. In the case of Python, the CPU value for Python applications was at a similar level regardless of the size of the database. One of the reasons for this state of affairs is an excellent optimization of data structures in Python, so that access to data was fast and did not require a lot of calculations from the processor. This could be due to the fact that Python has a very convenient

type of data object called a dictionary, which was used to build the tree structure. The dictionary is equivalent to an associative table in other languages. The dictionary type is implemented as a hash table and is additionally optimized, which makes the data extraction quite fast. Attribute access in an object uses dictionary access behind the scenes; dicts are the fastest and have a reasonable memory footprint.

It should also be noted that both Golang and Python also have very different approaches to handling concurrency. Golang has built-in concurrency support and is incredibly resource-efficient, using goroutines that are less demanding for CPU and memory. On the other hand, Python has no built-in concurrency support based on the concurrency library and is less resource-efficient. Here, significant numbers of libraries can be substantial, with which Python efficiently performs basic programming tasks, while Golang has built-in functions and is suitable for micro-services software architectures.

5. Conclusions

Considering the needs and challenges faced by engineers and programmers in the context of a more comprehensive realization of the concept of Industry 4.0, various methods and means should be sought to make this possible. Of course, the problem indicated is extensive and cannot be addressed in one publication. Therefore, the authors focused only on using selected programming languages to implement the decision tree algorithm in the context of their use in decision-making processes dedicated to solutions that enable customers to analyze the selection of elements in a remote rapid prototyping service. This service's choice as a source of data was not accidental. Considering the current epidemic situation and related limitations and difficulties in transport, communication, and cooperation seemed essential and exciting to the authors. However, nothing stands in the way of other data from the area of economics, CNC production systems, or sensor and control systems.

Given that the vast majority of hardware solutions used in industrial environments, particularly Industry 4.0, are usually embedded systems, the intervention in their architectural design is restricted, including in the area of memory expansion and processor replacement. Thus, optimization of the code, e.g., its size, required memory, and processor load, is an important issue. Obviously, specialized server systems, especially cloud computing, offer great possibilities of scalability of necessary resources. Still, such solutions in the area of industry can be used only to a narrower scale.

Analyzing the experiments' results, it can be concluded that the Golang programming language is an alternative to the more widely used Java and Python languages so far. It has excellent implementation potential and is characterized by a fresh approach to programming. At the same time, it is easy to learn. In terms of calculation time and resource management, it does not differ significantly from Java and Python applications, and in many cases, it performs better. From the point of view of resource usage, both languages, Go and Python is very similar.

Nevertheless, it is important from a programming and performance point of view to emphasize the following aspects of Go compared to other languages: syntax simplicity, C-based syntax, cross-platform and cross processor, as well as native compiles with no runtime installation required. Go can be an attractive alternative in the area of DevOps tools. It is attractive to build something small–medium that works natively without using a lot of RAM and which runs fast with many things needed for this task in the language itself. However, in Go's case, the most significant innovation is probably the efficient concurrency implemented with channels and goroutines: synchronous yet non-blocking IO use goroutines, which are essentially functions that can work simultaneously and independently. Goroutines are very scalable because they take up little memory and run many processes simultaneously, while Java threads are inherently blocked. Golang goroutines are the opposite of what a Java thread is, which turns out to be a very memory consuming technique. This feature of Java became apparent during the research, as this language had the worst results from the point of view of the memory used. On the other hand, Python uses concurrency, but it is not built-in; it introduces parallelism utilizing threads. This means that if you are going to work with large datasets, Golang seems to be a more appropriate choice and much more scalable.

Modern software must be high-speed, and delays of milliseconds are of great importance, especially in the case of large data sets. However, in embedded and dedicated systems under IoT and Industry 4.0, it is certainly more critical to optimize a given program's operation and effectively manage resources. One of Golang's characteristics in terms of its efficiency is channels. Channels are a robust solution and can quickly and easily share the state between threads instead of overloaded and performance-lowering locks. This aspect of the Go language will be further investigated in future publications.

As a result, both languages, Go and Python can be widely used in the IoT infrastructure and in specific implementations of Industry 4.0, which require speed and relatively low absorption of resources such as memory, CPU, etc. Thanks to this, the software created in the future, among others, to support decision-making, production, and analytical processes, will be useful and can also be implemented directly in the design and production infrastructure devices. Moreover, it is worth noting that the boundary between programming for desktop devices (PCs, laptops, servers) and mobile or IoT devices is increasingly fading. Therefore, programming languages should not be subject to this classification: desktop or mobile; but instead comply with a specific environment called ecosystem (Google, AWS, Apple, etc.), an example of which is presented by Google's Go language. Thus, modern distributed processing, as well as allowing one to improve development tools for building services in the cloud with the Go language.

Author Contributions: P.D., A.P., researched the literature; P.D., A.P., formulated the problems and constructed the research framework, P.D., A.P., conceived the methodologies and designed the experiments; P.D., A.P., calculations conducted; P.D., A.P., investigation and developing the application of results in a real production environment; P.D., A.P., contributed to contributed to writing of this paper. All authors have read and agreed to the published version of the manuscript.

Funding: This project is financed by the Minister of Science and Higher Education of the Republic of Poland within the "Regional Initiative of Excellence" program for years 2019–2022. Project number 027/RID/2018/19, amount granted 11,999,900 PLN.

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

References

1. Godina, R.; Ribeiro, I.; Matos, F.; Ferreira, B.T.; Carvalho, H.; Peças, P. Impact Assessment of Additive Manufacturing on Sustainable Business Models in Industry 4.0 Context. *Sustainability* **2020**, *12*, 7066. [CrossRef]
2. Pereira, A.C.; Romero, F. A review of the meanings and the implications of the Industry 4.0 concept. *Procedia Manuf.* **2017**, *13*, 1206–1214. [CrossRef]
3. Strange, R.; Zucchella, A. Industry 4.0, global value chains and international business. *Multinatl. Bus. Rev.* **2017**, *25*, 174–184. [CrossRef]
4. Barosz, P.; Gołda, G.; Kampa, A. Efficiency Analysis of Manufacturing Line with Industrial Robots and Human Operators. *Appl. Sci.* **2020**, *10*, 2862. [CrossRef]
5. Tsafnat, G.; Glasziou, P.; Choong, M.K.; Dunn, A.; Galgani, F.; Coiera, E. Systematic review automation technologies. *Syst. Rev.* **2014**, *3*, 74. [CrossRef]
6. Kusiak, A. Smart manufacturing. *Int. J. Prod. Res.* **2018**, *56*, 508–517. [CrossRef]
7. Schumacher, A.; Nemeth, T.; Sihn, W. Roadmapping towards industrial digitalization based on an Industry 4.0 maturity model for manufacturing enterprises. In Proceedings of the 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME, Gulf of Naples, Italy, 18–20 July 2018; pp. 409–414.
8. Zhou, J. Digitalization and intelligentization of manufacturing industry. *Adv. Manuf.* **2013**, *1*, 1–7. [CrossRef]
9. Sjödin, D.R.; Parida, V.; Leksell, M.; Petrovic, A. Smart Factory Implementation and Process Innovation. *Res. Technol. Manag.* **2018**, *61*, 22–31. [CrossRef]
10. Rojko, A. Industry 4.0 Concept: Background and Overview. *Int. J. Interact. Mobile Technol.* **2017**, *11*, 77–89. [CrossRef]
11. Saucedo-Martínez, J.A.; Pérez-Lara, M.; Marmolejo-Saucedo, J.A.; Salais-Fierro, T.E.; Vasant, P. Industry 4.0 framework for management and operations: A review. *J. Ambient Intell. Humaniz. Comput. Vol.* **2018**, *9*, 789–801. [CrossRef]

12. Mazur, D.; Paszkiewicz, A.; Bolanowski, M.; Budzik, G.; Oleksy, M. Analysis of possible SDN use in the rapid prototyping process as part of the Industry 4.0. *Bull. Pol. Acad. Sci. Tech. Sci.* **2019**, *67*, 21–30.
13. Haseeb, M.; Hussain, H.I.; Ślusarczyk, B.; Jermsittiparsert, K. Industry 4.0: A Solution towards Technology Challenges of Sustainable Business Performance. *Soc. Sci.* **2019**, *8*, 154. [CrossRef]
14. Piccarozzi, M.; Aquilani, B.; Gatti, C. Industry 4.0 in Management Studies: A Systematic Literature Review. *Sustainability* **2018**, *10*, 3821. [CrossRef]
15. Aslam, F.; Aimin, W.; Li, M.; Ur Rehman, K. Innovation in the Era of IoT and Industry 5.0: Absolute Innovation Management (AIM) Framework. *Information* **2020**, *11*, 124. [CrossRef]
16. Hecklau, F.; Galeitzke, M.; Flachs, S.; Kohl, H. Holistic approach for human resource management in Industry 4.0. *Procedia CIRP* **2016**, *54*, 1–6. [CrossRef]
17. Xu, L.D.; Duan, L. Big data for cyber physical systems in industry 4.0: A survey. *Enterp. Inf. Syst.* **2019**, *13*, 148–169. [CrossRef]
18. Yan, J.; Meng, Y.; Lu, L.; Li, L. Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes, and Applications for Predictive Maintenance. *IEEE Access* **2017**, *5*, 23484–23491. [CrossRef]
19. Mourtzis, D.; Vlachou, E.; Milas, N. Industrial Big Data as a Result of IoT Adoption in Manufacturing. *Procedia CIRP* **2016**, *55*, 290–295. [CrossRef]
20. Trstenjak, M.; Cosic, P. Process Planning in Industry 4.0 Environment. *Procedia Manuf.* **2017**, *11*, 1744–1750. [CrossRef]
21. Paszkiewicz, A.; Bolanowski, M.; Budzik, G.; Przeszłowski, Ł.; Oleksy, M. Process of Creating an Integrated Design and Manufacturing Environment as Part of the Structure of Industry 4.0. *Processes* **2020**, *8*, 1019. [CrossRef]
22. Dymora, P.; Mazurek, M.; Mroczka, B. Code optimization of advanced applications on the example of selected development technologies. In *VII Konferencja Naukowa "Symbioza Techniki i Informatyki"*; Wydawnictwo Uniwersytetu Rzeszowskiego: Rzeszów, Poland, 2017.
23. Li, Y.; Fu, Y.; Tang, X.; Hu, X. Optimizing the Reliability and Efficiency for an Assembly Line That Considers Uncertain Task Time Attributes. *IEEE Access* **2019**, *7*, 34121–34130. [CrossRef]
24. Semeria, L.; Ghosh, A. Methodology for hardware/software co-verification in C/C++. In *Proceedings 2000. Design Automation Conference, Yokohama, Japan*; IEEE Cat. No.00CH37106; IEEE: Piscataway, NJ, USA, 2000; pp. 405–408. [CrossRef]
25. Myalapalli, V.K.; Geloth, S. Minimizing impact on JAVA virtual machine via JAVA code optimization. In *2015 International Conference on Energy Systems and Applications*; IEEE: Piscataway, NJ, USA, 2015; pp. 19–24.
26. Dymora, P.; Mazurek, M.; Maciąg, P. Implementation and analysis of self-reconfigurable routing protocol for wireless sensor networks. In *VII Konferencja Naukowa "Symbioza Techniki i Infromatyki"*; Wydawnictwo Uniwersytetu Rzeszowskiego: Rzeszów, Poland, 2017.
27. Liang, M.; Wang, X. The design of intelligent robot based on embedded system. In Proceedings of the 2011 International Conference on Advanced Mechatronic Systems, Zhengzhou, China, 11 August 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 23–28.
28. Al Hayajneh, A.; Bhuiyan, M.Z.A.; McAndrew, I. Improving Internet of Things (IoT) Security with Software-Defined Networking (SDN). *Computers* **2020**, *9*, 8. [CrossRef]
29. Kiran, M.; Pouyoul, E.; Mercian, A.; Tierney, B.; Guok, C.; Monga, I. Enabling intent to configure scientific networks for high performance demands. *Future Gener. Comput. Syst.* **2018**, *79*, 205–214. [CrossRef]
30. Foster, N.; Harrison, R.; Freedman, M.J.; Monsanto, C.; Rexford, J.; Story, A.; Walker, D. Frenetic: A network programming language. *ACM Sigplan Not.* **2011**, *46*, 279–291. [CrossRef]
31. Voellmy, A.; Kim, H.; Feamster, N. Procera: A language for high-level reactive network control. In Proceedings of the 1st Workshop Hot Topics Software Defined Networks, Helsinki, Finland, 13 August 2012; Association for Computing Machinery: New York, NY, USA, 2012; pp. 43–48.
32. Wang, J.; Cheng, S.; Fu, X. SDN Programming for Heterogeneous Switches with Flow Table Pipelining. *Sci. Program.* **2018**, *2018*, 2848232. [CrossRef]
33. Wang, X.; Tian, Y.; Zhao, M.; Li, M.; Mei, L.; Zhang, X. PNPL: Simplifying programming for protocol-oblivious SDN networks. *Comput. Netw.* **2018**, *147*, 64–80. [CrossRef]
34. Lehmhus, D.; Wuest, T.; Wellsandt, S.; Bosse, S.; Kaihara, T.; Thoben, K.-D.; Busse, M. Cloud-Based Automated Design and Additive Manufacturing: A Usage Data-Enabled Paradigm Shift. *Sensors* **2015**, *15*, 32079–32122. [CrossRef]

35. Zhang, Y.; Zhang, G.; Liu, Y.; Hu, D. Research on services encapsulation and virtualization access model of machine for cloud manufacturing. *J. Intell. Manuf.* **2017**, *28*, 1109–1123. [CrossRef]
36. Howard, R.; Björk, B.-C. Use of standards for CAD layers in building. *Autom. Constr.* **2007**, *16*, 290–297. [CrossRef]
37. Kim, D.B.; Shin, S.-J.; Shao, G.; Brodsky, A. A decision-guidance framework for sustainability performance analysis of manufacturing processes. *Int. J. Adv. Manuf. Technol.* **2015**, *78*, 1455–1471. [CrossRef]
38. Bruccoleri, M.; La Diega, S.N.; Perrone, G. An Object-Oriented Approach for Flexible Manufacturing Control Systems Analysis and Design Using the Unified Modeling Language. *Int. J. Flex. Manuf. Syst.* **2003**, *15*, 195–216. [CrossRef]
39. Morel, G.; Panetto, H.; Zaremba, M.; Mayer, F. Manufacturing Enterprise Control and Management System Engineering: Paradigms and open issues. *Annu. Rev. Control* **2003**, *27*, 199–209. [CrossRef]
40. Ward, T.L.; Ralston, P.A.S.; Davis, J.A. Fuzzy logic control of aggregate production planning. *Comput. Ind. Eng.* **1992**, *23*, 137–140. [CrossRef]
41. Noorul Haq, A.; Kannan, G. Design of an integrated supplier selection and multi-echelon distribution inventory model in a built-to-order supply chain environment. *Int. J. Prod. Res.* **2006**, *44*, 1963–1985. [CrossRef]
42. Wang, S.; Wan, J.; Li, D.; Liu, C. Knowledge Reasoning with Semantic Data for Real-Time Data Processing in Smart Factory. *Sensors* **2018**, *18*, 471. [CrossRef]
43. Idris, I. *Python Data Analysis*, 2nd ed.; Packt Publishing: Birmingham, UK, 2014.
44. Dymora, P.; Mazurek, M. Performance Assessment of Selected Techniques and Methods Detecting Duplicates in Data Warehouses, Theory and Applications of Dependable Computer Systems. DepCoS-RELCOMEX 2020. In *Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2020; Volume 1173. [CrossRef]
45. Dymora, P.; Mazurek, M.; Łannik, D. Badanie efektywności tworzenia wielowymiarowych zestawów danych w wybranych środowiskach analitycznych, Monografia pt. In *Social and Technical Aspects of Security*; Oficyna Wydawnicza Politechniki Rzeszowskiej: Rzeszów, Poland, 2019; pp. 37–57, ISBN 978-83-7934-352-2.
46. Mabkhot, M.M.; Al-Ahmari, A.M.; Salah, B.; Alkhalefah, H. Requirements of the Smart Factory System: A Survey and Perspective. *Machines* **2018**, *6*, 23. [CrossRef]
47. Jaskó, S.; Skrop, A.; Holczinger, T.; Chován, T.; Abonyi, J. Development of manufacturing execution systems in accordance with Industry 4.0 requirements: A review of standard- and ontology-based methodologies and tools. *Comput. Ind.* **2020**, *123*, 103300.
48. Bansal, A.K. *Introduction to Programming Languages*; CRC Press: Boca Raton, FL, USA, 2013.
49. Ajah, I.A.; Nweke, H.F. Big Data and Business Analytics: Trends, Platforms, Success Factors and Applications. *Big Data Cogn. Comput.* **2019**, *3*, 32. [CrossRef]
50. Esquembre, F. Easy Java Simulations: A software tool to create scientific simulations in Java. *Comput. Phys. Commun.* **2004**, *156*, 199–204. [CrossRef]
51. Boisvert, R.F.; Moreira, J.; Philippsen, M.; Pozo, R. Java and numerical computing. *Comput. Sci. Eng.* **2001**, *3*, 18–24. [CrossRef]
52. The 10 Most Popular Programming Languages to Learn in 2020. Available online: https://www.northeastern.edu/graduate/blog/most-popular-programming-languages/ (accessed on 10 September 2020).
53. Hao, J.; Ho, T.K. Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. *J. Educ. Behav. Stat.* **2019**, *44*, 348–361. [CrossRef]
54. Raschka, S.; Patterson, J.; Nolet, C. Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information* **2020**, *11*, 193. [CrossRef]
55. Borcherds, P.H. Python: A language for computational physics. *Comput. Phys. Commun.* **2007**, *177*, 199–201. [CrossRef]
56. Tan, S.W.B.; Naraharisetti, P.K.; Chin, S.K.; Lee, L.Y. Simple Visual-Aided Automated Titration Using the Python Programming Language. *J. Chem. Educ.* **2020**, *97*, 850–854. [CrossRef]
57. Gabriel, R.P. *Performance and Evaluation of Lisp Systems*; MIT Press: Cambridge, MA, USA, 1985.
58. Dujmović, J.; Luis, N. Benchmarking the Efficiency of Array Processing for Various Types of Language Processory. In Proceedings of the 2010 Fifth International Conference on Software Engineering Advances, Nice, France, 22–27 August 2010; IEEE: Piscataway, NJ, USA, 2010.
59. Aruoba, S.B.; Fernández-Villaverde, J. A comparison of programming languages in macroeconomics. *J. Econ. Dyn. Control* **2015**, *58*, 265–273. [CrossRef]

60. Fourment, M.; Gillings, M.R. A comparison of common programming languages used in bioinformatics. *BMC Bioinform.* **2008**, *9*, 82. [CrossRef]
61. Bull, J.M.; Smith, L.A.; Pottage, L.; Freeman, R. Benchmarking Java against C and Fortran for Scientific Applications. In Proceedings of the 2001 Joint ACM-ISCOPE Conference on Java Grande, Palo Alto, CA, USA, 2–4 June 2001; ACM Press: New York, NY, USA, 2001.
62. Jääskeläinen, P.; Kultala, H.; Viitanen, T.; Takala, J. Code Density and Energy Efficiency of Exposed Datapath Architectures. *J. Sign. Process. Syst.* **2015**, *80*, 49–64. [CrossRef]
63. Constable, R.L.; Borodin, A.B. Subrecursive Programming Languages, Part I: Efficiency and program structure. *J. Assoc. Comput. Mach.* **1972**, *19*, 526–568. [CrossRef]
64. Wilson, L.B.; Clark, R.G. *Comparative Programming Languages*, 3rd ed.; Addison-Wesley: Harlow, UK, 2001.
65. Sebesta, R.W. *Concepts of Programming Languages*, 10th ed.; Pearson Education, Inc.: Upper Saddle River, NJ, USA, 2012.
66. Cezzar, R. *A Guide to Programming Languages: Overview and Comparison*; Artech House Publishers: Norwood, MA, USA, 1995.
67. Gadre, M. Comparing object-oriented programming languages. In *Object-Oriented Software for Manufacturing Systems*; Adiga, S., Ed.; Springer: Dordrecht, The Netherlands, 1993; pp. 125–166.
68. Lincke, R.; Lundberg, J.; Löwe, W. Comparing software metrics tools. In Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2008, Seattle, WA, USA, 20–24 July 2008.
69. Reddy, K.P.N.; Geyavalli, Y.; Sujani, D.; Rajesh, S.M. Comparison of Programming Languages: Review. *Int. J. Comput. Sci. Commun.* **2018**, *9*, 113–122.
70. Fan, J.; Han, F.; Liu, H. Challenges of Big Data analysis. *Natl. Sci. Rev.* **2014**, *1*, 293–314. [CrossRef]
71. Assous, F.; Chaskalovic, J. Data mining techniques for scientific computing: Application to asymptotic paraxial approximations to model ultrarelativistic particles. *J. Comput. Phys.* **2011**, *230*, 4811–4827. [CrossRef]
72. Li, H.; Zhang, Z.; Zhao, Z.-Z. Data-Mining for Processes in Chemistry, Materials, and Engineering. *Processes* **2019**, *7*, 151. [CrossRef]
73. Olson, D.L.; Delen, D. *Advanced Data Mining Techniques*; Springer: Berlin, Germany, 2008.
74. Tan, P.-N.; Steinbach, M.; Karpatne, A.; Kumar, V. *Introduction to Data Mining*, 2nd ed.; Pearson: London, UK, 2018.
75. Craven, M.W.; Shavlik, J.W. Using neural networks for data mining. *Future Gener. Comput. Syst.* **1997**, *13*, 211–229. [CrossRef]
76. Wu, X.; Kumar, V.; Ross Quinlan, J.; Ghosh, J.; Yang, Q.; Motoda, H.; McLachlan, G.J.; Ng, A.; Liu, B.; Yu, P.S.; et al. Top 10 algorithms in data mining. *Knowl. Inf. Syst.* **2008**, *14*, 1–37. [CrossRef]
77. Kingsford, C.; Salzberg, S. What are decision trees? *Nat. Biotechnol.* **2008**, *26*, 1011–1013. [CrossRef]
78. Quinlan, J.R. Induction of decision trees. *Mach. Learn.* **1986**, *1*, 81–106. [CrossRef]
79. Loh, W.Y. Classification and regression trees. *Data Min. Knowl. Discov.* **2011**, *1*, 14–23. [CrossRef]
80. Li, M.; Xu, H.; Deng, Y. Evidential Decision Tree Based on Belief Entropy. *Entropy* **2019**, *21*, 897. [CrossRef]
81. Tangirala, T. Evaluating the Impact of GINI Index and Information Gain on Classification using Decision Tree Classifier Algorithm. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 612–619. [CrossRef]
82. Karim, M.; Rahman, R.M. Decision Tree and Naïve Bayes Algorithm for Classification and Generation of Actionable Knowledge for Direct Marketing. *J. Softw. Eng. Appl.* **2013**, *6*, 196–206. [CrossRef]
83. Jovanović, M.M.; Kašćelan, L.; Joksimović, M.; Kašćelan, V. Decision tree analysis of wine consumers' preferences: Evidence from an emerging market. *Br. Food J.* **2017**, *119*, 1349–1361. [CrossRef]
84. Canbolat, Y.B.; Chelst, K.; Garg, N. Combining decision tree and MAUT for selecting a country for a global manufacturing facility. *Omega* **2007**, *35*, 312–325. [CrossRef]
85. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
86. Brissaud, J.-B. The meanings of entropy. *Entropy* **2005**, *7*, 68–96. [CrossRef]