ADVANCED FUNCTIONAL MATERIALS
www.afm-journal.de

Unleashing the Power of Machine Learning in Nanomedicine Formulation Development

Thomas L. Moore, Cristiano Pesce, Antonietta Greco, Claudia Pisante, Greta Avancini, Valentina Di Francesco, Yosi Shamay, and Paolo Decuzzi*

Artificial intelligence (AI) is being integrated into nearly every aspect of modern life, and machine learning (ML)–a subfield of AI–has the potential to accelerate the development of nanomedicines. Here, a machine learning workflow is presented to optimize the microfluidic-based formulation of nanomedicines. A database of ≈200 unique nanomedicine formulations with over 550 total measurements is curated by producing liposomes, lipid nanoparticles, and poly(lactic-*co*-glycolic acid) nanoparticles, either empty or loaded with the model therapeutic agent, curcumin. Nanoparticle production parameters are systematically varied, and the resulting particles are characterized for their diameter, polydispersity index, and encapsulation efficiency. These data are used to train and validate 13 different ML models using open-source libraries, with the task of returning the most accurate prediction of nanomedicine attributes. The most accurate models, based on random forest regression, are implemented to yield particles with user-specified attributes. Finally, the proposed ML workflow, MicrofluidicML, is compared against generative large language models–OpenAI ChatGPT, Google's Gemini, and DeepSeek. MicrofluidicML provides a workflow where the researcher has complete governance and control of the input data with a relatively low computational overhead, and represents a step toward implementing a computationally lightweight ML framework to accelerate nanomedicine development.

1. Introduction

Recent advances in artificial intelligence (AI) have sparked excitement across many scientific disciplines due to their potential to revolutionize medicine and laboratory sciences. Machine learning (ML), a subset of AI where computers are capable of adapting without explicit programming based on algorithms and statistical models, is particularly suitable for nanoparticle production and formulation development.[1–6] Indeed, in preliminary examples, ML has been used to engineer multifunctional peptides to boost the melanin binding rates in ocular delivery,[7] optimize the uptake of poly(lactic-co-glycolic acid)-polyethylene glycol (PLGA-PEG) nanoparticles in breast cancer cells,[8] and predict the transfection efficiency of lipid nanoparticles deploying nucleic acids.[9] However, building effective and generalizable ML models relies on curating a substantial amount of high-quality data.

In this context, there is an ever-growing interest in utilizing microfluidics as a scalable and reproducible approach to fabricate nanomedicines.[10,11] Independently of the various microfluidic-based mixing techniques available–hydrodynamic flow focusing, mixing via a staggered herringbone micromixer, bifurcating mixers, baffle mixers, T-junction mixing–the production of nanoparticles relies on the precise control of a few crucial engineering parameters (input features), such as the total flow rate

T. L. Moore
Department of Pharmacy
Università degli Studi di Napoli Federico II
Via Domenico Montesano, 49, Naples 80131, Italy
E-mail: thomaslee.moore@unina.it

T. L. Moore, C. Pesce, A. Greco, C. Pisante, G. Avancini, V. Di Francesco, P. Decuzzi
Laboratory of Nanotechnology for Precision Medicine
Istituto Italiano di Tecnologia
Via Morego, 30, Genoa 16163, Italy

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/adfm.202514387

DOI: 10.1002/adfm.202514387

C. Pesce
Department of Pharmaceutical and Pharmacological Sciences
Università degli Studi di Padova
Via Francesco Marzolo, 5, Padua 35131, Italy

Y. Shamay
Faculty of Biomedical Engineering
Technion–Israel Institute of Technology
Haifa, Israel

P. Decuzzi
Division of Oncology
Department of Medicine and Department of Pathology
Stanford University School of Medicine
269 Campus Drive, Stanford, CA 94305, USA

(TFR), the flow rate ratio (FRR) between aqueous and organic phases, and reagents' concentrations–previously documented by the group of Yvonne Perrie using design-of-experiments (DoE) and multivariate data analysis in the context of liposome production.[12]

However, at the outset of developing a new nanomedicine formulation, identifying the optimal input features to achieve a nanoparticle with the desired hydrodynamic diameter (d_H), polydispersity index (PdI), and drug encapsulation efficiency (EE) is still largely a game of expertise (prior know-how), literature research (replicating prior published works), and hit-or-miss experimentation (systematically testing different values of the input features). That stated, the ability to produce nanomedicines with specified physico-chemical properties remains a priority, as these properties are understood to mediate biological interactions.[13–17] There is no direct regulatory guidance for specific attributes of nanoparticles for clinical use,[18,19] however, it is clear that nanoparticles must be produced with consistent properties to exert a consistent and predictable biological effect.

Advanced statistical models or ML-developed models have significant potential to streamline this process and generate new revolutionary formulations based on high-quality data. Nonetheless, the implementation of these advanced techniques can be challenging for the broader community that is unaccustomed to such computational tools. Here, we aim to present a workflow that can, in part, provide a roadmap toward the facile integration of ML tools for the microfluidic development of nanomedicines.

2. Results and Discussion

2.1. The MicrofluidicML Workflow

The MicrofluidicML workflow, at its core, consists of a database of nanomedicine formulations comprised of lipid nanoparticles (LNP), liposomes, and PLGA nanoparticles. The experimental data are aggregated by particle type, identifying a subset for lipid-based nanoparticles, including LNP and liposomes, and a subset for polymer-based (PLGA) nanoparticles. The rationale for combining LNP and liposomes is based on the fact that they are both made out of lipids, and their combination further tests the generalizability of the program. Furthermore, PLGA nanoparticles were trained as a separate class because, as a material, polymers are fundamentally different from lipids. All nanoparticle types were fabricated using a Precision NanoSystems Benchtop NanoAssemblr instrument, equipped with a microfluidic micromixer cartridge characterized by a "Y"-like channel with staggered herringbone geometric features in the channel to facilitate fluid mixing (**Figure** 1A).

Figure 1B details how the experimental data are employed to build predictive ML models. Models were trained on subsets of broad particle classes (lipid vs polymer based) rather than specific particle types. First, the data were randomly divided into training data and testing data, corresponding to 85% and 15% of all the experimental data, respectively. The training data were employed for training the model, and the testing data were used to validate the final output of the model directly on 'unseen' data. Data were split so that the mean and standard deviation of the target attribute were approximately equal between training and testing data (Figure S1, Supporting Information).

Then, out of a library of available models, several different ML models were selected and their hyperparameters were tuned via 10-fold cross-validation combined with a random 10-point search of model hyperparameters. From these results, the most performant model was selected for further tuning, and this "optimally fit" model is then evaluated against the testing data, corresponding to 15% of the experimental data that were not seen by the ML model during the training process.

2.2. Microfluidic-Based Production of Nanomedicines

Almost 200 unique formulations of nanoparticles, either empty or loaded with an active pharmaceutical ingredient (API), were realized using a continuous-flow microfluidic-based mixing apparatus (Figure 1A). Specifically, during the production of the nanomedicines, two liquid phases, one aqueous and one organic, are pushed through microchannels with cross-sectional dimensions in the sub-millimetric range. Based on the unique geometry of the microchannels, the two phases are mixed, enabling the self-assembly of nanoparticles. To produce lipid nanoparticles (LNP), a mixture of lipids was dissolved in ethanol (the organic phase), which was then mixed with an aqueous buffer at a lower pH to enable the protonation of ionizable lipids (and generally their subsequent complexation of a nucleic acid payload). Liposomes were similarly produced; however, the species of lipids, their ratios, and the aqueous buffer differed. To produce PLGA nanoparticles, the polymer was dissolved in an organic solvent (i.e. acetonitrile) and then mixed with water. Hydrophobic-hydrophobic polymer interactions then induce the self-assembly of the nanoparticles. Incorporation of a hydrophobic drug in the organic phase would enable its entrapment within the polymeric matrix. The microfluidic-based apparatus offers a flexible and highly controllable approach to produce several different types of nanoparticles in a relatively short period of time.

In this work, nanoparticles were classified in three different categories: empty lipid nanoparticles (LNP); liposomes, either empty or entrapping the hydrophobic molecule curcumin (CURC) within the lipid bilayer; and PLGA nanoparticles, either empty or loaded with CURC. Among the mixing parameters, the aqueous: organic flow rate ratio (FRR) and the total flow rate (TFR) were systematically changed, while among the composition parameters, the total molar concentration for the lipids or polymers (materials) as well as the ratio between the API and nanoparticle mass were systematically modified. The actual values used for the four input features are listed in **Table** 1, while Figure S2 (Supporting Information) shows the so-called 'experimental box' in a 3D space, where each dot corresponds to a unique particle formulation. For simplicity, only the cases of empty LNP, liposomes, and PLGA nanoparticles are displayed.

Figure 2A provides a general schematic of the three particle types together with the percent content in lipids and polymers. Note that while the total mass of lipids was systematically changed for the LNP and liposomes, the relative ratios among the different lipid components were fixed. Scatter plots in Figure 2B show the effect of the mixing and composition features–TFR,

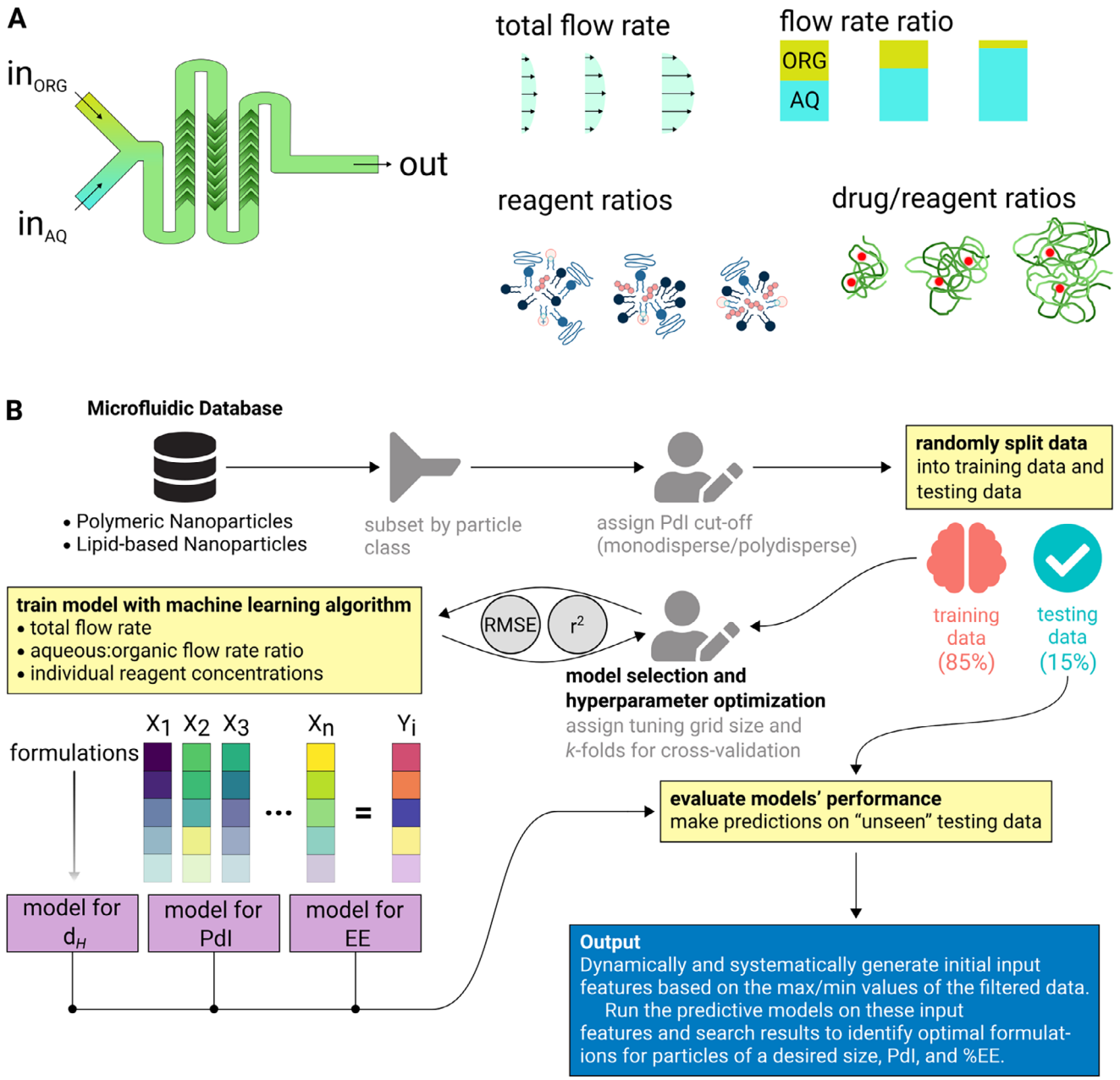

Figure 1. Schematic of microfluidic-base nanoparticle synthesis and the machine learning workflow (MicrofluidicML). A) A "Y"-shaped microfluidic channel with a staggered herringbone structure was employed for mixing organic (ethanol) and aqueous phases. Materials and flow parameters (input features) included the total flow rate (TFF), aqueous: organic flow rate ratio (FRR), and concentration ratios for the reagents. B) Experimental data were subset into relevant measurements based on particle class, and defined as either monodisperse (polydispersity index, PdI $\leq$ 0.250) or polydisperse (PdI > 0.250). These data were randomly split into training data (85%) and testing data (15%). Models were trained to predict particle hydrodynamic diameter (d_H), PdI, and encapsulation efficiency (EE) of some API on dynamically generated input features, based on the composition and microfluidic parameters of the subset database. Model hyperparameters were optimized via a grid search over a range of values and further evaluated via stratified *k*-fold cross-validation. Model accuracy was evaluated on "unseen" testing data. Optimized models were used to dynamically generate a table of the best formulations to yield particles of a desired size, below a PdI threshold, and with an optimal EE.

FRR, total molar concentration–on the hydrodynamic particle diameter, d_H. For example, LNP produced at a TFR of 2 ml/min, a FRR equal to 3:1, and at a total lipid concentration of 20 mM returned a hydrodynamic diameter of 85 ± 8 nm with a PdI = 0.201 ± 0.056. By comparison, empty liposomes produced at a TFR of 1 ml/min, a FRR of 3:1, and a total lipid concentration of 24 mM had a diameter of 103 ± 2 nm with a PdI = 0.128 ± 0.004, while PLGA particles produced with a TFR of 3 ml/min, a FRR of 3:1, and 0.2174 mM were 97 ± 0.4 nm in diameter with a PdI = 0.142 ± 0.015. Note that a threshold of 0.250 was implemented for the PdI, above which particles were deemed "polydisperse." This was necessary as the training of the ML models

Table 1. Input Features for each nanoparticle type. Almost 200 unique formulations of nanoparticles, either empty or loaded with an active pharmaceutical ingredient (curcumin), were realized using a continuous-flow microfluidic-based mixing apparatus.

	LNP	Liposomes	PLGA nanoparticles
TFR (ml/min)	2, 4, 8	1, 4, 8	3, 6, 13
FRR (aq:org)	1, 3, 5	3, 6, 9	3, 4, 5
Total reagent concentration (mM)	20, 40, 60	12, 24, 48	0.109, 0.217, 0.435
Aqueous phase	50 mM citrate buffer, pH 3.5	1× PBS, pH 7.4	10 mM Tris buffer, pH 7.2
Organic phase	Ethanol	Ethanol	Acetonitrile
Total no. measurements	81	162	324
No. unique formulations	27	54	108

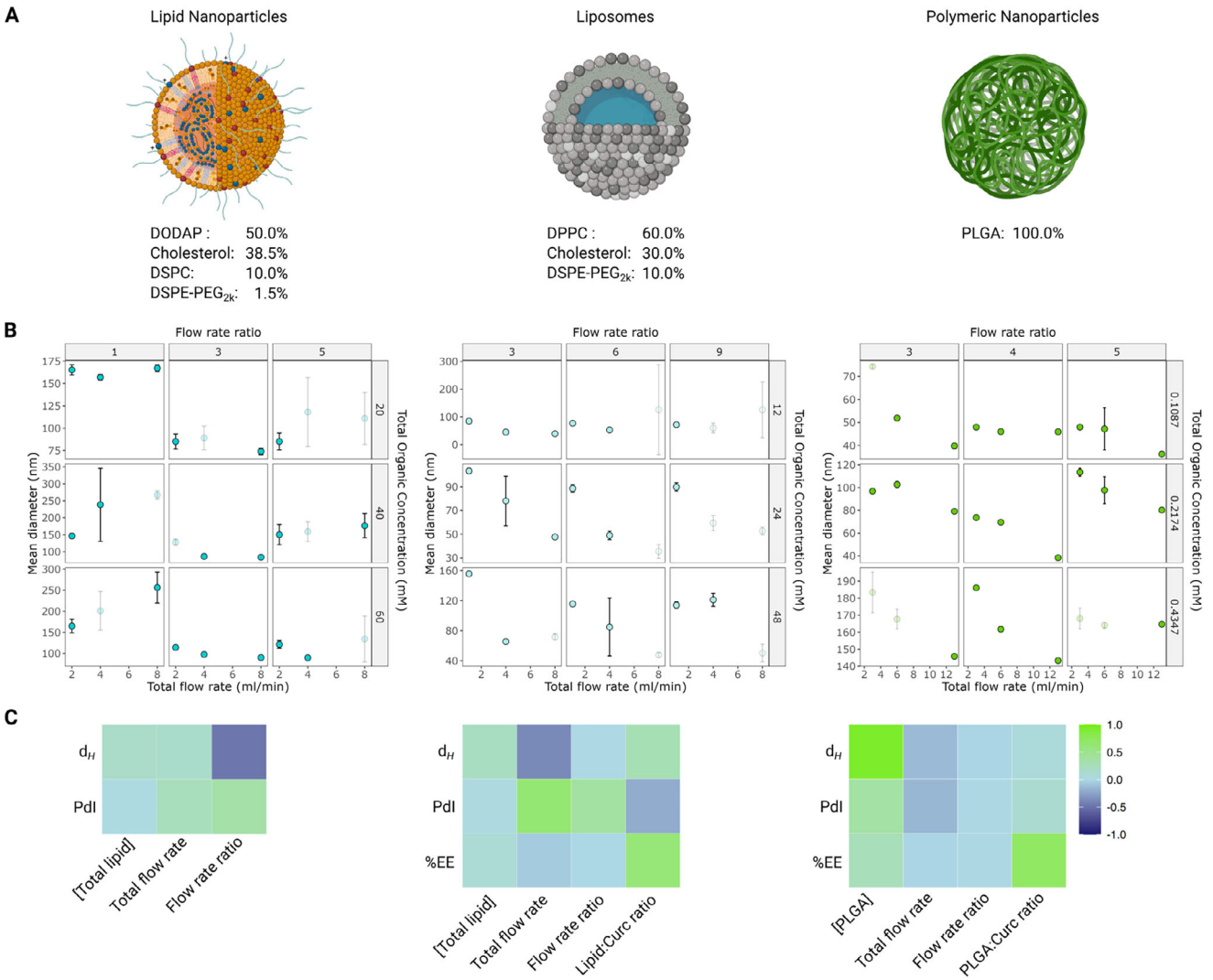

Figure 2. Selected summary of nanoparticle attributes obtained via microfluidic production. A) Illustrations showing the structure of the different types of nanoparticles: lipid nanoparticles, liposomes, and PLGA nanoparticles, as well as their molar composition (Created in BioRender. Pesce, C. (2025) https://BioRender.com/g35k993). B) Scatter plots showing the relationship between total flow rate (TFR) and the average hydrodynamic diameter (d_H) as measured by dynamic light scattering. Translucent points are formulations that were classified as polydisperse (i.e., PdI > 0.250). C) Pearson correlation plots showing how each formulation production feature (i.e., total reagent concentration, total flow rate, or flow rate ratio) is linearly correlated to the d_H, the PdI, or curcumin encapsulation efficiency (EE) for lipid nanoparticles, liposomes, or PLGA nanoparticles. Strong positive correlation (i.e., closer to 1.0) indicates a direct relationship between the feature and the target, while a strong negative correlation (i.e., closer to −1.0) indicates an indirect relationship.

to predict payload loading and d_H was restricted solely to the so-called 'monodisperse' particle configurations (i.e., PdI $\leq$ 0.250). For models to predict PdI, the entire training data set (including monodisperse and polydisperse formulations) was included. In Figure 2B, translucent dots indicate polydisperse samples.

Perhaps more interesting are the Pearson's correlation plots in Figure 2C, where a color map indicates the type of correlation between the input features and the output parameters. In estimating the Pearson's coefficient, a linear relationship is assumed to exist between two groups. When considering LNP, the flow rate ratio (FRR) has a significant impact on the hydrodynamic diameter d_H, as an increase in FRR leads to a decrease in d_H (purple box Figure 2C, left–inverse correlation). For all the other input features, only modest correlations were observed with a Pearson's coefficient close to 0 (light green and cyan boxes). Importantly, these are empty LNP, and results may change in the presence of an encapsulated biologically active agent (i.e., siRNA or mRNA).

For the liposomes, the Pearson's plot is more complex. The flow rate ratio has only a modest effect on the output parameters, which in this case also includes the loading of curcumin in the lipid bilayer. On the other hand, the total flow rate (TFR) tends to dominate: an increase in TFR is associated with a reduction in d_H (purple box Figure 2C, center–inverse correlation) and a significant increase in PdI (green box Figure 2C, center–direct correlation), and a modest reduction in EE. Also, the lipid:CURC ratio significantly affects the output parameters: an increase in lipid:CURC ratio leads to a decrease in PdI (purple box Figure 2C, center–inverse correlation) and an increase in EE (green box Figure 2C, center–direct correlation). The behavior observed for the PLGA nanoparticles is quite straightforward, as an increase in PLGA content positively correlates with the size of the nanoparticles d_H (green box Figure 2C, right–direct correlation), whereas an increase in the PLGA:CURC ratio increases the curcumin encapsulation efficiency (green box Figure 2C, right–direct correlation).

While these correlation maps give indications as to the relationship between input features and output parameters and could be extremely helpful in selecting the most effective input features to optimize the nanoparticle formulations, more quantitative results are desired. Thus, we turned toward open-source libraries in the R language to build machine learning models for predicting the hydrodynamic particle diameter, PdI, and encapsulation efficiency as a function of the four input features.

2.3. Machine Learning Model Selection

Thirteen different, freely available ML algorithms were evaluated to find the most accurate models to independently predict d_H, PdI, and EE (i.e. unique models were trained to predict nanoparticle attributes for each class, for 6 unique models in total). These were the linear regression, bagged decision trees (bagged trees), bagged multilayer perceptron neural network (bagged MLP), bagged multivariate adaptive regression splines (bagged MARS), xgboosted decision trees (xgboost), k-nearest neighbors, least absolute shrinkage and selection operator (LASSO), ridge regression (RIDGE), Poisson regression, support vector machines (SVM) with linear, radial, and polynomial kernel functions, and random forest. For all models, before training or validating testing data, input features were pre-processed by dropping any zero-variance features and max-min scaling (i.e., normalizing values between 0 and 1). Moreover, data were reshaped so that each reagent for each formulation had its own column (with the value as the reagent concentration in mM), and other input features were dynamically calculated, such as reagent:payload ratio, and payload chemical properties.

Following the workflow depicted in Figure 1B, the 13 ML models were first trained for predicting particle d_H, PdI, and EE, and their performance is shown in **Figure 3**A for the lipid-based nanoparticles and Figure 3B for the PLGA nanoparticles, presented in terms of the root-mean-square error (RMSE) and r^2 values. The ML models were ranked based on their performance across the three different output parameters (d_H, PdI, and EE). Figure 3 reports max-min normalized RMSE values, while the non-normalized RMSE values and precise r^2 values for all algorithms are reported in Tables S1 and S2 (Supporting Information). The random forest model presented, on average, the lowest RMSE per predicted output (dark purple boxes). Specifically, for the lipid-based nanoparticles, the random forest model returned average RMSE values of 16 $\pm$ 2 nm for d_H, 0.067 $\pm$ 0.005 for PDI, and 9.8 $\pm$ 1.2% for EE, after selecting for optimal hyperparameters with k fold cross validation on training data. These corresponded to r^2 values of 0.860, 0.749, and 0.929 for d_H, PdI, and EE, respectively. For the PLGA nanoparticles, the random forest model returned even lower average RMSE (and higher r^2) values corresponding to 11 $\pm$ 1 nm (0.935) for d_H, 0.053 $\pm$ 0.002 (0.629) for PDI, and 3.9 $\pm$ 0.5% (0.987) for EE. Note that the RMSE were related to an average diameter of 98 and 104 nm for lipid-based and PLGA nanoparticles, respectively.

A more granular look at the performance of the different ML models across the considered output parameters shows a wide variability. Interestingly, of all the output parameters, predicting EE seems to be the most straightforward, as for all the 13 ML models considered, a r^2 value larger than 0.75 is returned (yellow boxes in the r^2 plots for both lipid-based and polymeric nanoparticles in Figure 3). Conversely, predicting PdI appears to be the most difficult task, as for all the 13 considered ML models, a r^2 value varies largely between 0 and 1 (Figure 3).

Also, it was generally observed that if a specific ML model performed poorly at predicting PdI, the application of the same algorithm toward predicting d_H likewise performed poorly. This is confirmed by **Figure 4**, where the correlation between percent error in predicted for PdI and d_H is shown. The likely explanation of this is that the prediction of d_H depends on so-called monodisperse samples. Thus, if an independent model performs poorly in predicting the particle PdI with widespread variability in the PdI, then the underlying particles are unreliable in their size distribution. Thus, it follows that the prediction of particle diameter is similarly unreliable. It must be emphasized that the various models (i.e., for predicting d_H or PdI) are independently trained on the underlying data and are not themselves correlated. Thus, this correlation seen in Figure 4 is due to the actual physical nanoparticle population. One should further note the high variability of the individual predictions with each model type, which is reflected in the large standard deviation of the averaged points in Figure 4. It is understood, though, that a high percent error in the PdI prediction is thus likely an effect of a highly variable size population, which leads to further errors in predicting the particle size.

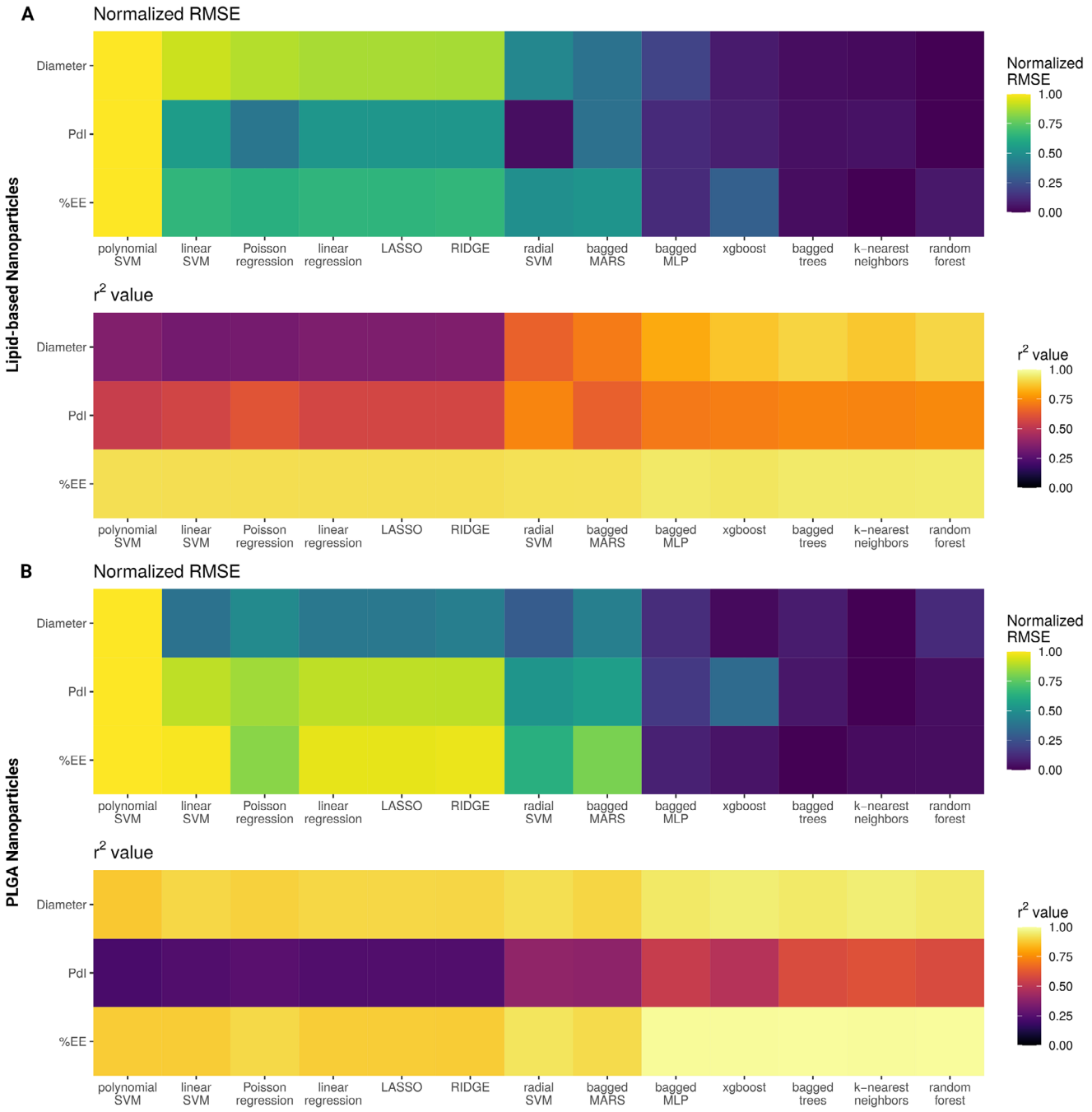

Figure 3. Performance of various machine learning models for predicting nanoparticle target attributes. Models were trained on thirteen different machine learning algorithms for predicting diameter, polydispersity index (PdI), and percent encapsulation efficiency (EE) for A) lipid-based nanoparticles and B) PLGA nanoparticles. Model accuracy was evaluated based on the root-mean-square error (RMSE), as well as the coefficient of determination (r^2 value). RMSE values were max-min normalized. Models are sorted by the minimum combined RMSE for predicting diameter, PdI, and EE for each particle class. Models were placed in order of the best average (minimum) RMSE for all models, for lipid-based nanoparticles.

The application of machine learning models toward formulation development is rapidly becoming a hot topic in the field of microfluidic formulation development. Recently, the authors reported the application of the open-source machine learning library scikit-learn in Python, coupled with an artificial neural network, toward predicting liposome size and dispersity.[20] In this previous work, a series of classification algorithms were similarly screened to determine whether a formulation was predicted to be "monodisperse" or "polydisperse", as well as "stable" (i.e., remain "monodisperse" over time) or "unstable." Recently, Eugster et al.[21] implemented machine learning toward streamlining the development of liposomal drug delivery systems and likewise

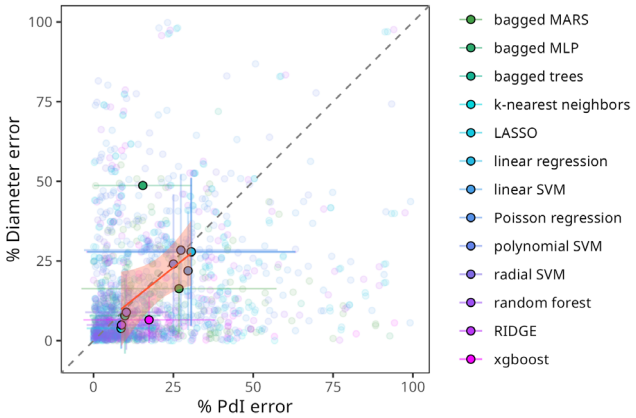

Figure 4. Correlation between the percent error for various machine learning models in predicting PdI and diameter. Translucent points represent the percent error (% error) for every measurement, with colors matched to the model. Solid points represent the average percent error for each model, with X-Y error bars representing the standard deviation for the percent error for PdI predictions and diameter predictions, respectively. The dashed diagonal line represents a perfect correlation between percent PdI error and percent Diameter error. The red line represents the best fit curve for the solid points.

screened various regression algorithms for predicting whether liposomes would form, the optimal FRR, and resultant liposome size. A 2022 study by Rebollo et al.[22] investigated the effect of cholesterol concentration, ionic strength of the aqueous phase (i.e., NaCl concentration), TFR, FRR, and temperature on the production of liposomes, and employed artificial neural networks to predict size and PdI. Finally, Wu et al.[23] reported the application of twelve different machine learning algorithms toward predicting the size and dispersity of chitosan nanoparticles produced by a multi-inlet vortex mixer. Thus, this systematic approach of screening various algorithms is relatively straightforward to implement programmatically and can assist in rapidly identifying the best algorithm for a classification or regression task. Machine learning has also been implemented to predict how combinations of drugs could spontaneously self-assemble into stable nanoparticles.[24,25]

2.4. Random Forest Model Hyperparameter Optimization

While the hyperparameters for all models screened were somewhat optimized via a 10-point random search, once the random forest model was selected, we performed a systematic tuning of model hyperparameters via a 10-point grid search. That is, we screened every combination over a range of 10 different values each for each of the models' hyperparameters (mtry, min n, and number of trees). Figure S3 (Supporting Information) shows the RMSE for each model predicting d_H, PdI, and EE for each of these combinations. It is apparent that the effect of the number of trees quickly attenuates, and after ca. 250 trees, there is no improvement in model performance. For *min n*, it appears that a lower value (e.g., *min n* = 2) yields the best performing models. Finally, in all cases, the largest *mtry* value results in the best-performing models. In all cases, the best hyperparameters are selected and locked into the final model.

2.5. Validation of the Random Forest Models

The most accurate model, the random forest model, was then evaluated first on the training data and then on the (unseen) testing data. A summary of all results on training and testing data are presented in **Table 2**. The random forest algorithm is a tree-based approach in which an uncorrelated collection of decision trees is evaluated as an ensemble in order to, for regression models, average the accuracy of the prediction.[26,27] The benefit of this approach is a reduced risk of overfitting at the cost of increased computational load, represented by an increasing number of "trees" in the forest.[28]

In order to evaluate the influence of training data size (i.e. percent of total dataset) on model accuracy, percentages of training data were tested on a range from 50–85% with 10 different randomized splits of the data. **Figure 5** shows the mean RMSE and r^2 value for the optimized random Forest model in predicting nanoparticle diameter for both lipid-based and PLGA nanoparticles. Likewise, Figure S4 (Supporting Information) shows the effect of percent training data on Random Forest model accuracy in predicting PdI and %EE, while Figure S5 (Supporting Information) shows the number of measurements within either the training or testing data, depending on what percent of the database is used for training data. It is important to note that RMSE and r^2 values remain relatively consistent across all training data percentages, which indicates that the model is not over-fitting.

To investigate how the size of the database (i.e. total number of measurements) affected the performance of the random forest models, the entire database was randomly sampled to create subsets ranging from 50 to 230 measurements. These smaller subsets were then processed through the MicrofluidicML workflow, and the model performance was reported as RMSE and r^2 value for both lipid-based (Figure S6, Supporting Information) and PLGA nanoparticles (Figure S7, Supporting Information). This process was repeated using 20 random seeds to vary the data sampling and splitting procedure. By repeatedly sampling with different random seeds, variability was observed in the model performance depending on which formulations were selected and how the data were separated into subsets. For certain attributes (i.e., predicting diameter or PdI for lipid-based nanoparticles, or PdI for PLGA nanoparticles), increasing the size of the database directly correlated with improved model performance. For other attributes (e.g. EE for both lipid-based and PLGA nanoparticles), there was no significant improvement in model performance after ~100 measurements. However, it is clear that increasing the total number of measurements (i.e. size of the database) reduces the variability introduced by the random sampling (Figure S8, Supporting Information). This implies that increasing the number of measurements results in models that are more predictive. The fact that the RMSE standard deviations do not plateau indicates that the critical number of measurements required for optimal model performance has not yet been reached.

Figure 6A,B show the correlation between outcomes predicted by the random forest model and those measured experimentally for the training data. The performance of the random Forest model on the training data is summarized in terms of RMSE and r^2. It appears that the models for PLGA nanoparticles perform better at predicting particle diameter (r^2 = 0.985) as compared to the model obtained for the lipid-based particles (r^2 = 0.950).

Table 2. Overall summary of random forest model performance. Random forest models' accuracy for predicting hydrodynamic diameter (d_H), PdI, and % encapsulation efficiency (EE) as reported by the root-mean-square error (RMSE) and coefficient of determination (r^2 value) for both training and testing data.

		Lipid-based nanoparticles			PLGA nanoparticles		
		d_H (nm)	PdI	EE (%)	d_H (nm)	PdI	EE (%)
RMSE	training data	9.8	0.046	12.9	5.4	0.033	3.1
	testing data	9.5	0.053	10.6	7.5	0.047	2.2
r^2 value	training data	0.950	0.903	0.668	0.985	0.849	0.970
	testing data	0.922	0.787	0.472	0.964	0.560	0.984

However, this is somewhat intuitive as the PLGA nanoparticles are comprised of a single material, while the lipid-based particles are comprised of two different particle architectures (LNP vs. liposomes) with different lipid compositions, including the relative lipid ratios, aqueous: organic phase ratios, and total lipid concentrations.

Testing data were randomly subset from the entire database and withheld from the model training process. The models constructed on the training data were then run on the input features of the testing data, and **Figure** 7A,B show the correlation between the predicted and measured values for d_H, PdI, and EE. The models for predicting d_H performed reasonably well with r^2 values of 0.926 and 0.964 for lipid-based nanoparticles and PLGA nanoparticles, respectively. Likewise, the RMSE for predicting d_H for the testing data was 9.5 and 7 nm for lipid-based nanoparticles and PLGA nanoparticles, respectively.

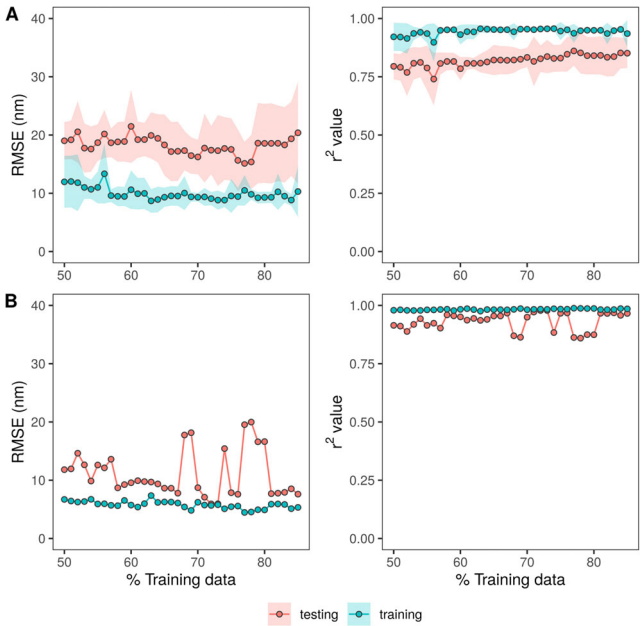

Figure 5. Effect of training data size (%) on random forest model performance in predicting diameter. Model performance is reported as either the root-mean-squared error (RMSE) or coefficient of determination (r^2 value) for A) lipid-based nanoparticles or B) PLGA nanoparticles. Lines and points represent mean values over 10 different randomized splits in the data, and shaded regions represent the mean value ± 1× standard deviation.

That stated, the models for predicting lipid-based nanoparticle outcomes perform quite well on the training data for predicting d_H and PdI (r^2 = 0.950 on training data), and were least performant on predicting the EE for the testing data (r^2 = 0.472 on testing data, Figure 7). However, it should be noted that only the liposomes are loaded with curcumin, thus skewing the distribution of empty versus CURC-loaded lipid-based nanoparticles within the training data set. For a brief comparison, models were trained using the random forest algorithm on a dataset comprised of only the liposomes. Following hyperparameter optimization via 10 fold cross-validation and a systematic 10-point grid search, liposome-only models were more performant when compared to the combined lipid-based nanoparticle models: RMSE of 9 ± 2 nm (r^2 = 0.871) for predicting d_H, RMSE of 0.061± 0.008 (r^2 = 0.824) for predicting PdI, and RMSE of 11.1 ± 1 nm (r^2 = 0.920) for predicting EE. Thus, it appears, at least initially, that model generality is traded for accuracy. On the other hand, for the polymeric particles, the RMSE for the diameter predictions was under 10 nm, against an average diameter ranging from 34 to 267 nm.

Admittedly, the models for predicting PdI for both PLGA and lipid-based nanoparticles performed relatively poorly (r^2 values of 0.787 and 0.560 for lipid-based and PLGA nanoparticles, respectively) for the testing data. Speculatively, this could be due to the variability of PdI data when particle populations are polydisperse.[29,30] That is, when particles with polydisperse populations (or perhaps multi-modal populations), their measurement via light scattering becomes unreliable; thus, modeling predictions on these "unreliable" data generate unreliable models. In fact, it is for this reason that only the monodisperse formulations are used in training the models for predicting d_H and EE. In order to test this hypothesis, random forest models were trained to predict PdI on only monodisperse (PdI ≤ 0.250) or polydisperse (PdI > 0.250) formulations, for both lipid-based and PLGA nanoparticles. Indeed, when trained on only monodisperse formulations, the capability to predict PdI on testing data was stronger (RMSE 0.019 and RMSE 0.040) compared to when trained on only polydisperse formulations (RMSE 0.060 and RMSE 0.057, for lipid-based and PLGA nanoparticles, respectively).

For the prediction of EE, lipid-based particles were somewhat worse compared to PLGA nanoparticles (RMSE of 12.9% versus 3.1% on training data). There are two explanations for this discrepancy: on one hand, the encapsulation of curcumin was only evaluated in the liposomes, as lipid nanoparticles are better suited for the delivery of nucleic acid payloads; on the other hand, the liposomes are comprised of multiple lipid types, whereas the PLGA nanoparticles were comprised of a singular material. To

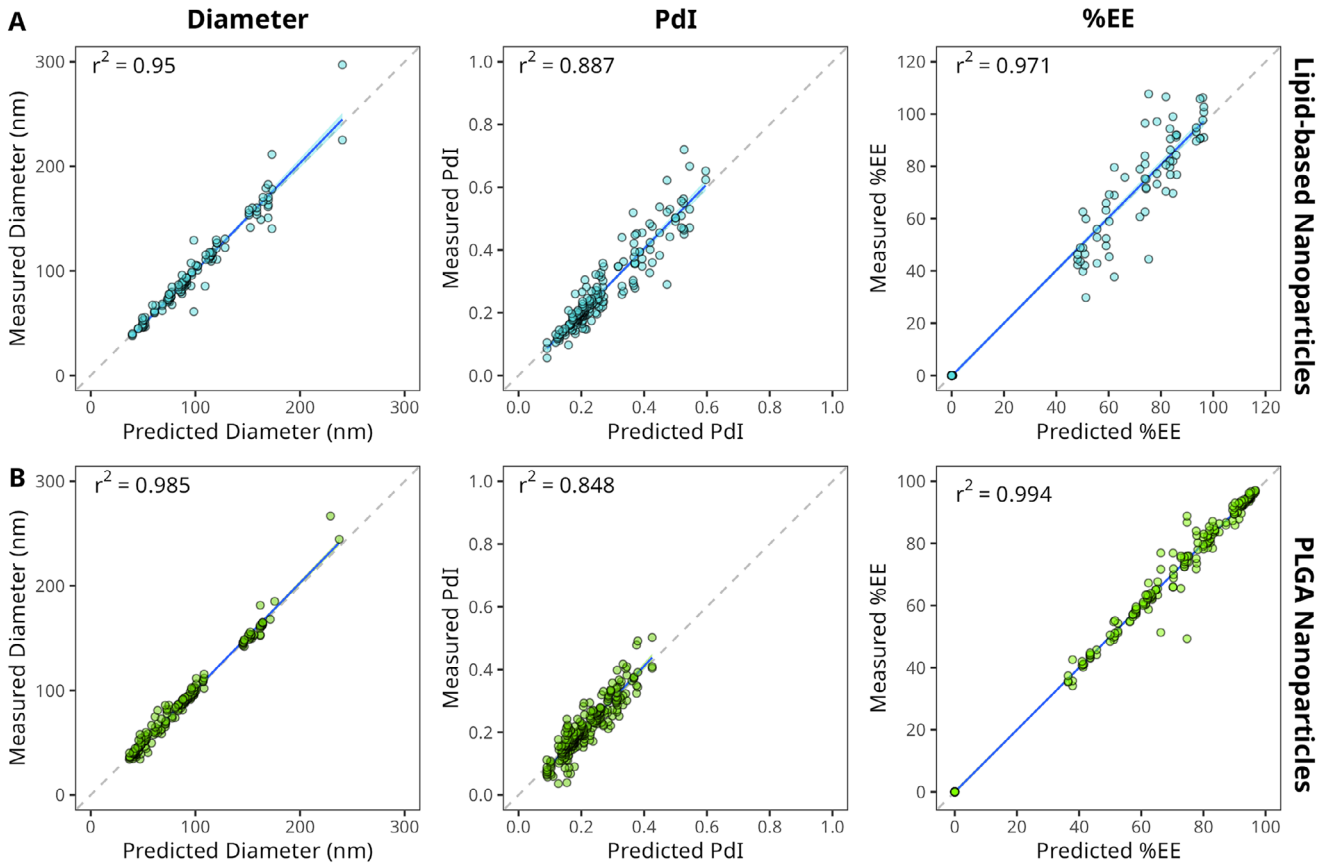

Figure 6. Optimized random forest model performance on training data. Random forest model accuracy in predicting diameter, PdI, and %EE for training data of A) lipid-based nanoparticles (comprised of lipid nanoparticles and liposomes combined), as well as B) PLGA nanoparticles. Solid blue lines represent best fit linear models between predicted and measured values, with shaded areas representing the standard error for the model. Dashed gray lines represent perfect correlation between predicted and measured values.

the first point, the loading of CURC into PLGA nanoparticles, a hydrophobic small molecule adapted for loading into a hydrophobic polymer, is ideal (r^2 value of 0.970 and 0.984 for training and testing data, respectively). Liposomes are not the ideal vector for loading a hydrophobic small molecule, and future work will surely necessitate investigating the loading efficiency of payloads more suitable for lipid-based carriers (i.e., nucleic acids, hydrophilic small molecules).

In order to test whether the same workflow could be applied in a completely "nanoparticle agnostic" fashion, random forest models for predicting d_H, PdI, and EE were compiled without first separating the particles into different subsets (i.e., lipid-based or PLGA nanoparticles). That is, all formulations were kept together, and models were built by subsetting the entire nanoparticle database into training and testing data with 10 fold cross-validation, preprocessing input features, and then optimizing random forest model hyperparameters.

These models were then utilized to make predictions on either the training data or testing data (all nanoparticle formulations combined). Figure S9 and Table S3 (Supporting Information) detail the results of these models, and it is apparent that models for predicting size and PdI perform relatively well, with RMSE of 11.1 nm and 0.051, respectively, on testing data. However, the model for predicting EE becomes non-performant (RMSE of 73.4% on testing data). This is presumably because of the material differences between lipids and PLGA, and the factors governing loading efficiency.

2.6. Assessing the Relevance of Input Features in the Design of Nanomedicines

In order to further elucidate the weight of each input feature in the microfluidic-based fabrication of nanoparticles, a KernelSHAP method was adopted.[31,32] Specifically, the SHapley Additive exPlanations (SHAP) is a method for computing the contribution of each input feature toward the output parameters by assessing the Shapley value,[33] which was computed via the *kernelshap* package in R.[34]

Figure 8 shows typical "beeswarm" plots where the input features are aligned vertically and ranked in order of their SHAP value. For each individual measurement, corresponding to a single-colored dot on the plot, the SHAP value is reported by the data point's position along the horizontal axis, with positive SHAP values indicating a significant contribution toward determining the output parameter, negative SHAP values indicating an inverse correlation between that input feature and the output parameter, and 0 SHAP values indicating the absence of any

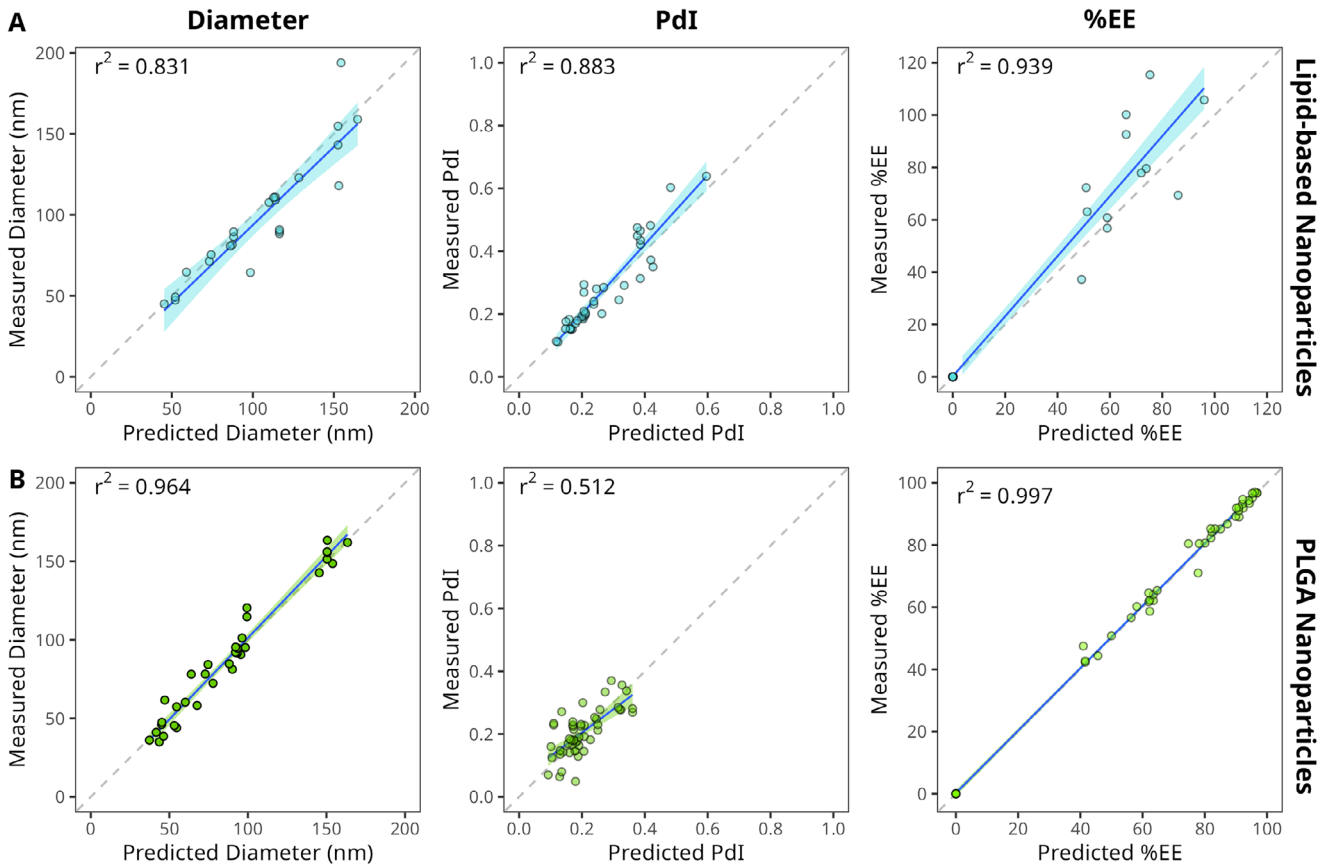

Figure 7. Models' performance on testing data. For testing data, which have not been seen by the models, plots show the correlation between predicted and measured diameter (d_H), polydispersity index (PdI), and encapsulation efficiency (%EE) for A) lipid-based nanoparticles and B) PLGA nanoparticles. Solid blue lines represent best fit linear models between predicted and measured values, with shaded areas representing the standard error for the model. Dashed gray lines represent perfect correlation between predicted and measured values.

influence of the input feature and the output parameter. The colors of the dots are associated with the max-min normalized value for each input feature. For instance, dark purple points on the TFR line correspond to low total flow rate values, while yellow points on the same line correspond to high flow rate values.

Some of the features reported, even if shown to strongly influence a prediction, are less powerful in this study. For example, in Figure 8 (%EE), the importance of the XlogP3, molecule polar surface area, and molecular weight of the API (Payload MW) is overstated, as only one API (i.e. CURC) was evaluated. Thus, in this case, there exist only two states for those features: 0 for no CURC and 1 for nanoparticles loaded with CURC. However, with a view toward the future, as more API are tested and added to the database, these features will become more relevant toward the predictions.

The plots can be interpreted as in this example, related to d_H in Figure 8B. The concentration of PLGA (Resomer RG 504H) is strongly predictive of d_H, and it can be clearly elucidated by the distinct separation between three feature levels and the distance between the low feature value cluster and the high feature value cluster. Figure S10 (Supporting Information) shows the mean absolute SHAP value computed by taking the average of the absolute value for all SHAP values: there is a 6-fold increase in mean absolute SHAP value for PLGA concentration compared to the next closest feature, TFR.

Of interest is the comparison between the SHAP analysis and the Pearson's correlation coefficient (PCC) in Figure 2C. While PCC assumes a linear correlation between the input feature and the output parameters, it is evident that there is overlap with the SHAP importance represented in Figure 8 and Figure S10 (Supporting Information). It is important to note that the SHAP values in Figure 8A reflect the feature importance for the mix of LNP and liposomes, whereas the PCC is individualized for LNP and liposomes. Moreover, the SHAP values consider the individual concentration of each constituent reagent and can therefore give a more detailed estimation for each input feature's effect on the predicted outcome. Thus, for the lipid-based nanoparticles, it appears that the cholesterol concentration is a main driver for d_H, followed by FRR and TFR, and the other input features. Meanwhile, TFR and FRR appear to drive lipid-based nanoparticle PdI, and the [total lipid]:[CURC] ratio drives EE.

For PLGA nanoparticles, PLGA concentration is the most important feature for determining d_H, which corresponds precisely with the PCC. Similarly, PLGA concentration is strongly determinant of PLGA nanoparticle PdI, which likewise corresponds with the PCC. Finally, flow parameters have little effect on CURC entrapment in the PLGA nanoparticles, and the main feature

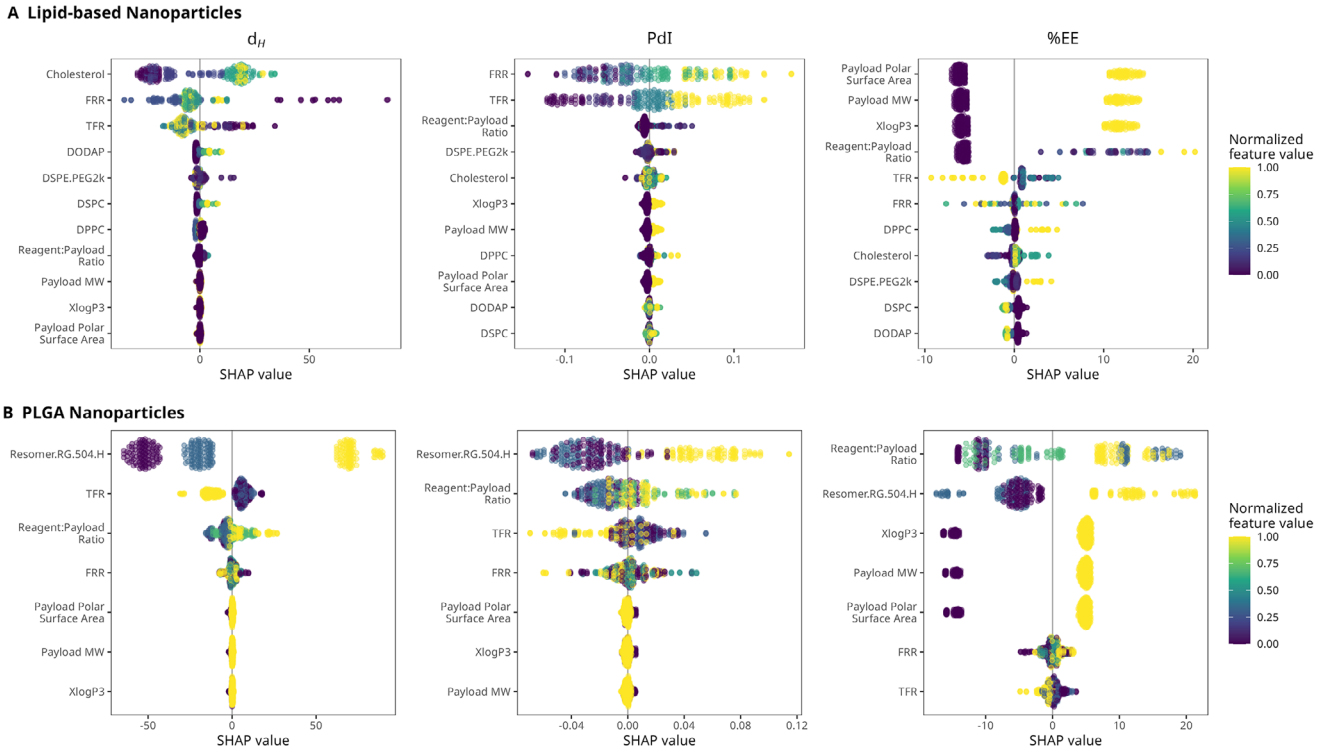

Figure 8. Evaluating feature importance for the random forest model. Feature importance was elaborated using the KernelSHAP method to deconstruct the contribution of each feature toward predicting hydrodynamic diameter (d_H), polydispersity index (PdI), and encapsulation efficiency (EE). Beeswarm plots for A) lipid-based nanoparticles and B) PLGA nanoparticles show the relative importance of each input feature ranked from most to least important along the y-axis. Each point on the plot is an individual measurement, and colors indicate the max-min normalized value of that input feature. Positive SHAP values indicate a strong influence of that feature on the predicted outcome, while negative SHAP values indicate an inverse relationship. Points (measurements) are jittered along the y-axis to illustrate the density of measurements around the SHAP value for each feature.

determining EE is the [PLGA]:[CURC] ratio, which fits nicely with the PCC. Thus, while these relationships may not be precisely linear, the kernelSHAP method provides an alternative perspective with regard to feature importance and is another valuable tool to understand how production parameters are related to the properties of the resultant nanoparticles.

Returning to the so-called "total models" where nanoparticles were not subset by class (e.g., lipid-based nanoparticles and PLGA nanoparticles) but rather all formulations in the database were retained for training and splitting (Figure S9 and Table S3, Supporting Information), the SHAP analysis provides insight into why the model for predicting EE failed (Figure 8). For both lipid-based nanoparticles and PLGA nanoparticles, the reagent: payload ratio is a major determinant for predicting EE. However, detailed examination of the SHAP values and the clustering of normalized feature values for both nanoparticle classes shows differences: for lipid-based particles, lower reagent: payload ratios have negative SHAP values, while higher reagent: payload ratios have positive SHAP values. For PLGA nanoparticles, all levels of reagent: payload ratios have positive SHAP values. There are other differences, such as TFR having a stronger effect on EE for lipid-based nanoparticles while not having a significant effect for PLGA nanoparticles. Thus, it is likely that with such dramatic differences in material characteristics and factors affecting loading, models for predicting EE could not perform. However, it remains to be seen whether building a database orders of magnitude larger, with more diverse combinations, could overcome this limitation.

2.7. Validation of Data on Unseen Formulations from Literature

To test how "general" the models were in predicting formulations never seen by the model, a sample of formulations was taken from the literature, and model accuracy was tested on these never-before-seen formulations from completely independent studies.[12,35–40] To maintain consistency in the microfluidic system employed to generate particles, we only tested data from articles where a Precision Nanosystems Benchtop NanoAssemblr with the same microfluidic chip was used. One of the limitations of the MicrofluidicML approach is that it is limited by the amount of data–it was only possible to produce so many formulations, using different materials, microfluidic parameters, and combinations thereof. Thus, articles in the literature did not have the exact same materials, but since the MicrofluidicML models do not yet factor in materials properties (molecular weight, log P values) or solvent properties (viscosity, volatility, miscibility with water), data were converted to material classes (rather than precise materials). For example, rather than testing specific PLGA (e.g., Resomer RG 504 H, Resomer RG 502 H, PURAC 5004A, etc.), all these polymers were classed simply as PLGA. Likewise, instead of testing for specific lipids, lipids were classed as either

"structural/helper" (e.g., DSPC, DPPC, or DOPE) or ionizable (e.g., DODAP, DLin-MC3-DMA, SM-102, etc.) lipids, and thus lipid concentrations were reported by their lipid type (i.e., helper lipid, ionizable lipid, PEG-lipid, sterol lipid).

Figure S11 (Supporting Information) shows the capability of models to predict LNP (Figure S11A, Supporting Information) or PLGA (Figure S11B, Supporting Information) nanoparticle size or PdI with formulations pulled from literature. It is apparent that the lipid-based model underperforms, and this is due to the wide variability of materials used in those formulations reported in the literature. Lipid-based nanoparticles are comprised of different types of lipids (e.g., helper/structural lipids, sterol lipids, PEG-lipids, or ionizable lipids) in different molar ratios, total lipid concentrations, etc. In order to improve the models' general performance, high-throughput formulation screening will be necessary in the future to produce and characterize a wider variety of lipid compositions.

Figure S11B (Supporting Information) shows how well the models for PLGA nanoparticles predict size and PdI, and while there are differences in polymer brand, what is interesting is that the formulations that were most accurately predicted fell within the range of the production parameters tested in the original MicrofluidicML database. The best performing predictions were made on a PLGA nanoparticle comprised of LACTEL B6010-2, which was compositionally similar to the Resomer RG 504 H used in the present study (lactide:glycolide ratio of 50:50, molecular weight range of 30–60 kDa), the organic phase was acetonitrile, the total PLGA concentration was 0.222 mM, and the microfluidic parameters were TFR of 5, 10, 15 mL min^{-1} and FRR of 1, 3, 5.[40] What is telling is that, even within the same published work, PLGA nanoparticles with different lactide: glycolide ratios (e.g., 75:25 or 85:15 instead of 50:50) are not well predicted by the MicrofluidicML models. These results emphasize the limitations of this approach–significantly more data are needed to account for variability in materials, and future iterations will need to account for more input features such as material properties (molecular weight, hydrophobicity, solubility, chemical composition), solvent properties (viscosity, volatility, water miscibility), etc. However, when compared with literature values, models are effective at interpolating nanoparticle attributes but are less effective when trying to extrapolate or match imprecise input features.

2.8. Machine Learning Models Compared to Generative Artificial Intelligence

The ultimate test would be the deployment of the optimal models toward providing the production parameters that would yield particles of a specified class with a desired diameter and underneath a defined PdI limit. The biggest competitor to the workflow proposed here, and so-called "conventional" machine learning in general, is the rise of generative large language models (LLM). Such models are based on the input of incomprehensible amounts of data, able to synthesize logical-sounding solutions to very general questions. Comparatively, the algorithm-based machine learning models proposed here are less flexible. In order to compare how such platforms may perform given a similar task, we prompted two such AI engines, OpenAI's ChatGPT[41] with the GPT-3.5 model and Google's Bard (now Gemini)[42] with the Bard/LaMDA-based model, to provide the optimal input features to produce either lipid nanoparticles or PLGA nanoparticles with a size of 100 nm. Full transcripts of those conversations, along with Tables S4 and S5 (Supporting Information) detailing the specifics of those formulations, can be found in the Supporting Information. Considering the rapid evolution of the AI field, these platforms were re-prompted at a later date to provide updated solutions, and another platform, DeepSeek[43] was included in the study.

This exercise in evaluating generative AI platforms versus the machine learning models developed in-house provided quite interesting results. Initially, both generative AI platforms were hesitant to provide a precise formulation. Rather, both tended to outline the important factors that dictate particle size with regard to the microfluidic production of particles. However, when pressed, both provided exact formulations. Further inputs were necessary to specify different, compatible solvents (i.e., using ethanol in the organic phase for lipid nanoparticles rather than chloroform). ChatGPT was equally cagey when asked what molecular weight PLGA should be used, but when pressed, it provided exact values for molecular weight, total flow rate, and the other parameters.

What was interesting about the Gemini platform is that it provided references to support its recommendations for the parameters it provided. However, the first reference was from Dasa et al.,[44] which does indeed refer to the formulation of liposomes. However, these liposomes were produced not with microfluidics, but via lipid film hydration and syringe extrusion. Moreover, the four references provided further in the conversation were completely fabricated, and we were unable to find any record of them in the archives of *Pharmaceutics*, the *International Journal of Pharmaceutics*, the *Journal of Microfluidics and Nanofluidics*, or the *Journal of Nanoparticle Research* (all real journals). While the Bard/Gemini platform did fabricate these references, it was interesting to note that it performed very well with regard to providing formulations that yielded both lipid-based nanoparticles and PLGA nanoparticles of 100 nm.

Meanwhile, MicrofluidicML was similarly employed to generate optimal formulations to yield lipid-based or PLGA nanoparticles with a mean d_H of 100 nm and PdI $\leq$ 0.250. MicrofluidicML is unable to generate dynamic responses in the same way as the generative LLM. Rather, a large array of formulations is synthetically screened–A matrix of production parameters was systematically generated, and then the random forest models were run against these input parameters. The optimal formulation, which matched the required nanoparticle attributes, was selected and tested. Thus, the MicrofluidicML approach in its current iteration relies to a degree on the user's expertise in the matter of microfluidic nanoparticle production.

Figure 9 shows the sizes and PdI after following the recommended formulation from all four platforms, both with the old prompts (e.g., GPT-3.5 and Bard) and the new prompts (GPT-4o, Gemini Flash 2.0, and DeepSeek-V3). For the "old" models, Bard/Gemini was very close to the 100 nm target, with a d_H of 93 ± 7 nm (PdI = 0.119) and 98 ± 1 nm (PDI = 0.211) for lipid-based and PLGA nanoparticles, respectively. Likewise, MicrofluidicML provided formulations that yielded 90 ± 5 nm in diameter (PDI = 0.061) and 97 ± 7 nm (PDI = 0.218) for lipid-based nanoparticles and PLGA nanoparticles, respectively. ChatGPT with GPT-3.5 was a bit farther off-target with the lipid-based

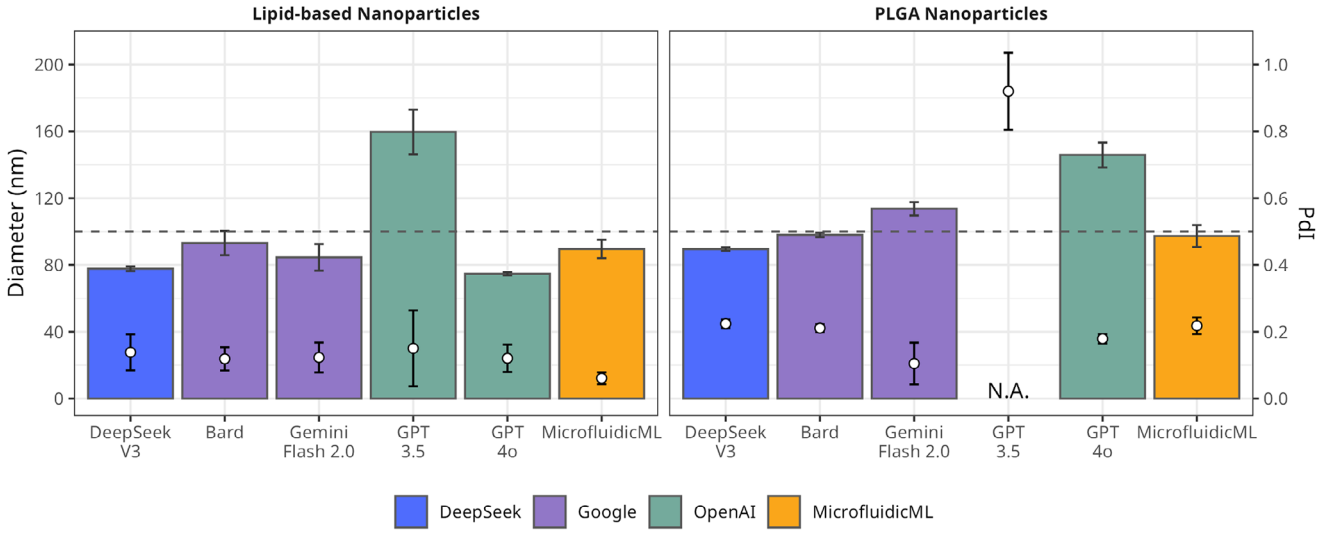

Figure 9. Comparing MicrofluidicML against generative artificial intelligence engines. Comparison between the results of formulations recommended by ChatGPT (OpenAI), Bard/Gemini (Google), and MicrofluidicML for producing lipid-based or PLGA nanoparticles. All platforms were prompted to provide parameters to produce 100 nm nanoparticles (dashed horizontal line), and bars show the mean diameter of particles generated following these conditions. Points show the mean PdI measured, and error bars show 1 standard deviation for either diameter or PdI. After some time, ChatGPT and Gemini were re-prompted (with their new AI models), and DeepSeek was added as a generative AI model platform, and these new formulations were attempted.

nanoparticle formulation, yielding particles with a diameter of 160 ± 13 nm (PDI = 0.150). However, the output of the ChatGPT formulation for PLGA nanoparticles was a cloudy-white solution with large, flocculated bodies with a high polydispersity (PdI = 0.920). However, one could anticipate this result because the ChatGPT platform recommended an aqueous: organic FRR of 1:3, i.e., a substantially higher volume of organic solvent. Thus, one could reasonably predict that nanoparticles would not form due to the significant presence of the organic solvent.

With the rapid development of LLM, we re-prompted the models (using a new, unassociated account) on updated AI engines, and there were significant changes in their responses. Both ChatGPT (GPT-4o) and Gemini (Flash 2.0) were much more direct in providing a solution. GPT-4o included citations as references, and these papers were real and on topic. Initially, GPT-4o provided microfluidic parameters not compatible with our platform (a total flow rate of 0.4 mL min^{-1}) for the production of lipid-based nanoparticles. Interestingly, it provided a formulation with an unconventional combination of lipids: DOPE, cholesterol, and DOTAP at molar ratios of 50:45:5. After prompting to provide a higher flow rate, GPT-4o provided a formulation much more in line with what was provided by Flash 2.0 and DeepSeek-V3, which were in line with conventional formulations of lipid nanoparticles (i.e. Dlin-MC3-DMA, DSPC, Cholesterol, and DMG-PEG_{2k} at percent molar ratios of 50: 10: 38.5: 1.5).

Formulations recommended by DeepSeek V3 yielded 78 nm (PdI = 0.139) and 90 nm (PdI = 0.224) LNP and PLGA nanoparticles, respectively. Google's new LLM engine, Gemini Flash 2.0, yielded 85 nm (PdI = 0.123) LNP and 114 nm (PdI = 0.105) PLGA nanoparticles. OpenAI's GPT-4o provided formulations that returned LNP of 75 nm (PdI = 0.121) and PLGA nanoparticles of 146 nm (PdI = 0.179). Interestingly, this PLGA nanoparticle formulation recommended by GPT-4o was the least "accurate," however was a big improvement over the earlier generation. Notably, the older LLM versions (GPT-3.5 and Bard) provided more "unique" solutions for lipid-based nanoparticle formulations, while all LNP formulations recommended by the newer LLM generations (GPT-4o, Gemini Flash 2.0, and DeepSeek) were quite similar in composition and diameter (≈79 ± 5 nm). Thus, it appears the types of recommendations and inputs generated by the LLM were converging for LNP. On the other hand, MicrofluidicML is able to generate innovative formulations because it is more simplistic than an LLM and is constrained by the input data. This can be seen as the MicrofluidicML approach generated an unconventional LNP formulation.

While generative LLM are more "creative" and flexible with regards to proposing formulations, there remains the issue of LLM hallucination.[45,46] Furthermore, with the updated LLM engines, all formulations converged to a similar formulation. This is likely due to the fact that this formulation is statistically generated based on those formulations most frequently reported in the literature. The MicrofluidicML approach, employing supervised machine learning algorithms to train predictive models, is constrained by the underlying data. Thus, it can only make predictions based on the (quality) data underpinning the platform. By enriching that data, adding more formulations and variables, it may be possible to sustainably grow the platform into a significant tool for formulation development. The generative LLM also requires a significant amount of data to train, and the scraping of this data from the web comes with risks, such as from AI models trained on AI-generated data[47] and potential ethical/copyright issues.[48] By contrast, the MicrofluidicML approach guarantees that there are no ethical concerns, as all data is generated and owned by the user, and the computational workload is minimal–i.e., able to run on a personal computer. In fact, screening the different machine learning algorithms requires < 10 seconds to

compile per model (Figure S12, Supporting Information), with relatively low memory usage (Figure S13, Supporting Information). Compilation of the random forest model, including systematic hyperparameter optimization, is accomplished in <5 min per model (Table S6, Supporting Information) with relatively low peak memory usage (Table S7, Supporting Information). Notably, these values are dependent on the hardware employed to compile the models.

3. Conclusion

AI is at the cutting-edge of research, and powerful AI-driven engines are frequently being applied to new problems. In the past, the utilization of machine learning and the associated complex computational tasks either required significant computing power or in-depth knowledge of statistics and mathematics. While not intending to undermine the importance of learning and understanding the fundamental underpinnings of these methods, machine learning frameworks and libraries are being developed in accessible programming languages, such as Python and R, that can be implemented on personal computers (i.e., not require powerful GPU or computer clusters), which can enable researchers in non-machine learning research fields to apply these algorithms to niche or specialized use-cases.

We reported the application of open-source machine learning libraries to develop MicrofluidicML, a workflow aimed at guiding the development of nanomedicines produced via microfluidics. This builds on a database comprised of three different nanoparticle types (liposomes, lipid nanoparticles, and PLGA nanoparticles) to develop relatively agnostic machine learning models that can predict d_H, PdI, and payload EE based on the microfluidic input features (i.e., total flow rate, flow rate ratio, individual reagent concentrations, and payload chemical properties). The selection of a random forest model was based on the screening of numerous machine learning algorithms. The final random forest model hyperparameters were optimized via a 10-point grid search with *k*-fold cross-validation. MicrofluidicML is able to accurately predict nanoparticle target attributes and provide (albeit general) formulation parameters to achieve particles of a specific class (i.e., lipid-based or PLGA nanoparticles) with specific properties (i.e., d_H, PdI, and EE). The model, evaluated on the unseen testing data, was quite accurate in predicting d_H and EE, however struggled to make predictions on PdI. However, this approach is flexible and scalable–the input of new data would mean a new model could easily and dynamically be built and re-optimized.

This work represents a foundational stone upon which a larger, comprehensive workflow is envisioned. Through the implementation of automated nanoparticle production and characterization methods, one can imagine a highly automated and data-rich workflow in which machine learning models are continually improved upon. With sustained activity, the underlying database can be greatly expanded upon, and these models can grow and adapt to incorporate even more input features (i.e., solvent type and properties), new particle types (i.e., hybrid polymer-lipid nanoparticles), and an increasing diversity in formulations (i.e., new API or particle/API combinations).

While this is a somewhat niche field (i.e., the application of AI and machine learning toward nanomedicine formulation development), we believe that further development can be of great utility toward the community working in the biomedical application of nanoparticles and can eventually streamline the future development of nanoparticle-based therapeutic platforms. This initial database serves as a proof-of-concept, but with continued input, these models have the potential to grow as an increasingly more general and more powerful machine learning-driven resource in the nanomedicine field.

4. Experimental Section

Materials: 1,2-distearoyl-*sn*-glycero-3-phosphocholine (DSPC), 1,2-dipalmitoyl-*sn*-glycero-3-phosphocholine (DPPC), methoxy- or carboxyl-terminated 1,2-distearoyl-*sn*-glycero-3-phosphoethanolamine-*N*-(polyethylene glycol)$_{2k}$ (DSPE-PEG$_{2k}$), and 1,2-dioleoyl-3-dimethylammonium-propane (DODAP) were purchased from Avanti Polar Lipids. Cholesterol and poly(lactide-*co*-glycolide) (Resomer RG 504 H, 50:50 lactide: glycolide, MW = 38-54 kDa) were purchased from Merck.

Microfluidic-Based Nanoparticle Fabrication: Nanoparticles were fabricated using a Precision NanoSystems Benchtop NanoAssemblr instrument. The system was equipped with a microfluidic micromixer cartridge characterized by a "Y"-like channel with staggered herringbone geometric features in the channel to facilitate fluid mixing.

Lipid nanoparticles were produced by preparing mixtures of different lipids (DODAP: DSPC: Cholesterol: DSPE-PEG$_{2k}$-OMe) at the molar ratio of 50:10:38.5:1.5 in ethanol. Three different total lipid concentrations were considered: 20, 40, and 60 mM. These were mixed with citrate buffer (50 mN, pH 3.6) at varying aqueous: organic flow rate ratios (FRR: 1:1, 3:1, and 5:1) and total flow rates (TFR: 2, 4, and 8 mL min^{-1}). The output LNP was dialyzed against 1× PBS (pH 7.4) overnight to remove traces of ethanol and restore a neutral pH.

Liposomes were prepared by dissolving DPPC, cholesterol, and DSPE-PEG$_{2k}$-COOH at molar ratios of 60:30:10 in ethanol. Three different total lipid concentrations were considered: 12, 24, 48 mM. For curcumin-loaded liposomes, curcumin was dissolved in ethanol with the lipids at varying [CURC]:[total lipid] ratios. This organic phase was mixed with 1 × PBS (pH 7.4) at varying aqueous: organic flow rate ratios (FRR: 3:1, 6:1, 9:1) and varying total flow rates (TFR: 1, 4, 8 mL min^{-1}). Output liposomes were dialyzed against 1× PBS (pH 7.4) at 4 °C overnight to remove excess ethanol (and free CURC).

PLGA nanoparticles were made by dissolving PLGA in acetonitrile at varying concentrations (5, 10, and 20 mg mL^{-1}, or 109, 217, and 438 µM, respectively). PLGA nanoparticles were self-assembled via microfluidic micro-mixing by systematically varying the flow rate ratio between the aqueous (Tris HCl buffer, pH 7.2, 10 mM) and organic (acetonitrile) phases at FRR = 3:1, 4:1, and 5:1. The total flow rate (TFR) was also systematically varied at 4, 6, and 13 mL min^{-1}. For CURC-loaded PLGA nanoparticles, CURC was dissolved in acetonitrile at various [PLGA]:[CURC] ratios. PLGA nanoparticles were synthesized in the NanoAssemblr Benchtop system, and the final particles were collected and washed via centrifugation at 18,213 rcf (12,700 rpm) for 30 min at 4 °C.

Nanoparticle Physico-Chemical Characterization: Particle d_H and PdI were characterized via dynamic light scattering (DLS) on a Malvern Zetasizer Nano S platform. Lipid-based nanoparticles were diluted 1:100 in distilled water before measurement. PLGA nanoparticles were evaluated directly after microfluidic production.

To evaluate the EE of CURC, PLGA nanoparticles were collected via centrifugation (18,213 rcf for 30 min at 4 °C). The supernatant was collected and analyzed by UV–vis on a Tecan Sark spectrophotometer at λ_{Abs} = 425 nm. The CURC concentration in the supernatant was determined via a calibration curve in the range of 0–15.625 µg mL^{-1}. EE was determined through Equation 1 as

$$\%EE = \frac{C_i - C_s}{C_i} \cdot 100 \quad (1)$$

where C_i is the initial CURC concentration used in the synthesis (i.e.: the concentration in the organic phase pre-microfluidic mixing) and C_s is the CURC concentration detected in the supernatant following production and centrifugation.

The Pearson's coefficient (Equation 2) was used to assess possible correlation between input features and target attributes:

$$PCC = \frac{\sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \quad (2)$$

where x_i and y_i are the individual values for every observation, and $\bar{x}$ and $\bar{y}$ are the average values for each variable.

Machine Learning to Predict Nanoparticle Properties: ML models were built in the R language (Version 4.3.3 "Angel Food cake")[49] using the *tidymodels* package,[50] an R package geared toward streamlining the process of training ML models by enabling data splitting, data pre-processing, and model tuning. Specifically, the resulting machine learning workflow, dubbed MicrofluidicML, employs various ML models within the *tidymodels* and *parsnip*[51] workflow to train models.

In the particle fabrication process, multiple independent parameters (input features) were considered including composition, such as the type of reagents (lipid, polymer), the reagent concentrations, the ratio between the concentration of the active pharmaceutical ingredient (API) and the reagents, as well as flow parameters, such as the total flow rate (TFR) and flow rate ratio (FRR). Likewise, API properties such as molecular weight, theoretical logP value (XlogP3),[52] and the estimated topological polar surface area (in Å^2) were implemented as input features.[53] Prior to building the models, the database, including all the experimental values of the input features, was subset into the relevant data. Specifically, nanoparticles were subdivided into the 'lipid-based nanoparticle' category, comprising lipid nanoparticles and liposomes, and the 'polymeric nanoparticle' category (i.e., PLGA nanoparticles). Moreover, data were randomly split into training data (85% of the subset data) and testing data (15% of the subset data). Three output parameters were considered, namely d_H, PdI, and EE of the nanoparticles.

Thirteen different ML algorithms were considered to build models for predicting the three target attributes, including *i*. linear regression (linear_reg with the 'lm' engine), *ii*. bagged decision trees (bag_tree via the 'rpart' engine),[54] *iii*. bagged multilayer perceptron (bagged MLP) neural networks (bag_mlp),[26,55] *iv*. bagged multivariate adaptive regression splines (MARS, bag_mars),[56] *v*. boosted decision trees (boost_tree with the 'xgboost' engine),[57] *vi*. k-nearest neighbors (nearest_neighbor),[58,59] *vii*. least absolute shrinkage and selection operator (LASSO) regression (linear_reg via the 'glmnet' engine with a Lasso penalty proportion of 1),[60–62] *viii*. Poisson regression (poisson_reg with the 'glm' engine),[63] *ix*. RIDGE regression (linear_reg via the 'glmnet' engine with a Lasso penalty proportion of 0), *x.–xii*. support-vector machines with linear, polynomial, and radial kernel functions (svm_linear, svm_poly, and svm_rbf, respectively, with the 'kernlab' engine),[64,65] and *xiii*. random forest (rand_forest with the 'randomForest' engine)[66] regression algorithms.

During the training phase, the different models' hyperparameters, which were used to tune the actual learning process, were optimized via a 10-point random search. Each model's hyperparameters were tuned by testing a range of hyperparameter values between a maximum and minimum ascribed by the ML engine. Prior to training, any zero-variance input features were dropped, and all input features were normalized (range 0 to 1) to eliminate potential bias. The performance of the model in predicting the target while testing the different hyperparameter values was evaluated via *k*-fold cross-validation (in the present work $k = 10$). The original training dataset is divided into *k* data subsets; the ML model is trained on the data contained in the combined *k*-1 subsets and eventually validated on the remaining data subset. This approach is repeated for each fold, and model performance is averaged to compute the accuracy of the method and its hyperparameter selection.[67] Each of the *k* subsets comprises training data where the mean and standard deviation for the target attribute (d_H, PdI, or EE) were roughly equal.

Two metrics were employed to report the models' performance: the root-mean-square error (RMSE) and the coefficient of determination (r^2 value), as computed by Equations 3 and 4, respectively:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \quad (3)$$

$$r^2 = \frac{SS_{total} - SS_{residual}}{SS_{total}} = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \quad (4)$$

where n is the sample size (number of measurements), y_i is the measured value of observation i, $\hat{y}_i$ is the predicted value of observation i, $\bar{y}$ is the sample mean for all observations, while SS_{total} and $SS_{residual}$ are the sum of squared prediction errors and sum of squared residuals, respectively. For selecting the optimal hyperparameters, the minimum RMSE was used as the determining metric.

For the models to predict the particle d_H or EE, only particle configurations associated with a PdI $\leq$ 0.250 were included, as these were considered suitably "monodisperse." To evaluate model performance across different ML algorithms, the RMSE value was max-min normalized as shown in Equation 5:

$$RMSE_{normalized} = \frac{RMSE_{i,j} - RMSE_{min,j}}{RMSE_{max,j} - RMSE_{min,j}} \quad (5)$$

where $RMSE_{i,j}$ represents the RMSE for algorithm i for predicting the target attribute j (i.e.: d_H, PdI or EE), $RMSE_{min,j}$ is the minimum RMSE of all algorithms for predicting the target attribute j, and $RMSE_{max,j}$ is the maximum RMSE of all algorithms for predicting the target attribute j.

Model performance was also evaluated in the context of the percent error (% Error), as calculated in Equation 6:

$$\%Error = \left|\frac{y_i - \hat{y}_i}{\hat{y}_i}\right| \cdot 100 \quad (6)$$

After selecting the optimal algorithm/model, hyperparameters were further optimized by a systematic 10-point grid search. Here, the entirety of the training data was pre-screened to select the range of hyperparameter values for the random forest model–that is, the *mtry* value (the number of randomly selected predictors), the number of trees in the "random forest," and the *minimum n* value (the minimum node size). Then, between the minimum and maximum hyperparameter values, 10 points were selected for each hyperparameter. A matrix of values is created with every possible combination of each hyperparameter (i.e. 1000 combinations), and models were trained on the training data via *k*-fold cross-validation. From these cross-validated results, the best-performing model for each attribute was selected.

To test whether the model was sufficiently general based on the dataset size, a script was written that employed the above hyperparameter optimization step across various sizes of training data (i.e. 50–85% training data). A loop was written where 10x seeds were randomly generated, and these were used as points to randomly split the database into training data and testing data. For each of the ten random seeds, the data were split between 50–85% training data, and the random forest model hyperparameters were optimized, and the model performance was evaluated on the testing data.

Elaborating on Feature Importance from the Machine Learning Model: The relative contribution of each input feature to the output parameters was ranked via SHAP values using the *kernelshap* and *shapviz* packages.[34,68] kernelshap provides a "model agnostic" implementation of the KernelSHAP method that can be integrated into a *tidymodels* workflow, through which it is possible to obtain SHAP values from the models. A "kernelshap" object is generated based on the best model (i.e., optimal hyperparameter conditions), and computed with the entire relevant database (i.e., subsets of all lipid-based nanoparticle or PLGA nanoparticle measurements). This can then be piped into a "shapviz" object, which,

through the *shapviz* package, enables visualization of the SHAP values for each model.

Validating the Model on Formulation Data from the Literature: To test how general the models were in predicting formulation parameters, data were taken from recent literature in which nanoparticles were formulated on a Precision Nanosystems Benchtop NanoAssemblr. Precise data from freely available databases (e.g., when provided in Supporting Information) were used when possible; otherwise, experimental values (generally mean values) were extracted from figures. Briefly, figures were imported into GNU Image Manipulation Program (GIMP), and pixel locations for graphical axes markers (i.e., y-values) were mapped to reported y-axis values. From these data (i.e., y-axis pixel location vs. y-axis marker value), a linear correlation was created, and mean nanoparticle attributes (e.g., size, PdI, etc.) were extrapolated from the graphical information.

Data were re-shaped to fit the same format as the MicrofluidicML database, and models were run on input features in order to make predictions on available nanoparticle attributes. Because there was rarely a one-to-one match in materials, reagents were defined by material "type" rather than precisely matched in order to be processed by the model. For example, rather than generating columns for each unique type of PLGA (e.g., LACTEL B6010-2, Resomer RG 502 H, PURAC 5004A, etc.), all polymers were classed together as simply PLGA. This was done because currently material properties (e.g., molecular weight, log P value, etc) were not features in the MicrofluidicML model. Likewise, solvent properties were not considered. Predictions on nanoparticle size and PdI were then made using the random forest models.

Comparison Against Generative Large Language Models: In order to identify the "optimal microfluidic formulation" via the MicrofluidicML models, a large matrix of input features (e.g. TFR, FRR, reagent concentrations) was generated based on the related data subset. Each feature was dynamically selected–for example, if making a prediction with "lipid-based nanoparticles" as the nanoparticle class, the program will generate a column for every reagent used to produce lipid-based nanoparticles and generate a range of n = 3 values evenly spaced between the maximum and minimum concentrations for each reagent. This is likewise done for TFR, FRR, and (if applicable) the ratio of total reagent concentration to API concentration. Then, the platform will programmatically generate every potential combination of all of these features, dynamically. The model for predicting PdI is then run against these formulation parameters, and only those formulations with a predicted PdI below the input threshold (0.250) were kept. The input features from these remaining formulations were then entered into the models for predicting d_H and EE. These results were sorted in order of the offset error from the desired diameter, and the top formulation is output.

For soliciting optimal formulations from large language models, different generative AI platforms (OpenAI's ChatGPT, Google's Gemini, and DeepSeek) were prompted to provide formulation parameters to produce lipid-based or PLGA nanoparticles with a size of 100 nm, as detailed in the Supporting Information. Improvements in AI engine performance were evaluated by again prompting these platforms with updated AI engines.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

This work was supported in part with funding from the European Union's Horizon 2020 Research and Innovation programme under the Marie Skłodowska-Curie grant agreement number 754490 within the MINDED project. The authors acknowledge icons used in Figure 1 provided by "Font Awesome by Dave Gandy–http://fontawesome.io," freely available under a CC BY 4.0 license (https://creativecommons.org/licenses/by/4.0/).

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Keywords

lipid nanoparticles, liposomes, machine learning, microfluidics, nanomedicine

Received: June 6, 2025
Revised: July 30, 2025
Published online:

[1] G. Yamankurt, E. J. Berns, A. Xue, A. Lee, N. Bagheri, M. Mrksich, C. A. Mirkin, *Nat. Biomedical Eng.* **2019**, *3*, 318.
[2] S. A. Damiati, D. Rossi, H. N. Joensson, S. Damiati, *Sci. Rep.* **2020**, *10*, 19517.
[3] F. Mekki-Berrada, Z. Ren, T. Huang, W. K. Wong, F. Zheng, J. Xie, I. P. S. Tian, S. Jayavelu, Z. Mahfoud, D. Bash, K. Hippalgaonkar, S. Khan, T. Bbuonassisi, Q. Li, X. Wang, *Npj Computational Mater.* **2021**, *7*, 55.
[4] H. Tao, T. Wu, M. Aldeghi, T. C. Wu, A. Aspuru-Guzik, E. Kumacheva, *Nat. Rev. Mater.* **2021**, *6*, 701.
[5] X. Chen, H. Lv, *NPG Asia Mater.* **2022**, *14*, 69.
[6] P. Tan, X. Chen, H. Zhang, Q. Wei, K. Luo, *Seminar Cancer Biol.* **2023**, *89*, 61.
[7] H. T. Hsueh, R. T. Chou, U. Rai, W. Liyanage, Y. C. Kim, M. B. Appell, J. Pejavar, S. Jayavelu, Z. Mahfoud, D. Bash, K. Hippalgaonkar, S. Khan, T. Buonassisi, Q. Li, X. Wang, *Nat. Commun.* **2023**, *14*, 2509.
[8] A. Ortiz-Perez, D. van Tilborg, R. van der Meel, F. Grisoni, L. Albertazzi, *Digital Discov.* **2024**, *3*, 1280.
[9] D. Y. Ding, Y. Zhang, Y. Jia, J. Sun, *Preprint at arXiv.* **2023**.
[10] S. J. Shepherd, D. Issadore, M. J. Mitchell, *Biomater.* **2021**, *274*, 120826.
[11] H. Zhang, J. Yang, R. Sun, S. Han, Z. Yang, L. Teng, *Acta Pharm. Sinica B.* **2023**, *13*, 3277.
[12] E. Kastner, R. Kaur, D. Lowry, B. Moghaddam, A. Wilkinson, Y. Perrie, *Int. J. Pharm.* **2014**, *477*, 361.
[13] F. Alexis, E. Pridgen, L. K. Molnar, O. C. Farokhzad, *Mol. Pharmaceutics* **2008**, *5*, 505.
[14] A. E. Nel, L. Mädler, D. Velegol, T. Xian, E. M. V. Hoek, P. Somasundaran, F. Klaessig, V. Castranova, M. Thompson, *Nat. Mater.* **2009**, *8*, 543.
[15] E. Blanco, H. Shen, M. Ferrari, *Nat. Biotechnol.* **2015**, *33*, 941.
[16] C. Kinnear, T. L. Moore, L. Rodriguez-Lorenzo, B. Rothen-Rutishauser, A. Petri-Fink, *Chem. Rev.* **2017**, *117*, 11476.
[17] K. Park, A. Otte, T. Li, *Mol. Pharmaceutics* **2024**, *21*, 3732.
[18] U.S. Food and Drug Administration (FDA), *Drug Products, Including Biological Products, that Contain Nanomaterials: Guidance for Industry*, U.S. Department of Health and Human Services, Silver Spring, MD **2022**.
[19] European Medicines Agency, *Data Requirements for Intravenous Liposomal Products Developed with Reference to an Innovator Liposomal Product: Scientific Guideline*, European Medicines Agency, Amsterdam, Netherlands **2021**.
[20] V. Di Francesco, D. P. Boso, T. L. Moore, B. Schrefler, P. Decuzzi, *Biomed. Microdevices.* **2023**, *25*, 29.

[21] R. Eugster, M. Orsi, G. Buttitta, N. Serafini, M. Tiboni, L. Casettari, J-L. Reymond, S. Aleandri, P. Luciani, *J. Controlled Release.* **2024**, *376*, 1025.
[22] R. Rebollo, F. Oyoun, Y. Corvis, M. M. El-Hammadi, B. Saubamea, K. Andrieux, N. Mignet, K. Alhareth, *ACS Appl. Mater. Interfaces.* **2022**, *14*, 39736.
[23] H. Wu, J. He, H. Cheng, L. Yang, H. J. Park, J. Li, *Int. J. Biol. Macromol.* **2022**, *222*, 1229.
[24] Y. Shamay, J. Shah, M. Işık, A. Mizrachi, J. Leibold, D. F. Tschaharganeh, D. Roxbury, J. Budhathoki-Uprety, K. Nawaly, J. L. Sugarman, E. Baut, M. R. Neiman, M. Dacek, K. S. Ganesh, D. C. Johnson, R. Sridharan, E. L. Chu, V. K. Rajasekhar, S. W. Lowe, J. D. Chodera, D. A. Heller, *Nat. Mater.* **2018**, *17*, 361.
[25] C. Zhang, Y. Yuan, Q. Xia, J. Wang, K. Xu, Z. Gong, J. Lou, G. Li, L. Wang, L. Zhou, Z. Liu, K. Luo, X. Zhou, *Adv. Sci.* **2025**, *12*, 2415902,
[26] L. Breiman, *Mach. Learning* **1996**, *24*, 123.
[27] M. Kuhn, K. Johnson, *Applied Predictive Modeling*, Springer, New York, USA **2016**.
[28] G. Biau, E. Scornet, *Test.* **2016**, *25*, 197.
[29] S. Bhattacharjee, *J. Control. Release.* **2016**, *235*, 337.
[30] D. Langevin, E. Raspaud, S. Mariot, A. Knyazev, A. Stocco, A. Salonen, A. Luch, A. Haase, B. Trouiller, C. Relier, O. Lozano, S. Thomas, A. Salvati, K. Dawson, *NanoImpact.* **2018**, *10*, 161.
[31] S. M. Lundberg, S-I. Lee, *Advances in Neural Information Processing Systems*, (Eds: I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett), Cuirran Associates Inc, Red Hook, NY, USA **2017**.
[32] I. Covert, S-I. Lee, *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, (Eds.: A. Banerjee, K. Fukumizu), The Proceedings of Machine Learning Research, NY, USA **2021**.
[33] L. S. Shapley, *Contributions to the Theory of Games*, (Eds.: H. W. Kuhn, A.W. Tucker), Princeton University Press, Princeton, NY, USA, **1953**.
[34] M. Mayer, D. Watson, kernelshap: Kernel SHAP v0.7.0, https://CRAN.R-project.org/package=kernelshap, (accessed: August 2024).
[35] V. M. Shah, D. X. Nguyen, P. Patel, B. Cote, A. Al-Fatease, Y. Pham, M. G. Huynh, Y. Woo, A. W. Alani, *Nanomedicine* **2019**, *18*, 146.
[36] C. Webb, S. Khadke, S. T. Schmidt, C. B. Roces, N. Forbes, G. Berrie, Y. Perrie, *Pharmaceutics.* **2019**, *11*, 653,
[37] C. B. Roces, G. Lou, N. Jain, S. Abraham, A. Thomas, G. W. Halbert, Y. Perrie, *Pharmaceutics.* **2020**, *12*, 1095,
[38] H. K. Mandl, E. Quijano, H. W. Suh, E. Sparago, S. Oeck, M. Grun, P. M. Glazer, W. M. Saltzman, *J. Control. Release* **2019**, *314*, 92.
[39] M. Cruz-Acuña, H. Kakwere, J. S. Lewis, *J. Biomed. Mater. Res. A.* **2022**, *110*, 1121.
[40] C. B. Roces, D. Christensen, Y. Perrie, *Drug Delivery Transl. Res.* **2020**, *10*, 582.
[41] OpenAI, Optimizing Lipid Nanoparticles, **2023**, https://chat.openai.com/share/891f04ea-5d9c-4763-b1ee-e5fea3036a1c, (accessed: February 2025).
[42] Google AI Gemini, Microfluidic Synthesis of 100 Nm Lipid Nanoparticles, **2023**, https://g.co/gemini/share/54b65fd8362e, (accessed: February, 2025).
[43] DeepSeek Chat (DeepSeek V3), https://www.deepseek.com, (accessed: February 2025)
[44] S. S. K. Dasa, G. Diakova, R. Suzuki, A. M. Mills, M. F. Gutknecht, A. L. Klibanov, J. K. Slack-Davis, K. A. Kelly, *Theranostics.* **2018**, *8*, 2782.
[45] G. Spitale, N. Biller-Andorno, F. Germani, *Sci Adv.* **2023**, *9*, adh1850.
[46] W. H. Walters, E. I. Wilder, *Sci. Rep.* **2023**, *13*, 14045.
[47] I. Shumailov, Z. Shumaylov, Y. Zhao, N. Papernot, R. Anderson, Y. Gal, *Nature.* **2024**, *631*, 755,
[48] P. Samuelson, *Science.* **2023**, *381*, 158.
[49] The R Project for Statistical Computing, https://www.R-project.org/, (accessed: June 2025).
[50] M. Kuhn, H. Wickham, Tidymodels: A Collection of Packages for Modeling and Machine Learning Using Tidyverse Principles v1.3.0, https://www.tidymodels.org, (accessed: June 2025).
[51] M. Kuhn, D. Vaughan. Parsnip: A Common API to Modeling and Analysis Functions v.1.2.1, https://CRAN.R-project.org/package=parsnip, (accessed: June 2025).
[52] T. Cheng, Y. Zhao, X. Li, F. Lin, Y. Xu, X. Zhang, Y. Li, R. Wang, L. Lai, *J. Chem. Inf. Model.* **2007**, *47*, 2140.
[53] P. Ertl, B. Rohde, P. Selzer, *J. Med. Chem.* **2000**, *43*, 3714.
[54] T. Hothorn, B. Lausen, A. Benner, M. Radespiel-Tröger, *Statist. Med.* **2004**, *23*, 77.
[55] M. Kuhn, baguette, Efficient Model Functions for Bagging v1.0.2, https://CRAN.R-project.org/package=baguette, (accessed: June 2025).
[56] J. H. Friedman, *Ann. Stat.* **1991**, *19*, 1.
[57] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, et al. Xgboost: Extreme Gradient Boosting v1.7.8.1, https://CRAN.R-project.org/package=xgboost, (accessed: 2024).
[58] K. Hechenbichler, K. Schliep, Collaborative Research Center 386, *Weighted k-Nearest-Neighbor Techniques and Ordinal Classification*, EconStor, Kiel, Germany **2004**.
[59] K. Schliep, K. Hechenbichler, Knn: Weighted K-Nearest Neighbors v1.3.1, https://CRAN.R-project.org/package=kknn, (accessed: June 2025).
[60] J. Friedman, R. Tibshirani, T. Hastie, *J. Stat. Softw.* **2010**, *33*, 1.
[61] R. Tibshirani, *J. Royal Stat. Soc. Ser. B: Stat. Methodol.* **1996**, *58*, 267.
[62] J. K. Tay, B. Narasimhan, T. Hastie, *J. Stat. Softw.* **2023**, *106*, 1.
[63] M. Kuhn, H. Frick, poissonreg: Model Wrappers for Poisson Regression. V1.0.1, https://CRAN.R-project.org/package=poissonreg, (accessed: June 2025).
[64] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, V. Vapnik, *Advances in Neural Information Processing Systems*, (Eds.: M. C. Mozer, M. Jordan, T. Petsche), MIT Press, Cambridge, Massachusetts **1997**.
[65] A. Karatzoglou, A. Smola, K. Hornik, A. Zeileis, *J. Stat. Softw.* **2004**, *11*, 1.
[66] A. Liaw, M. Wiener, *R. News* **2002**, *2*, 18.
[67] N. A. Diamantidis, D. Karlis, E. A. Giakoumakis, *Artificial Intelligence.* **2000**, *116*, 1.
[68] M. Mayer, shapviz: SHAP Visualizations v0.9.6, https://CRAN.R-project.org/package=shapviz, (accessed: June 2025).