nanomaterials

Article

Comparative Analysis of Machine Learning Models for Nanofluids Viscosity Assessment

Mohammadhadi Shateri [1], Zeinab Sobhanigavgani [1], Azin Alinasab [1], Amir Varamesh [2], Abdolhossein Hemmati-Sarapardeh [3,4,*], Amir Mosavi [5,6,7,*] and Shahab S [8,9,*]

1 Department of Electrical & Computer Engineering, McGill University, Montreal, QC H3A 2K6, Canada; mohammadhadi.shateri@mail.mcgill.ca (M.S.); zeinab.sobhanigavgani@mail.mcgill.ca (Z.S.); azin.alinasab@polymtl.ca (A.A.)
2 Department of Chemical & Petroleum Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada; amir.varamesh@ucalgary.ca
3 Department of Petroleum Engineering, Shahid Bahonar University of Kerman, Kerman 7616913439, Iran
4 College of Construction Engineering, Jilin University, Changchun 130600, China
5 Faculty of Civil Engineering, Technische Universität Dresden, 01069 Dresden, Germany
6 School of Economics and Business, Norwegian University of Life Sciences, 1430 Ås, Norway
7 Institute of Automation, Kando Kalman Faculty of Electrical Engineering, Obuda University, 1034 Budapest, Hungary
8 Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam
9 Future Technology Research Center, College of Future, National Yunlin University of Science and Technology, 123 University Road, Section 3, Douliou, Yunlin 64002, Taiwan
* Correspondence: hemmati@uk.ac.ir (A.H.-S.); amir.mosavi@mailbox.tu-dresden.de (A.M.); shamshirbandshahaboddin@duytan.edu.vn (S.S.)

Received: 23 June 2020; Accepted: 31 August 2020; Published: 7 September 2020

Abstract: The process of selecting a nanofluid for a particular application requires determining the thermophysical properties of nanofluid, such as viscosity. However, the experimental measurement of nanofluid viscosity is expensive. Several closed-form formulas for calculating the viscosity have been proposed by scientists based on theoretical and empirical methods, but these methods produce inaccurate results. Recently, a machine learning model based on the combination of seven baselines, which is called the committee machine intelligent system (CMIS), was proposed to predict the viscosity of nanofluids. CMIS was applied on 3144 experimental data of relative viscosity of 42 different nanofluid systems based on five features (temperature, the viscosity of the base fluid, nanoparticle volume fraction, size, and density) and returned an average absolute relative error (AARE) of 4.036% on the test. In this work, eight models (on the same dataset as the one used in CMIS), including two multilayer perceptron (MLP), each with Nesterov accelerated adaptive moment (Nadam) optimizer; two MLP, each with three hidden layers and Adamax optimizer; a support vector regression (SVR) with radial basis function (RBF) kernel; a decision tree (DT); tree-based ensemble models, including random forest (RF) and extra tree (ET), were proposed. The performance of these models at different ranges of input variables was assessed and compared with the ones presented in the literature. Based on our result, all the eight suggested models outperformed the baselines used in the literature, and five of our presented models outperformed the CMIS, where two of them returned an AARE less than 3% on the test data. Besides, the physical validity of models was studied by examining the physically expected trends of nanofluid viscosity due to changing volume fraction.

Keywords: nanofluid viscosity; experimental data; machine learning; deep learning; nano; nanomaterials; nanofluid; artificial neural network; data science; big data; ensemble models; artificial intelligence; computational fluid dynamics; material design; computational mechanics

1. Introduction

Conventional working mediums, such as water, ethylene glycol, etc., cannot provide enough efficiency in industrial processes since they suffer from relatively poor heat transfer characteristics. Investigations have proved that the suspension of solid particles into traditional fluids enhances their heat transfer capability. Based on this idea and with advances in nanotechnology, a new generation of heat transfer fluid, named as nanofluids, was invented. Nanofluids are the suspension of nanoparticles (such as metals, metal oxides, non-metals, etc.) whose dimensional size is typically in the range of 1–100 nm in a base fluid (such as water, oil, ethylene glycol, etc.) [1,2]. In comparison to conventional working mediums, nanofluids can provide much more efficient thermo-physical properties, which are the controlling factors for flow behavior and heat transfer ability of working fluids [3–6]. Therefore, the thermo-physical properties of nanofluids, including viscosity, are important parameters that must be evaluated before any application of them.

The viscosity of fluids is a measure of its resistance to flow. Nanofluids viscosity directly impacts the required pumping power and associated pressure drop in any energy system. In addition, the amount of heat augmentation in convection is strictly influenced by nanofluid viscosity. Moreover, the viscosity of nanofluids should be determined accurately since the value of critical dimensionless numbers, including Brinkman number, Prandtl number, and Reynolds number, are related to the value of viscosity [7,8]. Considering the importance of the viscosity of nanofluids, over the past years, different methods and equations were introduced to predict that. Einstein [9] was the first one who proposed an equation for the estimation of the viscosity of floating small particles with low volume fraction into a fluid. After Einstein's [9] pioneering work, many other authors tried to propose new methods for the prediction of viscosity of suspended particles, either by modifying Einstein's [9] equation or developing a new method. Brinkman [10], Lundgren [11], Frankel and Acrivos [12], Batchelor [13], Thomas and Muthukumar [14], Chen et al. [15], Maiga et al. [16] are among the most well-known developed models. A recent review of the available models for the prediction of viscosity of nanofluids was published by Varamesh and Hemmati-Sarapardeh [17]. They reviewed almost all the important models for the viscosity of nanofluids and categorized them into three main types, theoretical models, empirical equations, and computer-aided models.

Recently, artificial intelligent models, such as an artificial neural network (ANN), radial basis function neural network (RBF-NN), etc., which have powerful nonlinear regression ability and can theoretically model complex relations, have been widely utilized to model thermo-physical properties of nanofluids. These strong data-driven modeling tools can determine the complex nonlinear dependency of an output parameter to its input variables with high speed and low computational cost [6,18]. Karimi et al. [19] firstly proposed an artificial neural network based on a genetic algorithm (GA). They gathered 381 experimental data from eight different types of nanofluids for the development of the model by considering the input variables, including particle volume concentration, temperature, the viscosity of fluid base, particle size, and density ratio of base fluid to the nanoparticle. Their statistical results showed that the predicted values and experimental data were in good agreement. The mean average relative error of the model was 2.48%. Since then, several other computer-aided data-driven models have been developed by considering different intelligent modeling approaches, including fuzzy C-means clustering-based adaptive neuro-fuzzy system (FCM-ANFIS), hybrid self-organizing polynomial neural networks (PNN) based on group method of data handling (GMDH), least-square support vector machine (LSSVM), radial basis function neural networks (RBF-NN), genetic algorithm-polynomial neural network (GA-PNN), multilayer perceptron neural networks (MLP-NNs), gene expression programming (GEP) [8,20–35]. However, the most accurate model with a wide range of applicability for the prediction of viscosity of nanofluids was developed by Hemmati-Sarapardeh et al. [1], based on a committee machine intelligent system (CMIS). They introduced a machine learning model based on the combination of seven baselines, including four MLP-NNs, two RBF-NNs, and an LSSVM. Each of the MLP-NNs was optimized with different algorithms, including Levenberg–Marquardt (LM), resilient backpropagation (RB), Bayesian regularization (BR), and scaled conjugate gradient (SCG). The RBF-NNs were optimized by utilizing particle swarm optimization (PSO) and the genetic algorithm (GA). The LSSVM model was optimized with coupled simulated annealing (CSA). The proposed CMIS by Hemmati-Sarapardeh et al. [1] was applied on 3144 experimental data of relative viscosity of 42 different nanofluid systems by considering temperature, the viscosity of the base fluid, nanoparticle volume fraction, size, and density as input variables to predict relative viscosity as the output variable. The obtained results by Hemmati-Sarapardeh et al. [1] showed good agreement with experimental data with an average absolute relative error of 3.95% between predicted relative viscosity values and corresponding experimental data. Besides, the developed CMIS outperformed all of the investigated available models, and unlike the previously available models, the developed CMIS showed high accuracy over the whole range of input variables. Hemmati-Sarapardeh et al. [1] also analyzed the quality of the gathered 3144 experimental data points and showed that all of the data points had very good reliability except a small percent of them.

This study aimed to improve the accuracy and efficiency of the CMIS model developed by Hemmati-Sarapardeh et al. [1], which is the best available model. For this purpose, in this study, using the same data set gathered by Hemmati-Sarapardeh et al. [1], the two main baselines used by them, including a multilayer perceptron (MLP) network and LSSVM, were considered, and the hyperparameters of each were tuned. Moreover, a deeper MLP network, decision tree (DT) model, a random forest (RF) model, and extra trees model were used to improve the performance of the CMIS model developed by Hemmati-Sarapardeh et al. [1].

2. Data Collection

In this study, the most comprehensive data bank of nanofluid viscosity was used to develop reliable and accurate models. This data bank, which was already used in our previous study [1], covers 3144 experimental data points of nanofluid viscosity of 42 various types of nanofluid samples. Nanoparticle size, temperature, particle volume fraction, the density of the nanoparticles, and the viscosity of the base fluid were selected as the inputs of models, whereas the relative viscosity of nanofluid was assumed as the output. Details for the used data are summarized in Table 1. For more information about this data bank, readers can refer to our previous study [1]. The reason why the number of data points for some particles was low is that there are only limited studies regarding the evaluation of the viscosity of these particles, and we could not find more experimental data for these particles.

Table 1. The data bank of nanofluids used in this study.

	Al_2O_3	CuO	SiO_2	SiC	TiO_2	Fe_3O_4	MgO	$Mg(OH)_2$	Co_3O_4	Nanodiamond	ZnO
References	[36–51]	[39,49,52–55]	[41,56–60]	[61]	[36,44,46,47,62–65]	[66,67]	[24]	[68]	[69]	[70,71]	[72,73]
Base fluid	Water, DI water, Transformer oil, R11 refigerant, Polyalphaolefins, EG, EG/W 20:80 wt %, EG/W 40:60 wt %, EG/W 20:80 wt %, W/EG 60:40 vol %, W/EG 50:50 vol %, W/EG 40:60 vol %	Water, EG, PG/W 30:70 vol %, EG/W 60:40 wt %	Water, Ethanol, DI water, Transformer oil, EG, EG/W 25:75%, EG/W 50:50%, BG/W 20:80 vol %, BG/W 30:70 vol %	DI water	Water, DI water, EG, EG/W 20:80 wt %, BG/W 20:80 vol %, BG/W 30:70 vol %	Water, Toluene	EG	EG	EG	water, EG/W 20:80 wt %, EG/W 60:40 wt %, EG/W 40:60 wt %	EG
T (°C)	0–72	−35–67	19–80	30	9.85–80	20–60	20–70	23–65	10–50	0–60	10–50
φ (%)	0.01–10	0–9	0–8.4	0–3	0.2–10	0.04–2	0.1–5	0.1–2	0.9–5.7	0.2–1	0.25–5
d_p (nm)	8–120	11–152	7–190	100	6–50	10–13	21–125	20	17	11.83–19.27	4.6–48
ρ_P (gr/cm^3)	3.69–4	6.31	2.22–2.65	3.21	4.18–4.23	5.17–5.81	3.58	2.34	6.11	3.1	5.61–13.61
μ_{nf} (cp)	0.44–610.46	0.46–447.35	0.59–37.36	0.93–1.60	0.46–28.41	0.32–1.65	3.70–30.60	4.82–23.02	8.06–44.76	25.51	6.14–49.30
μ_{bf} (cp)	0.39–452.60	0.42–99.54	0.54–18.53	0.8	0.42–23.01	0.3–0.79	3.63–21.11	1.02–1.60	1.02–1.44	0.24–13.74	6.08–35.44
No. of data points	1197	500	278	5	308	121	198	35	25	357	122

W: Water, DI: Deionized, PG: Propylene Glycol, EG: Ethylene Glycol, BG: BioGlycol.

3. Model Development

In this section, all the models used in this study are introduced.

3.1. Multilayer Perceptron Network

The perceptron learning rule introduced by Rosenblatt [74] corresponds to a simple model consisting of one neuron (illustrated in Figure 1a) in which the output is a function of the sum of weighted inputs modified by an activation function or transfer function (f).

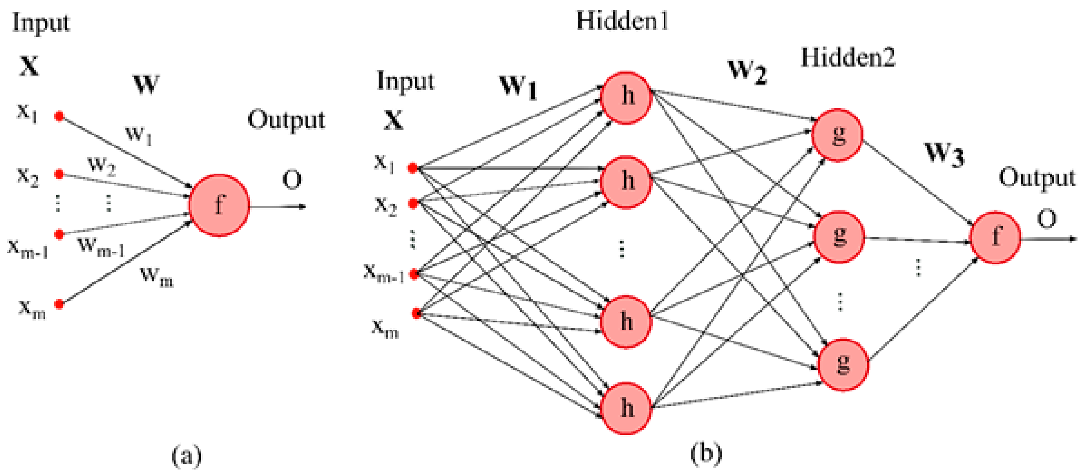

Figure 1. (**a**) Perceptron model, (**b**) Multilayer perceptron (MLP) with two hidden layers (Note that f, g, and h are activation functions, while the w represents the weights that the model needs to learn. x is also input of the model).

The multilayer perceptron (MLP) is a system of interconnected perceptron models. It is proved that the MLP can be trained to model every smooth, measurable function without concerning the data distribution [75]. In other words, using nonlinear activation functions gives the MLP networks the ability to approximate non-linear functions, while MLP with linear activation function can only model linear functions. For MLP networks, four kinds of hyperparameters, including the type of activation functions used in hidden and output layers, number of hidden layers, number of neurons in each hidden layer, and optimizer method, can be assumed and need to be determined before using the network for prediction. Tanh and Sigmoid are the most commonly used activation functions used in MLPs [76]. The optimizer has a pivotal role in the performance of MLP. In this study, four types of optimizers were used, and each of them has been explained in the next section.

3.2. Support Vector Machine for Regression

The support vector machine (SVM) was largely developed at AT&T Bell Laboratories. The support vector machine for regression (SVR) was first introduced in 1997 by H. Drucker et al. [77], following Cortes and Vapnik's work on support vector machines (SVM) [78]. SVR uses the same principles as SVM, with only a few minor differences. The SVR is a method to estimate a function that maps the input to a continuous number, which is the target output. SVM for classification does not apply a penalty on the points far away from the hyperplane as long as the class is predicted correctly; however, SVR needs the estimated function to be as close as possible to all target points. Therefore, SVR will apply a penalty on all points out of a predefined margin, ε, from the estimated function. In other words, SVR does not care about the errors as long as they are less than ε, but will not accept any deviations higher than ε. In the case of a linear function, $f(X) = w{\cdot}x + b$, we can describe the problem as a convex optimization problem:

$$minimize\ \frac{1}{2}\| w \|^2 \quad subject\ to \begin{cases} y_i - w{\cdot}x_i - b \le \varepsilon \\ w{\cdot}x_i + b - y_i \le \varepsilon \end{cases} \tag{1}$$

where y_i represents the true value of the ith sample, while the w and b are the weight and bias associated with the model, respectively. The value x is also input of the model.

Soft SVM: Similar to SVM, one can modify the optimization loss function by introducing slack variables, ζ_i and $\zeta_i{}^*$, to allow a soft margin when finding a function that can map all samples within the predefined margin, which is not feasible. The optimization problem will change as follows:

$$minimize\ \frac{1}{2}\| w \|^2 + C\sum_{i=1}^{l}\left(\zeta_i + \zeta_i^*\right) \quad subject\ to \begin{cases} y_i - w{\cdot}x_i - b \le \varepsilon + \zeta_i \\ w{\cdot}x_i + b - y_i \le \varepsilon + \zeta_i^* \\ \zeta_i, \zeta_i^* \ge 0 \end{cases} \tag{2}$$

where $C \ge 0$ is the variable that specifies how much deviation more than ε is allowed. Greater C is closer to hard margin SVR. This corresponds to introducing the ε-insensitive loss function. Figure 2 shows the concept of soft margin and the ε-insensitive loss function.

$$|\zeta|_\varepsilon = \begin{cases} 0 & if\ |\zeta| \le \varepsilon \\ |\zeta| - \varepsilon & otherwise \end{cases} \tag{3}$$

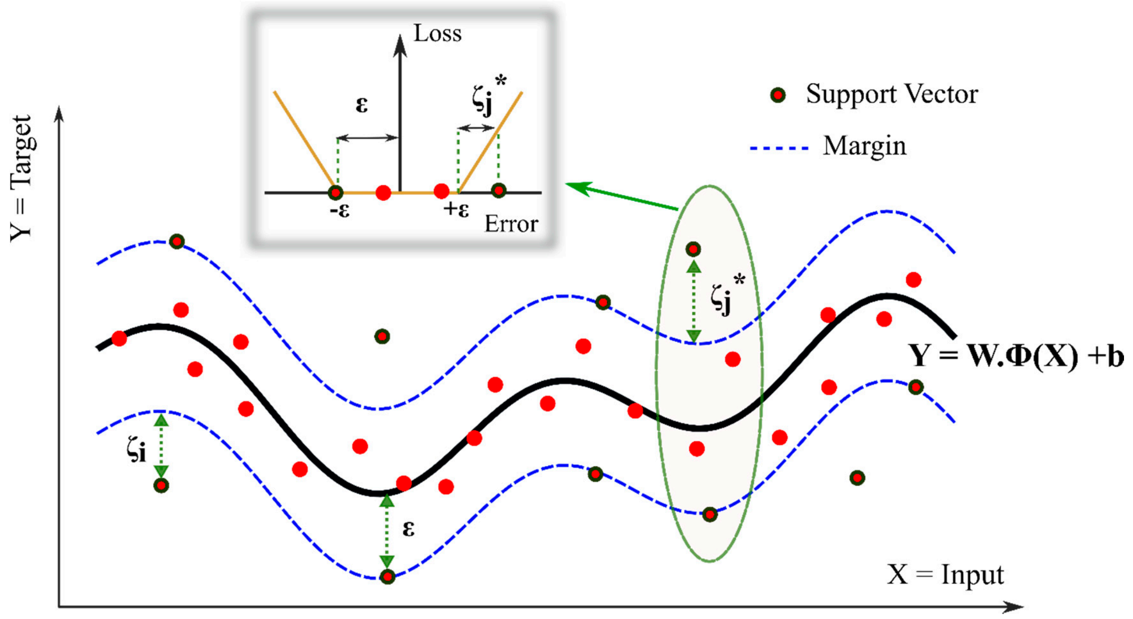

Figure 2. Soft margin nonlinear SVR (support vector machine for regression) with ε-insensitive loss function ($\phi(.)$ is a feature mapping function, W is the weight matrix, and b is bias vector).

The quadratic optimization problem (2) can be modeled and solved in two different ways, including primal and dual.

Primal: Introducing Lagrangian multipliers η_i, η_i^*, α_i, α_i^*, the Equations (4) and (5) show the primal form of the optimization problem:

$$\begin{aligned} & minimize\ \frac{1}{2}\| w \|^2 + C\sum_{i=1}^{l}\left(\zeta_i + \zeta_i^*\right) - \sum_{i=1}^{l}\left(\eta_i\zeta_i + \eta_i^*\zeta_i^*\right) \\ & - \sum_{i=1}^{l}\alpha_i(\varepsilon + \eta_i + y_i + w{\cdot}x_i + b) \\ & - \sum_{i=1}^{l}\alpha_i^*\left(\varepsilon + \eta_i^* + y_i + w{\cdot}x_i + b\right) \quad subject\ to \begin{cases} \alpha_i, \alpha_i^* \geq 0 \\ \eta_i, \eta_i^* \geq 0 \end{cases} \end{aligned} \tag{4}$$

Dual: The dual form of the optimization problem can be derived using the primal objective as below:

$$\begin{aligned} & maximize \sum_{i=1}^{l} y_i\left(\alpha_i - \alpha_i^*\right) - \varepsilon\sum_{i=1}^{l}\left(\alpha_i + \alpha_i^*\right) - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\left(\alpha_i - \alpha_i^*\right)\left(\alpha_j - \alpha_j^*\right)x_ix_j \\ & subject\ to \begin{cases} \sum_{i=1}^{l}\left(\alpha_i - \alpha_i^*\right) = 0 \\ 0 \leq \alpha_i, \alpha_i^* \leq C \end{cases} \end{aligned} \tag{5}$$

Kernels: When the data is not linearly separable (classification), one can use the kernel functions to transform the data into a higher dimensional feature space, where it turns to a linearly separable data. For the regression case, non-linear SVR can be done by kernelizing the data. Common kernel functions are summarized as follows:

$$\begin{cases} Polynomial:\ k(x, x') = (< x, x' > + c)^d \\ Tanh:\ k(x, x') = tanh(\theta + \varnothing < x, x' >) \\ RBF: k(x, x') = exp\left(\frac{\|x - x'\|^2}{2\sigma^2}\right) \end{cases} \tag{6}$$

3.3. Decision Tree

A decision tree that can be used for both regressions and classification problems is a non-parametric supervised learning algorithm. Messenger and Mandell in 1972 [79] proposed the first classification tree algorithm, which was named as THAID. The decision tree is a hierarchical tree-like flowchart composed of a root node, internal nodes, leaf nodes, and branches. The topmost node with no incoming branch, which represents the entire sample space, is called the root node. The nodes with one incoming branch and two or some outgoing edges are known as internal or test nodes, and all other nodes representing the final results are leaves, also named as terminal nodes. Splitting, stopping, and pruning are the main steps for building a decision tree [80]. Splitting the data means recursively partitioning the input data into two or more subsets based on testing the most significant attribute, which can separate the training instances as well as possible. The significant attribute is determined by different criteria, including the Gini index, entropy, classification error, gain ratio, information gain, and towing [81] for classification trees, and variance reduction or standard deviation-reduction for regression trees. Figure 3 represents an example of a decision tree for classification and regression problems. Splitting the data starts from the root node and continues on internal nodes until predefined homogeneity or stopping criteria is satisfied. Determining the stopping criteria, such as the minimum number of records in a node prior to splitting, the minimum number of records in a leaf, and the depth of any leaf from the root node, reduces the complexity of the tree, which prevents overfitting. Without stopping criteria, the splitting continues until a complex tree is constructed, in which the records in each node are 100% pure; such a tree would be fitted very well on the training data, but will not perform well on the unseen data. Therefore, these stopping criteria are normally tuned during training the model to choose the best values. Pruning is another method to avoid overfitting when stopping methods do not perform satisfactorily. In pruning, a complete tree is grown and then pruned back to a smaller tree by eliminating nodes, which have less information gain on the validation set.

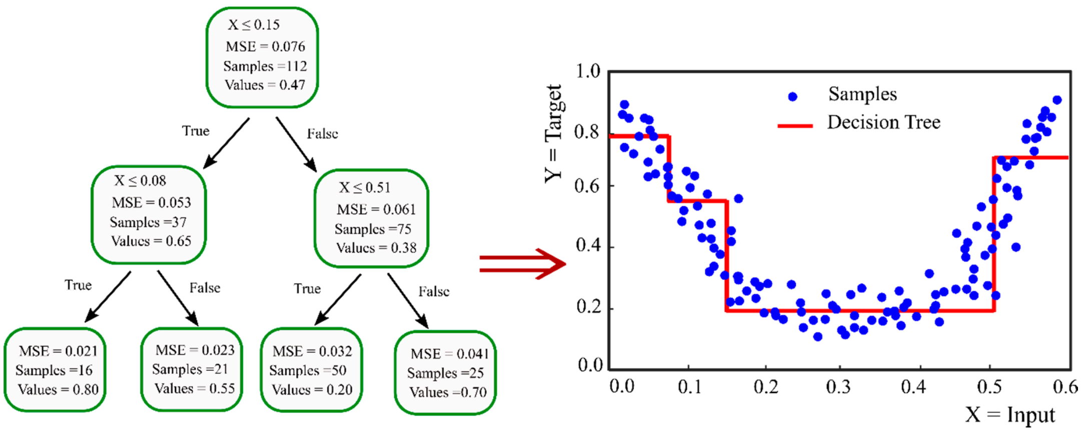

Figure 3. Examples of using a decision tree for regression.

3.4. *Random Forest and Extra Trees*

Random forest (RF) is an ensemble method, covering both classification and regression tasks. This algorithm creates a forest of multiple decision trees. The random forest algorithm, proposed by Leo Breiman [82], combines Breiman's idea of bagging with the random decision forests algorithm developed by Tin Kam Ho [83]. The random forest uses the randomly created trees approach. For creating each tree of the forest, k features of total m features are selected randomly (where k < m), the best feature is chosen by the splitting methods for the root node, and the internal nodes tests are selected by the same splitting approach until reaching the leaves. To predict unseen data with the trained model, each tree predicts and stores the outcome. The final forest output will be the majority voted class by different classification trees in case of classification, or the average of outcomes by different regression trees in case of regression. Random forest robustness against overfitting is one of the most important advantages of this algorithm, comparing the decision tree [84]. There is another tree-based ensemble model, which is called the extra trees (ET) algorithm. In contrast to the RF, which tests all possible splits over a fraction of features, the ET method tests random splits (cut point) over a fraction of features [85]. Besides accuracy, the ET method is computationally efficient compared to the RF. In this study, both methods were used and compared with other methods.

3.5. *Optimization Methods*

The gradient descent (GD) algorithm is one of the most well-known optimization methods. Considering the objective function $J(\theta)$ parameterized by model parameter θ, the GD updates the θ in the opposite direction of the gradient of $J(\theta)$ as follows:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{7}$$

where η is the learning rate. There are other versions of GD called stochastic gradient descent (SGD) and mini-batch gradient descent, which are faster than SGD [86]. Choosing proper learning rate, learning rate schedule, and trapping in (escaping) the local minima are some challenges with SGD and its other versions. To address these challenges, the following optimization algorithms that are extensively used in deep learning are provided.

Momentum: When the SGD method encounters ravines, which are common around local optima, it starts to oscillate across the slope of the ravine. The momentum method [87] helps the SGD to damp the acceleration toward the local optima in the true direction by adding a fraction of previous updates.

$$\begin{cases} v_t = \gamma v_{t-1} + \eta \cdot \nabla_\theta J(\theta) \\ \theta = \theta - v_t \end{cases} \tag{8}$$

where γ is called the momentum term.

NAG: Nesterov accelerated gradient or NAG is similar to the momentum method, except it calculates the gradient with regards to the future position of the parameter as follows [88]:

$$\begin{cases} v_t = \gamma v_{t-1} + \eta \cdot \nabla_\theta J(\theta - \gamma v_{t-1}) \\ \theta = \theta - v_t \end{cases} \tag{9}$$

Adagrad: Adaptive gradient algorithm (Adagrad), introduced by Duchi et al. [89], modifies the learning rate η for parameter θ_i at time step t using the past gradients used for θ_i as follows:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} g_{t,i} \tag{10}$$

where $g_{t,i} = \nabla_{\theta_i} J(\theta_{t,i})$ and G_t is a diagonal matrix, where each diagonal element ii is the sum of squared gradients with regards to θ_i up to time step t.

Using this modification, the infrequent parameters have a larger update, while there is a smaller update for the frequent parameter. The Adagrad has been used in different learning applications; however, the accumulation of the squared gradients in the denominator of the learning rate eventually shrinks the learning rate and makes the update rule ineffective.

Adadelta: To address the issue mentioned in Adagrad, the Adadelta method is introduced by Zeiler [90]. The main idea is that, instead of accumulating all the past squared gradients, just a fixed number of them, w, are used. This prevents the monotonical shrinkage of the learning rate. In practice, to avoid storing w previous squared gradients, an exponentially decaying average of squared gradients is used as follows:

$$\begin{cases} E\left(g^2\right)_t = \eta E\left(g^2\right)_{t-1} + (1-\eta) {g_t}^2 \\ \Delta\theta_t = \frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} \cdot g_t \end{cases} \tag{11}$$

where $RMS[x]_t = \sqrt{E(x^2)_t + \varepsilon}$ for any x.

Adam: Adaptive moment (Adam) is another optimization method, which is a combination of the momentum method and Adadelta [91]. This method uses the exponentially decaying average of previously squared gradients (similar to Adadelta) and the exponentially decaying average of gradients (similar to momentum).

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t \tag{12}$$

where $\hat{m}_t = \frac{m_t}{1-{\beta_1}^t}$, $\hat{v}_t = \frac{v_t}{1-{\beta_2}^t}$, $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$ and $v_t = \beta_2 v_{t-1} + (1-\beta_2) {g_t}^2$.

AdaMax: AdaMax is a variant of the Adam method, which scales the gradients based on the infinite norm, instead of the l_2 norm [91].

$$\theta_{t+1} = \theta_t - \frac{\eta}{u_t} \hat{m}_t \tag{13}$$

where $u_t = {\beta_2}^\infty v_{t-1} + (1 - {\beta_2}^\infty) {g_t}^\infty = \max\left(\beta_2 v_{t-1}, \left|g_t\right|\right)$.

Nadam: Nesterov accelerated adaptive moment (Nadam) is a combination of the Adam and NAG methods [92].

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} (\beta_1 \hat{m}_t + \frac{(1-\beta_1) g_t}{1 - {\beta_1}^t}) \tag{14}$$

In this work, four of these optimizers, including Adagrad, Adadelta, AdaMax, and Nadam, were used in optimizing the MLP network parameters.

4. Results and Discussion

Generally, two kinds of baseline models, including an MLP network with two hidden layers and an LSSVM model, were used in [1]. In this part, hyperparameters of these two baselines were tuned using validation data set for better performances. More specifically, the data sets were split into train and test sets with a ratio of roughly 85:15, while 10% of the training data was used as the validation set, which was used to set the hyperparameters. In this work, the MLP network with two hidden layers and four different optimizers was used as a baseline in [1]. In all of them, the Tanh, Sigmoid were used as activation functions for first and second hidden layers, respectively. Moreover, in all of the MLP networks, pure linear activation was used for the output layer. Among all of these MLP networks, MLP with Bayesian regularization (BR) optimizer returned the lowest average absolute relative error (AARE) of 4.931% on the test data. We believed by tuning the number of neurons in each hidden layer and selecting an appropriate optimizer method, the performance of MLP could be improved. Another option for improving the performance of MLP was increasing the number of hidden layers (deeper MLP), and we have used this option in the next section as one of our suggested models. Therefore, in this work, the same architecture, as [1], was used, but the number of neurons in each hidden layer and the type of optimizer were tuned as a validation set using a grid search method. As the preprocessing, the input data were normalized, i.e., each feature was centered by its mean value and scaled by its standard deviation. It should be noted that in this work, the MLP form was generally considered as (Input: number of neurons)(Hidden 1: number of neurons, activation function)(Hidden l: number of neurons, activation function)(Output: number of neurons, activation function)-optimizer. The result of the Grid search showed that MLP with form (5)(Tanh,32)(Sigmoid,64)(Linear,1)-Nadam had the best performance on the validation data set. Another modification that might improve the performance of the MLP used in [1] was the type of activation function. More specifically, in their MLP, the output layer had a pure linear activation function, and if be replaced with a non-linear activation function, they might improve the performance. Therefore, MLP with sigmoid activation function at the output layer was tuned for the appropriate number of neurons in hidden layers and type of optimizer. The Grid search result returned MLP with the form (5)(Tanh,32)(Sigmoid,64)(Sigmoid,1)-Nadam as the best model. Another baseline used in [1] was the least square SVM (LSSVM), which returned an AARE of 6.630% on the test data. In this work, the SVM regression with radial basis function (RBF) kernel was used. Our assumption was that since Gaussian models are the most commonly used models for real-world data, Gaussian kernel should be used in SVM. Two hyperparameters, including penalty parameter C and kernel coefficient gamma, were considered and tuned using the Grid search algorithm. The result showed that RBF-SVM with C = 1 and gamma = 2.3 provided the lowest AARE on validation data. Other models, including decision tree, random forest, extra trees, and an MLP with three hidden layers, could be used to improve the performance of the CMIS model used as the main model in [1]. The results of the CMIS model on different ranges of input features showed that in some ranges (for example, range 50–75 for nanoparticle size), the model had poor performance (large AARE), while in some other ranges, it had better performance. This behavior suggested that algorithms, such as decision tree (DT) or an ensemble of them (random forest and extra trees), which goes through each feature and finds the best test split based on the different ranges of features, might be a good candidate. To this end, four hyperparameters, including a maximum number of features, maximum depth, the minimum number of sample split, and a minimum number of leaf nodes, were considered and tuned using Grid search. The results of the Grid search suggested a DT with maximum depth equal to14, the maximum number of features equal to 4, the maximum number of leaf nodes of 450, and the minimum number of sample split of 3. As another attempt to improve the results, random forest (RF) was used as an ensemble of DTs. Hyperparameters, including a maximum number of features and the maximum depth, were tuned using the Grid search, and the results based on the best parameters maximum depth 18 and a maximum number of features 4 were provided. Extra trees (ET) was another ensemble of DTs. Similar to the DT, four hyperparameters were tuned, and the optimum values were the maximum depth of 20, the maximum number of features of 5, the maximum number of leaf nodes of 1000, and the minimum number of sample split of 4. The MLP network with three hidden layers was the last model used to improve the performance. Similar to the MLP with two hidden layers, hyperparameters (including the number of neurons in hidden layers and the type of optimizer) of MLP with three hidden layers were tuned using the Grid search. Again, the last activation function was changed to a Sigmoid function to see whether any improvement had been achieved. Table 2 lists the performance of each mentioned model using statistical parameters, including average absolute relative error (AARE (%)), average relative error (ARE (%)), root mean square error (RMSE), and standard deviation (SD).

From this table, it can be seen that the baselines model SVM and two layers MLP compared to baselines [1] were improved by fine-tuning. In addition, the suggested models, including random forest (RF), extra trees (ET), and three layers MLP (both with and without nonlinear activation at the output layer), provided better performance than the CMIS model. To see the performance of the models visually, the cross plot of the training and test data for each model are represented in Figure 4. In this figure, closer data points to the unit slope line indicated a better prediction of viscosity. For example, comparing Figure 4a,b, it can be seen how a nonlinear activation at the output layer improved the performance of MLP, or by comparing the trees-based methods in Figure 4d–f, the RF and ET outperformed the DT by providing more prediction close to the unit slope line.

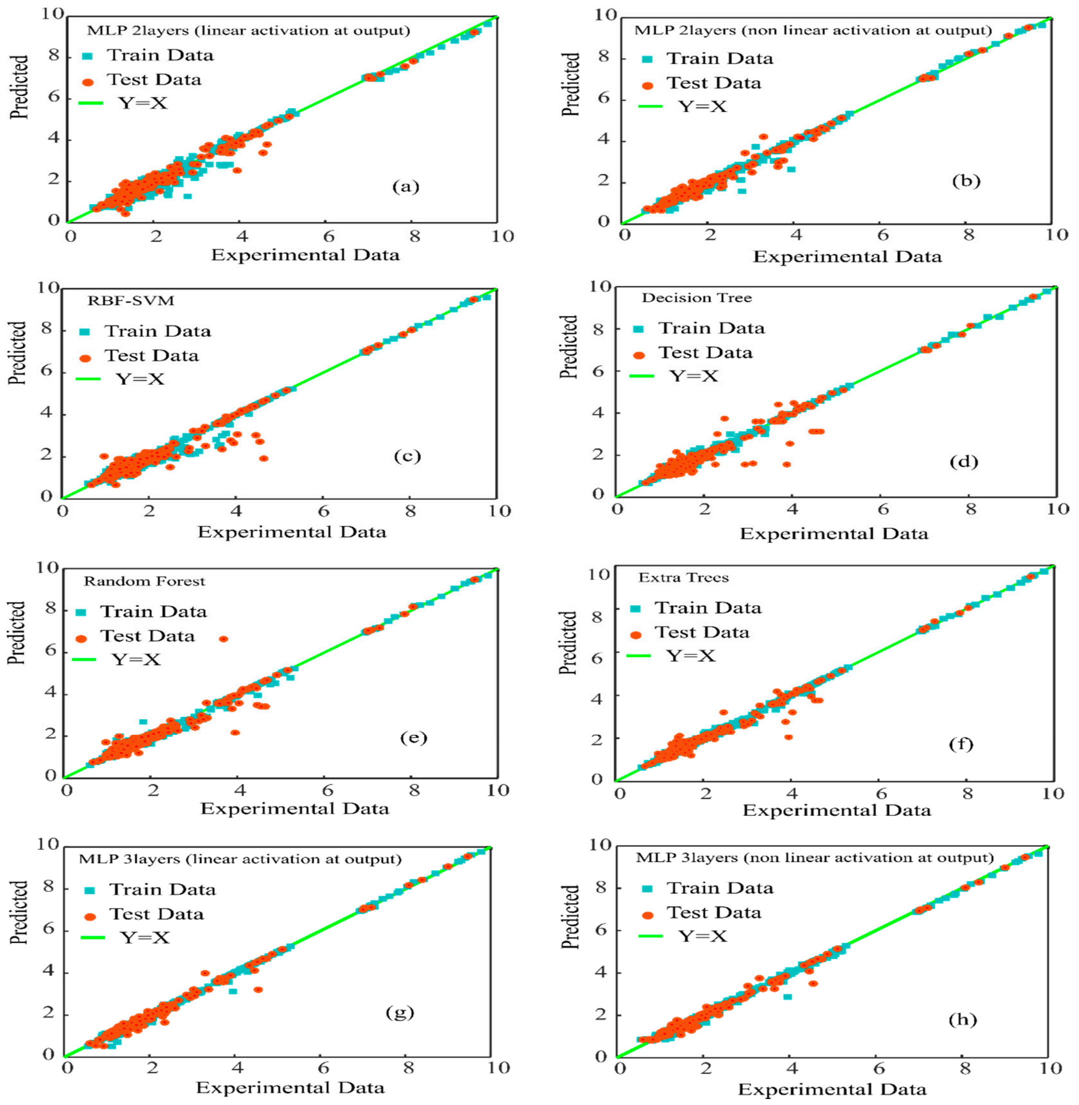

Figure 4. Cross plot of predicted viscosity versus experimental value, (**a**): MLP 2 layers (linear activation at output), (**b**): MLP 2 layers (non linear activation at output), (**c**): RBF-SVM, (**d**): Decision Tree, (**e**): Random Forest, (**f**): Extra Trees, (**g**): MLP 3 layers (linear activation at output), (**h**): MLP 2 layers (non linear activation at output).

Table 2. Statistical error analysis for prediction of the relative viscosity of nanofluids.

Model	**ARE (%)**			**AARE (%)**			**RMSE**			**SD**		
	Train	***Test***	***Total***	***Train***	***Test***	***Total***	***Train***	***Test***	***Total***	***Train***	***Test***	***Total***
CMIS [1]	−0.382	−0.515	−0.409	3.933	4.036	3.954	0.094	0.088	0.093	0.062	0.061	0.062
MLP [1]: (5)(Tanh,12)(Sigmoid,8)(Linear,1)-BR	−0.440	−0.179	−0.387	4.557	4.931	4.632	0.100	0.113	0.103	0.069	0.074	0.070
LSSVM [1]: Optimized by CSA	−0.921	−1.029	−1.011	5.342	6.630	5.488	0.187	0.047	0.193	0.070	0.017	0.108
MLP: (5)(Tanh,32)(Sigmoid,64)(Linear,1)-Nadam	1.596	1.555	1.587	4.076	4.818	4.238	0.012	0.015	0.013	0.064	0.080	0.067
MLP: (5)(Tanh,32)(Sigmoid,64)(Sigmoid,1)-Nadam	−0.206	−0.457	−0.260	2.369	3.876	2.697	0.008	0.012	0.009	0.040	0.062	0.046
RBF-SVM: C = 1; gamma = 2.3	0.089	−0.131	0.041	2.120	4.740	2.690	0.010	0.023	0.014	0.051	0.096	0.064
Decision Tree: max depth = 14, max feature = 4, min samples split = 3, max leaf nodes = 450	−0.103	0.321	−0.011	2.043	4.579	2.595	0.005	0.022	0.011	0.032	0.087	0.050
Random Forest: max depth = 18, max feature = 4	−0.204	−0.499	−0.268	1.746	3.945	2.225	0.006	0.020	0.011	0.036	0.080	0.049
Extra Trees: max depth = 20, max feature = 5, min samples split = 4, max leaf nodes = 1000	−0.149	−0.335	−0.189	1.244	3.597	1.756	0.004	0.016	0.008	0.023	0.070	0.038
MLP: (5)(Tanh,64)(Sigmoid,128)(Sigmoid,16)(Linear,1)-AdaMax	1.063	1.181	1.088	1.632	2.914	1.911	0.005	0.010	0.006	0.030	0.051	0.036
MLP: (5)(Tanh,64)(Sigmoid,128)(Sigmoid,16)(Sigmoid,1)-AdaMax	−0.507	−0.635	−0.535	1.583	2.855	1.860	0.005	0.009	0.006	0.029	0.049	0.035

A better comparison of the presented models was done using the cumulative frequency as a function of absolute relative error (%) in Figure 5. As this figure shows, the MLP (with three layers and sigmoid at the end) could predict 85% of the data points with an absolute relative error of less than 3%, which outperformed all the models.

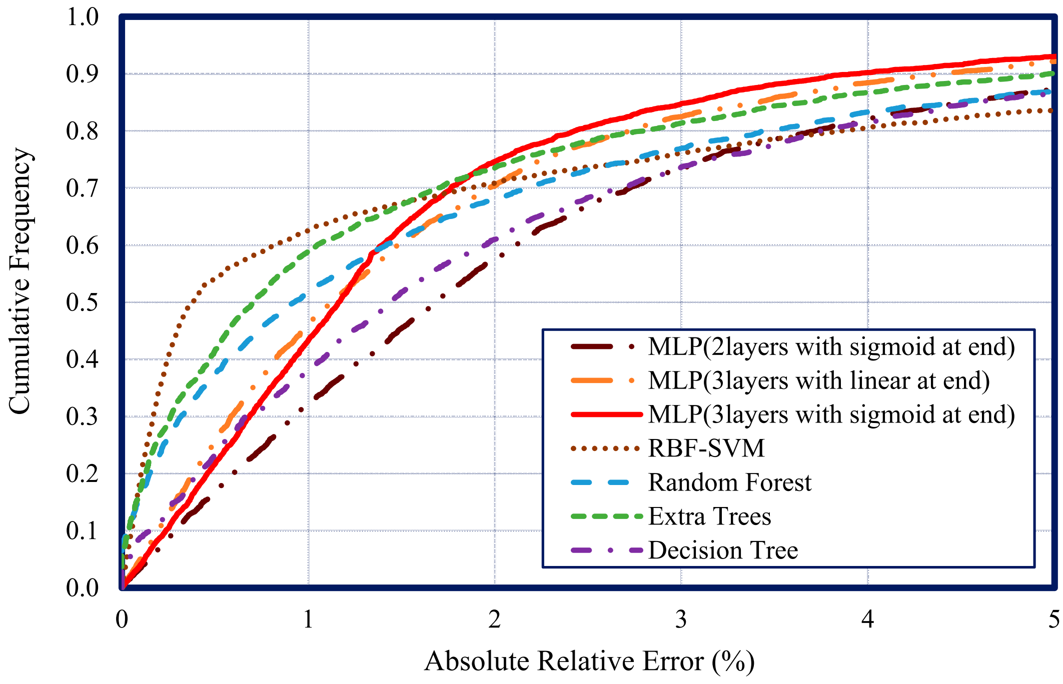

Figure 5. Cumulative frequency versus absolute relative error for all the mentioned models.

Moreover, from this figure, it can be seen that for the low absolute relative error (%), the RBF-SVM had the best performance where it could predict 63% of the data points with an absolute relative error less than 1%. The extra trees (ET) model showed consistent and acceptable performance in all absolute relative error ranges.

Figures 6–10 represent the trend of AARE (%) at different input ranges for all the presented models. In addition to determining the performance of each model by changing the input, these figures were used to specify which model was appropriate for a specific input range. For example, Figure 6 shows that the general trend of AARE (%) for all the models was decreasing by temperature until about 35 °C. It can be seen that for this temperature range (<35 °C), the MLP (with three layers and sigmoid at the end) had the best performance. For the other inputs, the MLP (with three layers and sigmoid at the end) showed almost the best performance.

To examine if the provided models can follow the physically expected trends of nanofluid viscosity by changing volume fraction, the predicted values by these models are represented in Figure 11. As can be seen in this figure, for two nanofluid samples, the experimental relative viscosity values increased with increasing the nanoparticles volume fraction. All the models could capture the expected trend with a variation of volume fraction, although some models had a slight deviation from the experimental data. This figure also proves that the proposed models were physically valid, with a variation of volume fraction as the most affecting parameter on the relative viscosity of nanofluids.

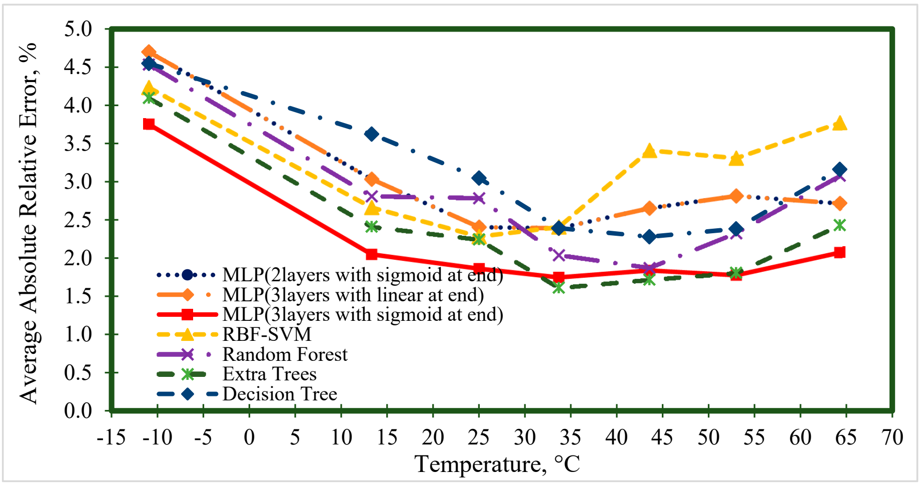

Figure 6. Average absolute relative error at different temperature ranges for prediction of the viscosity of nanofluids.

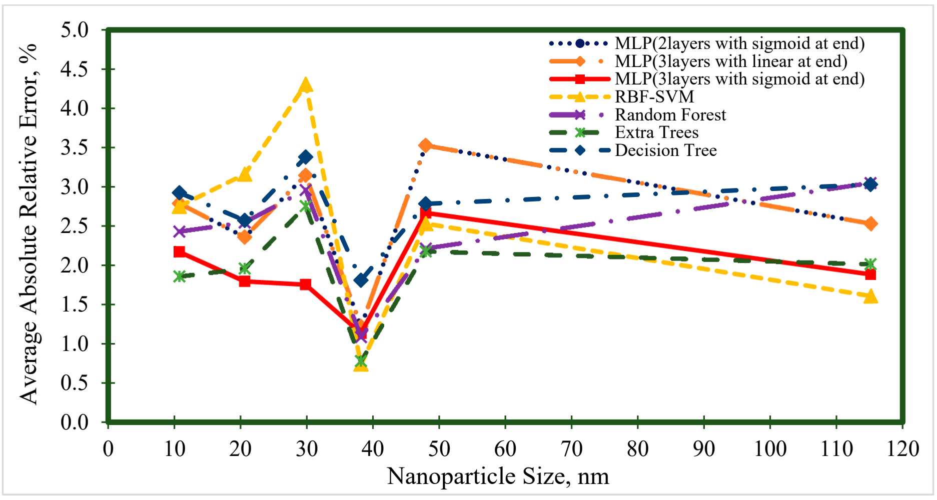

Figure 7. Average absolute relative error at different nanoparticle size ranges for prediction of the viscosity of nanofluids.

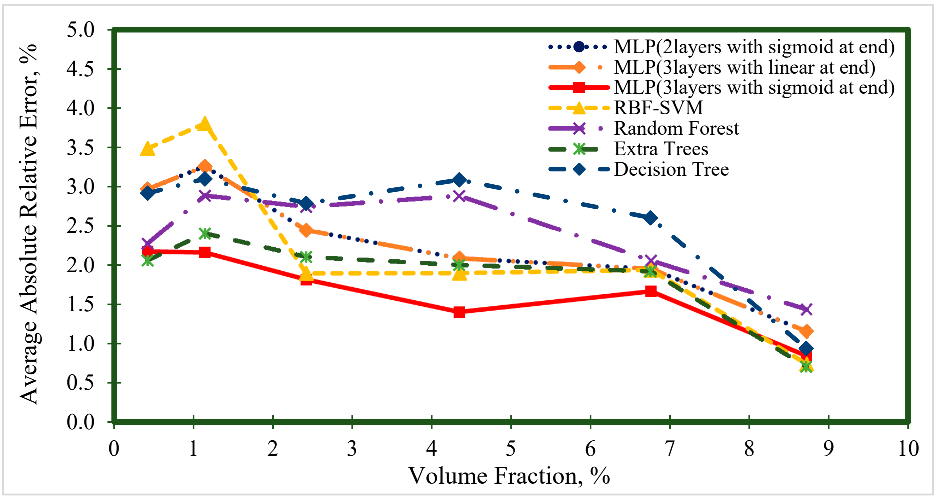

Figure 8. Average absolute relative error at different volume fraction ranges for prediction of the viscosity of nanofluids.

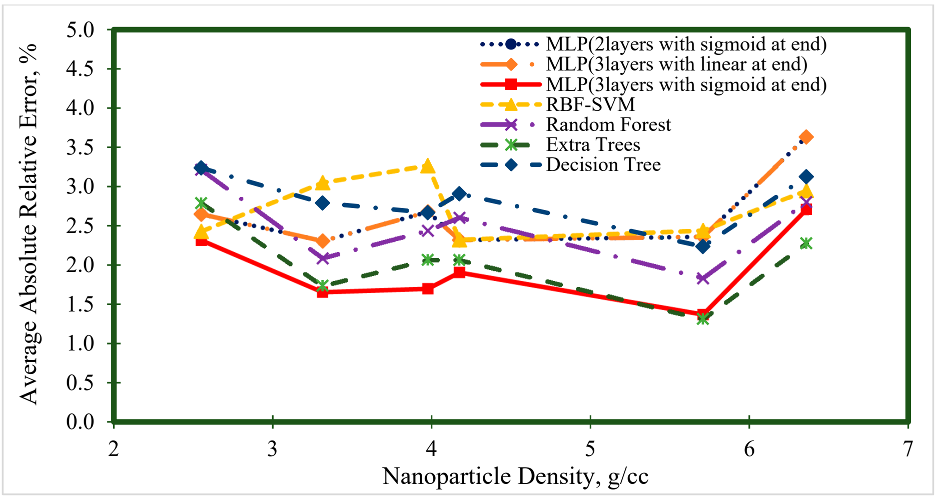

Figure 9. Average absolute relative error at different nanoparticle density ranges for prediction of the viscosity of nanofluids.

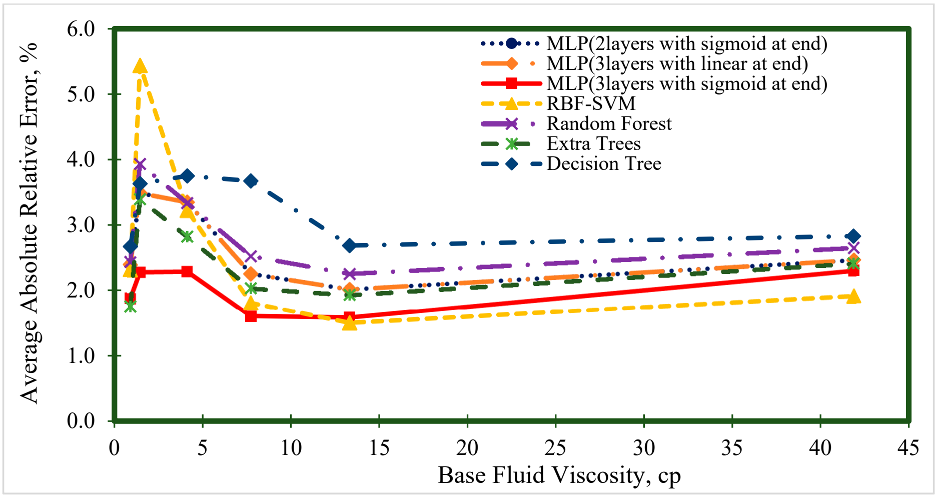

Figure 10. Average absolute relative error at different base fluid viscosity ranges for prediction of the viscosity of nanofluids.

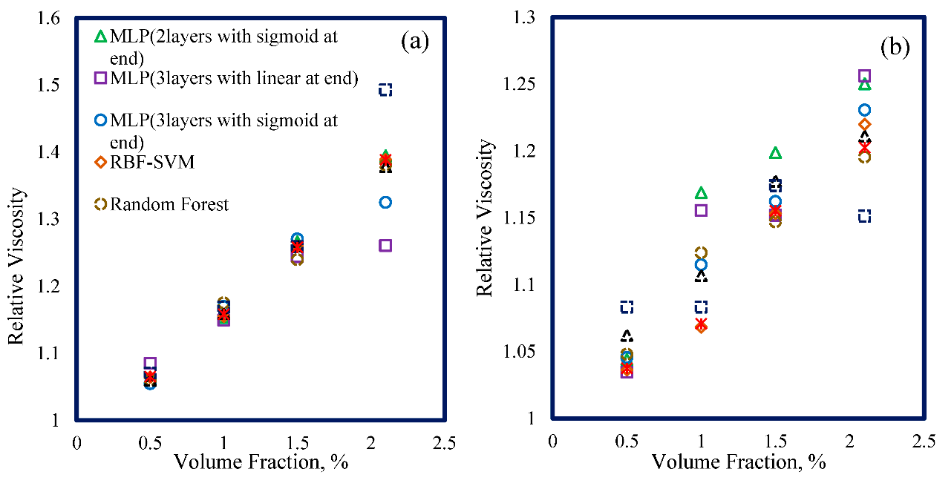

Figure 11. Variation of relative viscosity with a volume fraction of nanoparticles for two nanofluid samples, (**a**) ZnO (48 nm)—EG, (**b**) ZnO (4.6 nm)—EG [73].

5. Conclusions

In this study, the viscosity of various nanofluids was modeled using advanced computational frameworks. To this end, eight machine learning models were proposed, including two multilayer perceptron (MLP), each with Nesterov accelerated adaptive moment (Nadam) optimizer; two MLP, each with three hidden layers and Adamax optimizer; a support vector regression (SVR) with radial basis function (RBF) kernel; a decision tree (DT); two tree-based ensemble models, including random forest (RF) and extra tree (ET). The data bank, which was used for modeling, includes 3144 data points of nanofluids at different volume fraction, size, and density of nanoparticles, temperature, and viscosity of base fluids. The performance of these models at different ranges of input variables was evaluated and compared with the literature models. Based on our result, all the eight suggested models outperformed the baselines used in the literature, and five of our presented models outperformed the CMIS model, where two of them returned an AARE less than 3% on the test data. In addition, the physical validity of models was confirmed by examining the physically expected trends of nanofluid viscosity due to changing volume fraction.

Author Contributions: Conceptualization, A.H.-S.; methodology, M.S., Z.S. and A.A.; software, M.S., Z.S. and A.A.; validation, A.H.-S., A.M., and S.S.; formal analysis, A.V., M.S., Z.S. and A.A.; investigation, A.V.; resources, A.V., A.H.-S.; data curation, A.V., M.S., Z.S. and A.A.; writing—original draft preparation, A.H.-S., A.V., M.S., Z.S. and A.A.; writing—review and editing, A.H.-S., A.M., S.S.; visualization, M.S., A.V., Z.S. and A.A.; supervision, A.H.-S., A.M., S.S.; project administration, A.H.-S., A.M., S.S.; funding acquisition, A.M. All authors have read and agreed to the published version of the manuscript.

Funding: The research has been partly carried out within the EFOP-3.6.2-16-2017-00016 project in the framework of the New Szechenyi Plan. The completion of this project is also partly funded by the European Union and co-financed by the European Social Fund. Furthermore, the support of Alexander von Humboldt foundation is also acknowledged.

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

References

1. Hemmati-Sarapardeh, A.; Varamesh, A.; Husein, M.M.; Karan, K. On the evaluation of the viscosity of nanofluid systems: Modeling and data assessment. *Renew. Sustain. Energy Rev.* **2018**, *81*, 313–329. [CrossRef]
2. Yang, L.; Xu, J.; Du, K.; Zhang, X. Recent developments on viscosity and thermal conductivity of nanofluids. *Powder Technol.* **2017**, *317*, 348–369. [CrossRef]
3. Divandari, H.; Hemmati-Sarapardeh, A.; Schaffie, M.; Ranjbar, M. Integrating functionalized magnetite nanoparticles with low salinity water and surfactant solution: Interfacial tension study. *Fuel* **2020**, *281*, 118641. [CrossRef]
4. Rezaei, A.; Abdollahi, H.; Derikvand, Z.; Hemmati-Sarapardeh, A.; Mosavi, A.; Nabipour, N. Insights into the Effects of Pore Size Distribution on the Flowing Behavior of Carbonate Rocks: Linking a Nano-Based Enhanced Oil Recovery Method to Rock Typing. *Nanomaterials* **2020**, *10*, 972. [CrossRef] [PubMed]
5. Corredor-Rojas, L.M.; Hemmati-Sarapardeh, A.; Husein, M.M.; Dong, M.; Maini, B.B. Rheological behavior of surface modified silica nanoparticles dispersed in partially hydrolyzed polyacrylamide and xanthan gum solutions: Experimental measurements, mechanistic understanding, and model development. *Energy Fuels* **2018**, *32*, 10628–10638. [CrossRef]
6. Moghadasi, R.; Rostami, A.; Hemmati-Sarapardeh, A.; Motie, M. Application of Nanosilica for inhibition of fines migration during low salinity water injection: Experimental study, mechanistic understanding, and model development. *Fuel* **2019**, *242*, 846–862. [CrossRef]
7. Moldoveanu, G.M.; Ibanescu, C.; Danu, M.; Minea, A.A. Viscosity estimation of Al_2O_3, SiO_2 nanofluids and their hybrid: An experimental study. *J. Mol. Liq.* **2018**, *253*, 188–196. [CrossRef]
8. Gholami, E.; Vaferi, B.; Ariana, M.A. Prediction of viscosity of several alumina-based nanofluids using various artificial intelligence paradigms-Comparison with experimental data and empirical correlations. *Powder Technol.* **2018**, *323*, 495–506. [CrossRef]
9. Einstein, A. A new determination of molecular dimensions. *Ann. Phys.* **1906**, *19*, 289–306. [CrossRef]

10. Brinkman, H. The viscosity of concentrated suspensions and solutions. *J. Chem. Phys.* **1952**, *20*, 571. [CrossRef]
11. Lundgren, T.S. Slow flow through stationary random beds and suspensions of spheres. *J. Fluid Mech.* **1972**, *51*, 273–299. [CrossRef]
12. Frankel, N.; Acrivos, A. On the viscosity of a concentrated suspension of solid spheres. *Chem. Eng. Sci.* **1967**, *22*, 847–853. [CrossRef]
13. Batchelor, G. The effect of Brownian motion on the bulk stress in a suspension of spherical particles. *J. Fluid Mech.* **1977**, *83*, 97–117. [CrossRef]
14. Thomas, C.U.; Muthukumar, M. Three—body hydrodynamic effects on viscosity of suspensions of spheres. *J. Chem. Phys.* **1991**, *94*, 5180–5189. [CrossRef]
15. Chen, H.; Ding, Y.; He, Y.; Tan, C. Rheological behaviour of ethylene glycol based titania nanofluids. *Chem. Phys. Lett.* **2007**, *444*, 333–337. [CrossRef]
16. Maïga, S.E.B.; Nguyen, C.T.; Galanis, N.; Roy, G. Heat transfer behaviours of nanofluids in a uniformly heated tube. *Superlattices Microstruct.* **2004**, *35*, 543–557. [CrossRef]
17. Varamesh, A.; Hemmati-Sarapardeh, A. Viscosity of nanofluid systems—A critical evaluation of modeling approaches. In *Nanofluids and Their Engineering Applications*; CRC Press: Boca Raton, FL, USA; Taylor & Francis Group: Abingdon, UK, 2019.
18. Mazloom, M.S.; Rezaei, F.; Hemmati-Sarapardeh, A.; Husein, M.M.; Zendehboudi, S.; Bemani, A. Artificial Intelligence Based Methods for Asphaltenes Adsorption by Nanocomposites: Application of Group Method of Data Handling, Least Squares Support Vector Machine, and Artificial Neural Networks. *Nanomaterials* **2020**, *10*, 890. [CrossRef]
19. Karimi, H.; Yousefi, F.; Rahimi, M.R.J.H.; Transfer, M. Correlation of viscosity in nanofluids using genetic algorithm-neural network (GA-NN). *Heat Mass Transf.* **2011**, *47*, 1417–1425. [CrossRef]
20. Mehrabi, M.; Sharifpur, M.; Meyer, J.P. Viscosity of nanofluids based on an artificial intelligence model. *Int. Commun. Heat Mass Transf.* **2013**, *43*, 16–21. [CrossRef]
21. Atashrouz, S.; Pazuki, G.; Alimoradi, Y. Estimation of the viscosity of nine nanofluids using a hybrid GMDH-type neural network system. *Fluid Phase Equilibria* **2014**, *372*, 43–48. [CrossRef]
22. Meybodi, M.K.; Naseri, S.; Shokrollahi, A.; Daryasafar, A. Prediction of viscosity of water-based Al_2O_3, TiO_2, SiO_2, and CuO nanofluids using a reliable approach. *Chemom. Intell. Lab. Syst.* **2015**, *149*, 60–69. [CrossRef]
23. Zhao, N.; Wen, X.; Yang, J.; Li, S.; Wang, Z. Modeling and prediction of viscosity of water-based nanofluids by radial basis function neural networks. *Powder Technol.* **2015**, *281*, 173–183. [CrossRef]
24. Adio, S.A.; Mehrabi, M.; Sharifpur, M.; Meyer, J.P. Experimental investigation and model development for effective viscosity of MgO–ethylene glycol nanofluids by using dimensional analysis, FCM-ANFIS and GA-PNN techniques. *Int. Commun. Heat Mass Transf.* **2016**, *72*, 71–83. [CrossRef]
25. Atashrouz, S.; Mozaffarian, M.; Pazuki, G. Viscosity and rheological properties of ethylene glycol+water+Fe_3O_4 nanofluids at various temperatures: Experimental and thermodynamics modeling. *Korean J. Chem. Eng.* **2016**, *33*, 2522–2529. [CrossRef]
26. Barati-Harooni, A.; Najafi-Marghmaleki, A. An accurate RBF-NN model for estimation of viscosity of nanofluids. *J. Mol. Liq.* **2016**, *224*, 580–588. [CrossRef]
27. Heidari, E.; Sobati, M.A.; Movahedirad, S. Accurate prediction of nanofluid viscosity using a multilayer perceptron artificial neural network (MLP-ANN). *Chemom. Intell. Lab. Syst.* **2016**, *155*, 73–85. [CrossRef]
28. Longo, G.A.; Zilio, C.; Ortombina, L.; Zigliotto, M. Application of Artificial Neural Network (ANN) for modeling oxide-based nanofluids dynamic viscosity. *Int. Commun. Heat Mass Transf.* **2017**, *83*, 8–14. [CrossRef]
29. Bahiraei, M.; Hangi, M. An empirical study to develop temperature-dependent models for thermal conductivity and viscosity of water-Fe_3O_4 magnetic nanofluid. *Mater. Chem. Phys.* **2016**, *181*, 333–343. [CrossRef]
30. Barati-Harooni, A.; Najafi-Marghmaleki, A.; Mohebbi, A.; Mohammadi, A.H. On the estimation of viscosities of Newtonian nanofluids. *J. Mol. Liq.* **2017**, *241*, 1079–1090. [CrossRef]
31. Aminian, A. Predicting the effective viscosity of nanofluids for the augmentation of heat transfer in the process industries. *J. Mol. Liq.* **2017**, *229*, 300–308. [CrossRef]

32. Vakili, M.; Khosrojerdi, S.; Aghajannezhad, P.; Yahyaei, M. A hybrid artificial neural network-genetic algorithm modeling approach for viscosity estimation of graphene nanoplatelets nanofluid using experimental data. *Int. Commun. Heat Mass Transf.* **2017**, *82*, 40–48. [CrossRef]
33. Ansari, H.R.; Zarei, M.J.; Sabbaghi, S.; Keshavarz, P. A new comprehensive model for relative viscosity of various nanofluids using feed-forward back-propagation MLP neural networks. *Int. Commun. Heat Mass Transf.* **2018**, *91*, 158–164. [CrossRef]
34. Derakhshanfard, F.; Mehralizadeh, A. Application of artificial neural networks for viscosity of crude oil-based nanofluids containing oxides nanoparticles. *J. Pet. Sci. Eng.* **2018**, *168*, 263–272. [CrossRef]
35. Karimipour, A.; Ghasemi, S.; Darvanjooghi, M.H.K.; Abdollahi, A. A new correlation for estimating the thermal conductivity and dynamic viscosity of CuO/liquid paraffin nanofluid using neural network method. *Int. Commun. Heat Mass Transf.* **2018**, *92*, 90–99. [CrossRef]
36. Murshed, S.M.S.; Leong, K.C.; Yang, C. Investigations of thermal conductivity and viscosity of nanofluids. *Int. J. Therm. Sci.* **2008**, *47*, 560–568. [CrossRef]
37. Lee, J.-H.; Hwang, K.S.; Jang, S.P.; Lee, B.H.; Kim, J.H.; Choi, S.U.S.; Choi, C.J. Effective viscosities and thermal conductivities of aqueous nanofluids containing low volume concentrations of Al_2O_3 nanoparticles. *Int. J. Heat Mass Transf.* **2008**, *51*, 2651–2656. [CrossRef]
38. Singh, M.; Kundan, L. Experimental study on thermal conductivity and viscosity of Al_2O_3–nanotransformer oil. *Int. J. Theo. App. Res. Mech. Eng.* **2013**, *2*, 125–130.
39. Nguyen, C.T.; Desgranges, F.; Roy, G.; Galanis, N.; Maré, T.; Boucher, S.; Angue Mintsa, H. Temperature and particle-size dependent viscosity data for water-based nanofluids–Hysteresis phenomenon. *Int. J. Heat Fluid Flow* **2007**, *28*, 1492–1506. [CrossRef]
40. Chandrasekar, M.; Suresh, S.; Chandra Bose, A. Experimental investigations and theoretical determination of thermal conductivity and viscosity of Al_2O_3/water nanofluid. *Exp. Therm. Fluid Sci.* **2010**, *34*, 210–216. [CrossRef]
41. Tavman, I.; Turgut, A.; Chirtoc, M.; Schuchmann, H.; Tavman, S. Experimental investigation of viscosity and thermal conductivity of suspensions containing nanosized ceramic particles. *Arch. Mater. Sci.* **2008**, *34*, 99–104.
42. Zhou, S.-Q.; Ni, R.; Funfschilling, D. Effects of shear rate and temperature on viscosity of alumina polyalphaolefins nanofluids. *J. Appl. Phys.* **2010**, *107*, 054317. [CrossRef]
43. Mena, J.B.; Ubices de Moraes, A.A.; Benito, Y.R.; Ribatski, G.; Parise, J.A.R. Extrapolation of Al_2O_3–water nanofluid viscosity for temperatures and volume concentrations beyond the range of validity of existing correlations. *Appl. Therm. Eng.* **2013**, *51*, 1092–1097. [CrossRef]
44. Pak, B.C.; Cho, Y.I. Hydrodynamic and heat transfer study of dispersed fluids with submicron metallic oxide particles. *Exp. Heat Transf. Int. J.* **1998**, *11*, 151–170. [CrossRef]
45. Syam Sundar, L.; Venkata Ramana, E.; Singh, M.K.; Sousa, A.C.M. Thermal conductivity and viscosity of stabilized ethylene glycol and water mixture Al_2O_3 nanofluids for heat transfer applications: An experimental study. *Int. Commun. Heat Mass Transf.* **2014**, *56*, 86–95. [CrossRef]
46. Yiamsawas, T.; Dalkilic, A.S.; Mahian, O.; Wongwises, S. Measurement and correlation of the viscosity of water-based Al_2O_3 and TiO_2 nanofluids in high temperatures and comparisons with literature reports. *J. Dispers. Sci. Technol.* **2013**, *34*, 1697–1703. [CrossRef]
47. Yiamsawas, T.; Mahian, O.; Dalkilic, A.S.; Kaewnai, S.; Wongwises, S. Experimental studies on the viscosity of TiO_2 and Al_2O_3 nanoparticles suspended in a mixture of ethylene glycol and water for high temperature applications. *Appl. Energy* **2013**, *111*, 40–45. [CrossRef]
48. Chiam, H.W.; Azmi, W.H.; Usri, N.A.; Mamat, R.; Adam, N.M. Thermal conductivity and viscosity of Al_2O_3 nanofluids for different based ratio of water and ethylene glycol mixture. *Exp. Therm. Fluid Sci.* **2017**, *81*, 420–429. [CrossRef]
49. Anoop, K.; Kabelac, S.; Sundararajan, T.; Das, S.K. Rheological and flow characteristics of nanofluids: Influence of electroviscous effects and particle agglomeration. *J. Appl. Phys.* **2009**, *106*, 034909. [CrossRef]
50. Sekhar, Y.R.; Sharma, K. Study of viscosity and specific heat capacity characteristics of water-based Al_2O_3 nanofluids at low particle concentrations. *J. Exp. Nanosci.* **2015**, *10*, 86–102. [CrossRef]
51. Pastoriza-Gallego, M.; Casanova, C.; Páramo, R.; Barbés, B.; Legido, J.; Piñeiro, M. A study on stability and thermophysical properties (density and viscosity) of Al_2O_3 in water nanofluid. *J. Appl. Phys.* **2009**, *106*, 064301. [CrossRef]

52. Kulkarni, D.P.; Das, D.K.; Vajjha, R.S. Application of nanofluids in heating buildings and reducing pollution. *Appl. Energy* **2009**, *86*, 2566–2573. [CrossRef]
53. Naik, M.; Sundar, L.S. Investigation into thermophysical properties of glycol based CuO nanofluid for heat transfer applications. *World Acad. Sci. Eng. Technol.* **2011**, *59*, 440–446.
54. Pastoriza-Gallego, M.J.; Casanova, C.; Legido, J.L.; Piñeiro, M.M. CuO in water nanofluid: Influence of particle size and polydispersity on volumetric behaviour and viscosity. *Fluid Phase Equilibria* **2011**, *300*, 188–196. [CrossRef]
55. Namburu, P.K.; Kulkarni, D.P.; Misra, D.; Das, D.K. Viscosity of copper oxide nanoparticles dispersed in ethylene glycol and water mixture. *Exp. Therm. Fluid Sci.* **2007**, *32*, 397–402. [CrossRef]
56. Jia-Fei, Z.; Zhong-Yang, L.; Ming-Jiang, N.; Ke-Fa, C. Dependence of nanofluid viscosity on particle size and pH value. *Chin. Phys. Lett.* **2009**, *26*, 066202. [CrossRef]
57. Chevalier, J.; Tillement, O.; Ayela, F. Rheological properties of nanofluids flowing through microchannels. *Appl. Phys. Lett.* **2007**, *91*, 3103. [CrossRef]
58. Jamshidi, N.; Farhadi, M.; Ganji, D.; Sedighi, K. Experimental investigation on viscosity of nanofluids. *Int. J. Eng.* **2012**, *25*, 201–209. [CrossRef]
59. Rudyak, V.Y.; Dimov, S.V.; Kuznetsov, V.V. On the dependence of the viscosity coefficient of nanofluids on particle size and temperature. *Tech. Phys. Lett.* **2013**, *39*, 779–782. [CrossRef]
60. Abdolbaqi, M.K.; Sidik, N.A.C.; Rahim, M.F.A.; Mamat, R.; Azmi, W.H.; Yazid, M.N.A.W.M.; Najafi, G. Experimental investigation and development of new correlation for thermal conductivity and viscosity of BioGlycol/water based SiO_2 nanofluids. *Int. Commun. Heat Mass Transf.* **2016**, *77*, 54–63. [CrossRef]
61. Lee, S.W.; Park, S.D.; Kang, S.; Bang, I.C.; Kim, J.H. Investigation of viscosity and thermal conductivity of SiC nanofluids for heat transfer applications. *Int. J. Heat Mass Transf.* **2011**, *54*, 433–438. [CrossRef]
62. Duangthongsuk, W.; Wongwises, S. Measurement of temperature-dependent thermal conductivity and viscosity of TiO_2-water nanofluids. *Exp. Therm. Fluid Sci.* **2009**, *33*, 706–714. [CrossRef]
63. Chen, H.; Ding, Y.; Tan, C. Rheological behaviour of nanofluids. *New J. Phys.* **2007**, *9*, 367. [CrossRef]
64. Abdolbaqi, M.K.; Sidik, N.A.C.; Aziz, A.; Mamat, R.; Azmi, W.H.; Yazid, M.N.A.W.M.; Najafi, G. An experimental determination of thermal conductivity and viscosity of BioGlycol/water based TiO_2 nanofluids. *Int. Commun. Heat Mass Transf.* **2016**, *77*, 22–32. [CrossRef]
65. Khedkar, R.S.; Shrivastava, N.; Sonawane, S.S.; Wasewar, K.L. Experimental investigations and theoretical determination of thermal conductivity and viscosity of TiO_2–ethylene glycol nanofluid. *Int. Commun. Heat Mass Transf.* **2016**, *73*, 54–61. [CrossRef]
66. Singh, R.; Sanchez, O.; Ghosh, S.; Kadimcherla, N.; Sen, S.; Balasubramanian, G. Viscosity of magnetite–toluene nanofluids: Dependence on temperature and nanoparticle concentration. *Phys. Lett. A* **2015**, *379*, 2641–2644. [CrossRef]
67. Syam Sundar, L.; Singh, M.K.; Sousa, A.C.M. Investigation of thermal conductivity and viscosity of Fe_3O_4 nanofluid for heat transfer applications. *Int. Commun. Heat Mass Transf.* **2013**, *44*, 7–14. [CrossRef]
68. Esfe, M.H.; Saedodin, S.; Asadi, A.; Karimipour, A. Thermal conductivity and viscosity of Mg (OH) 2-ethylene glycol nanofluids. *J. Therm. Anal. Calorim.* **2015**, *120*, 1145–1149. [CrossRef]
69. Mariano, A.; Pastoriza-Gallego, M.J.; Lugo, L.; Mussari, L.; Piñeiro, M.M. Co_3O_4 ethylene glycol-based nanofluids: Thermal conductivity, viscosity and high pressure density. *Int. J. Heat Mass Transf.* **2015**, *85*, 54–60. [CrossRef]
70. Sundar, L.S.; Hortiguela, M.J.; Singh, M.K.; Sousa, A.C.M. Thermal conductivity and viscosity of water based nanodiamond (ND) nanofluids: An experimental study. *Int. Commun. Heat Mass Transf.* **2016**, *76*, 245–255. [CrossRef]
71. Sundar, L.S.; Singh, M.K.; Sousa, A.C.M. Enhanced thermal properties of nanodiamond nanofluids. *Chem. Phys. Lett.* **2016**, *644*, 99–110. [CrossRef]
72. Hemmat Esfe, M.; Saedodin, S. An experimental investigation and new correlation of viscosity of ZnO–EG nanofluid at various temperatures and different solid volume fractions. *Exp. Therm. Fluid Sci.* **2014**, *55*, 1–5. [CrossRef]
73. Pastoriza-Gallego, M.J.; Lugo, L.; Cabaleiro, D.; Legido, J.L.; Piñeiro, M.M. Thermophysical profile of ethylene glycol-based ZnO nanofluids. *J. Chem. Thermodyn.* **2014**, *73*, 23–30. [CrossRef]
74. Rosenblatt, F. *Principles of Neurodymanics: Perceptrons and the Theory of Brain Mechanisms*; Spartan Books; Cornell Aeronautical Laboratory, Inc.: Buffalo, NY, USA, 1962.

75. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
76. Karlik, B.; Olgac, A.V. Performance analysis of various activation functions in generalized MLP architectures of neural networks. *Int. J. Artif. Intell. Expert Syst.* **2011**, *1*, 111–122.
77. Drucker, H.; Burges, C.J.; Kaufman, L.; Smola, A.J.; Vapnik, V. Support vector regression machines. In *Advances in Neural Information Processing Systems*; MIT Press: Camberidge, MA, USA, 1997; pp. 155–161.
78. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
79. Loh, W.-Y. Fifty years of classification and regression trees. *Int. Stat. Rev.* **2014**, *82*, 329–348. [CrossRef]
80. Song, Y.-Y.; Ying, L. Decision tree methods: Applications for classification and prediction. *Shanghai Arch. Psychiatry* **2015**, *27*, 130.
81. Patel, N.; Upadhyay, S. Study of various decision tree pruning methods with their empirical comparison in WEKA. *Int. J. Comput. Appl.* **2012**, *60*, 20–25. [CrossRef]
82. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
83. Ho, T.K. Random decision forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 14–16 August 1995; pp. 278–282.
84. Friedman, J.; Hastie, T.; Tibshirani, R. *The Elements of Statistical Learning*; Springer: Berlin/Heidelberg, Germany, 2001.
85. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Mach. Learn.* **2006**, *63*, 3–42. [CrossRef]
86. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
87. Qian, N. On the momentum term in gradient descent learning algorithms. *Neural Netw.* **1999**, *12*, 145–151. [CrossRef]
88. Nesterov, Y. A method for unconstrained convex minimization problem with the rate of convergence O (1/k^2). In *Doklady AN USSR*; American Mathematical Society: Providence, RI, USA, 1983; pp. 543–547.
89. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
90. Zeiler, M.D. ADADELTA: An adaptive learning rate method. *arXiv* **2012**, arXiv:1212.5701.
91. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
92. Dozat, T. *Incorporating Nesterov Momentum into Adam*; Natural Hazards 3, no. 2; Stanford University: Stanford, CA, USA, 2016; pp. 437–453.