polymers

MDPI

Article

A Meta-Model to Predict the Drag Coefficient of a Particle Translating in Viscoelastic Fluids: A Machine Learning Approach

Salah A. Faroughi [1,*], Ana I. Roriz [2] and Célio Fernandes [1,2]

[1] Geo-Intelligence Laboratory, Ingram School of Engineering, Texas State University, San Marcos, TX 78666, USA; cbpf@dep.uminho.pt
[2] Department of Polymer Engineering, Institute for Polymers and Composites (IPC), Campus of Azurém, Engineering School of the University of Minho, 4800-058 Guimarães, Portugal; b12374@dep.uminho.pt
* Correspondence: salah.faroughi@txstate.edu

Abstract: This study presents a framework based on Machine Learning (ML) models to predict the drag coefficient of a spherical particle translating in viscoelastic fluids. For the purpose of training and testing the ML models, two datasets were generated using direct numerical simulations (DNSs) for the viscoelastic unbounded flow of Oldroyd-B (*OB-set* containing 12,120 data points) and Giesekus (*GI-set* containing 4950 data points) fluids past a spherical particle. The kinematic input features were selected to be Reynolds number, $0 < Re \leq 50$, Weissenberg number, $0 \leq Wi \leq 10$, polymeric retardation ratio, $0 < \zeta < 1$, and shear thinning mobility parameter, $0 < \alpha < 1$. The ML models, specifically Random Forest (RF), Deep Neural Network (DNN) and Extreme Gradient Boosting (XGBoost), were all trained, validated, and tested, and their best architecture was obtained using a 10-Fold cross-validation method. All the ML models presented remarkable accuracy on these datasets; however the XGBoost model resulted in the highest R^2 and the lowest root mean square error (RMSE) and mean absolute percentage error (MAPE) measures. Additionally, a blind dataset was generated using DNSs, where the input feature coverage was outside the scope of the training set or interpolated within the training sets. The ML models were tested against this blind dataset, to further assess their generalization capability. The DNN model achieved the highest R^2 and the lowest RMSE and MAPE measures when inferred on this blind dataset. Finally, we developed a meta-model using stacking technique to ensemble RF, XGBoost and DNN models and output a prediction based on the individual learner's predictions and a DNN meta-regressor. The meta-model consistently outperformed the individual models on all datasets.

Keywords: machine learning; deep learning; stacked learning; viscoelastic flows; Oldroyd-B fluid; Giesekus fluid; sphere drag coefficient

Citation: Faroughi, S.A.; Roriz, A.I.; Fernandes, C. A Meta-Model to Predict the Drag Coefficient of a Particle Translating in Viscoelastic Fluids: A Machine Learning Approach. *Polymers* **2022**, *14*, 430. https://doi.org/10.3390/polym14030430

Academic Editor: Mirta I. Aranguren

Received: 22 December 2021
Accepted: 18 January 2022
Published: 21 January 2022

1. Introduction

The flow of particle-laden complex fluids has been the centerpiece of many well-documented experimental, theoretical, and numerical approaches [1–4]. These fluids are non-Newtonian in character showing shear thinning, shear thickening, viscoplastic, time-dependent and viscoelastic behaviors under different flow conditions. Resolving the dynamics of particles within these fluids is extremely challenging both experimentally and computationally, especially when the matrix fluid is viscoelastic, e.g., a polymer solution or polymer melt [4–7].

Characterizing the dynamics of particles in complex fluids under different flow and/or environmental conditions requires comprehensive experimentation and simulation tools to resolve the nonlinear interplay of multiple physical variables, flow parameters and many-body interactions [8]. This is computationally expensive even when using high performance computing resources with robust parallelized algorithms [4]. In recent decades, many physics-based numerical models have been proposed to model complex fluids, see

the review by Maxey [9]. These approaches are limited to specific conditions and cannot be practically applied to large-scale engineering applications where hundreds or millions of particles are suspended, e.g., blood and other biological fluids, hydraulic fracturing, cementing, etc. [10,11]. In addition to the computing power, the insufficiency of comprehensive physics-based constitutive models to describe a broad range of complexities involved with such fluids, their use in realistic conditions encounter severe uncertainties or limitations. Therefore, the wealth of existing domain knowledge and scientific capabilities in this field need to be complemented with evolving technologies such as Machine Learning (ML) and Deep Learning (DL) to accelerate fundamental and applied research and close knowledge and computational gaps. For example, for a time intensive conventional computational model, both inner-loop (i.e., forward simulations) and the outer-loop (optimization and data assimilation) can be improved using the adaptivity and acceleration of ML- or DL-based models [12,13].

One main challenge for the application of ML and DL in the field of complex fluids is the lack of datasets. This challenge can be resolved by an in-line integration of traditional (e.g., CFD) and DL-based modeling, so-called a physics-informed DL, physics-guided DL, or digital-twin technique [14]. The ML or DL algorithms can be trained and integrated with traditional physics-based forward modeling to predict the flow dynamics under different conditions at a reduced computational cost. The latter is done by learning the solutions for ordinary and partial differential equations governing the system [13] or learning the closure laws for the pertinent physics, e.g., lift and drag forces, turbulence models, etc.

Such integration can be explored for the Eulerian-Lagrangian multi-phase model [6,15], as one of the main computational methods to resolve the flow of complex fluids. The ML/DL integration can drastically increase the robustness of this numerical algorithm that integrates the presence of multiple non-Brownian particles as the discrete material phase embedded in a viscoelastic fluid treated as a continuum phase. In particular, the momentum-exchange model, including drag, lift, hindrance, and retardation closure laws to couple the constituents [6,16,17], can be replaced with ML or DL models. In this approach, the fundamental goal is to provide reasonably accurate data-driven predictions that substitute expensive computational steps. These data-driven models learn the multitude of coupling within the complex fluids at the particle-level and enable accurate simulations of complex fluids at larger length and time scales.

In the present contribution, we propose to take the first step and complement the Eulerian-Lagrangian multi-phase approach with a data-driven drag model for the translation of a spherical particle in constant viscosity and shear thinning matrix-based viscoelastic fluids. In a constant viscosity elastic fluid, the drag coefficient decreases at low levels of elasticity, and increases at high elasticity due to the large elastic stresses developing on the surface and on the wake of the particles [1,18–20]. When the shear thinning effect is added, the drag coefficient decreases as elasticity increases. Recently, Faroughi et al. [6] developed a closure drag model for a single spherical particle translating through constant viscosity elastic fluids described by the Oldroyd-B constitutive equation. However, due to strong interaction of elasticity (e.g., high Weissenberg number) and kinematic parameters (shear thinning and thickening), a general solution for this problem that can integrate all dimensions of the data and perform well over a wide range of parameters, is still missing and cannot be formed using traditional approaches.

Accordingly, the present contribution is undertaken to achieve two goals. First, we generate and condition comprehensive datasets capturing the dynamics of a spherical particle translating through constant viscosity and shear thinning viscoelastic fluids. This is done using direct numerical simulations (DNSs) following the work of Faroughi et al. [6]. The data are processed and labeled for a set of operational conditions to be consumed by supervised ML and DL methodologies. Second, we develop data-driven drag models by examining several ML-based regression methods trained, validated and tested on the generated datasets. The performance, accuracy and productivity of different models are thoroughly evaluated based on common statistical measures.

The paper is organized in the following manner: in Section 2 we present the governing equations describing the transient, incompressible and isothermal laminar flows of viscoelastic matrix-based fluids. We also present the physical system and computational domain used to generate the training datasets. In Section 3, the ML-based regression algorithms employed to predict the viscoelastic drag coefficient are described. Next, in Section 4, we present the complexity of the training datasets in detail and compare the performance of each ML model employed to learn the characteristics of these datasets. A meta-model, based on different ML models, is then trained to fully describe the datasets. Finally, in Section 5, we summarize the main conclusions of this work.

2. Underlying Physics

The conservation equations governing transient, incompressible and isothermal laminar flow of viscoelastic fluids are the continuity, momentum balance and constitutive equations. The continuity and momentum balance equations read as follows,

$$\nabla \cdot (\rho \mathbf{u}) = 0 \tag{1}$$

$$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) + \nabla \cdot (p\mathbf{I}) - \nabla \cdot \boldsymbol{\tau} = 0 \tag{2}$$

where ρ is the fluid density, $\mathbf{u}$ is the velocity vector, t is the time, p is the pressure, $\mathbf{I}$ is the identity tensor and $\boldsymbol{\tau}$ is the total extra-stress tensor, which is split into solvent ($\boldsymbol{\tau}_S$) and polymeric ($\boldsymbol{\tau}_P$) contributions, such that $\boldsymbol{\tau} = \boldsymbol{\tau}_S + \boldsymbol{\tau}_P$. These stress terms are obtained by the following constitutive equations,

$$\boldsymbol{\tau}_S = \eta_S \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T \right) \tag{3}$$

$$\lambda \overset{\triangledown}{\boldsymbol{\tau}}_P + \boldsymbol{\tau}_P + \frac{\alpha \lambda}{\eta_P} \boldsymbol{\tau}_P \cdot \boldsymbol{\tau}_P = \eta_P \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T \right) \tag{4}$$

where η_S and η_P are the solvent and polymeric viscosities, respectively, λ is the fluid relaxation time, α is the mobility parameter and $\overset{\triangledown}{\boldsymbol{\tau}}_P$ indicates the upper-convective time derivative of the polymeric extra-stress tensor defined as,

$$\overset{\triangledown}{\boldsymbol{\tau}}_P \equiv \frac{\partial \boldsymbol{\tau}_P}{\partial t} + \mathbf{u} \cdot \nabla \boldsymbol{\tau}_P - \boldsymbol{\tau}_P \cdot \nabla \mathbf{u} - \nabla \mathbf{u}^T \cdot \boldsymbol{\tau}_P \tag{5}$$

Equation (4) is known as the Giesekus viscoelastic constitutive model [21]. For the particular case where the mobility parameter is zero, $\alpha = 0$, Equation (4) reduces to the well-known quasi-linear elastic dumbbell fluid, the Oldroyd-B fluid. When written in the continuum formulation, viscoelastic fluid flows are well known to introduce numerical convergence difficulties at high Weissenberg numbers. This is mainly related to the lack of sufficient resolution of the discretization methods to resolve the exponential growth of stresses near critical points as the Weissenberg number is incremented. To prevent this issue in the calculation of the polymeric extra-stress tensor components, we follow the log-conformation approach [22,23] implemented in the OpenFOAM [24] computational library.

We use the above governing equations to perform an extensive set of DNSs and calculate the drag coefficient for a spherical particle translating in viscoelastic fluids. Figure 1 schematically illustrates the computational domain used in this study to simulate the unbounded viscoelastic flow around a sphere [6]. The domain size in flow direction, L_x, is considered larger than other dimensions to allow enough space for the polymer chains to be relaxed. Our numerical model and solver were comprehensively tested against this computational challenge (see, e.g., Fernandes et al. [25] and Faroughi et al. [6] for the case of an Oldroyd-B fluid flow around a sphere). All the numerical simulations were performed in parallel using several High-Performance Computing facilities (see Acknowledgments section). On a system with a 2.30 GHz AMD EPYC 7742 64-Core processor, the computational time for a single run was 12 ± 1 h.

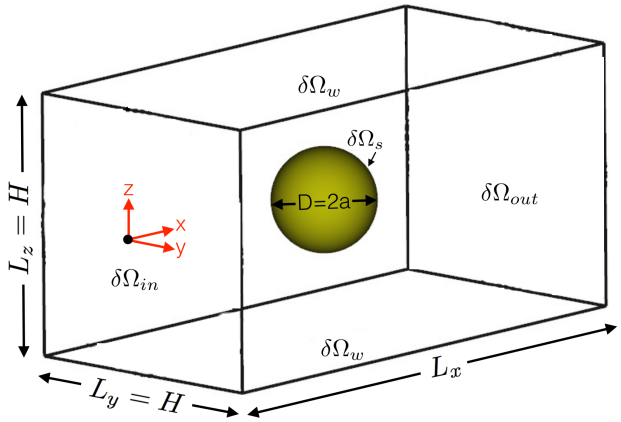


Figure 1. Schematic illustration of the computational domain (square duct) used to simulate the viscoelastic fluid flow past a sphere.

For the present problem, we define the Reynolds and Weissenberg dimensionless numbers as follows,

$$Re = \frac{2a\rho U}{\eta_0} \tag{6}$$

$$Wi = \frac{\lambda U}{a} \tag{7}$$

where U is the inlet average fluid velocity and a is the sphere radius ($D = 2a$ is the sphere diameter). Additionally, the other dimensionless numbers considered in this work are the shear thinning mobility parameter α, and the polymeric viscosity ratio ζ. The latter is also known as the characteristic retardation ratio defined as,

$$\zeta = \frac{\eta_P}{\eta_S + \eta_P} = \frac{\eta_P}{\eta_0} \tag{8}$$

where η_0 is the total fluid viscosity in the limit of vanishing shear rate and η_S and η_P are the solvent and polymeric contributions to the fluid viscosity, respectively.

Here, we carry out the calculations for the viscoelastic drag coefficient, C_D, using the surface integration of the total stress, $\boldsymbol{\tau} = \boldsymbol{\tau}_P + \boldsymbol{\tau}_S$, and pressure field, p, on the surface of the sphere as,

$$C_D = \frac{2}{\rho U^2 A} \int_{\delta\Omega_s} (\boldsymbol{\tau}_P + \boldsymbol{\tau}_S - p\mathbf{I}) \cdot \mathbf{n} \cdot \boldsymbol{x}\, dS \tag{9}$$

where A is the cross-sectional area of the sphere, $\mathbf{n}$ is the unit normal vector to the sphere surface, S, and $\boldsymbol{x}$ is the unit vector parallel to the flow direction. The results computed for the viscoelastic drag coefficient using Equation (9) are normalized by,

$$\chi = \frac{C_D}{C_D(Wi = 0)} \tag{10}$$

where χ is the viscoelastic drag coefficient correction factor [6].

3. Machine Learning Regression Algorithms

To relate the input features (i.e., a set of explanatory variables, Re, Wi, ζ and α) to the output features (i.e., the response variable, χ), different ML-based regression algorithms are employed in this work. These algorithms enable us to model multidimensional datasets which cannot be described using traditional techniques [6]. In the most basic form, the linear regression model explains a dependent variable y via a linear combination of the independent features, x_i $(i = 1, \ldots, n)$,

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n + \varepsilon \tag{11}$$

where ε is an additive error and β_j $(j = 0, \ldots, n)$ are the coefficients of the features. Despite its simplicity, this model is widely used as a baseline and a tool to analytically study the independent variables and understand the significance of the input features. To achieve more accurate estimates and prevent the overfitting issue, we consider more sophisticated regression models such as ensemble decision tree algorithms (Random Forest and Extreme Gradient Boosting) and a Deep Neural Network (DNN). These methods possess their own challenges and should be applied with special care in scenarios where the training data are sparse [26].

The Random Forest (RF) is an ensemble learning technique that alleviates the overfitting issue and offers excellent performance within the scope of the training data [27]. In this approach, multiple decision trees are constructed at training time and the mean of the individual predictions is reported as the output of the ensemble method. At each candidate splitting within each tree model, a randomly selected subset of feature space is used. This trick has proven to be very effective and the resulting models are usually robust to the overfitting problem [28]. The RF models have emerged as a versatile and highly accurate regression methodology requiring little tuning while providing interpretable outputs. In summary, the RF algorithm includes (i) randomly select n subsamples, (ii) train the regression tree for each sample, and finally (iii) average all prediction results from all trees. This algorithm has 16 main hyperparameters as listed in Table 1, and the most important ones to tune are the n_estimators that represents the total number of trees in the forest, and Max_feature that represents the number of features to consider when looking for the best split. The selection of the feature for node splitting from a random set of features decreases the correlation between different trees and, thus, the average prediction of multiple regression trees is expected to have lower variance than individual regression trees [28].

The Extreme Gradient Boosting (XGBoost) algorithm proposed by Chen and Guestrin [29] is an improved algorithm of gradient boosting to recognize complex, nonlinear patterns inside datasets. One of the differences between XGBoost and RF models is related to the way the trees are built. In RF, trees are built independent of each other, but, in XGBoost, a new tree is added to complement the already built ones [30]. A prediction value (y_i^*) from an ensemble model can be represented as,

$$y_i^* = h(x_i) = \sum_{k=1}^{K} f_k(x_i),\ i = 1, \ldots, N \tag{12}$$

where f_k is a regression tree, and $f_k(x_i)$ represents the score given by the k-th tree to the i-th observation in data. The goal in XGBoost is to minimize the regularized objective function expressed as [30],

$$L = \sum_{i=1}^{N} \Lambda(y_i, y_i^*) + \sum_{k=1}^{K} \Omega(f_k) \tag{13}$$

in order to choose functions f_k. Here, N is the number of observation (e.g., rows of data), Λ is the loss function which measures the accuracy and performance of the model in terms of its relationship between input (x_i) and output (y_i) features, and the penalty term Ω is included to prevent too large complexity of the model, being defined as [30]

$$\Omega(f_k) = \gamma T + \frac{1}{2}\beta||\omega||^2 \tag{14}$$

where γ and β are parameters controlling penalty for the number of leaves, T, and magnitude of leaf weights, ω, respectively. This penalty term makes XGBoost unique compared to general tree boosting methods. It has two main goals; (i) to prevent overfitting, and (ii) to simplify the end model produced by this algorithm. In addition to this regularized loss function, XGBoost is reinforced with two additional features that further prevent overfitting. First, the weights of each new tree can be scaled down reducing an impact of a single tree

on the final score, which provides more room for next trees to improve the model [30]. The second feature is a column sampling working in a similar way as RF where each tree is built using only a column-wise sample from the training dataset [31]. The XGBoost algorithm has 24 main hyperparameters as listed in Table 1, divided in three categories: (a) general parameters as a guide to the overall functioning, (b) booster parameters as a guide to the individual booster at each step, and (c) learning task parameters as a guide to the optimization performance.

Table 1. Tunable hyperparameters in different machine learning regression models applied in this study.

Model	Hyperparameters	Total Number
Random Forest	Bootstrap, criterion, max_depth, max_features, max_leaf_nodes, min_impurity_decrease, min_impurity_split, min_samples_leaf, min_samples_split, min_weight_fraction_leaf, n_estimators, n_jobs, oob_score, random_state, verbose, warm_start	16
XGBoost	base_score, booster, colsample_bylevel, colsample_bynode, colsample_bytree, gamma, importance_type, learning_rate, max_delta_step, max_depth, min_child_weight, missing, n_estimators, n_jobs, nthread, objective, random_state, reg_alpha, reg_lambda, scale_pos_weight, seed, silent, subsample, verbosity	24
DNN	activation, alpha, batch_size, beta_1, beta_2, early_stopping, epsilon, hidden_layer_sizes, learning_rate, learning_rate_init, max_iter, momentum, n_iter_no_change, nesterovs_momentum, power_t, random_state, shuffle, solver, tol, validation_fraction, verbose, warm_start	21

The Deep Neural Network (DNN) algorithms are one of the most commonly applied regression algorithms for stationary datasets [32]. The popular implementation is the multilayer perceptron (MLP), in which the architecture is optimized by iterating on various numbers of hidden neurons and layers that would lead to the best model with the highest accuracy on a dataset [33]. In MLP algorithm, the model is expressed as [34],

$$y = h\left(\varphi_0 + \sum_{j=1}^{N} \varphi_j g\left(\sum_{i=1}^{M} \theta_i x_i\right)\right) \quad (15)$$

where N and M represent the number of neurons in the hidden and input layers, respectively, g and h denote the transfer functions for the input layer and hidden layer, and the vector matrices of θ and φ represent the weight values for neurons in the input and hidden layers, respectively. A cost function is defined to measure the accuracy and performance of the model in terms of its relationship between input and output features. The objective in MLP is to minimize the cost function defined as [35],

$$Arg\ min : \frac{1}{2n} \sum_{i=1}^{n} (h(x_i) - y_i)^2 \quad (16)$$

where n is the number of samples and $h(x_i)$ represents the model prediction. The batch gradient descent technique and stochastic gradient descent are the well-known optimization algorithms used to minimize the cost function [26]. These algorithms find the direction (gradient) necessary to minimize the cost function and often they are known as a hill-climbing approach [36]. It is important to note that a DNN model might have the highest

accuracy in the training set obtained from multiple attempts, but it is prone to memorize the trend, noise, and detail in training set instead of intuitively understanding the trend in the dataset. Therefore, it loses the prediction capability. In order to avoid this, one may set a stoppage criteria for learning where the model tests its predictive capability on a validation set and stops training when validation accuracy departs from training accuracy. The DNN algorithm in total has 21 main hyperparameters as listed in Table 1, and the most important ones are hidden_layer_sizes and learning_rate [37].

4. Results and Discussion

This section reports the processes taken to generate training datasets and develop ML models that predict the drag coefficient correction of a spherical particle translating in viscoelastic fluids described by the Oldroyd-B (constant viscosity fluids) and Giesekus (shear thinning fluids) constitutive equations. Figure 2 shows a summary of the inputs and output data considered in the development of the ML models described in Section 3. The input features are Reynolds number, Weissenberg number, retardation ratio, and mobility parameter, and the output variable is the drag coefficient normalized by the Newtonian value, i.e., χ as defined in Section 2. Figure 2 shows a schematic architecture for the DNN model.

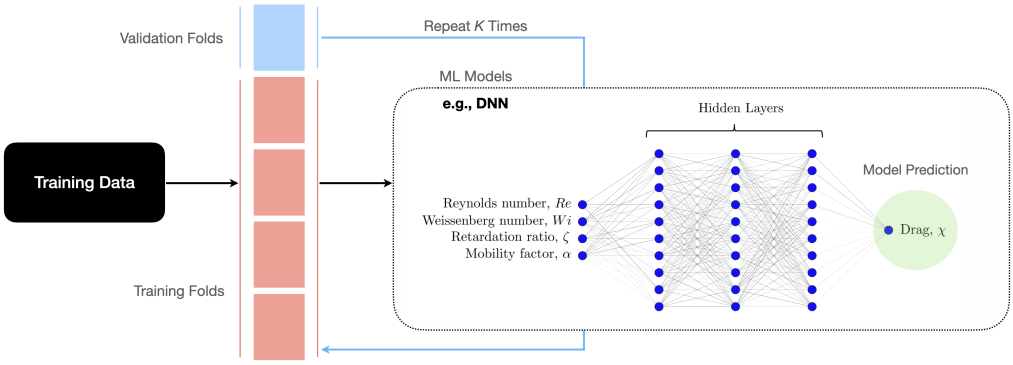


Figure 2. A summary of the training approach, input and output features to develop ML-based regression models (e.g., it is shown for a DNN model).

4.1. Data Collection and Analysis for Oldroyd-B Fluids

For constant viscosity viscoelastic fluids [38], the relaxation time, λ, and retardation ratio, ζ, are the two important characteristics that define the viscoelastic behaviors. These fluids are generally modeled using the Oldroyd-B constitutive equation [39], and best represent very dilute polymer solutions at low Weissenberg number. Direct numerical simulations (DNSs), following the methodology implemented by Faroughi et al. [6] on the physical system elaborated in Section 2, were employed to generate the training dataset for the viscoelastic drag coefficient correction of a sphere translating in Oldroyd-B fluids ($\alpha = 0$). For that purpose, the range of the input features varied within $0 < Re \leq 50$, $0 \leq Wi \leq 10$, and $0 < \zeta < 1$, which resulted in a total of 12,120 input values (hereafter we call this dataset *OB-set*). In addition to this dataset, we also generated a blind dataset, from a total of 60 DNSs, with an input feature coverage outside the scope or interpolated within the scope of *OB-set*. The blind dataset did not enter in the initial training, validation and testing phases and is used to scope the inference of the ML models beyond the limits of the training set, i.e., test the ML models' generalization as described in Section 4.4.

Figures 3 and 4 show the flow characteristics associated with the *OB-set*. In Figure 3a,b, the contours of the viscoelastic drag coefficient correction, χ, are presented for Reynolds numbers $0 < Re \leq 1$ and Weissenberg numbers $0 \leq Wi \leq 10$ at two polymeric retardation ratios $\zeta = 0.1$ and 0.9, respectively. The viscoelastic drag coefficient correction follows the same behavior for both $\zeta = 0.1$ and 0.9 across all considered Wi and Re numbers. As shown in Figure 3c,d, the viscoelastic drag coefficient correction slightly decreases at

low Wi number, hits a minimum and then sharply increases with Wi number. The drag enhancement is more significant at higher ζ values (star blue symbols).

In Figure 4a,b, the contours of the viscoelastic drag coefficient correction, χ, are presented for Weissenberg numbers $0 \leq Wi \leq 10$ and Reynolds numbers $Re \leq 50$ at two polymeric retardation ratios $\zeta = 0.05$ and 0.9, respectively. For both cases, the viscoelastic drag coefficient correction decreases with the increase of inertia and increases with Wi, being more noticeable for higher ζ. Figure 4c,d, show the behavior of χ at a fixed Reynolds number, $Re = 50$. The viscoelastic drag coefficient correction increases up to $Wi < 0.2$, then stays more or less constant up to $Wi \approx 2$, and then increases with Wi number, being this behavior more abrupt at higher retardation ratios.

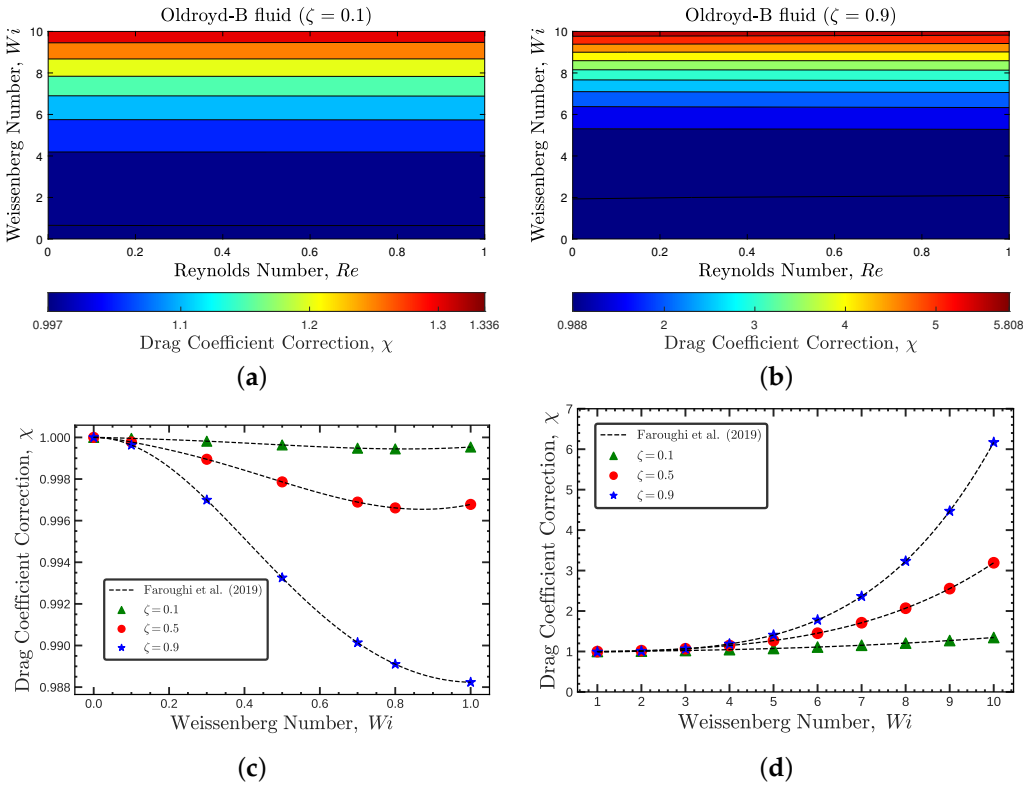


Figure 3. Contours of the viscoelastic drag coefficient correction, χ, for the Oldroyd-B fluid at $0 < Re \leq 1$, $0 \leq Wi \leq 10$ and (**a**) $\zeta = 0.1$ and (**b**) $\zeta = 0.9$. Panels (**c**,**d**) show the variation of χ with Weissenberg numbers for different values of ζ at $Re = 1$.

Figure 5 shows the contours of the normal component of the dimensionless polymeric stress, $\boldsymbol{\tau}_{xx}$, for different values of Wi and ζ at $Re = 1$. The left column in Figure 5 shows the $\boldsymbol{\tau}_{xx}$ contours at $Wi = 1$, and the right column shows the same but at $Wi = 10$. As expected, one observes that the magnitude of $\boldsymbol{\tau}_{xx}$ increases as ζ increases. In addition, the location at which the maximum value of $\boldsymbol{\tau}_{xx}$ occurs shifts from the top/bottom flow separation points to the wake of the particle as Wi increases. At $Wi = 10$, for both ζ values, a long wake was also observed in the downstream region of the flow, where extensional flow dominates due to the significant effects of the flow elasticity.

Due to the similarity observed in the behavior of χ under different flow conditions (Re, Wi and ζ), the *OB-set* is an ideal dataset to be used in developing the building blocks for the ML-based models predicting the viscoelastic drag coefficient correction. However, the *OB-set* does not represents a large body of fluids that are encountered in nature or industrial applications. This issue is rectified in the next section using the shear thinning Giesekus constitutive equation [21] to augment the data.

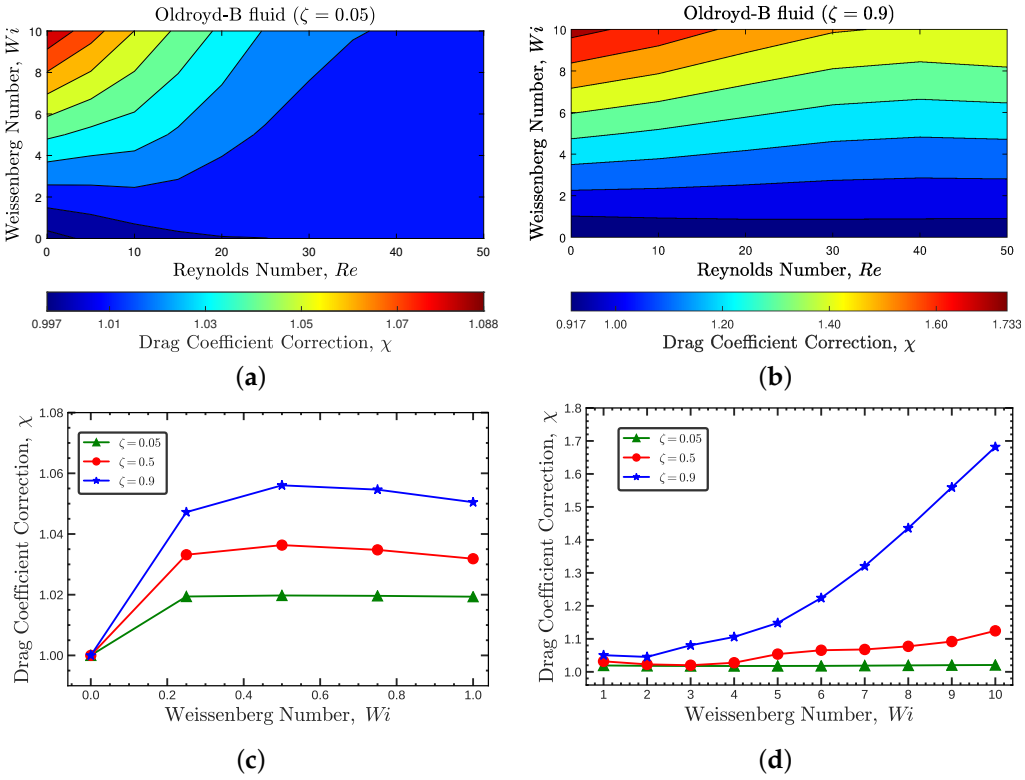


Figure 4. Contours of the viscoelastic drag coefficient correction, χ, for the Oldroyd-B fluid at $0 < Re \leq 50$, $0 \leq Wi \leq 10$ and (**a**) $\zeta = 0.05$ and (**b**) $\zeta = 0.9$. Panels (**c**,**d**) show the variation of χ with Weissenberg numbers for different values of ζ at $Re = 50$.

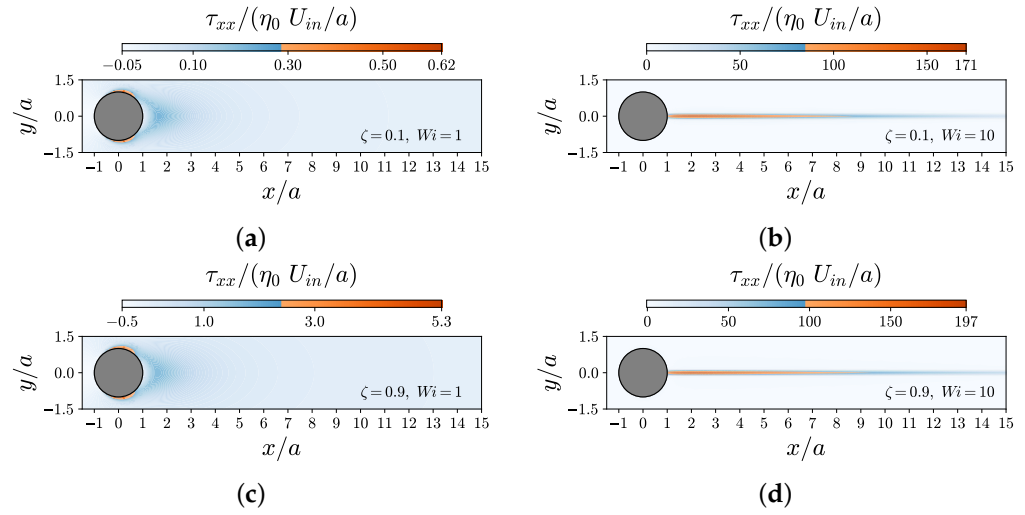


Figure 5. Contours of the dimensionless normal component of the polymeric extra-stress tensor $\boldsymbol{\tau}_{xx}$ for the Oldroyd-B fluid at $Re = 1$ for different values of ζ and Wi: (**a**) $\zeta = 0.1$ and $Wi = 1$, (**b**) $\zeta = 0.1$ and $Wi = 10$, (**c**) $\zeta = 0.9$ and $Wi = 1$, and (**d**) $\zeta = 0.9$ and $Wi = 10$.

4.2. Data Collection and Analysis for Giesekus Fluids

Most of the viscoelastic fluids show mid to strong shear thinning features. Shear thinning behavior leads to more complex and nonlinear dependencies at non-vanishing Weissenberg numbers, at which shear thinning effects become more pronounced. This behavior (neglected in the previous section) dramatically changes the behavior of the viscoelastic drag coefficient correction, χ. Therefore, the inference of the models trained based on the *OB-set* will certainly fail for shear thinning fluids. Hence, the *OB-set* must be augmented with data representing shear thinning fluids, or ML models developed based

on the *OB-set* must be further trained to also account for shear thinning effects. Several viscoelastic constitutive models have been developed over the past few decades to model shear thinning fluids [40]. The Giesekus fluid is the one generally used to best represent the polymer molecules contribution to the momentum exchange in dilute to semi-dilute polymer solutions [41]. This model is based on a concept of configuration-dependent molecular mobility, and thus, the viscoelastic component of the polymeric stress tensor is represented by λ and ζ as well as the mobility parameter, α. The mobility parameter varies between zero and unity (practically 0.5 is the upper limit [41]) and accounts for the shear thinning behavior of the viscoelastic fluids.

We again used DNSs to generate the training dataset for the viscoelastic drag coefficient correction of a sphere translating in Giesekus fluids. A total of 4950 numerical simulations for the unbounded flow of the shear thinning viscoelastic Giesekus fluid past a sphere (using the physical system described in Section 2) were performed. Hereafter, we call this dataset *GI-set*. Simulations were conducted under a wide range of numbers for the input features, specifically $0 < Re \leq 50$, $0 \leq Wi \leq 10$, $0 < \zeta < 1$, and $0 < \alpha < 1$. The end goal is to use *GI-set* to augment the *OB-set* and develop a ML-based meta-model that can be used by the scientific community to obtain a prediction of the dimensionless viscoelastic drag coefficient correction of a sphere translating in both Oldroyd-B and Giesekus fluids. We also generated a blind dataset using DNSs with flow features (Re, Wi, ζ and α) outside the ranges provided in the generation of *GI-set*. This dataset, consisting of 64 data points, is used to scope the accuracy of the ML models when inferred outside the limits of the training dataset, see Section 4.4.

Figures 6 and 7 present the flow characteristics associated with *GI-set*. Figure 6 shows the contours of the viscoelastic drag coefficient correction, χ, for $0 < Re \leq 50$ and $0 \leq Wi \leq 10$ at polymeric retardation ratios $\zeta = 0.05$ and 0.9, and mobility parameters $\alpha = 0.1$ and 0.9. As shown in Figure 6, increasing inertia (i.e., Re number) leads to a reduction in the viscoelastic drag coefficient correction, similar to the behavior observed for the Oldroyd-B fluid. However, interestingly, increasing the elasticity of the flow (increasing the Wi number) results in a sharp reduction of the viscoelastic drag coefficient correction. The reduction is more pronounced at higher retardation ratio results. This behavior is totally different than what was observed for constant viscosity fluids, where the viscoelastic drag coefficient correction increases with Wi. Additionally, when the mobility parameter is increased (i.e., stronger shear thinning effect), it promotes the drag reduction even further as shown in Figure 6d.

To better illustrate the complexity of the flow and the effects of all flow features (Re, Wi, ζ and α), we compare the contours of the normal component of the dimensionless polymeric stress, $\boldsymbol{\tau}_{xx}$, in Figure 7. These comparisons are shown for different values of Re (top line $Re = 0.01$ and bottom line $Re = 50$) and ζ (left column $\zeta = 0.05$ and right column $\zeta = 0.9$) at fixed $Wi = 5$ and $\alpha = 0.1$. One observes that the magnitude of $\boldsymbol{\tau}_{xx}$ increases as Re number increases. In addition, the change in the flow structure (i.e., flow separation and formation of symmetric eddies) due to Re number shifts around the location where the maximum of $\boldsymbol{\tau}_{xx}$ occurs. Figure 7 also shows that an increase in the retardation ratio ζ promotes an elongated wake in the downstream of the flow. In Figure 8, we show the contours of $\boldsymbol{\tau}_{xx}$ for the shear thinning Giesekus viscoelastic fluid for two different values of the mobility parameter, $\alpha = 0.1$ and 0.5, at fixed $Re = 1$, $Wi = 2$ and $\zeta = 0.5$. As expected and illustrated in Figure 8, increasing the mobility parameter decreases the stress overshoot on the surface as well as in the wake of the sphere, which in turn drastically hinders the enhancement of the viscoelastic drag coefficient correction due to elasticity (a behavior that was observed for Oldroyd-B fluids).

Figures 3–8 collectively show the presence of a complex, multidimensional dynamics associated with a single spherical particle flowing through a viscoelastic fluid. All flow features (Re, Wi, ζ and α) strongly affect the flow fields and hence the viscoelastic drag coefficient correction (χ). These effects can be hardly decoupled to derive an analytical/empirical or semi-empirical expression for the viscoelastic drag coefficient correction

prediction. A machine learning model, however, can be developed to learn these hidden features, in addition to features that are obvious to us in the data, to predict the viscoelastic drag coefficient of a spherical particle translating in an unbounded Oldroyd-B and Giesekus viscoelastic fluids.

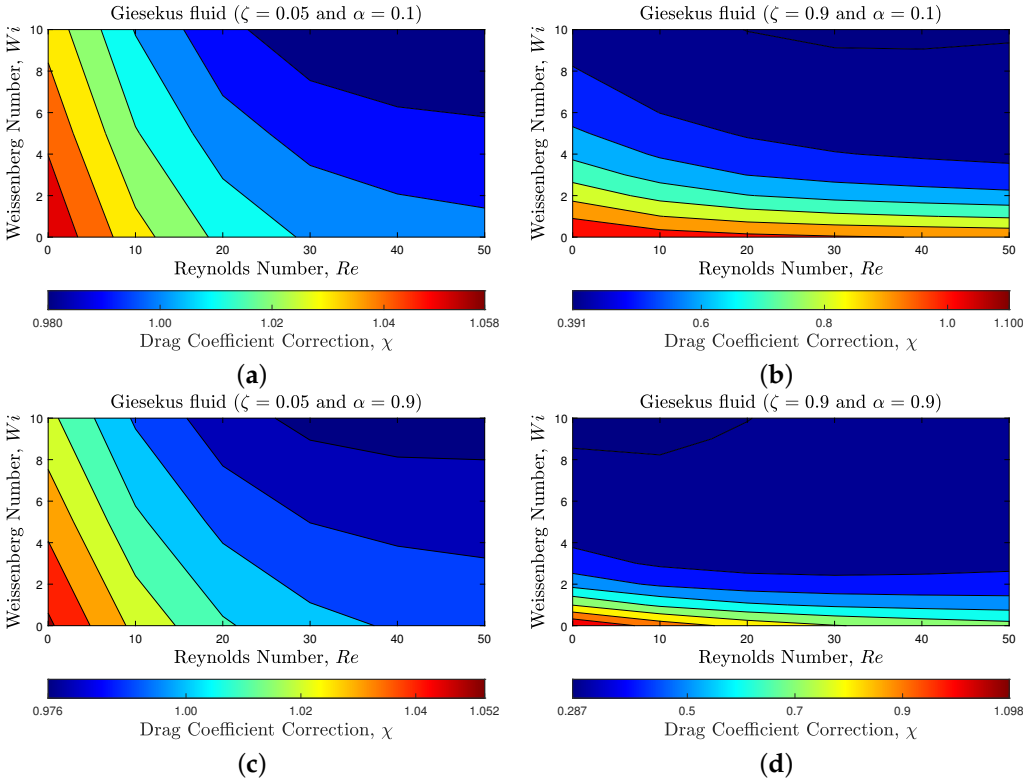


Figure 6. Contours of the viscoelastic drag coefficient correction, χ, for the shear thinning Giesekus viscoelastic fluid for $0 < Re \leq 50$, $0 \leq Wi \leq 10$ and different values of α and ζ: (**a**) $\zeta = 0.05$ and $\alpha = 0.1$, (**b**) $\zeta = 0.9$ and $\alpha = 0.1$, (**c**) $\zeta = 0.05$ and $\alpha = 0.9$, and (**d**) $\zeta = 0.9$ and $\alpha = 0.9$.

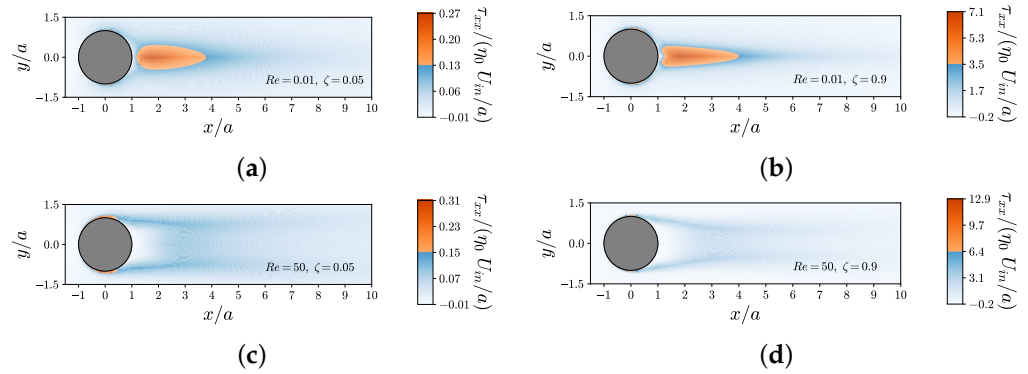
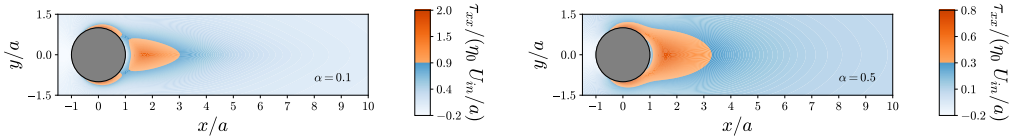


Figure 7. Contours of the dimensionless normal component of the polymeric extra-stress tensor $\boldsymbol{\tau}_{xx}$ for the shear thinning Giesekus viscoelastic fluid at fixed $Wi = 5$, $\alpha = 0.1$ and different values of Re and ζ: (**a**) $Re = 0.01$ and $\zeta = 0.05$, (**b**) $Re = 0.01$ and $\zeta = 0.9$, (**c**) $Re = 50$ and $\zeta = 0.05$, and (**d**) $Re = 50$ and $\zeta = 0.9$.



(**a**) (**b**)

Figure 8. Contours of the dimensionless normal component of the polymeric extra-stress tensor $\boldsymbol{\tau}_{xx}$ for the shear thinning Giesekus viscoelastic fluid at fixed $Re = 1$, $Wi = 2$, $\zeta = 0.5$ and different values of mobility parameter: (**a**) $\alpha = 0.1$ and (**b**) $\alpha = 0.5$.

4.3. ML Models Development

In this section, we leverage the *OB-set* and *GI-set* to train, validate and test the ML models discussed in Section 3. Based on these datasets, the design space for the input variables is defined as $0 < Re \leq 50$, $0 \leq Wi \leq 10$, $0 < \zeta < 1$ and $0 < \alpha < 1$. First, a normalization stage is followed to restrict the input value range, which transforms the original input feature x to $\tilde{x} = (x - x_{min})/(x_{max} - x_{min})$. This is a common practice which speeds up learning and leads to faster convergence, especially for the DNN model. Next, we split each dataset to a training set (consisted of 80% of the data) and a test set (consisted of 20% of the data) that are bundled randomly. One of the primary objectives in this section is to improve the performance score, based on data patterns and observed evidence. To achieve this objective, the ML model architecture needs to be optimized by tuning a specific set of hyperparameters defined for each model (see Table 1 for a complete list of hyperparameters of the ML models considered in this study).

4.3.1. Hyperparameter Tuning

Hyperparameter tuning relies more on experimental results than theory, and thus the best method to determine the optimal settings is to try many different combinations and evaluate the performance of each model. However, evaluating each model only on the training set can lead to overfitting (i.e., a model scores very well on the training set but performs poorly on the test set or blind dataset). Routinely, a subset of data from the training set, known as validation set, is reserved for this purpose. We adopt the K-Fold cross-validation (K-Fold CV) technique [42,43] to conduct hyperparameter tuning. In K-Fold CV technique, the training set is further split into K number of subsets, called folds, as schematically shown in Figure 2. The ML model is then iteratively fitted K times; each time, the training is done on K-1 of the folds and evaluation is done on the Kth fold (the validation set). At the very end of training, we average the performance on each of the folds to come up with final validation metrics for the model. The trained models each defined with specific hyperparameters are compared against each other, and the best one that offer the highest accuracy metrics is selected. In this study, unless otherwise stated, we apply 10-Fold CV, i.e., to assess a different set of hyperparameters, we split our training dataset into 10 folds and train and evaluate each model with selected hyperparameters 10 times. If we select X sets of hyperparameters using 10-Fold CV technique, which represents 10X training loops on the entire training dataset (e.g., X = 24 for XGBoost ML model). This process is thus computationally tedious. To facilitate that, K-Fold technique is coupled with RandomSearchCV algorithm to optimize selected hyperparameters [44]. This coupled approach tries random combinations within a range of values given for each parameter, with a defined number of iterations of random searches. The training time, using a workstation with 48 CPU cores and a NVIDIA RTX A8000 GPU, was on average 7 ± 0.5 h and 320 ± 8 h, respectively, for each iteration and all iterations required to perform hyperparameter tuning for a model.

To train and compare the performance of the ML models, the accuracy is evaluated based on three common statistical measures, R^2, RMSE, and MAPE. The latter, MAPE, represents the mean-absolute-value of the ratio of estimation errors to actual values. A lower MAPE value indicates that the predicted value is closer to the ground truth. The RMSE represents the root-mean-square error, which is also used to measure the differences between

actual and predicted values by a model. The R^2 coefficient represents the fitness performance, i.e., higher values of R^2, with a max value of 1, are preferred. The mathematical expressions for R^2, RMSE and MAPE are as follows [45,46],

$$\begin{aligned} R^2 &= 1 - \frac{\sum_{i=1}^{n}(y_i - y_i^*)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \\ \text{RMSE} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^*)^2} \\ \text{MAPE} &= \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - y_i^*}{y_i}\right| \times 100\% \end{aligned} \tag{17}$$

where n is the total number of observations, y_i is the actual value, y_i^* is the predicted value and $\bar{y}_i$ is the average of the actual values.

4.3.2. Training and Testing

We first train, validate and test three different ML-based regression models to predict the drag coefficient correction of a single spherical particle translating through a viscoelastic fluid described by the Oldroyd-B constitutive equation (using the *OB-set*). We used 10-Fold CV approach in conjunction with RandomSearchCV algorithm for hyperparameters tuning, and employed the statistical measures given in Equation (17) to analyze the accuracy of the regression models. Table 2 reports the best set of hyperparameters (i.e., best architecture) obtained for each ML model. Notice that only hyperparameters that have been tuned are reported. These architectures, tuned by cross-validation technique on the *OB-set*, offer the best statistical measures for the predictions as reported in Table 3. The accuracy between real and predicted values is remarkable for all ML models as represented by the large R^2 values in Table 3. For the *OB-set*, XGBoost is the model that presents the best R^2 with the lowest values of RMSE and MAPE.

Table 2. Optimized hyperparameters for different ML-based regression models trained, validated and tested on the *OB-set*.

Model	Hyperparameter Value
Random Forest	bootstrap = True, criterion = mse, max_depth = 110, max_features = auto, min_samples_leaf = 3, min_samples_split = 5, n_estimators = 800
XGBoost	colsample_bynode = 0.8, colsample_bytree = 0.8, learning_rate = 0.1, max_depth = 15, n_estimators = 300, objective = reg:gamma, reg_alpha = 1.2, reg_lambda = 1.3, subsample = 0.7
DNN	activation = relu, alpha = 0.0001, hidden_layer_sizes = (40, 20, 10), learning_rate = adaptive, max_iter = 8000, momentum = 0.9, n_iter_no_change = 10, solver = adam, tol = 0.0001

Table 3. Optimal statistical measures for different ML-based regression models trained, validated and tested on the *OB-set*.

	RF	XGBoost	DNN
R^2	0.9991	**0.9995**	0.9989
RMSE	0.0228	**0.0134**	0.0451
MAPE	0.0048	**0.0012**	0.0145

The residuals (or the prediction errors) and quantile-quantile (Q-Q) plots for the ML-based regression models trained, validated and tested on the *OB-set* are shown in Figure 9.

The residuals are computed as the difference between the actual value (in the test set) and the values predicted by the optimized ML models. Figure 9 shows that the data points are mainly scattered around the horizontal axis and the calculated error is mainly distributed around zero. Figure 9 also shows Q-Q plots for each model in which the probability distributions for errors are compared for both train and test sets by plotting their quantiles against theoretical quantiles [47]. The theoretical quantiles on the x-axis represents normal distribution as the base distribution. This plot readily depicts whether or not the residuals (errors) are normally distributed. If points are close to the normal line, $y = x$, then residuals are assumed to be normally distributed. It can be seen that, for all models, most of the errors lies on $y = 0$ line and data follow a heavy tail distribution [47]. Figure 9d illustrates the comparison between the actual and predicted values of the viscoelastic drag coefficient correction, χ, on the *OB-set* for the XGBoost model. As shown the best fit line coincides with the identity line, which corroborates the high R^2 value presented in Table 3.

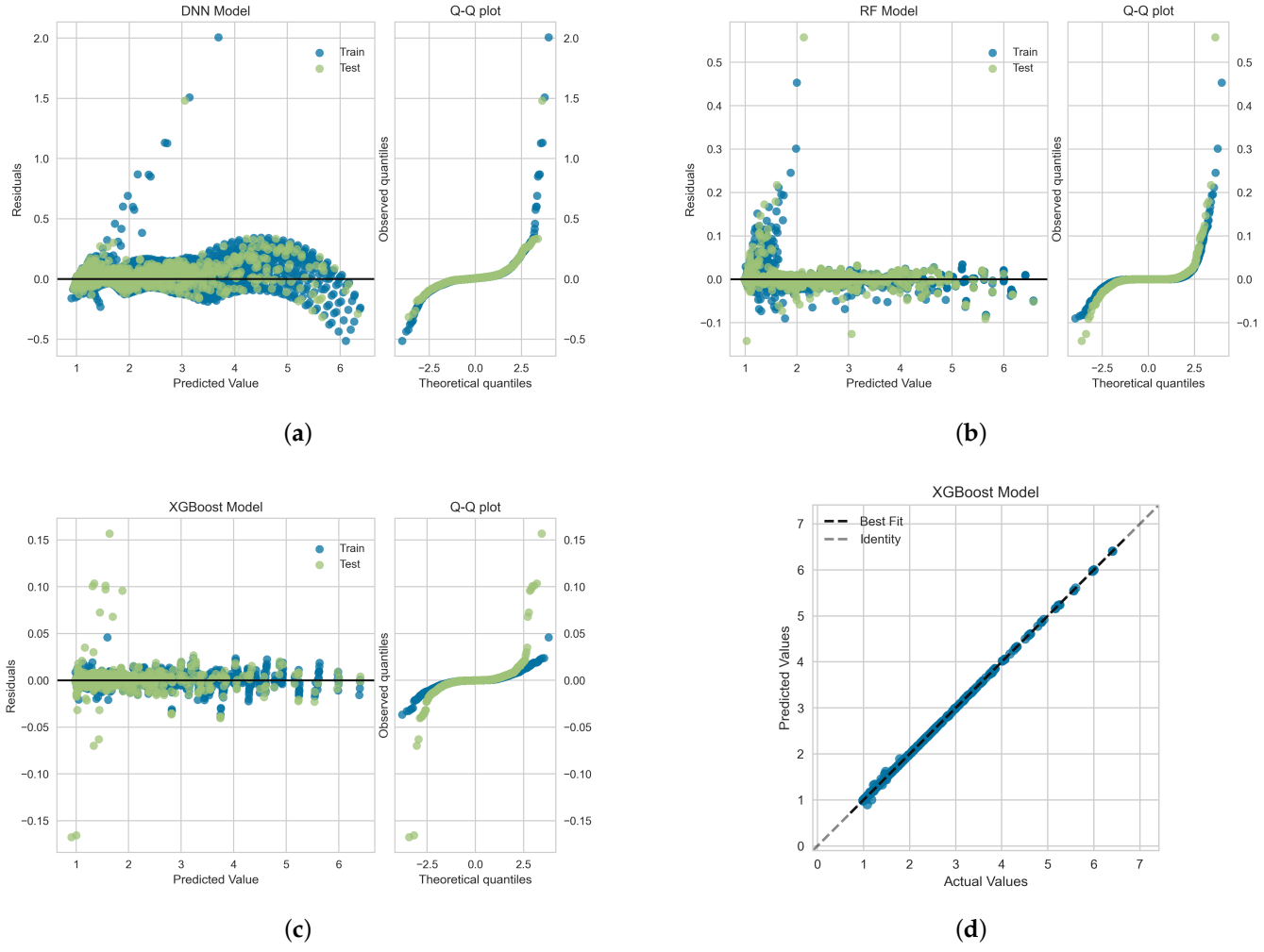


Figure 9. Residuals and quantile-quantile (Q-Q) plots obtained for the ML algorithms trained, validated and tested on the *OB-set*: (**a**) Neural Network, (**b**) Random Forrest, and (**c**) XGBoost models. Panel (**d**) shows the prediction error plot for XGBoost that yields the highest R^2 as reported in Table 3.

The ML models trained on the *OB-set* fail, as expected, when inferred against a dataset generated for shear thinning fluids (e.g., even at $\alpha = 0.1$, which is just slightly outside the scope of the *OB-set* where α is set to zero). To accurately predict the drag coefficient correction of a spherical particle translating through a more realistic viscoelastic fluid, the trained models require further augmentation. For that purpose, three different

approaches can be explored: (i) start training, validation and testing from scratch using a combination of the datasets developed for Oldroyd-B and Giesekus fluids (augmented *OB-set* and *GI-set*), (ii) use transfer learning technique [48] where models with knowledge gained on the *OB-set* are further reinforced using *GI-set*, or (iii) infuse physics in the models' architecture using the constitutive fluid models as loss or activation function broadening the range over which the ML models are valid [49]. The latter approach is outside the scope of the current study and will be explored elsewhere. The accuracy obtained for the model derived by the second approach (i.e., transfer learning) was found to be significantly lower than the first approach when tested on the blind datasets. This is mainly due to the difficulty associated with transfer learning in decision tree ML models (Random Forest in particular where there is a limited capacity to accommodate local changes [50]). Thus, we adopted the first approach to develop ML-based models that satisfies both Oldroyd-B and Giesekus fluids. This approach is also challenging because datasets are not balanced. The weight of the *OB-set* (bigger dataset with 12,120 data points) is a lot larger than the *GI-set* (smaller dataset with 4950 data points), and consequently ML models will be more biased towards the *OB-set* (e.g., undermines the effects of α on the models' predictability). To resolve this issue, we used synthetic minority over-sampling technique [51], SMOTE, which blends under-sampling of the majority set (*OB-set*) with a special form of over-sampling of the minority set (*GI-set*). In SMOTE, we synthesized elements for the minority set, based on the data that already exist. It works randomly by picking a point from the minority set and computing the k-nearest neighbors for this point. The synthetic points for minority set (*GI-set*) are placed between the chosen point and its neighbors. This process continues until we reach balanced states for both datasets, hereafter we call this dataset *SMOTE-set* containing 21,750 data points.

Again, we used 10-Fold CV approach in conjunction with the RandomSearchCV algorithm for the hyperparameters tuning of the ML models trained,validated and tested on *SMOTE-set*. Table 4 reports the best set of hyperparameters (i.e., best architecture) obtained for each one of the ML model employed in this work. These architectures offer the best statistical measures (R^2, RMSE and MAPE) for the ML-based regression models as reported in Table 5. The accuracy obtained for all ML models is acceptable as represented by the large R^2 and low RMSE values. For the *SMOTE-set*, again, the XGBoost model possesses the highest R^2, and the lowest values of RMSE and MAPE. This result is in agreement with the literature [44,52–54] and shows that the decision-tree models perform better than neural network models to learn hidden features on relatively midsize datasets. However, performing well on the test set still does not guarantee the accuracy of decision-tree-based regression models when inferred on blind datasets generated outside the scope of training set or interpolated within the training sets (see Section 4.4).

Table 4. Optimized hyperparameters for the different ML-based regression models trained, validated and tested on *SMOTE-set*.

Model	Hyperparameter Value
Random Forest	bootstrap = True, criterion = mse, max_depth = 45, max_features = auto, min_samples_leaf = 3, min_samples_split = 5, n_estimators = 950
XGBoost	colsample_bynode = 1, colsample_bytree = 0.8, learning_rate = 0.2, max_depth = 5, n_estimators = 400, objective = reg:gamma, reg_alpha = 1.2, reg_lambda = 1.3, subsample = 0.7
DNN	activation = relu, alpha = 0.0001, hidden_layer_sizes = (120, 50, 20, 3), learning_rate = adaptive, max_iter = 10,000, momentum = 0.95, n_iter_no_change = 15, solver = adam, tol = 0.0001

Table 5. Optimal statistical measures for different ML-based regression models trained, validated and tested on *SMOTE-set*.

	RF	XGBoost	DNN
R^2	0.9965	**0.9967**	0.9945
RMSE	0.0388	**0.0378**	0.0463
MAPE	0.0131	**0.0127**	0.0216

In Figure 10, we show the residuals and Q-Q plots for the ML models trained, validated and tested on *SMOTE-set*. The residuals are computed as the difference between the actual value (in the test set) and the values predicted by the optimized ML models. Figure 10 again shows that the data points are mainly scattered around the horizontal axis and the calculated error is mainly distributed around zero. The Q-Q plots for Random Forest and DNN models again depict that most of the errors lies on $y = 0$ line, within the standard deviation range, and data follow a heavy tail distributions [47]. For the XGBoost model, the residuals are closer to the normal line, $y = x$, and thus, they follow a distribution closer to normal distribution. Figure 9d illustrates the comparison between the actual and predicted values of the viscoelastic drag coefficient correction, χ, for the *SMOTE-set* using the XGBoost model. This comparison corroborates the high R^2 values presented in Table 5 and the fact that the residual plot for XGBoost model is symmetric around $y = 0$ as shown in Figure 9c.

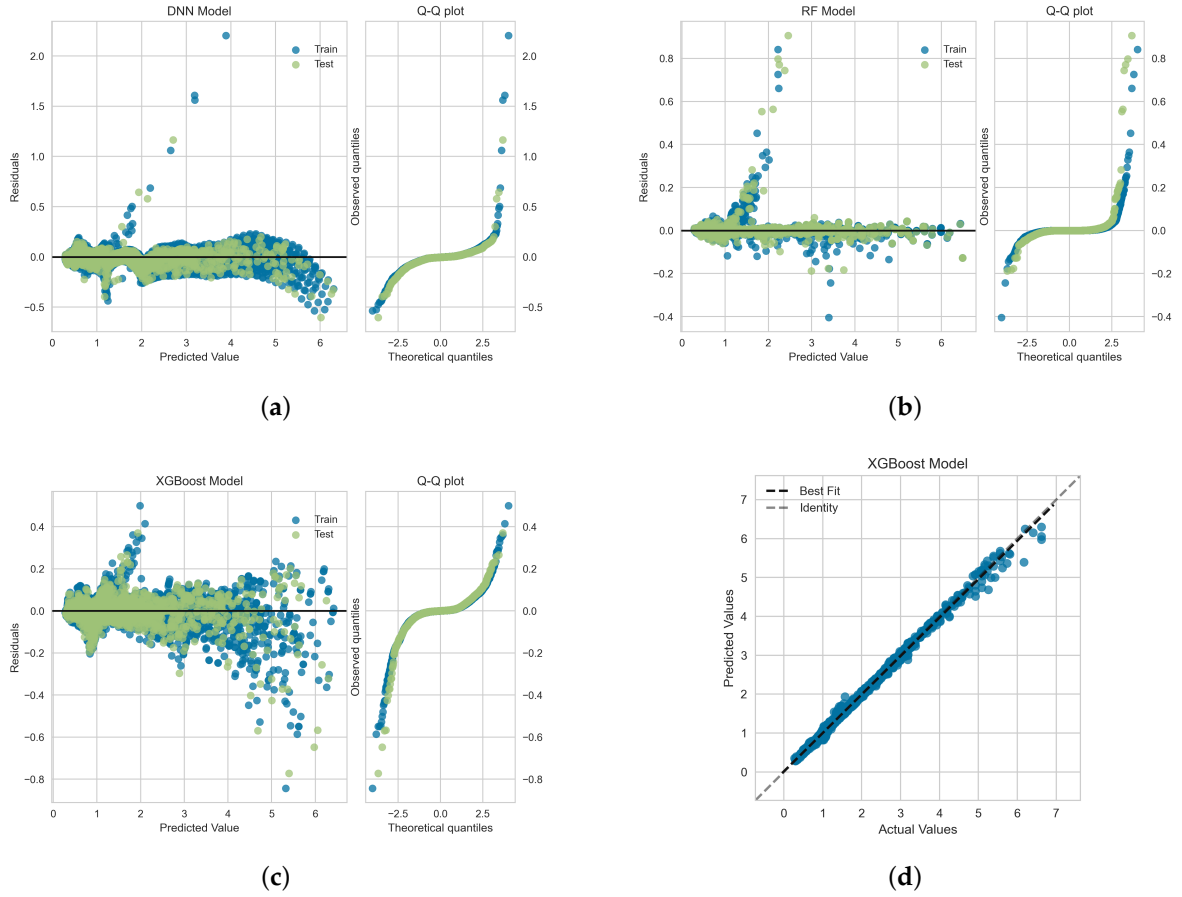


Figure 10. Residuals and quantile-quantile (Q-Q) plots obtained for the ML algorithms trained, validated and tested on *SMOTE-set*: (**a**) Neural Network, (**b**) Random Forrest, and (**c**) XGBoost models. Panel (**d**) shows the prediction error plot for XGBoost that yields the highest R^2 as reported in Table 5.

4.4. Models Performance on Blind Datasets

To further evaluate the performance of the ML models trained on *SMOTE-set* in Section 4.3, we test them against blind datasets. We designed the blind datasets to have an input feature coverage outside the scope of training set or interpolated within the training sets. The blind datasets are generated using DNSs on the physical system described in Section 2 following the work of Faroughi et al. [6] for both Oldroyd-B and Giesekus fluids.

The blind dataset for the Oldroyd-B fluid is constructed with a total of 60 DNS runs using $\zeta = \{0.25, 0.5, 0.75, 0.9\}$, $Wi = \{0.1, 0.3, 0.5, 0.65, 0.8, 1, 1.2, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$ at a constant $Re = 1$. Figure 11 shows the comparisons between the real values for the viscoelastic drag coefficient correction (obtained by DNSs and represented with a solid line) and the values predicted by the ML models (represented by symbols). The ML models used in this comparison are those trained, validated and tested based on the *OB-set*. As depicted in Figure 11, all models perform very well to predict the blind dataset. The statistical measures for ML models to predict this blind dataset are reported in Table 6. The XGBoost model performs superior than other models and its predictions are in a very good agreement with the numerical results.

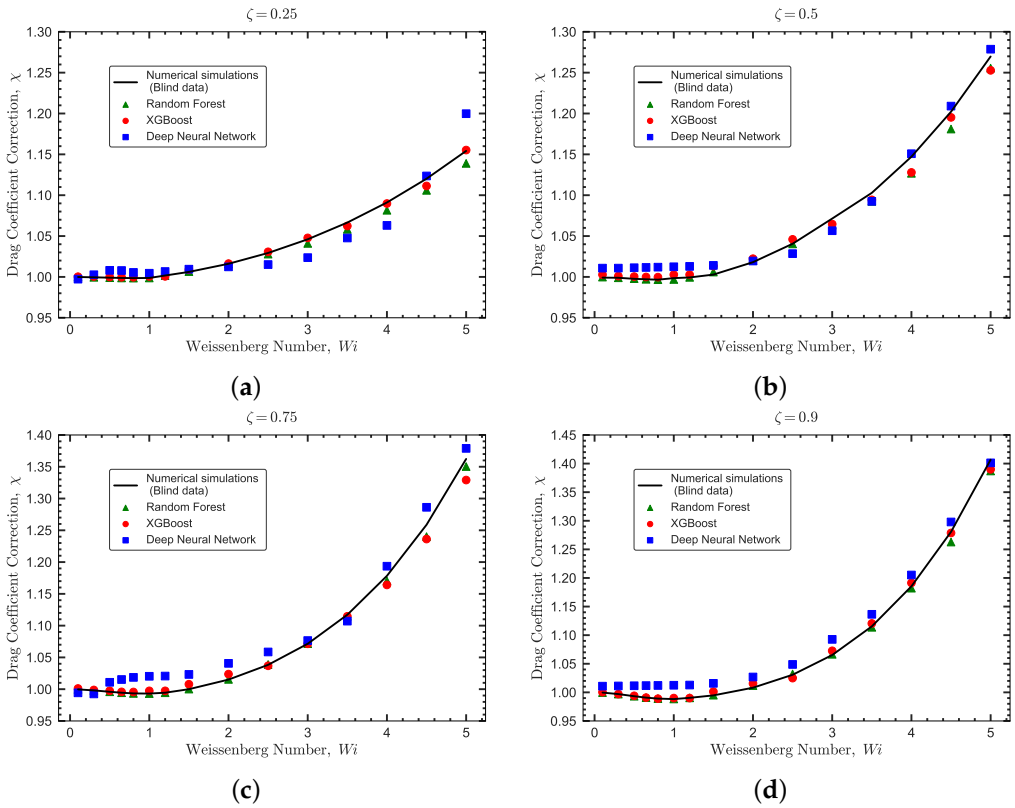


Figure 11. Validation of the ML models against the blind dataset generated for the Oldroyd-B fluid. The comparisons are between the DNSs (solid lines showing the real values for the viscoelastic drag coefficient correction) and the predicted values by the ML models. The comparisons are shown at $Re = 1$ for different values of ζ: (**a**) $\zeta = 0.25$, (**b**) $\zeta = 0.5$, (**c**) $\zeta = 0.75$ and (**d**) $\zeta = 0.9$. The predictions obtained with the Deep Neural Network, Random Forest and XGBoost models are represented by square, triangle and circle symbols, respectively.

Table 6. Statistical measures for different ML-based regression models tested against the blind dataset provided for Oldroyd-B fluids.

	RF	XGBoost	DNN
R^2	0.9926	**0.9931**	0.9621
RMSE	0.0085	**0.0074**	0.0173
MAPE	0.051	**0.0042**	0.0139

The blind dataset for Giesekus fluids is constructed with a total of 64 DNS runs using $Re = \{0.3, 75\}$, $\zeta = \{0.15, 0.8\}$, $\alpha = \{0.2, 0.4\}$, and $Wi = \{0.2, 0.4, 0.6, 0.8, 1, 2, 3, 4\}$. Figure 12 shows the comparisons for two sample sets between the real values of the viscoelastic drag coefficient correction obtained by DNS (solid lines) and the predicted values by the ML models (symbols). The ML models used in this comparison are trained, validated and tested based on *SMOTE-set*. The statistical measures to predict this blind dataset are reported in Table 7. A sharp reduction in prediction performance is noticed for all models. This is due to two reasons: (i) the presence of values of *Re* which are out of the limits of the *SMOTE-set*, and (ii) the sparsity of data points in the combined dataset, i.e., *OB-set* and *GI-set*. Even using the SMOTE technique to balance the data sets (enforce the effect of high *Re* numbers and α), the ML models trained on *SMOTE-set* show a relatively poorer performance in predicting the blind dataset compared to the same models trained and tested on the *OB-set* (see Figure 11 and Table 6).

As shown in Figure 12, the DNN model performs slightly better than the decision tree models for both *Re* and α values. In general, ensemble decision tree models (e.g., XGBoost) are easy to train and prevent overfitting to a great extent [44,52,55]; however, they do not perform well in predicting sparse datasets where interpolation between input features is required. On the other hand, deep neural networks models are hard to train, but offer a better performance when inferred outside the scope of the training dataset or when interpolation between input features is needed [55,56]. Therefore, the DNN model provides a better potential for the generality of the model. In addition, for a ML model to be fully predictive under any new or unseen conditions (e.g., flow features), physics must complement the model. This can only be achieved using deep learning models, known as physics-based neural networks [57] or physics-guided neural networks [58] that mimic an infinitely deep model. Incorporating physics in DNN is essential in the field of particle-laden fluid flow, because it is not a data-oriented domain (i.e., large datasets can be hardly found). Developing true physics-based neural network is outside the scope of the current work, and it will be presented elsewhere. Here, to resolve this issue and provide a meta-model for the viscoelastic drag coefficient correction that can be coupled with Eulerian-Lagrangian algorithms [4], we use stacking technique [59]. This technique leverages the superiority of all developed ML models (i.e., the fact that each model performs better in a different section of the data), and is a very powerful method to increase the generality of the model in predicting unseen data.

Table 7. Statistical measures for ML-based models tested against the blind dataset provided for Oldroyd-B and Giesekus fluids.

	RF	XGBoost	DNN
R^2	0.8664	0.8566	**0.9013**
RMSE	0.0495	0.0516	**0.0428**
MAPE	0.0326	0.0341	**0.0305**

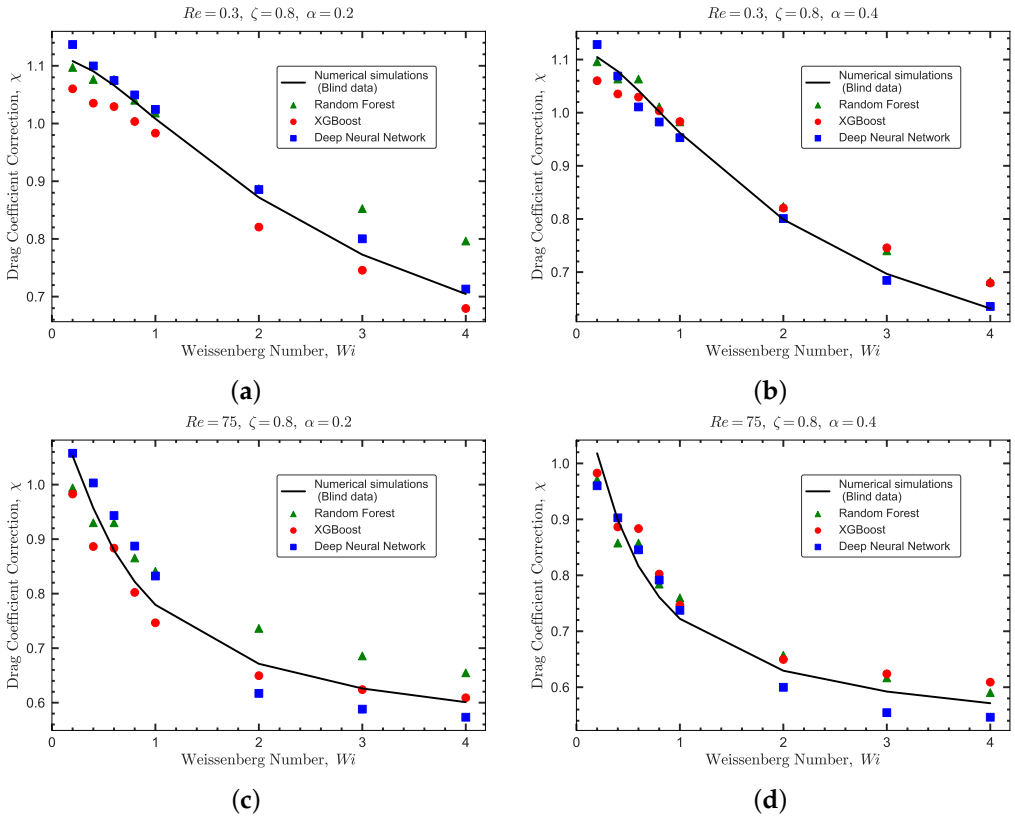


Figure 12. Validation of the ML models against blind datasets generated for Giesekus fluids. The comparisons are between the DNSs (solid lines showing the real values for the viscoelastic drag coefficient correction) and the predicted values by the ML models. The comparisons are shown at $\zeta = 0.8$ for different values of Re and α: (**a**) $Re = 0.3$ and $\alpha = 0.2$, (**b**) $Re = 0.3$ and $\alpha = 0.4$, (**c**) $Re = 75$ and $\alpha = 0.2$, and (**d**) $Re = 75$ and $\alpha = 0.4$. The predictions obtained with the Deep Neural Network, Random Forest and XGBoost models are represented by square, triangle and circle symbols, respectively.

4.5. Model Ensembling

In this section, we leverage stacking which is an ensemble learning technique to combine multiple ML-based regression models via a meta-regressor. The objective is to develop a meta-model with high accuracy when predicting the drag coefficient for a particle translating in viscoelastic fluids. In previous sections, we showed that different ML models perform better on different sections of the data when inferred against blind datasets. For example, the XGBoost performs better on the Oldroyd-B blind dataset (see Table 6), and the DNN model performs better on the Giesekus blind dataset (see Table 7). The hypothesis here is to leverage the superiority of all developed ML models (i.e., decision tree models to prevent overfitting and DNN model to learn complicated features in a sparse dataset) and increase the generality of the model in predicting unseen data.

A schematic architecture for the stack model is shown in Figure 13. We first use the ML models trained on *SMOTE-set* (with their best architectures found in the previous section) to provide the level-1 predictions. These predictions are then provided as input features to the second-level regressor, which is a DNN meta-regressor. The hyperparameters for DNN meta-regressor are also tuned again using the 10-Fold CV approach in conjunction with the RandomSearchCV algorithm, similar to other models. The stack model is trained, validated and tested on *SMOTE-set*. The optimized architecture obtained for the DNN meta-regressor is reported in Table 8. This meta-model developed using stacking generalizes better and provides more accurate predictions on unseen data when compared to the performance of the individual models. One example comparison is reported in Table 9. As reported, the R^2 value for the meta-model increased to 0.9472 from 0.9013, which was previously obtained for the DNN model, as the best model in Section 4.4 to predict the blind datasets.

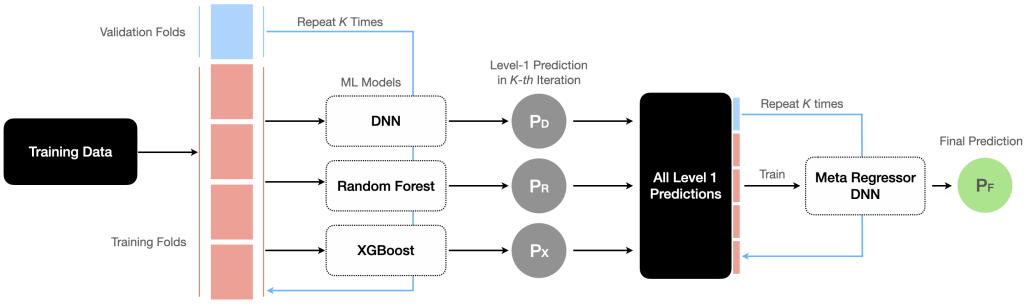


Figure 13. A schematic architecture for the meta-model to predict the viscoelastic drag coefficient using the stacking technique ensembling three optimized ML models and a meta-regressor.

Table 8. Optimized hyperparameters for the DNN meta-regressor trained, validated and tested on *SMOTE-set*.

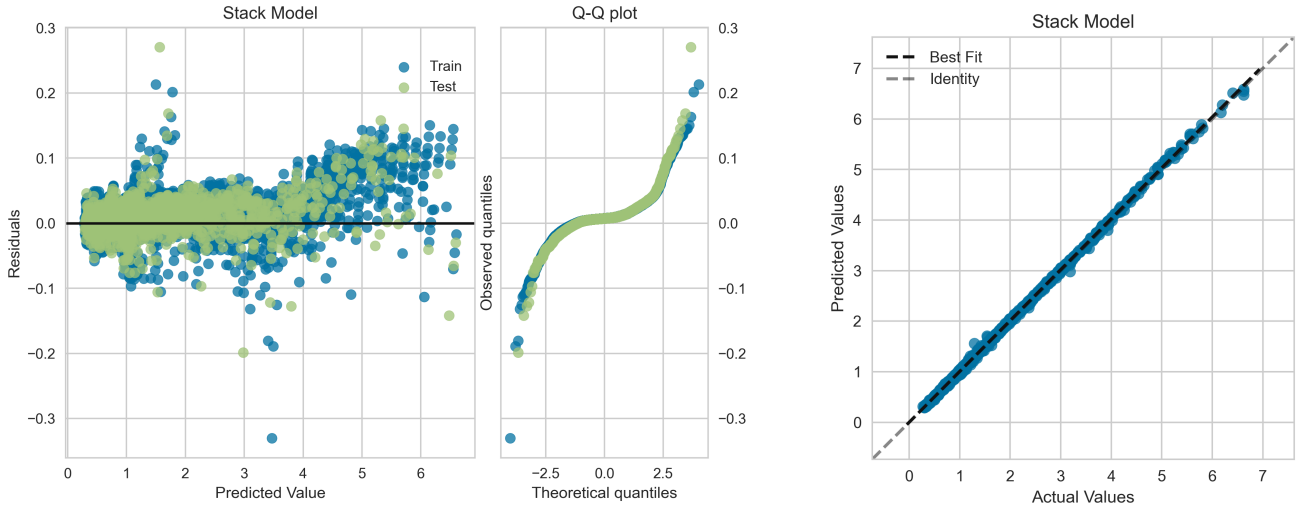
Model	Hyperparameter Value
DNN Meta-Regressor	activation = relu, alpha = 0.0001, hidden_layer_sizes = (20, 40, 10), learning_rate = adaptive, max_iter = 10,000, momentum = 0.95, n_iter_no_change = 15, solver = adam, tol = 0.0001

Table 9. Comparison of the statistical measures for the performance of the meta-model and DNN model against the *SMOTE-set* and blind datasets.

	Meta-Model		DNN Model	
	SMOTE-Set	**Blind Set**	**SMOTE-Set**	**Blind Set**
R^2	**0.9993**	**0.9472**	0.9945	0.9013
RMSE	**0.0178**	**0.0313**	0.0463	0.0428
MAPE	**0.0103**	**0.0209**	0.0216	0.0305

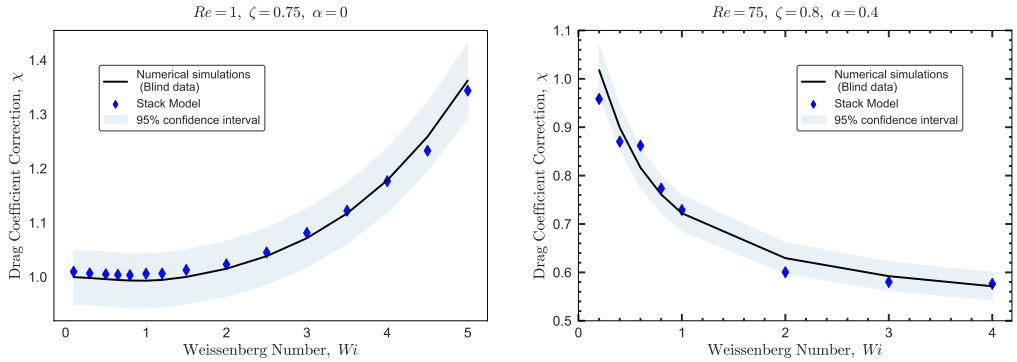
In Figure 14, we show the residuals and Q-Q plots for the meta-model trained, validated and tested on *SMOTE-set*. Figure 14a shows that the data points are mainly scattered around the horizontal axis and the calculated error is mainly distributed around zero. The Q-Q plot for meta-model depicts that most of the errors lies closer to $y = x$ line within the standard deviation range, and thus the data follow a distribution closer to normal distribution [47]. Figure 14b illustrates the comparison between the actual and predicted values of the viscoelastic drag coefficient correction using the meta-model. The very good agreement between the meta-model predictions and the actual values corroborates the high R^2 values presented in Table 9 and the fact that the residual plot for meta-model is relatively symmetric around $y = 0$ as shown in Figure 14a.

Figure 15 shows the performance of the meta-model against the blind datasets generated for Oldroyd-B and Giesekus viscoelastic fluids. The blind datasets are shown using solid lines for the flow of an Oldroyd-B fluid past a sphere at $Re = 1$, $\zeta = 0.75$ and $\alpha = 0$, and for the flow of a Giesekus fluid past a sphere at $Re = 75$, $\zeta = 0.8$ and $\alpha = 0.4$. The predictions obtained with the meta-model are represented by diamond symbols in Figure 15a,b. A 95% confidence interval region for the meta-model predictions is also illustrated. These comparisons and the statistical measures reported in Table 9 collectively show that the meta-model consistently outperforms the individual decision tree ML models as well as the DNN model on all unseen datasets.



(**a**) (**b**)

Figure 14. (**a**) Residuals and quantile-quantile (Q-Q) plots and (**b**) prediction error plot obtained for the meta-model shown in Figure 13 and trained, validated and tested on *SMOTE-set*. The statistical measures are reported in Table 9.



(**a**) (**b**)

Figure 15. Performance of the meta-model (stack-model) against the blind datasets generated for Oldroyd-B and Giesekus fluids. The blind datasets are shown using solid lines for the flow of an Oldroyd-B fluid past a sphere at (**a**) $Re = 1$, $\zeta = 0.75$ and $\alpha = 0$; and for the flow of a Giesekus fluid past a sphere at (**b**) $Re = 75$, $\zeta = 0.8$ and $\alpha = 0.4$. The predictions obtained with the meta-model are represented by diamond symbols. A 95% confidence interval region for the predictions is also shown.

This meta-model alongside the training datasets (*OB-set* and *GI-set*) are packaged and published with this paper, as supplementary materials. The viscoelastic fluid dynamics community can leverage this meta-model in their simulations and/or leverage the data to train new data-driven models.

5. Conclusions

This study presents a framework to predict the drag coefficient of a spherical particle translating in viscoelastic fluids. To this end, continuum simulations and Machine Learning (ML) models were employed to generate a data-driven meta-model. We first generated two datasets using direct numerical simulations; the *OB-set* (the dataset for the Oldroyd-B fluid) and the *GI-set* (the dataset for the Giesekus fluid) that include a total of 12,120 and 4950 data points, respectively. The kinematic input features were selected to be Reynolds number, $0 < Re \leq 50$, Weissenberg number, $0 \leq Wi \leq 10$, polymeric retardation ratio,

$0 < \zeta < 1$, and shear thinning mobility parameter, $0 < \alpha < 1$. Three ML regression models, Random Forest (RF), Deep Neural Network (DNN) and Extreme Gradient Boosting (XGBoost), were employed to predict the drag coefficient enhancement or reduction due to the fluids' elasticity and shear thinning effects. The ML models were all trained, validated, and tested on the *OB-set* and *SMOTE-set* (a balanced dataset combining the *OB-set* and *GI-set*), and their best architecture (i.e., tuned hyperparameters) were obtained using a 10-Fold cross-validation method. All the ML models presented remarkable accuracy when trained and inferred on these datasets; however the XGBoost model resulted in the highest R^2 and lowest RMSE and MAPE measures.

The trained ML models were also tested against a blind dataset where the input features coverage was outside the scope of the training set or interpolated within the training sets. A total of 124 data points were generated using DNSs for both Oldroyd-B and Giesekus fluids. The predictions obtained with the DNN model achieved the highest R^2 and lowest RMSE and MAPE measures when inferred on the blind test dataset. To leverage the power of all models (decision tree models to prevent overfitting and DNN model to learn complicated features), we developed a meta-model using stacking technique. The meta-model ensembles RF, XGBoost, and DNN models and outputs a prediction based on the individual learner's predictions and a DNN meta-regressor. The meta-learner model consistently outperformed the individual decision tree and DNN models on all datasets.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/polym14030430/s1.

Author Contributions: Conceptualization, S.A.F. and C.F.; Formal analysis, S.A.F. and C.F.; Investigation, S.A.F., A.I.R. and C.F.; Methodology, S.A.F. and C.F.; Resources, S.A.F. and C.F.; Software, S.A.F. and C.F.; Supervision, S.A.F. and C.F.; Validation, S.A.F., A.I.R. and C.F.; Writing—original draft, S.A.F., A.I.R. and C.F.; Writing—review & editing, S.A.F. and C.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by FEDER funds through the COMPETE 2020 Programme and National Funds through FCT (Portuguese Foundation for Science and Technology) under the projects UID-B/05256/2020, UID-P/05256/2020 and MIT-EXPL/TDI/0038/2019—APROVA—Deep learning for particle-laden viscoelastic flow modelling (POCI-01-0145-FEDER-016665) under MIT Portugal program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to acknowledge the University of Minho cluster under the project NORTE-07-0162-FEDER-000086 (URL: http://search6.di.uminho.pt), the Minho Advanced Computing Center (MACC) (URL: https://macc.fccn.pt) under the project CPCA_A2_6052_2020, the Consorzio Interuniversitario dell'Italia Nord Est per il Calcolo Automatico (CINECA) under the Project HPC-EUROPA3 (INFRAIA-2016-1-730897) with the support of the EC Research Innovation Action under the H2020 Programme, and PRACE—Partnership for Advanced Computing in Europe under the project icei-prace-2020-0009, for providing HPC resources that have contributed to the research results reported within this paper. The authors thank Professor Gareth Huw McKinley from the Hatsopoulos Microfluids Laboratory, Department of Mechanical Engineering at the Massachusetts Institute of Technology for insightful comments regarding this work.

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

References

1. Chhabra, R.P. *Bubbles, Drops, and Particles in Non-Newtonian Fluids*; CRC Press: Boca Raton, FL, USA, 2007; ISBN 978-0-4290-7480-6. [CrossRef]
2. Deshpande, A.P.; Krishnan, J.M.; Kumar, S. *Rheology of Complex Fluids*; Springer: New York, NY, USA, 2010; ISBN 978-1-4419-6493-9. [CrossRef]

3. Barbati, A.C.; Desroches, J.; Robisson, A.; McKinley, G.H. Complex fluids and hydraulic fracturing. *Annu. Rev. Chem. Biomol. Eng.* **2016**, *7*, 415–453. [CrossRef]
4. Fernandes, C.; Faroughi, S.A.; Carneiro, O.S.; Nóbrega, J.M.; McKinley, G.H. Fully-resolved simulations of particle-laden viscoelastic fluids using an immersed boundary method. *J. Non-Newton. Fluid Mech.* **2019**, *266*, 80–94. [CrossRef]
5. Faroughi, S.; Huber, C. A generalized equation for rheology of emulsions and suspensions of deformable particles subjected to simple shear at low Reynolds number. *Rheol. Acta* **2014**, *54*, 85–108. [CrossRef]
6. Faroughi, S.A.; Fernandes, C.; Nóbrega, J.M.; McKinley, G.H. A closure model for the drag coefficient of a sphere translating in a viscoelastic fluid. *J. Non-Newton. Fluid Mech.* **2020**, *277*, 104218. [CrossRef]
7. Fernandes, C.; Faroughi, S.A.; Ribeiro, R.; Roriz, A.I.; McKinley, G.H. Finite volume simulations of particle-laden viscoelastic fluid flows: Application to hydraulic fracture processes. *Eng. Comput.* **2021**. [CrossRef]
8. Faroughi, S.A.; Huber, C. Crowding-based rheological model for suspensions of rigid bimodal-sized particles with interfering size ratios. *Phys. Rev. E* **2014**, *90*, 052303. [CrossRef]
9. Maxey, M. Simulation methods for particulate flows and concentrated suspensions. *Annu. Rev. Fluid Mech.* **2017**, *49*, 171–193. [CrossRef]
10. Faroughi, S.A.; Huber, C. A self-similar behavior for the relative viscosity of concentrated suspensions of rigid spheroids. *Rheol. Acta* **2017**, *56*, 35–49. [CrossRef]
11. Faroughi, S.A.; Pruvot, A.J.-C.J.; McAndrew, J. The rheological behavior of energized fluids and foams with application to hydraulic fracturing. *J. Pet. Sci. Eng.* **2018**, *163*, 243–263. [CrossRef]
12. Kutz, J.N. Deep learning in fluid dynamics. *J. Fluid Mech.* **2017**, *814*, 1–4. [CrossRef]
13. Brunton, S.L.; Noack, B.R.; Koumoutsakos, P. Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.* **2020**, *52*, 477–508. [CrossRef]
14. Molinaro, R.; Singh, J.-S.; Catsoulis, S.; Narayanan, C.; Lakehal, D. Embedding data analytics and CFD into the digital twin concept. *Comput. Fluids* **2021**, *214*, 104759. [CrossRef]
15. Fernandes, C.; Semyonov, D.; Ferrás, L.L.; Nóbrega, J.M. Validation of the CFD-DPM solver DPMFoam in OpenFOAM through analytical, numerical and experimental comparisons. *Granul. Matter* **2018**, *20*, 64. [CrossRef]
16. Kelbaliyev, G.I. Drag coefficients of variously shaped solid particles, drops, and bubbles. *Theor. Found. Chem. Eng.* **2011**, *45*, 248–266. [CrossRef]
17. Faroughi, S.A.; Huber, C. Unifying the relative hindered velocity in suspensions and emulsions of nondeformable particles. *Geophys. Res. Lett.* **2015**, *42*, 53–59. [CrossRef]
18. Gheissary, G.; van den Brule, B.H.A.A. Unexpected phenomena observed in particle settling in non-Newtonian media. *J. Non-Newton. Fluid Mech.* **1996**, *67*, 1–18. [CrossRef]
19. Chilcott, M.D.; Rallison, J.M. Creeping flow of dilute polymer solutions past cylinders and spheres. *J. Non-Newton. Fluid Mech.* **1988**, *29*, 381–432. [CrossRef]
20. McKinley, G.H. Steady and transient motion of spherical particles in viscoelastic liquids. In *Transport Processes in Bubble, Drops, and Particles*; Chhabra, R., Kee, D.D., Eds.; Taylor & Francis: New York, NY, USA, 2002; pp. 338–375. ISBN 978-1-5603-2906-0.
21. Giesekus, H. A simple constitutive equation for polymer fluids based on the concept of deformation-dependent tensorial mobility. *J. Non-Newton. Fluid Mech.* **1982**, *11*, 69–109. [CrossRef]
22. Habla, F.; Tan, M.W.; Hablberger, J.; Hinrichsen, O. Numerical simulation of the viscoelastic flow in a three-dimensional lid-driven cavity using the log-conformation reformulation in OpenFOAM. *J. Non-Newton. Fluid Mech.* **2014**, *212*, 47–62. [CrossRef]
23. Pimenta, F.; Alves, M.A. Stabilization of an open-source finite volume solver for viscoelastic fluid flows. *J. Non-Newton. Fluid Mech.* **2017**, *239*, 85–104. [CrossRef]
24. OpenFOAM. The Open Source CFD Toolbox. 2004. Available online: https://www.openfoam.com/ (accessed on 5 December 2021).
25. Fernandes, C.; Araujo, M.S.B.; Ferrás, L.L.; Nóbrega, J.M. Improved both sides diffusion (iBSD): A new and straightforward stabilization approach for viscoelastic fluid flows. *J. Non-Newton. Fluid Mech.* **2017**, *249*, 63–78. [CrossRef]
26. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
27. Liaw, A.; Wiener, M. Classification and regression by randomForest. *R News* **2002**, *2*, 18–22.
28. Smith, P.F.; Ganesh, S.; Liu, P. A comparison of random forest regression and multiple linear regression for prediction in neuroscience. *J. Neurosci. Methods* **2013**, *220*, 85–91. [CrossRef]
29. Chen, T.; Guestrin, C. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794. [CrossRef]
30. Luckner, M.; Topolski, B.; Mazurek, M. Application of XGBoost algorithm in fingerprinting localisation task. In *Computer Information Systems and Industrial Management*; Saeed, K., Homenda, W., Chaki, R., Eds.; Springer: Cham, Switzerland, 2017; pp. 661–671. ISBN 978-3-319-59105-6. [CrossRef]
31. Brownlee, J. *XGBoost with Python: Gradient Boosted Trees with XGBoost and Scikit-Learn*; Machine Learning Mastery. 2016. Available online: https://books.google.pt/books?id=HgmqDwAAQBAJ&printsec=copyright&redir_esc=y#v=onepage&q&f=false (accessed on 5 January 2022).
32. Specht, D.F. A general regression neural network. *IEEE Trans. Neural Netw.* **1991**, *2*, 568–576. [CrossRef] [PubMed]

33. Mahmudul, M.; Mia, A.; Biswas, S.K.; Urmi, M.C.; Siddique, A. An algorithm for training multilayer perceptron (MLP) for image reconstruction using neural network without overfitting. *Int. J. Sci. Technol. Res.* **2015**, *4*, 271–275.
34. Jiang, H.; Zou, Y.; Zhang, S.; Tang, J.; Wang, Y. Short-term speed prediction using remote microwave sensor data: Machine learning versus statistical model. *Math. Probl. Eng.* **2016**. [CrossRef]
35. Palyam, R.K. Deep feature interpolation for image content changes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 22–25 July 2017; pp. 6090–6099. [CrossRef]
36. Trillos, N.G.; Morales, F.; Morales, J. Traditional and accelerated gradient descent for neural architecture search. In *Geometric Science of Information*; Nielsen, F., Barbaresco, F., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 507–514. ISBN 978-3-030-80209-7.
37. Asiltürk, I.; Çunkaş, M. Modeling and prediction of surface roughness in turning operations using artificial neural network and multiple regression method. *Expert Syst. Appl.* **2011**, *38*, 5826–5832. [CrossRef]
38. James, D.F. Boger fluids. *Annu. Rev. Fluid Mech.* **2009**, *41*, 129–142. [CrossRef]
39. Oldroyd, J.G. On the formulation of rheological equations of state. *Proc. R. Soc. Lond. Ser. Math. Phys. Eng. Sci.* **1950**, *200*, 523–541. [CrossRef]
40. Joseph, D.D. *Fluid Dynamics of Viscoelastic Liquids*; Springer: Berlin/Heidelberg, Germany, 1990; ISBN 978-1-4612-8785-8.
41. Cherizol, R.; Sain, M.; Tjong, J. Review of non-Newtonian mathematical models for rheological characteristics of viscoelastic composites. *Green Sustain. Chem.* **2015**, *5*, 6–14. [CrossRef]
42. Anguita, D.; Ghelardoni, L.; Ghio, A.; Oneto, L.; Ridella, S. The 'K' in K-fold cross validation. In Proceedings of the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), Bruges, Belgium, 25–27 April 2012; pp. 441–446.
43. Jung, Y.; Hu, J. A K-fold averaging cross-validation procedure. *J. Nonparametr. Stat.* **2015**, *27*, 167–179. [CrossRef] [PubMed]
44. Memon, N.; Patel, S.B.; Patel, D.P. Comparative analysis of artificial neural network and XGBoost algorithm for PolSAR image classification. In *International Conference on Pattern Recognition and Machine Intelligence*; Springer International Publishing: Cham, Switzerland, 2019; pp. 452–460. ISBN 978-3-030-34868-7.
45. Coleman, C.D.; Swanson, D.A. On MAPE-R as a measure of cross-sectional estimation and forecast accuracy. *J. Econ. Soc. Meas.* **2007**, *32*, 219–233. [CrossRef]
46. Zeraatpisheh, M.; Ayoubi, S.; Jafari, A.; Tajik, S.; Finke, P. Digital mapping of soil properties using multiple machine learning in a semi-arid region, central Iran. *Geoderma* **2019**, *338*, 445–452. [CrossRef]
47. Guha, P.; Chakraborty, B. On a multivariate generalization of quantile-quantile plot. In *International Conference on Robust Statistics*; 2009; p. 62, ISBN 978-88-903330-0-2. Available online: https://www.researchgate.net/profile/Tadeusz-Bednarski/publication/220363666_Frechet_Differentiability_in_Statistical_Inference_for_Time_Series/links/552502a80cf2caf11bfcf362/Frechet-Differentiability-in-Statistical-Inference-for-Time-Series.pdf#page=84 (accessed on 5 January 2022).
48. Yang, Q.; Zhang, Y.; Dai, W.; Pan, S.J. *Transfer Learning*; Cambridge University Press: Cambridge, UK, 2020; ISBN 978-1107016903.
49. Thuerey, N.; Holl, P.; Mueller, M.; Schnell, P.; Trost, F.; Um, K. Physics-Based Deep Learning. 2021. Available online: https://physicsbaseddeeplearning.org (accessed on 5 December 2021).
50. Segev, N.; Harel, M.; Mannor, S.; Crammer, K.; El-Yaniv, R. Learn on source, refine on target: A model transfer learning framework with random forests. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1811–1824. [CrossRef]
51. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
52. Fauzan, M.A.; Murfi, H. The accuracy of XGBoost for insurance claim prediction. *Int. J. Adv. Soft Comput. Its Appl.* **2018**, *10*, 159–171.
53. Giannakas, F.; Troussas, C.; Krouska, A.; Sgouropoulou, C.; Voyiatzis, I. XGBoost and deep neural network comparison: The case of teams' performance. In *International Conference on Intelligent Tutoring Systems*; Springer: Cham, Switzerland, 2021; pp. 343–349. ISBN 978-3-030-80420-6. [CrossRef]
54. Shehadeh, A.; Alshboul, O.; Mamlook, R.E.A.; Hamedat, O. Machine learning models for predicting the residual value of heavy construction equipment: An evaluation of modified decision tree, LightGBM, and XGBoost regression. *Autom. Constr.* **2021**, *129*, 103827. [CrossRef]
55. Ahmad, M.; Hippolyte, J.-L.; Mourshed, M.; Rezgui, Y. Random forests and artificial neural network for predicting daylight illuminance and energy consumption. In Proceedings of the 15th International Building Performance Simulation Association Conference, San Francisco, CA, USA, 7–9 August 2017; pp. 1949–1955. [CrossRef]
56. Kim, J.Y.; Cho, B.H.; Im, S.M.; Jeon, M.J.; Kim, I.Y.; Kim, S.I. Comparative study on artificial neural network with multiple regressions for continuous estimation of blood pressure. In *Conference Proceedings IEEE Engineering in Medicine and Biology Society*; IEEE: Piscataway, NJ, USA, 2005; pp. 6942–6945. [CrossRef]
57. Willard, J.; Jia, X.; Xu, S.; Steinbach, M.; Kumar, V. Integrating physics-based modeling with machine learning: A survey. *arXiv* **2020**, arXiv:2003.04919.
58. Lu, L.; Pestourie, R.; Yao, W.; Wang, Z.; Verdugo, F.; Johnson, S.G. Physics-informed neural networks with hard constraints for inverse design. *Soc. Ind. Appl. Math. J. Sci. Comput.* **2021**, *43*, B1105–B1132. [CrossRef]
59. Sill, J.; Takács, G.; Mackey, L.; Lin, D. Feature-weighted linear stacking. *arXiv* **2009**, arXiv:0911.0460.