Computational modeling and software design can assist with understanding the Ann Arbor 1,4-dioxane plume as well as creating tools capable of helping others in their sustainability efforts. Our computational modeling efforts are two pronged. Our first task was to design models capable of determining the biodegradability of a given compound. Our second task was to investigate the physical properties of water with real world concentrations of dissolved 1,4-dioxane.

Biodegradation Model

Motivation

Compounds used in industrial processes, like in the pharmaceutical industry, are often released into the ecosystem with disastrous consequences[1,2]. These incidents, accidental or otherwise, can carry risks to humans[1]. Therefore, it is worthwhile for entities to investigate if the compounds they utilize are biodegradable. Biodegradable compounds have the capability to be naturally degraded by common enzymes often found in the environment[2]. Biodegradable compounds represent a safer option for protecting communities[1].

There are a number of accepted tests for determining biodegradability. One of the most common of these is called the Modified MITI Test[2,3,4]. The MITI test uses compounds in aqueous bacterial stock and tests the oxygen demand over a period of 28 days[17]. Due to the time constraint nature of the test as well as other issues, high throughput testing is difficult. Therefore, using machine learning (ML) to reduce the number of samples needed to be tested could be a helpful technique[3].

Previous Research

ML approaches have been explored for biodegradability testing. Most of the previous research has been traditional in nature, i.e. not involving deep learning. However, a few instances of deep techniques have been implemented. Through these approaches, different feature generation methods have been explored as well as model architectures. The best current known models are Quantitative Structure-Activity Relationship (QSAR) models, the two most common techniques being Support Vector Machines (SVMs) or Graph Convolutional Networks (GCNs). However, there have been many attempts to model biodegradation which we notice seem to have a natural performance limit around improving the sensitivity.

Paper Name	**Year**	**Accuracy**	**AUC**	**Data source**	**Notes**
Concomitant prediction of environmental fate and toxicity of chemical compounds[1]	2020	Regressing Model	.879	NITE	-
Modeling the Biodegradability of Chemical Compounds Using the Online CHEmical Modeling Environment (OCHEM)[2]	2014	.876	-	NITE + University Dataset	Required refinement of structures in preprocessing

Multimodal Deep Neural Networks using Both Engineered and Learned Representations for Biodegradability Prediction[3]	2018	.875	-	MITI Test (NITE)	Combines a neural network with feature based approach
Modeling of ready biodegradability based on combined public and industrial data sources[4]	2019	.750	-	NITE, ECHA, VEGA, EPI Suite, OPERA, Solvay	Had access to large database compared to other papers
A Comparative Study of the Performance for Predicting Biodegradability Classification: The Quantitative Structure–Activity Relationship Model vs the Graph Convolutional Network[19]	2022	0.84	-	ECHA, NITE, VEGA, EPI Suite, OPERA	Tested a vast range of models including kNNs, SVMs, RFs, GB, and GCNs
Prediction of biodegradability from chemical structure: Modeling of ready biodegradation test data[16]	2009	0.83	-	NITE (MITI Test),	Studied fragments of the chemicals and their effects on the biodegradability
Development of models predicting biodegradation rate rating with multiple linear regression and support vector machine algorithms[20]	2020	0.77	-	NITE (MITI Test),	-
Biodegradability Prediction of Fragrant Molecules by Molecular Topology[14]	2016	0.808	-	Boethling	-

Machine Learning

1. **Dataset**

Data provides an incredibly important contribution when designing high fidelity models. We believe that the data published by Lunghini et. al. is the most thorough and well processed available biodegradability dataset[4]. They aggregate data from the ECHA database, NITE, VEGA, EPI Suite, and OPERA for public availability. Overall this data contains 2830 samples with 1097 being biodegradable and 1733 being

non-biodegradable[4]. We removed five molecules due to processing errors. We further split this into 1956, 559, 280 samples for training, testing, and validation purposes respectively (roughly a 70/20/10 split).

2. Machine learning and Software

We sought to explore a number of ML and artificial intelligence (AI) paradigms in order to thoroughly explore possible solutions. Our first challenge was representing molecules to our ML models. Molecules are real-world objects with properties and structural relationships that are not easily described with numbers. Additionally, ML models are generally strict such that the input they require is a vector of numbers which often correlate to real world properties. To accomplish this stringent task, many software packages have been developed to sufficiently represent molecules.

These softwares, which include DRAGON, Mordred, RDkit Descriptors, etc., generate large numbers of features to describe molecules[18]. Features quantifying or describing structural or chemical components of molecules can be represented in tabular formats with a known dimensionality. Due to the shape of the data being both universal and known, models can be more easily developed around these data. To that end, we implement feature generation with Mordred, which creates 1040 unique descriptors per molecule. We explored Random Forests (RFs), XGBoost, and Naive Bayes models. Additionally, we explored the application of Graph Neural Networks (GNNs) as they do not require strict input dimensions, which will be elaborated on later.

3. Tree based models

Both RF and XGBoost models are tree based. This means they use a number of decision trees to drive their predictions. A decision tree splits the dataset at random intervals (nodes), and as a sample meets a number of criteria, the model is able to generate a class prediction[5]. The predictive power of one decision tree is not great, which is where RF and XGBoost become helpful. RF models average a large number of independent trees, this is called bagging[6]. XGBoost implements boosting, which is where trees are iteratively designed to correct the errors from previous trees[6].

Both RF and XGBoost models are considered ensemble models, a class of models which combines the predictions of multiple constituent models. In general ensemble models are robust. However, we also explored the effects of intelligent feature selection to see if we could increase performance by presenting the models with the most relevant features. To this end we investigated three feature selection techniques. Mutual information characterizes how the knowledge of one variable reduces the uncertainty in the item characterization[7]. Fisher Score attempts to quantify how well a certain feature is able to separate the class of all the samples in a dataset[8]. Finally, we also investigated the impact of reducing the number of highly correlated features through multicollinearity analysis in an attempt to reduce redundant features[9].

Below we present the results of both RF Models:

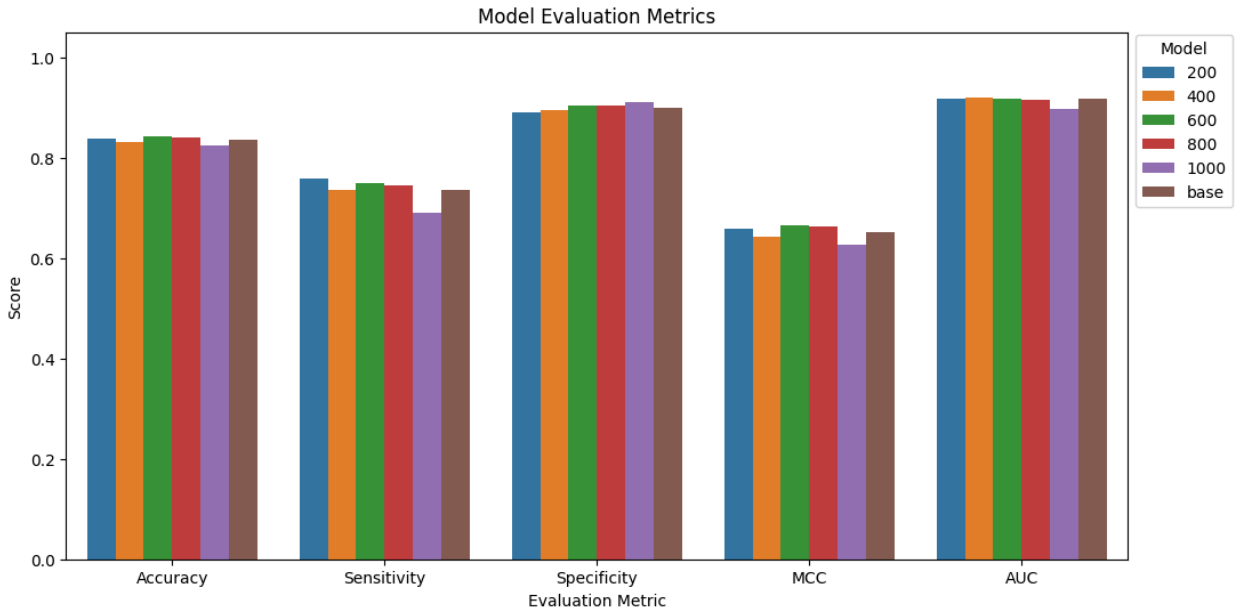

Fig 1. Performance of RF models with differing levels of Fisher Score cut-off. Metrics measured include accuracy, sensitivity, specificity, Matthews Correlation Coefficient (MCC), and area under the curve (AUC). Fisher Score cut-offs were set to 200, 400, 600, 800, and 1000.

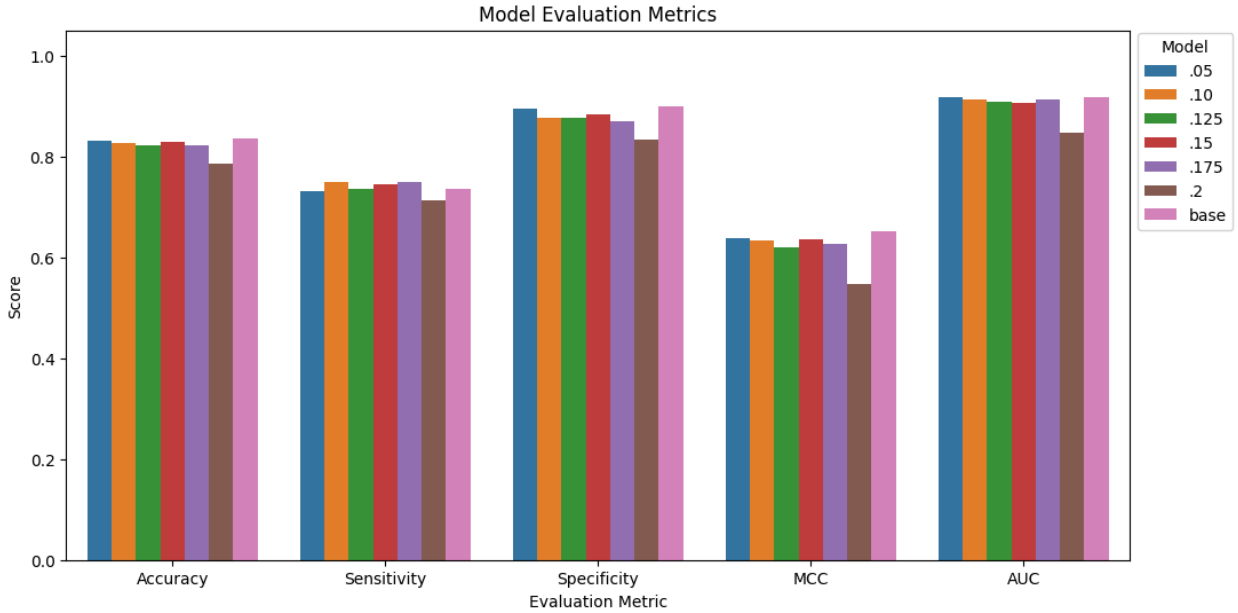

Fig 2. Performance of RF models with differing levels of Information Gain cut-off. Metrics measured include accuracy, sensitivity, specificity, MCC, and AUC. Information Gain cut-offs were set to 0.05, 0.10, 0.125, 0.15, 0.175, and 0.20.

The XGBoost Model results are presented now:

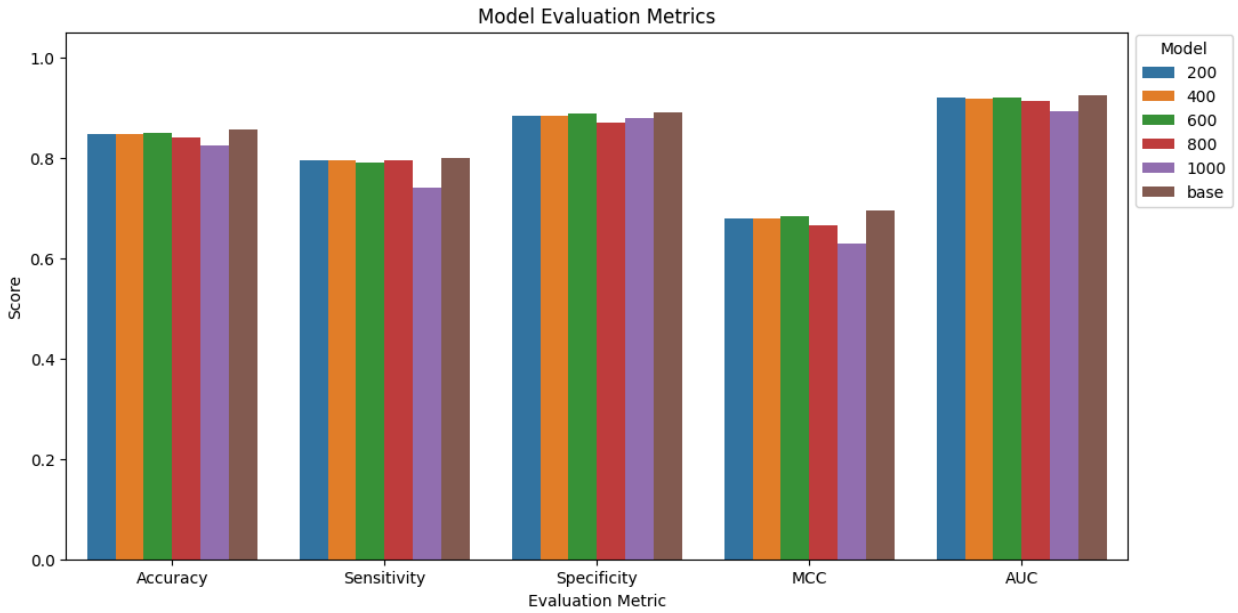

Fig 3. Performance of XGBoost models at varying levels of Fisher Score cut-off. Metrics measured include accuracy, sensitivity, specificity, MCC, AUC. Fisher Score cut-offs were set to 200, 400, 600, 800, and 1000.

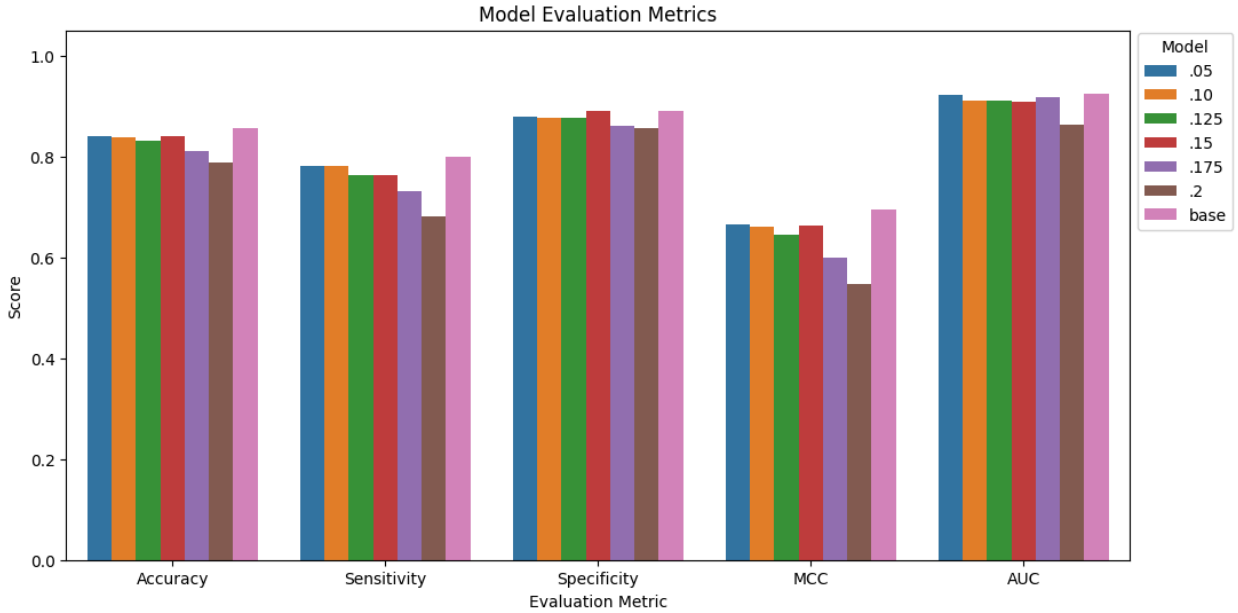

Fig 4. Performance of XGBoost at varying levels of Information Gain. Metrics measured include accuracy, sensitivity, specificity, MCC, AUC. Information Gain cutoffs were set to .05, .10, .125, .15, .175, .2.

4. Naive Bayes and Bayesian Averaging

The second broad class of models we investigated were Naive Bayes models. Naive Bayes models are predicated on the famous Bayes Theorem, which provides a convenient framework for evaluating

conditional probabilities. Naive Bayes models make assumptions that present critical challenges to the proper application of the model. First, Naive Bayes assumes that features are independent of each other - this is often not true and presents an intrinsic limitation for performance. Secondly, Naive Bayes models calculate probabilities based on a probability distribution manually specified. These distinct model classes include Gaussian Naive Bayes, Categorical Naive Bayes, and Bernoulli Naive Bayes based on their respective distribution titles[15].

We chose to implement three Naive Bayes models. First, we implemented a Gaussian Naive Bayes across the entire dataset. We believe this represents our most naive modeling approach as it makes no attempt to intelligently select features or model types. Secondly we chose to design and implement a custom Naive Bayes architecture capable of combining multiple Naive Bayes models using unique assumed probability distributions. To do this, we had to narrow the number of features used, which will be explained more thoroughly in the next paragraph. Our third model was a Gaussian Naive Bayes implemented across the same features as the custom Naive Bayes for a more fair comparison.

To select features for custom Naive Bayes architecture, we designed a series of tests to determine if a distribution was binary, categorical, or within a reasonable margin of being Gaussian. Specifically for Gaussian distribution, we used a Shapiro-Wilk test, which technically can only disprove data being drawn from a normal distribution. However, for modeling purposes we set the P-value threshold to 0.2. We believe that if data was greater than that threshold it could be modeled somewhat-confidently with a Gaussian distribution. We were able to identify 49 Bernoulli features, 90 categorical features, and 0 Gaussian features. For combining model classes, we used Bayesian Averaging, a popular and simplistic technique to combine the outputs of multiple bayesian classifiers. The simplicity of Bayesian Averaging lies in its definition as a linear combination of the outputs of multiple naive Bayes models, weighted by their respective posterior possibilities[15].

$$\widehat{Y}^{*} = \sum_{j=1}^{n} Y_j^{*} p\left(M_j | \text{data}\right)$$

Fig 5. Equation for Bayesian model averaging. Y^* denotes the probability of the next prediction; Y^*_j denotes the probability of the next prediction under model M^j; $p(M_j|data)$ denotes the posterior probability of the model M_j. The posterior probability, in turn, can be calculated via the following equation.

$$p\left(M_m \mid \text{data}\right) = \frac{p\left(\text{data} \mid M_m\right) p(M_m)}{\sum_{j=1}^{n} p\left(\text{data} \mid M_j\right) p(M_j)}$$

Fig 6. Equation for model posterior probability. $P(M_m|data)$ denotes the posterior probability of model M_m; $p(data|M_m)$ denotes the marginal likelihood of model M_m; and $p(M_m)$ denotes the prior probability of model M_m.

Finally, it is worth noting that while we compared models across a variety of metrics, no truly fair comparison could be achieved. Categorical Naive Bayes are unable to make predictions on classes which they were not exposed to during the training phase. Therefore, three samples of the testing data had to be excluded from the testing set due to introducing previously unseen classes. While we don't believe this severely modifies the testing metrics, it is a limitation.

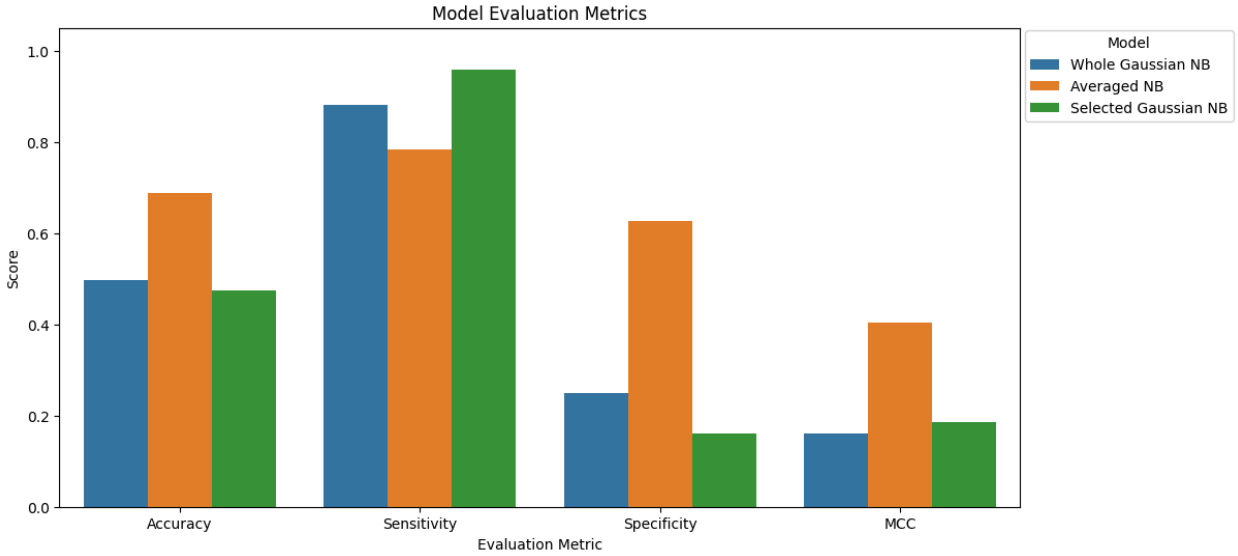

Fig 7. Results of the three implemented Naive Bayes models. Metrics measured include accuracy, sensitivity, sensitivity, specificity, and MCC. Whole Gaussian NB was trained across the entire dataset, Averaged and Selected Gaussian NB were trained and tested on characterized features.

5. Deep learning and Graph Neural Networks

An alternative approach to modeling can be done through deep learning. While many deep learning models such as multi-layer perceptrons (MLPs) and convolutional neural networks require geometrically constrained data, GNNs allow for graphs of any shape to be processed[10]. Graphs are a mathematical structure represented by a number of nodes and edges. Naturally molecules can be represented through these structures where atoms are interpreted as nodes and bonds as edges. Furthermore, a feature vector can be associated with each node[10]. This framework suggests a very natural way to do computational work with molecules as their true structure (or something much closer to) can be represented to the model.

We tried a variety of graph neural networks and feature generators. Ultimately, we landed on the combination of a graph convolution model with features generated by ConvMolFeaturizer, which generates a feature for each atom[12,18]. Both of these classes were provided by the Deepchem package[12,28].

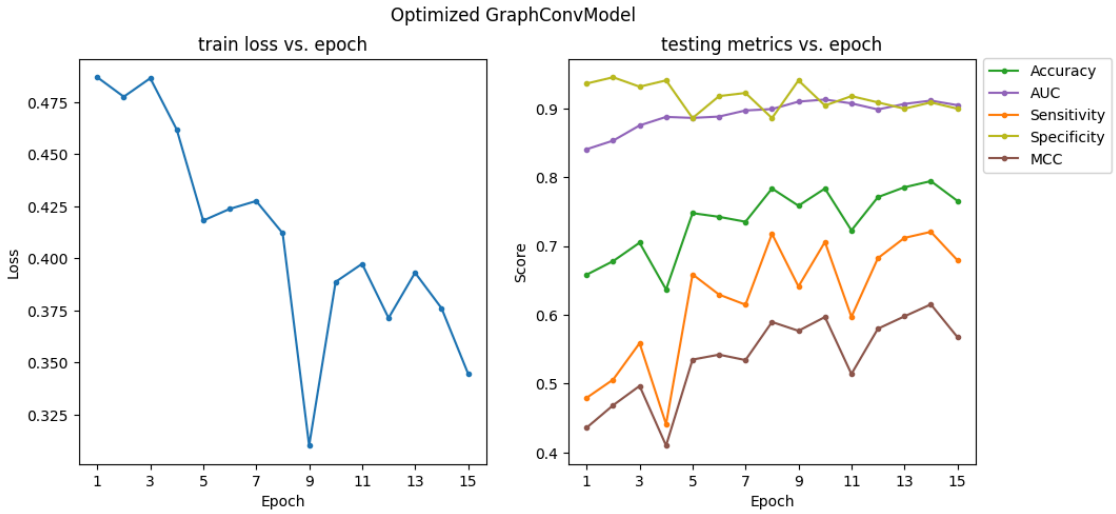

Fig 8. Performance of optimized Graph Convolutional Model trained solely on the biodegradability data across 15 epochs of training. Metrics measured include accuracy, AUC, sensitivity, specificity, and MCC.

Often in machine learning the technique of transfer learning is used to increase performance. While this is most often done in the area of computer vision, we also explored its application in biodegradability. Essentially, transfer learning works by training a model on tangentially related data with more samples which gives the model increased inferencing abilities when applied to the data of interest. To make the model predict on the key data, the model must be fine-tuned with a subset of the actual data[13].

To accomplish this for biodegradability, we trained a GraphConvModel using Tox21 data, a dataset which classifies molecules as toxic or non-toxic[28]. After training, we fixed the GNN weights and fed the intermediate vector into a MLP which we fine-tuned on the biodegradability data.

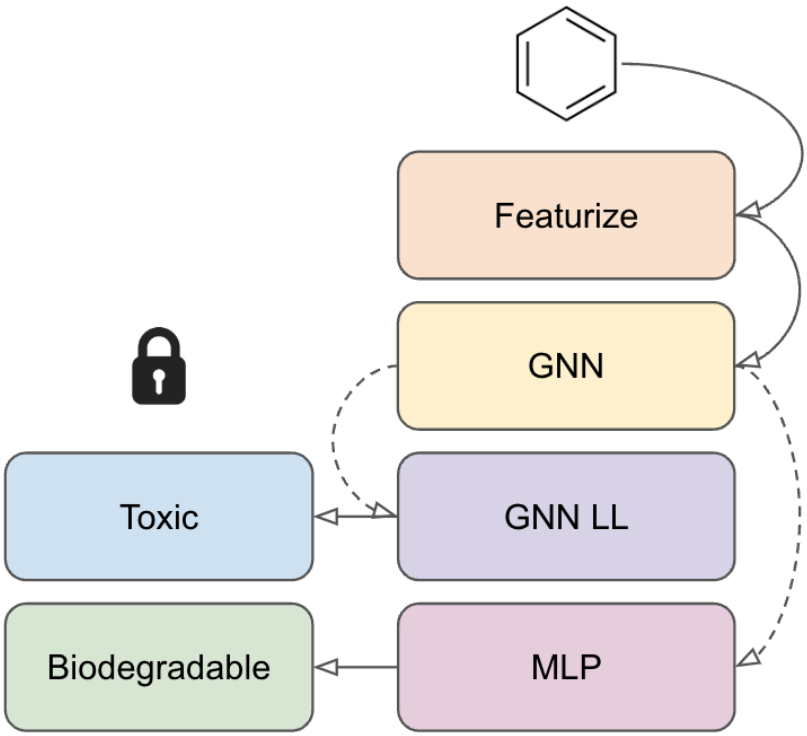

Fig 9. Diagram of transfer learning pipeline. During training, molecules are featurized and then toxicity is predicted using the GNN. For biodegradability, the GNN is frozen and feature embeddings are then fed into a MLP for training and then prediction once training is complete.

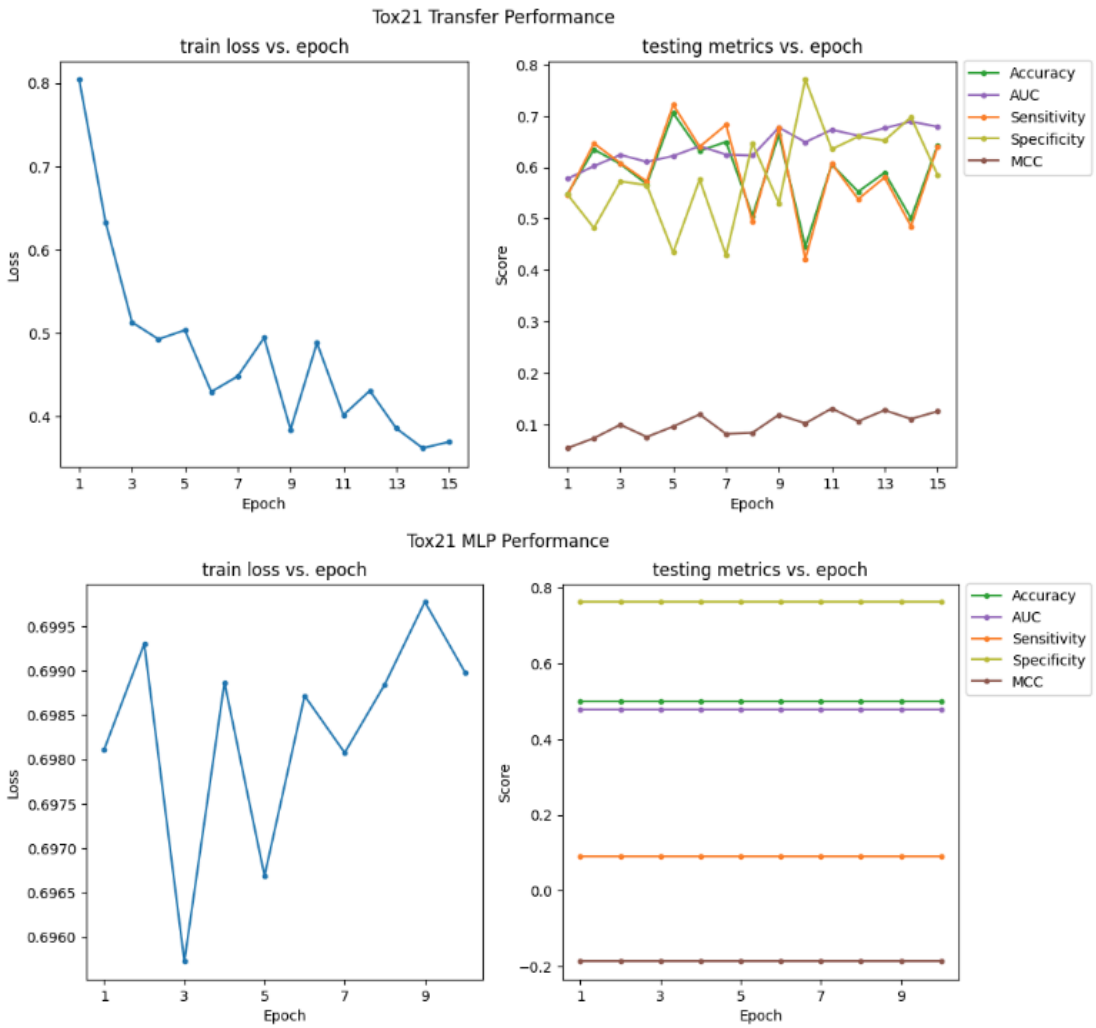

Fig 10. Performance of Graph Convolutional Model on Tox21 data as well as the performance of the MLP used for transfer learning applied to biodegradability data. Metrics measured include accuracy, AUC, sensitivity, specificity, and MCC.

Conclusion/Discussion

These experiments reveal a few trends in machine learning. Generally, we see robust ensemble models (XGBoost and RF) outperform the rest. Even when performing feature selection, base models tend to perform on par or better than models trained with selected data. This could be due to a number of reasons. First, the decision tree basis of these models inherently makes the models consider the conditional set of features of a molecule. Individual features may not have very high predictive power, but a conditional sequence of features might. Additionally, because there are many voters in ensemble modeling, these models tend to be robust in our experience and show higher immunity to variation in the data. Finally, we note that XGBoost outperforms RF and all other model classes. We believe that this is likely due to the boosted nature of XGBoost. As noted in previous sections, the dataset is skewed. By iteratively correcting for errors, the XGBoost may be less likely to vote naively after training.

Our Naive Bayes models performed relatively poorly, however it is clear that the method of Bayesian averaging boosts performance substantially. One reason for the less impressive performance of the averaged Naive Bayes model is due to the massive reduction in the number of features utilized. Because most features' distributions could not be characterized, the averaged model used roughly an order of magnitude less features than other model classes. This could hamper overall performance.

Unfortunately, our GNN models underperformed from what we were expecting. For the non-transfer learning approach, we did see learning occur and general convergence of performance after about 10 epochs of training. However, the model seems to be sensitive to epochs afterwards and does not continue increasing performance metrics convincingly. This may be due to the lack of a larger dataset, variability in the dataset, or simply the network may have had trouble making sense of the feature vectors initially supplied. One of the downsides of deep-learning is that models often act as a black-box: we cannot know/make sense of what happens internally. Transfer learning performed worse. While the GNN exhibited convergence on the Tox21 dataset, the MLP performed about randomly on the biodegradability dataset. This could be due to lack of overlap between the datasets, not powerful enough feature representation being achieved on Tox21, or improper transfer scheme. While the transfer performance was underwhelming it is not surprising as the GNN performance on the biodegradability dataset was not especially effective since our GNN did not perform especially when on the Tox21 dataset.

Overall, we believe that the base XGBoost model is our overall best. On the testing set data, we achieve the following performance: Accuracy : 0.855, Specificity : 0.891, Sensitivity : 0.799, AUC : 0.925, MCC Score : 0.694. To further verify performance, we applied this model to the validation dataset and achieved the following performance: Accuracy: 0.875, Specificity: 0.918, Sensitivity: 0.809, AUC: 0.934, and MCC Score: 0.736. Availability of our model is through a reproducible training pipeline in the Gitlab repository, which allows anyone to recreate our best performing model. This is available to through download and can be used to generate predictions on molecules characterized by the same mordred features we use.

Molecular Dynamic Simulation of 1,4-Dioxane Dissolved Water

In parallel with machine learning-based models to study potential biodegradability of compounds, our team also aimed to study any effect that 1,4-dioxane in our problem scope may affect the viscosity of water, thereby influencing the function of our proposed bioreactor. We hypothesized that the current highest concentration of 1,4-dioxane in water should not have an effect on the macroscopic physical properties of water. For this application, we used LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator), a versatile and widely used molecular dynamics simulation software designed to model particles in a wide range of materials capable of simulating complex systems with millions of particles. Additionally, LAMMPS supports a wide variety of force fields and potential models, making it flexible and adaptable to a wide range of scientific research applications[21].
Current steps:

First, we used the atb model builder to generate a structure for 1,4-dioxane that can be used for LAMMPS. Then, force field template (GROMOS 54) and molecule data file containing key information (charges, bond, pair, etc.) were extracted from atb into a Linux environment. Moltemplate, a LAMMPS-associated software package, was used to generate structure files in a LAMMPS-readable

format for 1,4-dioxane. SPCE water data file was then added to generate a water box of 2.5 million H2O molecules for 1 1,4-dioxane molecule to mirror the 1900 ppb of 1,4-dioxane in water (highest recorded concentration in Ann Arbor). Larger systems might be more accurate, though we expected minimal differences in the macro properties between water and water containing 1,4-dioxane of this concentration. Note that there is likely overlap, and moltemplate doesn't seem to have a function to check for molecule overlap/energy errors. This will need to be manually corrected in LAMMPS later on. A way to avert this issue is using the 1,4-dioxane and H2O mol files and randomly generate designated numbers of each molecule in a simulation box in LAMMPS - however, this requires significantly more computational resources. After this step, Moltemplate will produce several outputs, among which the system.data file and the parm.lammps files will be used as input for LAMMPS since they contain the simulation setup and parameters for corresponding atoms, respectively.

In LAMMPS, group H2O and 1,4-dioxane based on their assigned atom numbers from moltemplate. Delete H2O molecules that overlap with 1,4-dioxane. The setup is now ready for minimization and viscosity simulation. The method we are using is the periodic perturbation method, the primary built in viscosity calculation mode in LAMMPS. It works by measuring the momentum flux in response to an applied velocity gradient, it measures the velocity profile in response to applied stress. A cosine-shaped periodic acceleration is added to the system via the fix accelerate/cos command, and the compute viscosity/cos command is used to monitor the generated velocity profile and remove the velocity bias before thermostatting. The reciprocal of eta (viscosity) is computed within the script, and printed out as v_invVis in thermo_style command. Then eta is obtained from the reciprocal of time average of v_invVis to be about 0.75 eta, which is the typical value of water simulation at this time point. This value is expected to reach 1 at several hundred picoseconds as is the case for water viscosity simulation using this method, which can be verified using devices with higher computational power as the wall time for a system of this size is significant on personal computers.

Bibliography

1. Garcia-Martin, J. A., Chavarría, M., de Lorenzo, V., & Pazos, F. (2020). Concomitant prediction of environmental fate and toxicity of chemical compounds. Biology Methods and Protocols, 5(1). https://doi.org/10.1093/biomethods/bpaa025
2. Vorberg, S., & Tetko, I. V. (2013). Modeling the biodegradability of chemical compounds using the online chemical modeling environment (OCHEM). Molecular Informatics, 33(1), 73–85. https://doi.org/10.1002/minf.201300030
3. Goh, G. B., Sakloth, K., Siegel, C., Vishnu, A., & Pfaendtner, J. (2018). Multimodal deep neural networks using both engineered and learned representations for biodegradability prediction. arXiv preprint arXiv:1808.04456.
4. Lunghini, F., Marcou, G., Gantzer, P., Azam, P., Horvath, D., Van Miert, E., & Varnek, A. (2019). Modelling of ready biodegradability based on combined public and industrial data sources. SAR and QSAR in Environmental Research, 31(3), 171–186. https://doi.org/10.1080/1062936x.2019.1697360
5. Song, Y. Y., & Ying, L. U. (2015). Decision tree methods: applications for classification and prediction. Shanghai archives of psychiatry, 27(2), 130.
6. Shebl, A., Abriha, D., Fahil, A. S., El-Dokouny, H. A., Elrasheed, A. A., & Csámer, Á. (2023). Prisma hyperspectral data for lithological mapping in the Egyptian eastern desert: Evaluating the support Vector Machine, Random Forest, and XG Boost Machine Learning Algorithms. Ore Geology Reviews, 161, 105652. https://doi.org/10.1016/j.oregeorev.2023.105652
7. Beraha, M., Metelli, A. M., Papini, M., Tirinzoni, A., & Restelli, M. (2019, July). Feature selection via mutual information: New theoretical insights. In 2019 international joint conference on neural networks (IJCNN) (pp. 1-9). IEEE.
8. Gu, Q., Li, Z., & Han, J. (2012). Generalized fisher score for feature selection. arXiv preprint arXiv:1202.3725.
9. Tsangaratos, P., & Ilia, I. (2016). Comparison of a logistic regression and Naïve Bayes classifier in landslide susceptibility assessments: The influence of models complexity and training dataset size. Catena, 145, 164-179.
10. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., ... & Sun, M. (2020). Graph neural networks: A review of methods and applications. AI open, 1, 57-81.
11. Lee, M., & Min, K. (2022). A comparative study of the performance for predicting biodegradability classification: the quantitative structure–activity relationship model vs the graph convolutional network. ACS omega, 7(4), 3649-3655.
12. Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., & Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. Advances in neural information processing systems, 28.
13. Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. Journal of Big data, 3, 1-40.
14. Vincent, B., Jesús, G. S., María, G. L., Jorge, G., & Ramón, G. D. (2016). Biodegradability Prediction of Fragrant Molecules by Molecular Topology.
15. Clyde, M., Çetinkaya-Rundel, M., Rundel, C., Banks, D., Chai, C., & Huang, L. (2022a). An Introduction to Bayesian Thinking. Github.

16. Loonen, H., Lindgren, F., Hansen, B., Karcher, W., Niemelä, J., Hiromatsu, K., ... & Struijś, J. (1999). Prediction of biodegradability from chemical structure: modeling of ready biodegradation test data. Environmental Toxicology and Chemistry: An International Journal, 18(8), 1763-1768.
17. Hydrotox - Labor für Ökotoxikologie und Gewässerschutz GmbH. (n.d.). Modified miti (II) test. Hydrotox. https://www.hydrotox.de/en/services/laboratory-services/biological-degradation/inherent-biodegradability/modified-miti-ii-test.html#:~:text=The%20inherent%20biodegradability%20is%20measured,the%20testing%20of%20insoluble%20chemicals
18. Moriwaki, H., Tian, Y. S., Kawashita, N., & Takagi, T. (2018). Mordred: a molecular descriptor calculator. Journal of cheminformatics, 10, 1-14.
19. Lee, M., & Min, K. (2022). A comparative study of the performance for predicting biodegradability classification: the quantitative structure–activity relationship model vs the graph convolutional network. ACS omega, 7(4), 3649-3655.
20. Tang, W., Li, Y., Yu, Y., Wang, Z., Xu, T., Chen, J., ... & Li, X. (2020). Development of models predicting biodegradation rate rating with multiple linear regression and support vector machine algorithms. Chemosphere, 253, 126666.
21. LAMMPS - A flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, Comp. Phys. Comm. 271, 108171 (2022).
22. Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.
23. Wes McKinney (2010). Data Structures for Statistical Computing in Python . In Proceedings of the 9th Python in Science Conference (pp. 51 - 56).
24. Hunter, J. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95.
25. Michael L. Waskom (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021.
26. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
27. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32.
28. Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, & Zhenqin Wu (2019). Deep Learning for the Life Sciences. O'Reilly Media.