big data and cognitive computing

MDPI

Article

Edge Machine Learning: Enabling Smart Internet of Things Applications

Mahmut Taha Yazici, Shadi Basurra and Mohamed Medhat Gaber *

School of Computing and Digital Technology, Birmingham City University, Birmingham B5 5JU, UK; Mahmut.Yazici@mail.bcu.ac.uk (M.T.Y.); Shadi.Basurra@bcu.ac.uk (S.B.)
* Correspondence: Mohamed.Gaber@bcu.ac.uk

Received: 11 July 2018; Accepted: 17 August 2018; Published: 3 September 2018

Abstract: Machine learning has traditionally been solely performed on servers and high-performance machines. However, advances in chip technology have given us miniature libraries that fit in our pockets and mobile processors have vastly increased in capability narrowing the vast gap between the simple processors embedded in such things and their more complex cousins in personal computers. Thus, with the current advancement in these devices, in terms of processing power, energy storage and memory capacity, the opportunity has arisen to extract great value in having on-device machine learning for Internet of Things (IoT) devices. Implementing machine learning inference on edge devices has huge potential and is still in its early stages. However, it is already more powerful than most realise. In this paper, a step forward has been taken to understand the feasibility of running machine learning algorithms, both training and inference, on a Raspberry Pi, an embedded version of the Android operating system designed for IoT device development. Three different algorithms: Random Forests, Support Vector Machine (SVM) and Multi-Layer Perceptron, respectively, have been tested using ten diverse data sets on the Raspberry Pi to profile their performance in terms of speed (training and inference), accuracy, and power consumption. As a result of the conducted tests, the SVM algorithm proved to be slightly faster in inference and more efficient in power consumption, but the Random Forest algorithm exhibited the highest accuracy. In addition to the performance results, we will discuss their usability scenarios and the idea of implementing more complex and taxing algorithms such as Deep Learning on these small devices in more details.

Keywords: machine learning; edge computing; Internet of Things; IoT

1. Introduction

The Internet of Things (IoT) is rapidly changing the world from the way we drive to how we make purchases and even how we get energy for our homes [1]. The sophisticated sensors and chips are embedded in the physical things that surround us, each transmitting valuable data which lets us understand how these things work and work together [2]. In other words, machines are embedded with sensors that can relay data to each other with no human involvement. Essentially, it means all sorts of everyday items are connected to the Internet, which could potentially transform the way we live.

Another reason that gives IoT its current importance and makes its use ubiquitous throughout all industries is the fact that big businesses are already investing billions into the emerging technology. In 2008, there were officially more devices connected to the Internet than there were human beings, and by 2020 that number is expected to reach 50 billion [3].

In like manner, if corporations and policy-makers get it right, linking the digital and physical worlds could generate up to $11.1 trillion a year in economic value by 2025 [4]. Thus, firms will further invest in IoT to optimise distribution costs, redesign factory work-flows and improve tracking of

materials. For instance, both UPS (Multinational package delivery, Atlanta, USA) and John Deere (Manufacturer of agricultural, construction, and forestry machinery, Moline, USA) are already using IoT-enabled fleet tracking technologies to improve supply efficiency and cut operating costs [5].

Moreover, it has now become easier to digitise certain functions and key capabilities of industrial-age products, thanks to advances in microprocessor technologies, increasingly efficient power management, reliable memory, and broadband communication [6]. In addition, as stated above, IoT solutions typically combine information technology (IT) with physical things in the form of software and hardware. As a result, the primary thing-based physical functions of a thing can be augmented with further IT-based digital services, which can be accessed not only on a local basis but at a global level [7]. For example, the primary function of a light bulb is to provide light. If it was, however, enriched with IoT technology, it could also sense human presence and serve as a low-cost security system. Likewise, together with other energy consuming appliances in the house, it may be connected to a central unit or application for energy consumption detection and optimisation [8].

Thereafter, data extracted from IoT must be exploited using some sort of process. For that, traditional data analysis has been reliable so far at explaining data. Reports or models of what happened in the past or of what is happening today can be generated, pulling useful insights to apply to the organisation. Data analytics helps quantify and track goals, enable smarter decision making, and later provide the means for measuring success over time. On the contrary, data models that are typical of traditional data analytics are often static and of limited use in addressing fast-changing and unstructured data. When it comes to IoT, it is often compulsory to identify correlations between masses of sensor inputs and external factors that are, continuously and in real time, producing millions of data points.

In traditional data analysis a model woul be built on past data and expert opinion to establish a relationship between the variables. However, machine learning starts with the outcome variables (e.g., saving energy) and then automatically utilising predictor variables and their interactions. An eminent example is Google's recent application of machine learning on their data center cooling technology to maintain environmental conditions suitable for their servers operation. With the goal of increasing energy efficiency, Google applied machine learning and cut its overall energy consumption by 15%. This represents hundreds of millions of dollars in savings for Google in the coming years.

Predictive capabilities are extremely useful in an industrial setting. By drawing data from multiple sensors in or on machines, machine learning algorithms can "learn" what's typical and abnormal behaviour for a machine, sense soil moisture and nutrients in agriculture, manage smart homes, power wearables, revolutionise healthcare, and so on. The billions of sensors and devices that will continue to be connected to the Internet in the upcoming years will produce exponentially more data. This gigantic increase in data will steer great improvements in machine learning, unlocking countless opportunities for us to reap the benefits.

The objective of this research is to answer the following questions:

(a) How could machine learning algorithms be applied to IoT smart data?
(b) Is it feasible to run machine learning algorithms on IoT?
(c) What measures could be taken to enhance the execution of these algorithms on IoT devices?
(d) What would be the next step forward in applying machine learning to IoT smart data?

There are different types of IoT devices and each type depends on the application it is designed for. For example, automating and control of home devices or industry tasks. These devices vary in their processing capability, memory and energy requirement. One of the key platforms for IoT is the Raspberry Pi. This is a popular platform because it is reasonably cheap, and it offers a complete Linux server in a tiny device. We use Raspberry Pi because it is easy to connect various sensors into the device to run machine learning for prediction and classification. The scope of this paper is to understand how these devices can cope when running different types of machine learning algorithms. This will help us to quantify the processing power and the energy required to run various algorithm to

learn if these devices can survive on batteries when deployed in a remote location, and whether the algorithms can be performed in a reasonable time with reasonable accuracy.

To comprehend which algorithms are more suitable for processing the generated data from the IoT, visualising the following three concepts is essential. First, the IoT applications, second, the IoT data characteristics, and, third, the data-driven vision of machine learning algorithms. Then, we discuss the issues.

The research will use data sets that contain real data extracted from applications that would enjoy the leverage which IoT and machine learning could bring. These data sets range across various fields that include air quality and forest fires detection to autism screening and breast cancer exposure, where imminent response with utmost accuracy is critical. Therefore, for the applications' reliability, the selected algorithms must perform with acceptable accuracy levels and do so in a short time without consuming excessive amount of energy.

Having reviewed the state of the art about how IoT data is analysed, many noteworthy and insightful findings have been revealed regarding data characteristics. To have a profound insight into IoT smart data, patterns must be extracted, and the generated data to be understood. This will allow for obtaining the right accuracy score, enabling the IoT device to respond to events and consequently affecting IoT decision-making or controls. Moreover, to improve IoT speed and power consumption, computationally expensive processes of the machine learning algorithms can be isolated, and possibly executed on powerful machines e.g., the cloud, leaving only the necessary parts to run on the less capable lightweight IoT devices.

Assuming that positive results have been obtained through this research, more complex and demanding algorithms could be deployed and tested on these same devices to further test their capabilities, and find out about the opportunities this may offer. Thereafter, algorithms such as Deep Learning would extend the range of issues that can be tackled using IoT.

The rest of this paper is organised as follows. Related work in the area of edge analytics is discussed in Section 2. The machine learning methods used in this experimental profiling are discussed in Section 3. The chosen data sets are presented along with the experimental work in Section 4. The obtained results and observations from the experiments are viewed in Section 5, and the conclusions together with future research directions and open issues are presented in Sections 6 and 7, respectively.

2. Related Work

Realising the potential of handheld devices to run machine learning algorithms has been explored in the early years of the new century by Kargupta et al. [9,10] and Gaber et al. [11–13]. The work has provided evidence of the potential of edge analytics, even before the era of smartphones and tablet computers.

Distribution of machine learning methods running on smartphones, with the rise of Internet of Things, has been studied thoroughly in the area of Pocket Data Mining [14]. It is now evident that edge devices are complementary to cloud computing to scale out machine learning systems. Furthermore, the deployment of deep learning methods is a rising research area with compression of deep learning models through pruning and quantisation [15]. The other approach is to directly train small networks with the aim of decreasing latency. MobileNets models from Google is an exemplar in this approach [16]. Advances in this area will ultimately lead to a wide deployment of deep learning models at the edge to reduce latency. Shallower deep learning models and traditional shallow learning models can be viable options for edge devices, especially those with resource constraints.

In this work, we focus on profiling the performance of notably successful shallow learning methods. The choice of the methods was based on accuracy profiling of machine learning algorithms over a large number of data sets [17].

3. Machine Learning Methods

The aim is to deploy powerful Machine Learning algorithms (ML) on the IoT devices to accommodate solutions for various setbacks that the IoT devices can overcome with the right tools as the research targets to equip IoT devices with the best possible instruments, so they can tackle current and future real-world difficulties. Less powerful techniques such as linear regression and decision trees have already been implemented on IoT [18] and will not be considered. The goal is to stress the devices to the furthest extent, acquire utmost usage and extend their benefit in their fields of use.

In the early stages of this research, Logistic Regression and K-Nearest Neighbour algorithm had been implemented and tested with classification data set. Later, when ten data sets of two types (one half classification and the other regression) and with different use cases were selected for testing, LR and KNN were dropped as they could only be used for classification problems.

The remaining algorithms that were implemented and used in all conducted tests. These are Multi-Layer Perceptron, Support Vector Machine and Random Forests.

3.1. Multi-Layer Perceptron

Deep Learning and Neural networks are one of the hottest topics right now. Large corporations and young startups alike are gold-rushing into this state-of-the-art field. Artificial neural networks (ANNs) are currently driving some of the most ingenious inventions of the century, for the reasons mentioned in [19]. ANNs are the artificial representation of the working nervous system. They are an interconnected web of nodes, which are called neurons, that are connected with edges typically have a weight that adjusts as learning continues (See Figure 1).

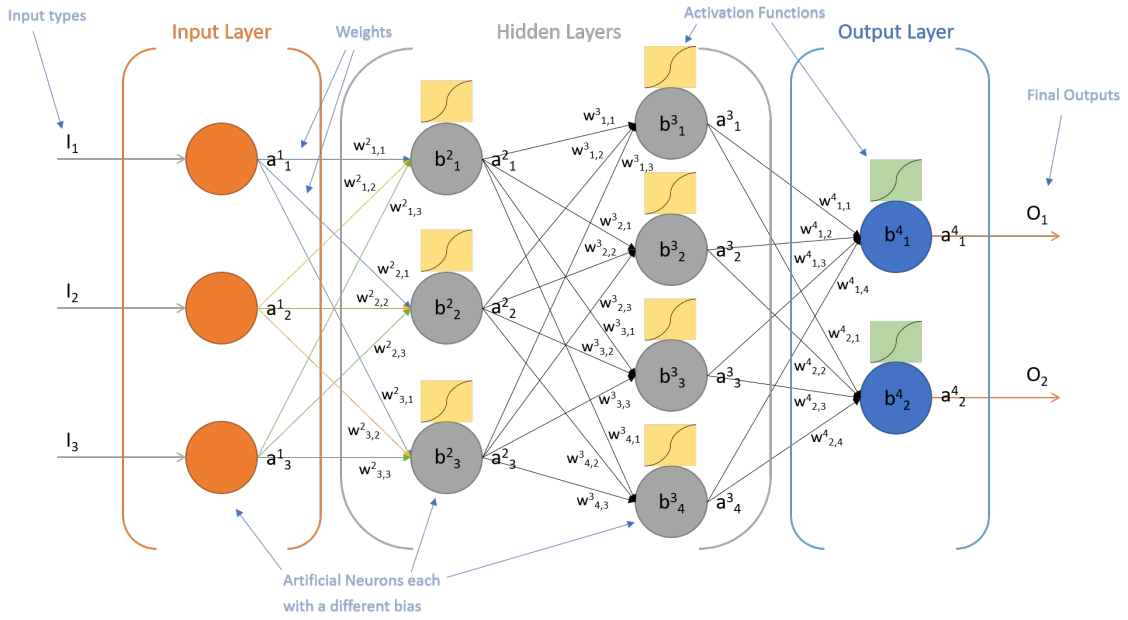

Figure 1. Visual Representation of an Artificial Neural Network.

ANNs are analytical algorithms that are widely used in the recognition and classification of patterns. They are being used effectively, under many different architectures, to undertake specific tasks—for example: forecasting, clustering, pattern recognition, decision making and management, data compression and approximation [20]. Thus, since ANNs are vital at numerous real-world applications, they could not be left out of this research. The ANN algorithm will be used for the testing phase as they can be developed using multiple different training algorithms, and they require less formal statistical training to develop. In addition, ANNs can detect nonlinear relationships between

dependent and independent variables and possess the ability to detect all possible interactions between predicator variables [21].

One of the biggest disadvantages of Artificial Neural Networks is their "black box" nature, which explicitly gives limited ability to identify possible casual relationships [22]. However, this will not affect this search, as the research's sole purpose is to test and benchmark the devices' capabilities rather than understand the metamorphosis of data while training. Another issue that ANNs bring along is that they require greater computational resources compared to more traditional ML algorithms [23]. The research will attempt to use this hindrance as a tool to discover the devices' furthest limits in terms of training time, computational power, storage capability and battery life.

The implementation of the ANN algorithm will resemble overclocking computer hardware for better performance using the same exact pieces. To begin with, a shallow neural network will be used. Incrementally, newer neurons and then new layers will be added to underpinning the algorithm performance, all the while watching how these incremental changes affect the performance of the IoT devices. Using the test data, the algorithm will be settled at the threshold where the best performance meets the best possible accuracy.

Artificial Neurons are the heart of a neural network. The neuron of a neural network is an activation node. The activation node takes the input from the presidency nodes, applies the learning parameters to the weighted sum and then passes that sum to an activation function that computes the composite prediction or probabilities. This is known as a perceptron which simply takes multiple inputs and produces one output. Therefore, the predictions are progressively processed until the final output is generated [24].

3.1.1. Random Forest

Random Forest (RF) is an easy to use and flexible machine learning algorithm. It is known to achieve high accuracy results, even without hyper-parameter tuning [25]. It was a primary consideration of the research because of its simplicity, and the fact that it can be used for both classification and regression problems.

Random Forest is a supervised learning algorithm, and, as the name suggests, it creates a forest and makes it in some way random. The "forest" is constructed from an assembly of Decision Trees, mostly trained with the "bagging" method. RF grows multiple trees as opposed to a single tree in court model to classify a new object based on attributes each tree gives. The classification is achieved by having the most votes overall in the forest, and in the case of regression, it takes the average of the outputs generated by different trees [26] (See Figure 2).

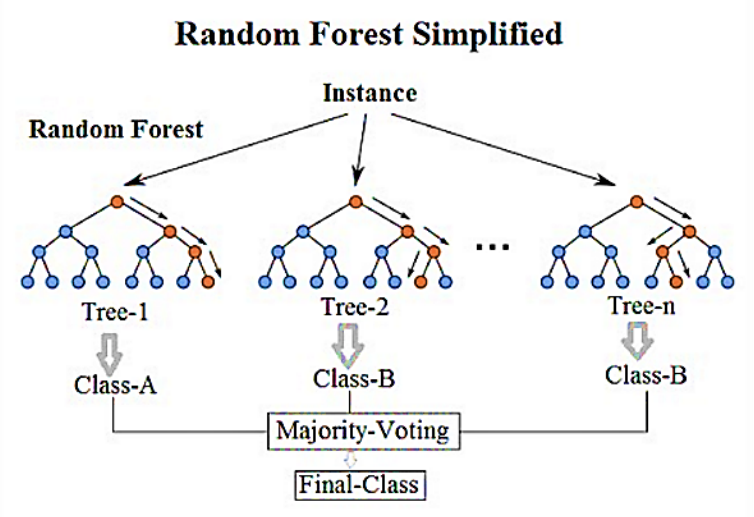

Figure 2. Random forest simplified representation.

Random Forest was picked here because it can be used for both classification and regression tasks and will not overfit the model. In addition, it can handle large data sets with higher dimensionality [25].

A major disadvantage of Random Forests is their complexity. They are less intuitive and are much harder and time-consuming to be built in comparison with decision trees. However, on the bright side, a single decision tree tends to overfit the data, while RF's process of averaging or joining the results of different decision trees aids to overcome the problem of overfitting. Moreover, they are extremely flexible and tend to offer high accuracy outputs [26].

3.2. Support Vector Machine

The final algorithm is another powerful and widely used learning algorithm, the Support Vector Machine (SVM). SVM looks at the extremes of the data sets and draws a decision boundary, also known as a hyperplane, near the extreme points in the dataset. Essentially, the SVM algorithm is a frontier which best segregates the two classes [27]. SVM can be considered as an extension of the perceptron. However, the perceptron algorithm minimises misclassification errors, whereas, in SVMs, the optimization objective is to maximize the margin (see Figure 3).

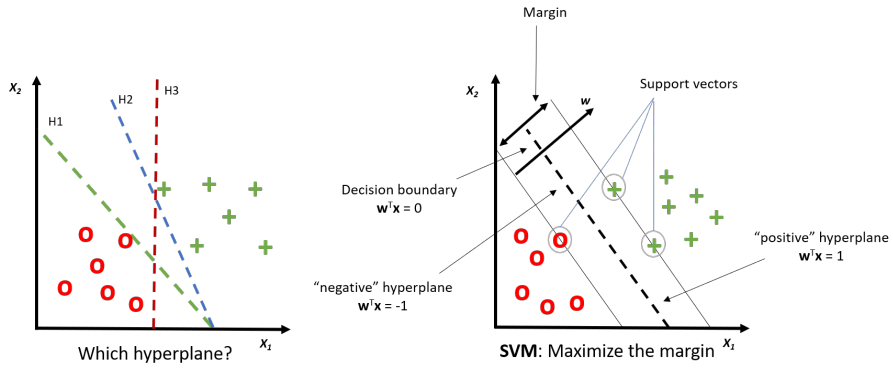

Figure 3. Demonstration of Support Vector Machine (SVM).

SVMs work with linearly separable datasets; hence, if the dataset is not linearly separable, it is transformed into a higher dimensional space, so the maximum-margin can be plotted [28]. The problem with transformation into higher dimensional feature space is that it is computationally expensive. This issue can be avoided by using a "kernel trick" to reduce the computational cost: a function that takes inputs as vectors in the original space and returns the dot product of the vectors in the feature space. This is called a kernel function, also referred to as the kernel trick. Using a kernel function, we can apply the dot product between two vectors so that every point is mapped into a high dimensional space via some transformation [29]. A drawback of SVM is that it does not directly provide probability estimates. On the other hand, SVM is versatile in specifying decision functions using different Kernels. It is also effective and memory efficient in high dimensional spaces. They provide high precision, making them applicable to datasets with a large number of features. Accordingly, they are highly used in studying the air quality in urban areas of cities, image interpolation, as well as medical classification [30].

4. Experimental Study

4.1. Data Sets

As the aim of this research is to test the capabilities of current IoT devices, they can be deployed in various sectors to aid and improve real-world operations. Each of the chosen data sets are real data collected by various recent research/study. For further information about the data type, size and source, check the Table 1 below.

The selected ten data sets [31] are divided into two categories, and these are classification data sets and regression data sets. All data sets are different from each other except for one: "Energy Efficiency", as this is being used both for classification and regression.

4.1.1. Regression Data Sets

1. Air Quality Data Set—The data set contains hourly response averages coming from a gas multi-sensor device deployed on the field in an Italian city. The goal is to estimate benzene levels in an urban pollution monitoring scenario.
2. Concrete Compressive Strength Data Set—The objective here is to use age and ingredients (cement, ash, water, etc.) found in the data set to determine the concrete material's strength.
3. Energy efficiency Data Set—The data set contains eight features of energy analysis of 12 buildings. The aim is to visualize and assess the performance of heating and cooling operations of houses.
4. Individual household electric power consumption Data Set—The data set is comprised of one-minute sampling rate measurements of electric power consumption of a household over four years. This data set will be used to detect bizarre usage patterns of the household.
5. Yacht Hydrodynamics Data Set—Data set containing dimensions and velocity of yachts for evaluating the ships performance and for estimating the required propulsive power.

4.1.2. Classification Data Sets

1. Autism Screening Adult Data Set—Data on autistic spectrum disorder screening of adults. It will be used to determine the state of a patient in the clinic for early diagnosis.
2. Breast Cancer Data Set—The data set contains instances described by nine attributes, some of which are linear, and some are nominal, and will be used to classify the type of cancer for a given patience at the spot.
3. Energy efficiency Data Set—The data set contains eight features of energy analysis of 12 buildings. The aim is to visualize and assess the performance of heating and cooling operations of houses.
4. Glass Identification Data Set—Data set that consists of six types of glass, defined in terms of their oxide content (i.e., Na, Fe, K, etc). It will be used to predict the types of glass motivated by criminology investigation.
5. Leaf Data Set—The data set contains shape and texture features extracted from 40 different plant leaf specimens. It will aid in the classification purposes of new species on the field.

The characteristics of all data sets are shown in Table 1.

Table 1. Number of instances and attributes for each data set used to test classification and regression.

Data Set Name	Category	No. of Instances	No. of Attributes
Air quality	Regression	9358	15
Concrete compressive strength	Regression	1030	9
Energy efficiency	Regression	768	8
Indidual household electric power consumption	Regression	2,075,259	9
Yacht hydrodynamics	Regression	308	7
Autism screening adult	Classification	704	21
Breast cancer	Classification	286	9
Energy efficiency	Classification	768	8
Glass identification	Classification	214	10
Leaf	Classification	340	16

4.2. Machine Learning Algorithms

The aim is to deploy the powerful ML algorithms on the IoT devices to accommodate solutions for various setbacks that the IoT devices can overcome with the right tools. The research targets

IoT devices equipped with the best possible instruments, so they can tackle current and future real-world difficulties.

Likewise, less powerful techniques such as linear regression and decision trees have already been implemented on IoT [18]. Hence, the goal is to stress the devices to the furthest extent, acquire utmost usage and extend their benefit in their fields of use.

4.3. Test Computer, IoT Device and Software

We ran the experiments on a PC with typical specification. It has an Intel Core i7-6700HQ (Intel Corporation, California, USA), which is a quad-core processor based on the Skylake architecture that was introduced in September 2015. It is a ubiquitous CPU running on most of the current personal computers, which makes it an ideal choice to develop the algorithms on. For the IoT device, Raspberry Pi 3 model B was chosen for the experiment. This is the latest version of Raspberry Pi available on the market today, which is a cheap, functional, bare metal platform ideal for the research. It is small, and has a Quad Core 1.2 Ghz at disposal with 1 GB of RAM and onboard connectivity through wireless LAN and Bluetooth. The Figure 4 shows the Raspberry Pi 3.

Figure 4. Raspberry Pi 3 model B was used for the experiment.

Anaconda platform was selected since it is one of the most popular Python data science platforms. Out of the box, it contains lead open source projects like NumPy and SciPy.

4.4. Measurement Tools

4.4.1. Accuracy

For classification, the 'accuracy score' metric from scikit-learn will be used. This uses two variables total and count, which are used to calculate the frequency at which predictions match the labels. This function simply divides the total by the count [32]:

$$accuracy = \frac{y\_predicted}{y\_true}. \tag{1}$$

For regression, the 'R^2' (coefficient of determination) function metric will be used. It is defined as $(1- u/v)$, where 'u' is the residual sum of squares and 'v' is the total sum of squares. The best possible score is 1.0 [32]:

$$R^2 = 1 - \frac{u}{v}, \tag{2}$$

$$u = \sum (y_{true} - y_{pred})^2 \text{ and } v = \sum (y_{true} - \mu_{pred})^2$$

4.4.2. Speed

As a basic measurement and to retain consistency, the speed of all algorithms will be measured in seconds. This will be measured using the time module from python.

On the Raspberry Pi that runs Linux, the Python module returns the current processor time as a floating point number expressed in seconds. The precision, and, in fact, the very definition of the meaning of "processor time" depends on that of the C function of the same name, but, in any case, this is the function to use for benchmarking Python or timing algorithms [33].

On the PC that runs Windows, this function returns wall-clock seconds elapsed since the first call to this function, as a floating point number, and based on the Win32 function QueryPerformanceCounter, the resolution is typically better than one microsecond [33].

In order calculate the common speed measurements with confidence, we ran each algorithm 20 times with the same data set, and then we calculated the average time which is reported in the results section of this article.

4.4.3. Power Consumption

The power consumption will be measured using the Muker USB multimeter (M. way USB Multimet Po, China). It is a small yet powerful device that provides all the information required for this use case and much more: Current (A), Voltage (V), Energy (Wh), Resistance (Ω), Capacity (mAh) and Power (W) going from the socket to the device that is charging from it in real time. It also shows the Cumulative time (Seconds), Internal Temperature (°C) of the device. Figure 5 depicts the Muker USB multimeter device whilst in operation.

Figure 5. The Muker USB multimeter which was used to measure the energy consumption of running MLs algorithms on the Raspberry Pi 3.

The power consumption of the IoT device for a given time while running the algorithm will be monitored. This is achieved by measuring the algorithm's excess usage of power in comparison to when the device runs in idle mode. This is calculated as amps per second. Then, this will be fed into another equation, resulting in its power usage over a specific time, which is measured in joules:

$$J = W \times s, \quad \text{where} \quad W = V \times A, \tag{3}$$

$$J = joules, \quad W = watts, \quad s = seconds, \quad V = volts, \quad A = amps. \tag{4}$$

In addition, in this case $V = 5$, as that is the electric potential difference of the charger used.

4.5. Implementation

The aim here was to use the time efficiently and develop the models swiftly as it involved three different models to be implemented, which would later be individually customised according to the ten data sets.

4.5.1. Models

At a glance, building a good machine learning model is no different than building any other product: starting with ideation, where the problem being tackled is investigated and some potential approaches are considered. Once a clear direction is identified, a prototype solution is formed. Later, this prototype get tested to check if it meets the requirements. The process can be broken down to these four stages:

Ideation: Align on the main problem to solve, and consider the possible data inputs for the solution.

Data preparation: Select and gather the data in a suitable format for the model to process and learn from.

Prototyping and testing: Build a model to solve the problem, test the performance and iterate until achieving satisfactory results.

Productisation: Stabilise and scale the model for all data sets to produce useful outputs in the testing environment.

Thus, the initial goal was to build the models and have them working with at least one of the data sets. This way, they would be ready for tuning, and so the testing phase would commence sooner.

Scikit-learn library will be used to build the three models.

- For Multi-Layer Perceptron 'MLPClassifier' is used for classification and 'MLPRegressor' for regression;
- For Support Vector Machine 'svm' and 'SVR' are used;
- and, finally, for Random Forest, 'RandomForestClassifier' and 'RandomForestRegressor' are used.

4.5.2. Pre-Processing and Tuning

The goal of this phase is to transform the data into a form that can be plugged as input into the built models and then tune the models to achieve utmost performance.

Data cleansing is a valuable process that can help save time and increase efficiency [34]. It is done by spotting and rectifying inaccurate or corrupt data from the data set. Furthermore, in this case, the null values of certain features found in the data sets were deleted as they were not in abundance and plentiful of data was left to be iterated on.

To further cleanse the data, feature selection techniques will be used to identify and remove redundant and irrelevant data without causing dramatic loss of data [35]. This is also useful on multiple fronts as it can reduce overall training times. Moreover, it can potentially minimise overfitting and increasing generalisability [36]. Hence, after thorough reading of the research papers that each data set used, several features that had very little impact were dropped. A good example for this is 'Autism Adult Screening' data set where features like the individual answers to particular questions were dropped, as the sum of the remaining of the questions was sufficient to make a decision.

Because some of the selected data sets are too large, which is time-consuming to process, and usually requires high computational power, the sampling technique was used on some data sets to select a sample from the data while maintaining accurate representation of the entire population [37]. For example, the 'Individual household electric power consumption' data set contained over two million entries and running through it on the Raspberry Pi would have been unfeasible. Therefore, in such data sets, data was hourly sampled into average and total power consumed. This sampling method reduced the number of entries to around ten thousand entries.

Tuning

Open-source packages like scikit-learn enable the use of powerful ML tools quickly and easily, but tuning these models is often a non-intuitive, time-consuming process. The tuneable hyperparameters of these models themselves can greatly influence their accuracy [38].

Accordingly, grid search was used to tune the three models on each data set. The same accuracy parameters (*accuracy* and R^2) were used as the performance metrics and it was measured by cross-validation on the data sets. For MLP, three of the hidden layers of size varying from 3 to 90 together with three solvers (lbfgs, sgd and adam) and four activation functions (identity, logistic, tanh, relu) were fed in the grid search to find the best hyper-parameters. For SVM, the following hyper-parameters were modified with grid search: C, coef0, degree, epsilon, kernel and tol. Lastly, for RF, only the number of trees ranging from 100 to 200 trees was fed into the tuning algorithm.

Consequently, with the modified hyper-parameters, an overall accuracy score of over 85% was achieved throughout the three models on all data sets.

Script Separation

Throughout the implementation process mentioned above, the code was broken down and separated into different scripts (files). This allowed concentrating into one problem at a time, and negated the single point of failure while debugging.

Therefore, pre-processing, tuning, saving the model through pickling and extracting the model using unpickling (more on this in the next subsection) has been separated into different scripts in the folder.

Pickling and Inference

Pickling (and unpickling) also known as "serialization" is a Python module that implements binary protocols for serializing and de-serializing an object structure [39]. Thus, using pickling, the finalised models were converted into binary files on the PC, which were then uploaded to the Raspberry Pi. On the Pi, they were unpickled and later used for inference for the testing phase.

5. Experimental Results

5.1. Testing

The testing phase stands as the main focal-point of the research where the capabilities of the IoT device (Raspberry Pi) will be measured and evaluated. Each of the three models will be tested on all data sets ten times, obtaining a reliable mean as the result. On each run, the accuracy of the model on the data set and the average execution time will be recorded on both devices. While excess power consumption per second and total power consumption throughout the execution time will only be recorded on the Raspberry Pi. This will consist of three stages.

The models run from scratch on the PC to test training and inference, measuring their accuracy and execution time. Training will be tested by running the complete model from start. This will give us a baseline to compare with and distinguish the difference of running the same algorithm on the smaller IoT device.

The models have been designed to run on the Raspberry Pi to test training and inference, measuring accuracy and execution time plus energy consumption. As mentioned in Section 4.4.3, their excess power usage (shown below) and the total power usage through the duration of the algorithm.

$$\text{Algorithm's Excess Usage} = \text{Total Power consumption of RasPi} - \text{Idle power consumption of RasPi.}$$

The inference will be tested on the IoT device inference by unpickling the earlier uploaded binary files with the models already trained. This test will resemble the real-life use case of the device where

it would collect data on the field and give a response by inference quickly, and without consuming excessive amount of power. This experiment will be conducted by running the ML algorithm on a selected random section/sample of 100 instances from the data set. Thus, to the execution time of inference on a single instance:

$$\text{Single Instance Inference} = \text{Execution time (100 instances)}/100.$$

5.2. Evaluation

Figures 6 and 7 depict the training times required by the computer and Raspberry Pi. As expected, the PC, running an i7 with a clock speed that is double the speed of the processor found in the IoT device, outperformed its smaller counterpart by a great amount.

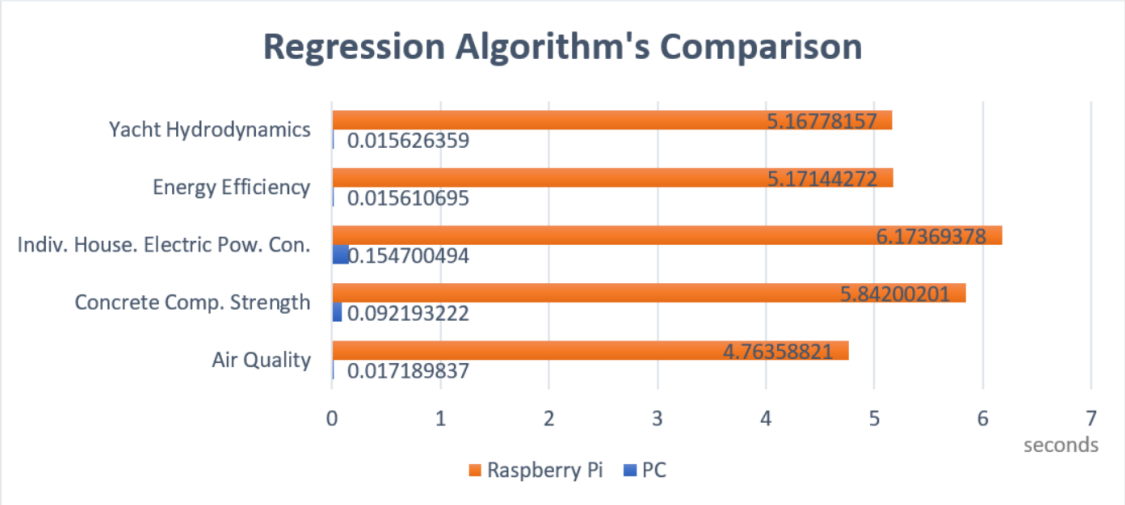

Figure 6. Random Forest-Execution Time-Raspberry Pi vs. PC.

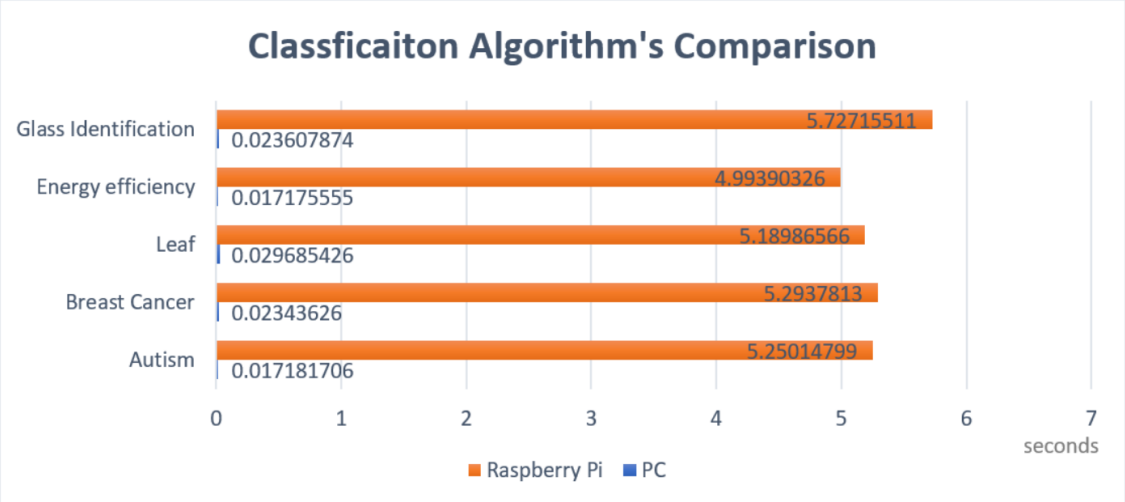

Figure 7. Random Forest-Execution Time-Raspberry vs. PC.

Figure 8 demonstrated that running only inference on the small IoT device is feasible as the average run time per instance is 0.05 s. The graph shows the run time for training and inference of the three different algorithms on the IoT device. It is apparent that algorithms such as SVM required a huge amount of time for training the algorithms to optimise the separating hyperplane to Multi-Layer

Perception and Random Forest. Interestingly, the time required for the inference is almost identical among all algorithms subject for this experiment.

Figure 8. Algorithm training and inference speed comparison.

Figure 9 displays the overall accuracy of the three models on classification and regression data sets. SVM proved to be better at classification problems, while MLP showed a better performance for regression in this case. However, on the other hand, RF outperformed both algorithms in classification and regression problems.

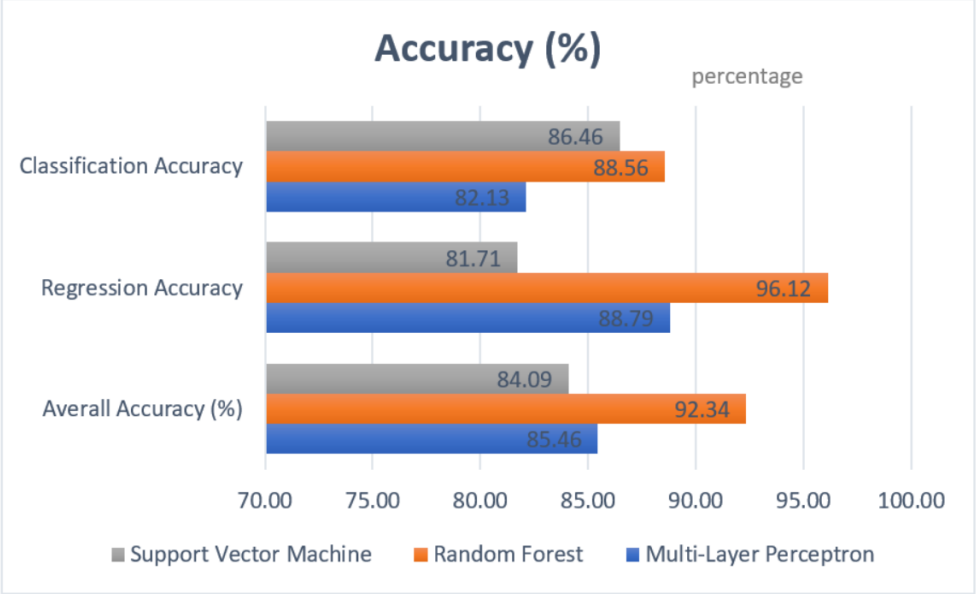

Figure 9. Algorithm accuracy comparison.

Figure 10 depicts the measurement of the operating electrical current running through the Raspberry Pi when running Multi-layer perception, Random Forest or SVM algorithms. The used Muker USB multimeter measure this amps per second. The algorithms' excess power consumption adds to the statement made earlier about the feasibility of running inference on the IoT device.

Their consumption is less than 15 amps per second, which is very feasible. Other tests were conducted where the power consumption was measured whilst browsing the web, watching videos or playing games, and all proved to consume at least 18 amps or more.

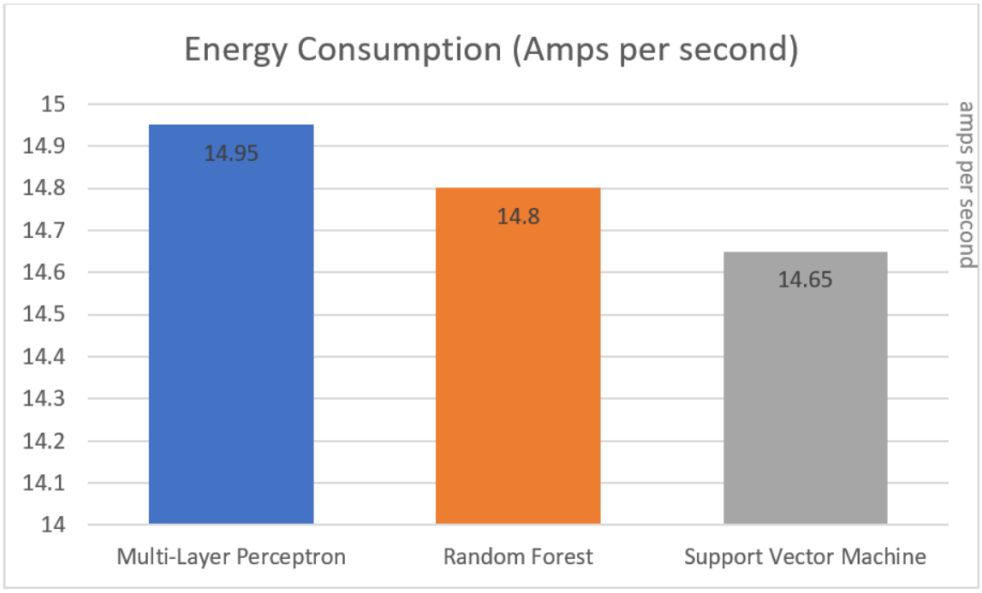

Figure 10. Algorithm energy consumption comparison.

Figure 11 displays the total energy consumption, which was calculated as discussed in Section 4.4.3. This is the total energy consumed by the Raspberry Pi by running the machine learning algorithms when training and inference on all the data sets. This is measured in Joules which is a unit of energy needed to move one ampere through one ohm of resistance for one second. In training, due to the time it takes, SVM has the biggest consumption by an extensive margin. Here again RF proved to be the best option as it surpasses its rival algorithms both in training and inference.

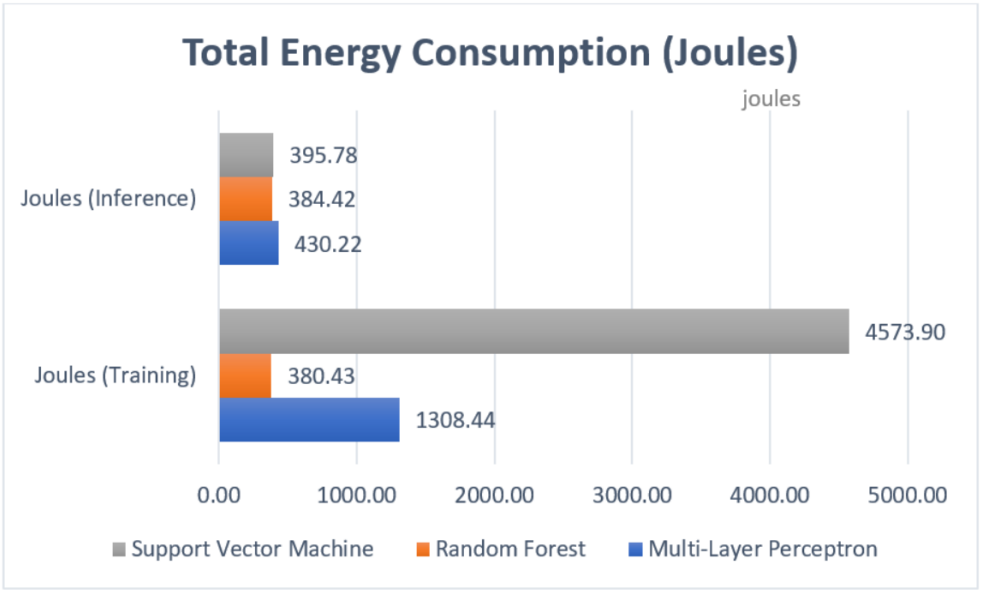

Figure 11. Algorithm total energy consumption comparison.

6. Discussion

Looking into the results, Random Forest is the overall winner in all categories. It is quick to train, versatile, provides a pretty good indicator of feature importance and it is an excellent benchmark model.

However, those concerned with fast inference can still consider the Support Vector Machine and the Multi-layer Perception. Although, their training is long, the inference is still fast and very close to the one of the Random Forest. In these cases, we suggest a hybrid approach where training takes place on a high performance computer to minimise the the time required for training. From the results, it is apparent that SVM and MLP are best to run on lightweight IoT devices, but at the cost of having slightly lower accuracy (8–14%).

These tests and results can also be considered as a starting guide for those looking to deploy machine learning models on IoT edge devices and are not sure which ML algorithms to choose. In addition, all information regarding the software and hardware is given, so the tests can be replicated with ease.

7. Conclusions

In the paper, the feasibility of running a number of ubiquitous machine learning algorithms on an IoT edge device was questioned. As a result of the conducted tests, comparing performances of all three algorithms namely Multi-layer perception, Random Forest and SVM algorithms, the Random Forest algorithm proved to be slightly faster in speed and widely better in accuracy. However, looking at the research from a wider perspective, all of the algorithms' accuracy exceeded 80%, the time required to run them for inference was below one millisecond and they all had moderately low energy consumption.

Hence, the conducted research proves that running the state-of-the-art machine learning algorithms is feasible to be run on edge IoT devices for all purposes.

As a recommendation, the idea of implementing more complex and taxing algorithms such as Deep Learning, using platforms like TensorFlow, on these small devices would be the next step in revealing their power in more detail. Work on deployment of pruned deep learning models in recent research looks promising.

Nevertheless, the future of IoT seems much more fascinating as billions of things will be communicating to each other and human intervention will become least. IoT will bring a macro shift in the way we live and work.

Author Contributions: For research articles with several authors, Conceptualization, S.B. and M.M.G.; Methodology, M.T.Y., S.B. and M.M.G.; Software, M.T.Y.; Validation, M.T.Y., S.B. and M.M.G.; Formal Analysis, M.T.Y.; Investigation, M.T.Y.; Resources, M.T.Y., S.B. and M.M.G.; Data Curation, M.T.Y.; Writing—Original Draft Preparation, M.T.Y.; Writing—Review & Editing, S.B. and M.M.G.; Visualization, M.T.Y.; Supervision, S.B. and M.M.G.; Project Administration.

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

References

1. Green, H. *The Internet of Things in the Cognitive Era: Realizing the Future and Full Potential of Connected Devices*; Technical Report; IBM Watson IoT: New York, NY, USA, 2015.
2. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
3. Evans, D. The Internet of Things: How the Next Evolution of the Internet is Changing Everything. *CISCO White Paper* **2011**. *1*, 1–11.
4. Manyika, J.; Chui, M.; Bisson, P.; Woetzel, J.; Dobbs, R.; Bughin, J.A.D. *Unlocking the Potential of the Internet of Things*; Technical Report; McKinsey Global Institute: New York, NY, USA, 2015.
5. Lee, I.; Lee, K. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Bus. Horiz.* **2015**, *58*, 431–440. [CrossRef]

6. Yoo, Y.; Henfridsson, O.; Lyytinen, K. The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research. *Inf. Syst. Res.* **2010**, *21*, 724–735. [CrossRef]
7. Wortmann, F.; Fluchter, K. Internet of Things: Technology and Value Added. *Bus. Inf. Syst. Eng.* **2015**. *57*, 221–224.
8. Fleisch, E.; Weinberger, M.; Wortmann, F. *Business Models for the Internet of Things-Bosch IoT Lab White Paper*; Universität St. Gallen: St. Gallen, Switzerland, 2014.
9. Kargupta, H.; Park, B.H.; Pittie, S.; Liu, L.; Kushraj, D.; Sarkar, K. MobiMine: Monitoring the stock market from a PDA. *ACM SIGKDD Explor. Newsl.* **2002**, *3*, 37–46. [CrossRef]
10. Kargupta, H.; Bhargava, R.; Liu, K.; Powers, M.; Blair, P.; Bushra, S.; Dull, J.; Sarkar, K.; Klein, M.; Vasa, M.; et al. VEDAS: A mobile and distributed data stream mining system for real-time vehicle monitoring. In Proceedings of the 2004 SIAM International Conference on Data Mining, Lake Buena Vista, FL, USA, 22–24 April 2004; pp. 300–311.
11. Gaber, M.M.; Philip, S.Y. A holistic approach for resource-aware adaptive data stream mining. *New Gener. Comput.* **2006**, *25*, 95–115. [CrossRef]
12. Gaber, M.M. Data stream mining using granularity-based approach. In *Foundations of Computational, Intelligence Volume 6*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 47–66.
13. Gaber, M.M.; Krishnaswamy, S.; Zaslavsky, A. On-board mining of data streams in sensor networks. In *Advanced Methods for Knowledge Discovery From Complex Data*; Springer: Goldaming, UK, 2005; pp. 307–335.
14. Gaber, M.M.; Gomes, J.B.; Stahl, F. Pocket Data Mining. In *Big Data on Small Devices*; Series: Studies in Big Data; Springer: Cham, Switzerland, 2014.
15. Anwar, S.; Hwang, K.; Sung, W. Structured pruning of deep convolutional neural networks. *ACM J. Emerg. Technol. Comput. Syst. (JETC)* **2017**, *13*, 32. [CrossRef]
16. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
17. Fernández-Delgado, M.; Cernadas, E.; Barro, S.; Amorim, D. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.* **2014**, *15*, 3133–3181.
18. Chavan, G.; Momin, B. An integrated approach for weather forecasting over Internet of Things: A brief review. In Proceedings of the 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 10–11 February 2017; pp. 83–88. [CrossRef]
19. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *arXiv* **2015**, arXiv:1510.00149.
20. Ata, R. Artificial neural networks applications in wind energy systems: A review. *Renew. Sustain. Energy Rev.* **2015**, *49*, 534–562. [CrossRef]
21. Raschka, S. *Python Machine Learning: Effective Algorithms for Practical Machine Learning and Deep Learning*, 2nd ed.; Packt Publishing Limited: Birmingham, UK, 2017.
22. Tu, J.V. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *J. Clin. Epidemiol.* **1996**, *49*, 1225–1231. [CrossRef]
23. Tang, J.; Sun, D.; Liu, S.; Gaudiot, J.L. Enabling Deep Learning on IoT Devices. *Computer* **2017**, *50*, 92–96, doi:. [CrossRef]
24. Ng, A. Neural Networks and Deep Learning-Coursera. 2017. Available online: https://www.coursera.org/learn/neural-networks-deep-learning/ (accessed on 27 August 2018).
25. Mestre, D.; Fonseca, J.M.; Mora, A. Monitoring of in-vitro plant cultures using digital image processing and random forests. In Proceedings of the 8th International Conference of Pattern Recognition Systems (ICPRS 2017), Madrid, Spain, 11–13 July 2017; pp. 1–6. [CrossRef]
26. Dogru, N.; Subasi, A. Traffic accident detection using random forest classifier. In Proceedings of the 2018 15th Learning and Technology Conference (L T), Jeddah, Saudi Arabia, 25–26 February 2018; pp. 40–45, doi:. [CrossRef]
27. Witten, I.H.; Frank, E.; Hall, M.A. *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed.; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2011.
28. Tian, Y.; Qi, Z.; Ju, X.; Shi, Y.; Liu, X. Nonparallel Support Vector Machines for Pattern Classification. *IEEE Trans. Cybern.* **2014**, *44*, 1067–1079. [CrossRef] [PubMed]

29. Kruczkowski, M.; Szynkiewicz, E.N. Support Vector Machine for Malware Analysis and Classification. In Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Warsaw, Poland, 11–14 August 2014; Volume 2, pp. 415–420. . [CrossRef]
30. Amin, S.; Singhal, A.; Rai, J.K. Identification and classification of neuro-degenerative diseases using statistical features and support vector machine classifier. In Proceedings of the 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Delhi, India, 3–5 July 2017; pp. 1–8. . [CrossRef]
31. Asuncion, A.; Newman, D. UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 27 August 2018).
32. Scikit-Learn. Documentation of Scikit-Learn 0.19.1. Available online: https://github.com/amueller/scipy-2017-sklearn (accessed on 27 August 2018).
33. Van Rossum, G. *Python Tutorial*; Technical Report CS-R9526; Centrum voor Wiskunde en Informatica (CWI): Amsterdam, The Netherlands, 1995.
34. Rahm, E.; Do, H.H. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*108 **2000**, *24*, 3–13.
35. Bermingham, M.L.; Pong-Wong, R.; Spiliopoulou, A.; Hayward, C.; Rudan, I.; Campbell, H.; Wright, A.F.; Wilson, J.F.; Agakov, F.; Navarro, P.; et al. Application of high-dimensional feature selection: Evaluation for genomic prediction in man. *Sci. Rep.* **2015**, *5*, 10312. [CrossRef] [PubMed]
36. Guyon, I.; Elisseeff, A. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
37. Choudhury, M.D.; Lin, Y.R.; Sundaram, H.; Candan, K.S.; Xie, L.; Kelliher, A. How Does the Data Sampling Strategy Impact the Discovery of Information Diffusion in Social Media? In Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media (ICWSM), Washington, DC, USA, 23–26 May 2010.
38. Kong, W.; Dong, Z.Y.; Luo, F.; Meng, K.; Zhang, W.; Wang, F.; Zhao, X. Effect of automatic hyperparameter tuning for residential load forecasting via deep learning. In Proceedings of the 2017 Australasian Universities Power Engineering Conference (AUPEC), Melbourne, Australia, 19–22 November 2017; pp. 1–6. [CrossRef]
39. Rossum, G. Python Reference Manual. Available online: https://docs.python.org/2.0/ref/ref.html (accessed on 27 August 2018).

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).