November 2023

Machine Learning Modeling of Polymer Coating Formulations: Benchmark of Feature Representation Schemes

Nelson I. Evbarunegbe
University of Massachusetts Amherst

MACHINE LEARNING MODELING OF POLYMER COATING FORMULATIONS: BENCHMARK OF FEATURE REPRESENTATION SCHEMES

A Thesis Presented by

NELSON EVBARUNEGBE

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements of the degree of

MASTER OF SCIENCE IN CHEMICAL ENGINEERING

September 2023

Department of Chemical Engineering

MACHINE LEARNING MODELING OF POLYMER COATING FORMULATIONS: BENCHMARK OF FEATURE REPRESENTATION SCHEMES

A Thesis Presented By

NELSON EVBARUNEGBE

Approved as to style and content by:

_____

Peng Bai, Chair

_____

Dimitrios Maroudas, Member

_____

Hui Guan, Member

_____

Russell Tessier, Acting Department Head

Department of Chemical Engineering

ACKNOWLEDGMENTS

Special thanks to everyone who contributed to this work both financially and emotionally, especially my loving parents, Engr. and Mrs. Thomas and Patience Evbarunegbe who encouraged me to come this far in my academic pursuit. Special thanks also to my advisor, Peng Bai, who made it possible for me to embark on this exciting project that created a significant landmark in my career progression.

Huge thanks also to the Bai group for their added support that helped me scale through the first stage of graduate school, most especially Anne Le, Joseph Tapia, Yachan Liu, Shubham Malviya, Namrata Masli, Jiacheng Wang and Harshavardhan Chilukuri.

I would also love to extend my gratitude to my committee members for supporting this exciting work and taking the time to be a part of this current milestone position of my life.

ABSTRACT

MACHINE LEARNING MODELING OF POLYMER COATING FORMULATIONS:

BENCHMARK OF FEATURE REPRESENTATION SCHEMES

SEPTEMBER 2023

NELSON EVBARUNEGBE

B.ENG.Ch.E, UNIVERSITY OF BENIN

M.S.Ch.E, UNIVERSITY OF MASSACHUSETTS, AMHERST

Directed by: Professor Peng Bai

Polymer coatings offer a wide range of benefits across various industries, playing a crucial role in product protection and extension of shelf life. However, formulating them can be a non-trivial task given the multitude of variables and factors involved in the production process, rendering it a complex, high-dimensional problem. To tackle this problem, machine learning (ML) has emerged as a promising tool, showing considerable potential in enhancing various polymer and chemistry-based applications, particularly those dealing with high dimensional complexities.

Our research aims to develop a physics-guided ML approach to facilitate the formulations of polymer coatings. As the first step, this project focuses on finding machine-readable feature representation techniques most suitable for encoding formulation ingredients. Utilizing two polymer-informatics datasets, one encompassing a large set of 700,000 common homopolymers including epoxies and polyurethanes as coating base materials while the other a relatively small set of 1000 data points of epoxy-diluent formulations, four featurization schemes to represent polymer coating molecules were benchmarked. They include the molecular access system, the extended connectivity fingerprint, molecular graph-based chemical graph network, and graph convolutional network (MG-GCN) embeddings. These representation schemes were used with

ensemble models to predict molecular properties including topological surface area and viscosity. The results show that the combination of MG-GCN and ensemble models such as the extreme boosting machine and random forest models achieved the best overall performance, with coefficient of determination (r^2) values of 0.74 in topological surface area and 0.84 in viscosity, which compare favorably with existing techniques. These results lay the foundation for using ML with physical modeling to expedite the development of polymer coating formulations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML – Machine Learning

MG – Molecular Graph

CGN – Chemical Graph Network

GCN – Graph Convolutional Network

MLP – Multilayer Perceptron

ECFP – Extended Connectivity Fingerprint

MACCS – Molecular Access System

SMILES – Simplified Molecular-Input Line-Entry System

XGBoost – Extreme Gradient Boosting

RF – Random Forest

VR – Voting Regressor

TPSA – Topological Polar Surface Area

MAE – Mean Average Error

MSE – Mean Squared Error

r^2 – Coefficient of Determination

HS – Hard Segment

SS – Soft Segment

df – DataFrame

CHAPTER 1

INTRODUCTION

Background and Motivation

Polymer coatings are widely used in various sectors including the food industry, pipeline coating for natural gas transportation, and container manufacturing. They function as a barrier layer to prevent the intrusion of environmental elements and contact with the substrate, which can be in the form of polymers or metal enclosing, thus preserving the content of the container they are embedded in.

Without good coatings, contact with the substrate can lead to undesirable contaminants being introduced into the product content, and in the case of acidic product content, it can also result in the erosion of the substrate. As opposed to free-standing polymer coatings, the substrate materials are also important as they hold these polymers in place, as well as proper ease of transportation. The performance of a coating is greatly dependent on the type of substrate. For example, a coating may possess good adhesion on wood but not on metals. This work focuses on the polymer coating on metallic and polymeric substrate applications but does not necessarily address the additional complexity from the type of substrate in which the coating is being applied.

These coatings play a pivotal role but formulating them can be challenging. Conventionally, coatings are synthesized from fossil-based polymers, but their usage raises concerns about sustainability and the finite nature of fossil resources. In response to these challenges, there is a growing interest in exploring bio-based alternatives, such as furan-based polyurethanes, which offer more sustainable, renewable, and environmentally friendly options for the future. However, it becomes imperative to intricately integrate these bio-based alternatives into formulations alongside other materials to make them comparable to fossil-based polymers. This offers the added

advantage of injecting additional properties into the coating that make up for deficiencies that may exist in the base ingredient(s). This may include better adhesive strength to the substrate material, lesser permeability of content through the coatings, and resistance to temperature and pressure, amongst others.

In efforts to achieve this, traditional experimental techniques are used to determine the best combination of ingredient materials for different curing conditions. However, a significant challenge arises due to the vast number of variables to be tested, resulting in a high-dimensional problem. As a consequence, the formulation process becomes Edisonian and resource-intensive. Another technique is the physics-based approach. They are however unable to handle this problem, as predicting the final materials property given just the input formulation is currently not feasible. For example, the polymerization stage of preparing these polymer coatings could require density functional theory calculations, the cross-linking stage may be required to be modeled by kinetic Monte Carlo, and the morphology evolution stage is a mesoscale process usually modeled by molecular or coarse-grained simulations. All these expensive and time-consuming physics-based calculations make it a multi-scale multi-physics problem.

Machine learning (ML) offers a potential solution to this problem through quantitative structure-property relationships as it can bridge the wide space between input formulations, the processing conditions involved, and the final property of the complete material. Furthermore, once trained on a related system, they may be faster and less resource-intensive to apply for a new application and hence can assist the experimental formulation efforts.

The drawback of this data-driven approach, however, lies in the scarcity of available data, causing it to be a challenge in finding the specific required information to train these ML models. Hence, it is noteworthy that this thesis represents the initial stage of a larger project, wherein the

ultimate objective is to develop a physics-guided ML approach that assists in facilitating the formulations of polymer coatings. This can be further approached by setting physics constraints to the ML framework and hence ensuring ML experiments are tuned to obey scientific bounds in their prediction.[1] This approach, in combination with an expert technique such as active learning which helps to iteratively guide experiments in the desired direction, can be potentially utilized in generating accurate ML predictions with small datasets. It is also important to note that the physics-based-guided calculations involved in this overall project would be the second stage, hence no physics-based calculations were carried out in this thesis.

As the first step, the objective of this thesis is to identify the most appropriate feature representation methods and pre-train them using the available data. This pre-training is crucial to enable easy transferability to new applications where experimental data are being collected.

The existing data, however, is sparse, further necessitating a representation that can effectively reveal important details about the starting materials. This information is crucial for the successful pre-training of ML models. Moreover, we face the challenge of exploring the high-dimensional space of variables and conditions (which we collectively classify as features) that influence polymer properties. Streamlining this space is essential to improve the accuracy of predictions. Therefore, we seek suitable feature representations that can capture the most information for making precise predictions while working with a limited dataset space.

This thesis aims to select a feature representation that best fits the polymer coating formulation problem. Additionally, we will utilize a downstream ensemble model that excels in handling small datasets, resulting in a well-generalized algorithm with high adaptability.

Research Objective

The goal of this study is to investigate the most suitable featurization scheme – ML ensemble model combination that best generalizes on a polymer coating formulation problem in contributing information to the prediction of the formulation properties. This algorithm should also be able to be transferred and generalized to new similar polymer coating instances.

Existing methods of featurization particularly performed on polymer coating formulations, the target application for this study, are usually too high dimensional (generating too many features that could potentially exceed the number of training samples), what most researchers would term 'the curse of dimensionality', further leading to overfitting to the training dataset. Hence, our work tends to explore this by benchmarking a suitable featurization scheme that produces low dimensional learned embeddings and still reserves comparable predictability among other featurization schemes, using two distinct polymer informatics datasets to achieve this.

Research Hypothesis

Among our selected fingerprint and featurization schemes, we hypothesize that the molecular graphs, in combination with the graph convolutional networks (GCNs), are currently the best representation and model for representing polymer coating information and should have comparable or better performances with current state-of-the-art chemical feature representations. We also hypothesize that due to the excessive requirement of data from the graph model, ensemble models can help reduce chances of overfitting, hence using the graph model as a feature representation-learning algorithm for the ensemble models would not only help to reduce the overfitting problem but ensure advanced generalizability to new instances due to the combination of both powerful models i.e., a graph-based neural network and the ensemble model. This presents to us the potential to explore the high dimensional space of the independent variables that affect

the target variable (i.e., property of the polymer to be determined) with only fewer features (embeddings) learned from the graph models.

CHAPTER 2

LITERATURE REVIEW

2.1. Polymer Coatings

Polymer coatings have seen applications in various industries. Various kinds of polymer coatings have emerged and have presented multiple properties that are paramount in the protection and safer transportation of fluids, and the prevention of contact from the substrate enclosing the product. These coatings are usually prepared by a process known as curing. According to the definition from CorrosionPedia (2019), curing in a chemical system is known as a process that occurs during a chemical reaction, where heat is being used to catalyze or initiate chemical and molecular-level structural changes within a polymeric material.[2] In this case, the cured polymer coating becomes a final hardened material and hence possesses a more stable linkage to be applied as a coating.

This study focuses on inner polymer coatings (coatings between the substrate and the product itself). Prominent polymer coating base materials that have been identified include polyurethanes and epoxies.[3,4] Polyurethanes are made from the reaction of isocyanates and polyols. Their efficacy as barrier coatings has shown tremendous progress in various coating applications. However, they are reported to not perform well at high temperatures, most especially when not properly cured.[5,6]

Developing polyurethane polymers most times also includes the addition of a chain extender, usually a shorter chain diol. The chain extender with the isocyanate together forms the hard segment (HS) of the formulation, while the polyol (also known as macro-diol) serves as the soft segment (SS). The hard segment, embedded within the soft segment and supplements elasticity, contributes to the tensile strength and hardness of the final polyurethane coating.

According to the works of Pugar et al. (2020), they developed a machine-learning model that predicts the glass transition temperature for linear polyurethanes.[7] They reported that the hard segment contributed to the glass transition temperature prediction the most from their training and this is intuitive as most of the final properties of these polymer materials are related to the glass transition temperature, which from this study, the HS is highly related with.

As proof of concept, epoxy-based coatings will be used in this study. Epoxy coatings are usually made from epoxy resin, a diluent, and additives including a hardener. These types of coatings have been known to possess high moduli and strength which are important features desired for a finish coating material.[4, 8]

As introduced earlier, the cured epoxy coating is highly driven by its formulations. For example, If the curing agent component in the formulation is too low, curing may not properly occur, and the final material may not possess good tensile strength, amongst other mechanical properties that may also be marred by inadequate curing. On the other hand, if the curing agent is too high, the formulation production would be difficult to control, hence likely leading to undesirable results.[4] Hence, there exists the need to find an optimal combination of formulation ingredients and conditions, which all together yield the desired outcome formulation.

2.2. Formulations

The issue of the high dimensionality of formulations poses a current problem in cheminformatics and ML–assisted science. There are many variables that could potentially affect the material properties and finding these features can be non-trivial. These features may include the ingredient material types, their individual properties, the processing conditions used in the synthesis of coating, and the molecular information of the starting materials, amongst many others. Also, data is sparse for specific systems, hence we do not have access to the desired amount of

data sufficient to train ML models. The dimensionality further worsens when we tend to add feature representations to the already-available feature set of process variables and concentrations of the ingredient molecules to provide more feature information to the computer.

To bypass the intensive calculations and effort of getting properties from molecule states or formulation ingredients (e.g., through physics-based calculations), a quantity-structure-property-prediction (QSPR) can be utilized to bridge the formulations and molecular information to the property prediction directly. Hence, the formulations are selected and modeled using ML techniques to predict the best value for the property of both the intermediate and the final polymer coating material. Intermediate polymer coating properties could include the topological polar surface area (TPSA), molecular weight, and rotatable bonds while final coating properties could include glass transition temperature, abrasion resistance, and adhesive strength of the coatings to attach to the substrate materials.

Some related studies to the aforementioned include the works of Pugar et al. (2020) where they inferred that characteristics of isocyanate monomers strongly influenced the glass transition temperature (which is in turn known to strongly affect the mechanical and flow characteristics of the final material) of a thermoplastic polyurethane which can be used as a coating material.[7] They achieved this utilizing ML models including the random forest and linear regressors.

Another interesting work that utilized ML to tackle formulation problems was that of Gribova et al. (2021) where functional properties of multilayer coatings, e.g. thickness and adhesive properties, were predicted using an ML approach.[9] The formulations in this study include the presence of various functional groups and the concentrations of the components amongst others.

Formulations will be addressed in our future works as this study is focused primarily on fingerprints and how we can use them to carry out predictions on features or properties that influence polymer coatings.

2.3. Fingerprinting and Feature Representation

A chemical fingerprint is a computer-readable feature representation of chemical information. In the selection of a proper representation, the study of epoxy and diluent combination as well as jointly utilizing the information of the molecules themselves (i.e., fingerprints) in the prediction of the properties of the final coating material, to the best of our knowledge, has not been paid attention to. However, in a general case, there exist some works that have implemented polymer featurization schemes for other polymer informatics problems. Some works like Patel et al. (2022) investigated various datasets of a vast number of copolymers using different featurization scheme strategies on the repeating units of the polymers.[10] From their studies, they suggested directly encoding polymer sizes in polymer representations which could also be applied to homopolymers. Some other works have performed similar procedures but used the base monomer of the polymers. For this study, we decided to work with the base monomer of the polymers as our structure representation for our end-to-end analyses (i.e., molecule structure directly to final cured coating performance).

In representing these monomers, there exist various approaches from different researchers. Most generally successful feature representations have been demonstrated using fingerprints like the Morgan fingerprints, and especially a variant of this which is the extended connectivity fingerprint (ECFP).[11,12] Kokabi et al. (2020) represented chemical structures of drug compounds using the ECFP, among other fingerprints, to maximize their quantitative structure-activity relationship model performance in predicting whether a compound was active or inactive by its

individual effectiveness on Wnt (wingless/integrated)-signaling inhibition. They had recorded area-under-curve (AUC) values of about 70%.[13] On a general basis, however, it is important to note that the choice of fingerprints must match the choice of the machine learning model, hence the need for a series of guided trial-and-error experiments.

A notable work that has demonstrated this evaluation, also using the ECFP fingerprint in their fingerprint framework, is that of Ju et al. (2021).[14] They trained their model (stack of ensemble ML models) using a database of 1970 molecules, recording a mean absolute error (MAE) of 0.00504 a^{-1} (good agreement to their ground truth labels) on the prediction of a range separation parameter (w). The unit a^{-1} is the inverse of the Bohr radius which is a measure of the size of an electron cloud around a nucleus of the hydrogen atom. In their combined fingerprint framework, they implemented the ECFP with a length of 1024 bits and a maximum diameter of 4 (the unit of this diameter is the number of bonds away from the central atom that the ECFP algorithm considers when generating circular substructures) for the substructures included within the fingerprint which significantly contributed to the performance of their results with most of the models.

Outside the ECFP, the molecular access system (MACCS) has also demonstrated success. However, they are unable to encode chemical information and insight as it is only based on a dictionary of key-value pairs of chemical element components. The MACCS would be implemented in our work as they possess the advantages of simplicity, understandability, and low computational cost, especially when knowledge of chemical connectivity and more chemical intuition are not necessarily required. Furthermore, the works of Pattanaik and Coley (2020) included the MACCS fingerprint in their multiple-fingerprints featurization approach where they combined 24 fingerprints, hence generating 71,375 dimensions of features.[15] This is however

highly dimensional and would not be suitable for relatively small datasets, as the nature of the MACCS fingerprint alone is highly dimensional (166 features).

Neural networks, with fingerprints of the molecules as inputs, have also been used to represent the featurization space, stemming from the dimensionality problem with the fingerprints in relation to the limited amount of data available. Hu et al. (2022) proposed an ML-assisted materials genome approach for designing novel epoxy thermosets which possess excellent mechanical properties.[8] Using a sample of 238,574 small molecules, they implemented a gate-augmented GCN in conjunction with a multilayer perceptron (MLP) which outperformed other models chosen for their study in generalizability. The r^2 predictions of the mechanical properties (modulus, strength, and elongation) were recorded to be 0.81, 0.73, and 0.84 respectively. However, this is still considered a small amount of data, hence the need to introduce more intelligent systems that can harness the most information from these few samples.

In their study on predicting drug-related molecular properties, Deng et al. (2021) replaced the output layer of their directed message-passing neural network with an XGBoost base learner to make predictions.[16] They observed significant improvements compared to fingerprint-based features, having indicated this performance boost in both classification-based (area under the curve analysis) and regression-based (root mean squared error) datasets.

Our approach for neural network modeling, in a similar overall approach to the works of Deng et al. (2021), performs the utilization of the features learned and extracted from both a vanilla MG-MLP and an MG-GCN to train a downstream random forest learner.

CHAPTER 3

METHODS AND MODELS

3.1. Dataset

We selected two datasets for our study. The specific aim is to test our hypothesis on both datasets to determine which featurization both performed and generalized best. Hence, for the scope of this thesis, we have utilized both datasets for regression tasks only. The datasets chosen are discussed as follows:

3.1.1. 700,000 Polymers Dataset from the PI1M Database

The PI1M database translates to "~1 million polymers for polymer informatics". The database is open-source and was contributed by Ma and Luo (2020) in their work on polymer informatics.[17] Amongst many polymers, the database contains various potential coating base polymers such as polyurethanes and epoxies. To generate the dataset (which contains the simplified molecular-input line-entry system (SMILES) strings of various base polymers), they trained a generative model on ~12000 polymers which they had manually collected from the PolyInfo database.[18]

For our dataset, we performed training and testing individually on 7 batches of 100,000 polymers (the total dataset equals 70 percent of the total PI1M database) and trained our models in a hold-out validation analysis for our downstream ML tasks.

For our analysis, we used the cheminformatics tool RDKit via Python to generate the TPSA for each SMILES in the dataset to serve as the dependent variable for our ML operations. The TPSA is a relevant property that can influence the adhesion properties of the polymer coating. A higher TPSA indicates a larger polar surface area, and this further enhances adhesion interaction

between the coating material and the surfaces/substrate materials. Also, TPSA can affect the solubility behavior of polymers as polymers with a higher TPSA value tend to be more hydrophilic and hence may possess better solubility in polar solvents. This can be useful in applications where solubility is a desired final property.

It is however crucial to note that this is only a proof of concept to design the ML framework before utilizing it on a major dataset (see section 3.1.2) where there is no direct relationship between the input variables and the target variable.

Hence, for the prediction of TPSA, we used our various fingerprints and featurization schemes as the independent variables (X_i).

3.1.2. Epoxy-Diluent Dataset for Coating Formulation

This dataset was specifically collected to test our formulation hypothesis i.e., how well combining fingerprints with base formulation information helps the prediction performance of polymer coating properties. We shortlisted an epoxy formulation dataset collected from the study by Qiu et al. (2022).[19] The dataset consisted of 1000 sample data points, and the features (X_i) recorded included as follows:

- epoxy resin type
- the diluent type
- their additional groups (for both epoxy and diluent)
- the ratio of the concentration of the epoxy resin to the diluent
- the temperature of the formulation before curing.

Also, the viscosity of each formulated sample was recorded by the dataset authors via collation from various literature sources where they were calculated. This dataset was selected as the number of samples was sufficient for our modeling and the epoxy formulation resin data

provided a good basis for the quantitative-structure-property relationship (QSPR) study, the property here being the viscosity of the final coating material. In the author's original work, they used the group contribution method as their QSPR model. The group contribution method accounts for the presence of functional groups that make up the individual epoxies and diluents and further represents them using a binary scheme i.e., the presence of a certain epoxy e.g., bisphenol A gives a one (1) while the absence gives a zero (0). This scheme, however, is incapable of encoding low-level and important chemical information, and improving this is one of the objectives of this current study. Statistical information on the datasets can be found in Table 3.*1* below, which was constructed and summarized using the Pandas DataFrame tool via Python.

Table 3.1: Statistical information of the independent variable (temperature) and dependent variables (viscosity of final cured coating and TPSA) for both datasets

	Temperature (°C)	**Viscosity (MPa.s)**	**TPSA**
Dataset count	999	999	7×10^5
Mean	42.99	2180.66	61.56
Standard deviation	17.72	7286.81	39.71
Minimum	20	0	0
25 %	25.24	35.51	29.46
50 %	39.78	152.54	58.20
75 %	55	980.39	87.30
Maximum	100	1.37×10^5	295.85

Pertaining to the second dataset, it should be noted that a dataset of 1000 samples is still relatively small and hence, there exists the need to develop unique featurization schemes that can help encode as much information as possible from the base epoxy and diluent monomers. This in

turn facilitates training and reduces the chances of overfitting during ML modeling. These featurization schemes and procedures are explained in subsequent subsections. Figure 3.1 shows the overall framework of our implementation. Dataset 1 was used as an initial proof of concept to model the relationship between the inputs and an intermediate feature, TPSA, hence we could acquire a long list of polymers and label them using RDKit to calculate their TPSA values. The TPSA variable was then set as the dependent variable (y_i).

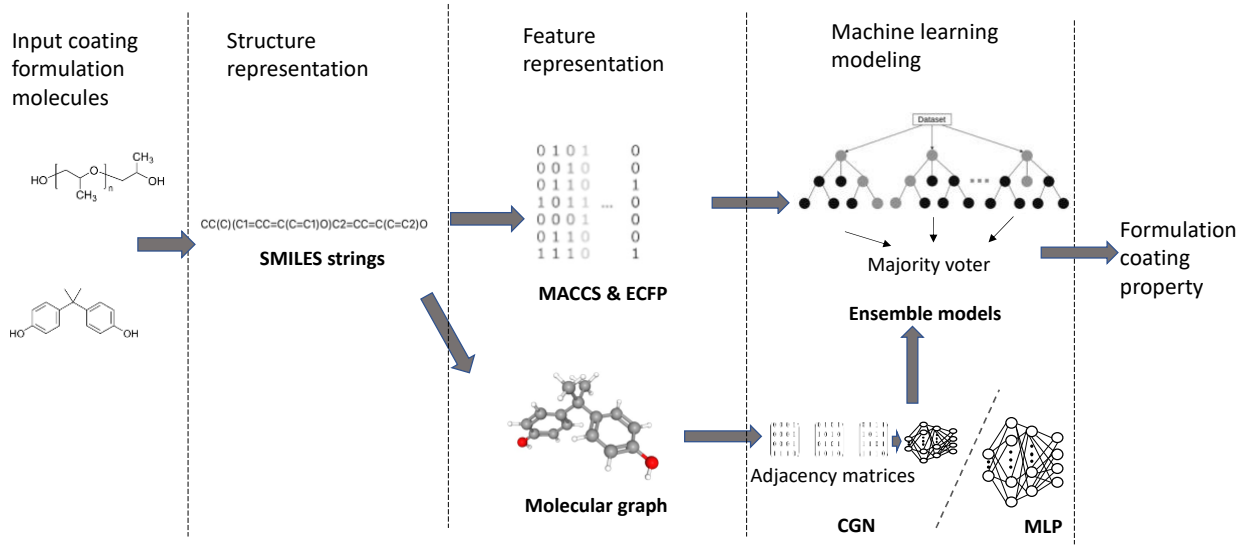


Figure 3.1: Overall methodology framework. Each arrow represents the direction of a featurization scheme comparison test. MACCS, ECFP, MG-CGN, and baseline MG-MLP direction

3.2. Structure Representation

After the dataset collection, the SMILES strings,[20] first introduced by Weininger in 1988, were collected. For dataset 1, the SMILES representation of the molecules was as found from the original database, while for dataset 2, the epoxy and diluent monomers in the original dataset were manually converted to SMILES string representations. The SMILES string is a textual representation of a molecule or polymer that helps to encode the structural information of the

molecules in a machine-readable format. Specifically for dataset 2, we simply used the SMILES strings of their individual monomers for processing to represent the polymers. The SMILES strings for all molecules in this dataset were collected from the PubChem open-source library.[21]

3.3. Feature Representation

Four featurization schemes were selected for our benchmarking study to investigate which of them best represents our molecules of the polymer property prediction (i.e., MACCS, ECFP, MG-MLP, and MG-CGN). Fingerprint schemes (i.e. MACCS and ECFP) were computed using RDKit[22] via Python and deep network-based featurization schemes (MG-CGN, MG-GCN, and the comparison baseline MG-MLP embedding) were computed using PyTorch via Python, Jupyter Notebook being the integrated development environment (IDE) of choice for all coding experiments. Furthermore, the SMILES strings were used as a structural information source to generate these features. Given below is a brief overview of these feature representations implemented in our study and subsequent sections would highlight the methods on how they were utilized with ML models.

3.3.1. Molecular Access System

Developed by Molecular Design Limited, the MACCS system is known to be used for chemical structure similarity and substructure searching in chemical databases.[23] The chemical structures are represented as a set of binary fingerprints. These fingerprints are generated using a dictionary set of predefined structural keys. Each key corresponds to a particular structure present in the molecule and the presence and absence of that structure in each molecule are classified as either 1 or 0 respectively. The issue with this key-based system is that the chemical connectivity information is not being accounted for. Also, there is a limit of flexibility as information for important substructures may not be captured, since they are based on a predefined set of keys. The

advantage of this featurization technique is its simplicity and relatively low computational cost. It is suitable for systems where chemical connectivity does not contribute as much detail to the predictions from the overall cheminformatics architecture.

The reason for selecting MACCS in our benchmarking study is based on its simplicity and low computational cost. Hence, with good generalizability performance with MACCS on our coating dataset, there may not be the need to evaluate more complex featurization schemes and MACCS may be sufficient for our coating problem.

3.3.2. Extended Connectivity Fingerprint

Known to be the most popular fingerprint representation of molecules, the ECFP is based on the concept of molecular subgraphs and it uses a circular algorithm via varying radii.[24] It is designed to capture information about a molecule's topography and connectivity features. Unlike the MACCS, the ECFP possesses the advantage of encoding a deeper level of chemical information and connectivity about the molecule. They can be generated by first assigning unique identifiers to each atom in a molecule. Then a series of graph traversal rules are applied to construct a set of molecular subgraphs. These graph traversal rules refer to the process of navigating through the atoms and bonds in the molecule in a systematic way to identify important features and substructures within the molecule.

After the application of these rules, the subgraphs are then converted into a set of binary fingerprints, where each bit corresponds to the presence or absence of a particular subgraph or feature within the molecule. The process of this conversion into binary fingerprints is done by hashing the information and any hash type can be implemented for this.

The disadvantage of the ECFP is its production of very high dimensional features which makes it difficult to accurately model relatively small amounts of data. For example, the base

number of bits is usually set to 1028 bits, and this will be insufficient for a dataset of only 1000 samples. With a greater number of bits, more information is being captured properly as there are fewer chances of bit collision. Bit collision is the process of two different structural information being captured by a single bit. Hence, the larger the number of bits, the lesser the chances of this occurrence. When the number of features exceeds the number of training samples, there tends to be data overfitting, and this can lead to generalizability problems in the model architecture.

Similar to the reason given on the selection of MACCS even though more computationally expensive, we also selected the ECFP as it encodes more intrinsic information about the molecules which can help reduce the chances of overfitting. We also benchmarked this scheme due to its proven performance in past works and literature.

3.3.3. Molecular Graphs

Molecular graphs are graphical representations of molecules where atoms are represented as nodes (or vertices) and chemical bonds are represented by edges. By representing molecules as graphs, it makes it possible to apply graph theory, mathematical and computational techniques in analyzing these molecules, and ultimately help to identify and represent structural features that are important for predicting the properties of interests of that molecule. It is usually used in conjunction with graph neural networks and their variants. It possesses the advantages of encoding the chemical information of the molecule as well as reducing the feature dimensionality as opposed to the ECFP. Moreover, it is general knowledge that ML models trained on molecular graphs are less likely to overfit to small datasets compared to those trained on ECFP which produces a high-dimensional feature space.[25] The disadvantage however is that computing them is relatively more computationally expensive. However, this is not a huge point of concern when dealing with small datasets.

Aside from their performance in literature, molecular graphs were benchmarked for this study as they best represent molecules with sufficient information that can be used to train ML models.

3.4. Modeling

After representing the polymers in machine-readable formats, we pass them to ML models to benchmark the best featurization-algorithm combination. For dataset 2 specifically, feature representations for both epoxy and diluent were concatenated with the numerical features of the original dataset (temperature and ratio of epoxy to diluent) to form the input space. Below is a brief overview of the ML models used in this study:

3.4.1. Random Forest

This is a popular ensemble ML algorithm, first introduced by Breiman (2001) that combines several decision trees to make better predictions, further reducing the chances of model overfitting.[26] It is a widely used algorithm in both regression and classification problems and is openly accessible via the sci-kit-learn library. In addition to its popularity and performance, it is well known for its high accuracy, robustness, and interpretability.

3.4.2. XGBoost

Short for extreme gradient boosting, the XGBoost is a scalable gradient-boosting ML algorithm that uses decision trees as a base model.[27] It is popularly known for its scalability, speed, and high performance. This model was first created by Chen and Guestrin (2016) from the University of Washington and was developed to handle the limitations of traditional gradient boosting which had suffered from the problems of lack of scalability, slow computation time, and issues of overfitting. This model is also well-known for winning several Kaggle competitions and

has since demonstrated exceedingly good performance even in recent times of large datasets (i.e., big data problems).

The XGBoost model works by iteratively building an ensemble of decision trees, and each tree is designed to correct the errors made by the previous trees. This process is known as boosting. The algorithm, therefore, implements a gradient boosting framework where it optimizes the loss function of the model by minimizing the gradient with respect to the model parameters.

3.4.3. Graph Convolutional Networks and Chemical Graph Network

This is a deep learning algorithm designed to work with graph-structured data, like the molecular graphs of our current study. Introduced by Kipf and Welling in 2017, the GCN uses a convolutional approach to learn and extract meaningful information from graphs.[28] In comparison to the traditional convolutional neural network which involves the sliding of filters over the input dimensions, the GCN's convolutional operation involves the aggregation of information from a node's neighbors and further uses this information to update the node's representation. What makes the GCN unique is its use of adjacency matrices which capture the graph structure, and the message-passing mechanism propagates information between nodes, hence providing the possibility to collect both local and global information in the graph.

For dataset 1, we will be using a variant that does not utilize a convolutional layer and hence we will be terming it a 'chemical graph network (CGN)' instead. Our reason for this is baseline comparison with reduced computational time for processing 700,000 molecules.

3.4.4. Voting Regressor

The voting regressor combines predictions from multiple ML models, called base learners, to make a final prediction with improved accuracy and robustness. It usually consists of base learners

and a meta learner which combines the output of models (i.e., the base learners) to produce an intelligent final prediction.

In general, this study shows that ML can easily match chemical inputs to polymer coating properties by learning this relationship explicitly by itself.

3.5. Feature Engineering

To prepare the input data of the epoxy and diluent, the molecular graphs were used to compute the graph adjacency matrices and were padded with additional 'zero' layers to the length of the molecule with the greatest number of atoms for both epoxy and diluent after computation of both their adjacency matrices. The epoxy and diluent adjacency matrices were then concatenated and fed into the CGNs in the form of adjacency matrices. This is the feature engineering step for our MG-CGN architecture computed using RDKit, a Python-based cheminformatics tool. All featurization schemes were appended with the base dataset features (temperature of formulation before curing and the epoxy-to-diluent ratio) to form the independent variable set.

After feature engineering, the GCNs were built using the PyTorch framework. (see Figure 3.2 and Table 3.2). This was compared to a benchmark vanilla deep neural network i.e., MLP which was made of 7 linear layers with Leaky rectified linear unit (ReLU) activation functions located on the output of every layer similarly also done for the GCN's architecture as well.
The GCN serves as an unsupervised pre-trainer model that feeds learned features into the ensemble models.

3.6. Ensemble Learning

After training the GCN, as for dataset 2, the learned feature representations were then fed to a voting regressor made up of both an XGBoost and a random forest model, which are the base

models. Our voting regressor implements a linear regressor as a meta learner (i.e., the algorithm that combines the predictions of the base models), and equal weights were assigned to both base models. Our reason for introducing the voting regressor was primarily due to the following reasons:

1. Better handle class imbalance in the given dataset.
2. Improve model performance by the collective contribution of the advantages of the individual base learners and reducing the effects of the weaknesses of these models.

For dataset 1, the CGN's result was directly passed to a simple random forest regressor. Details of this architecture can be found in the Appendix section.

3.7. Model Architecture

The pseudocode for all the algorithm architectures can be found in the Appendix section for work reproducibility. Table 3.2 also highlights the specific hyperparameters for all the models used in this work. For dataset 1, a baseline ECFP of 1024 bits and a radius of 2 was selected for analysis. Also, for the deep net featurization schemes, the output dimension of the deep nets (embeddings to be used as features for downstream ML modeling) was set to 4 dimensions. Also, both deep learning frameworks comprise 2 hidden layers of dimensions 32 and 64 (see Appendix for details). Furthermore, no convolutions were implemented to allow for faster and a fair baseline comparison, hence we termed the intended GCN a 'chemical graph network' (CGN). The downstream ML model was a simple random forest regressor with 300 trees and a maximum depth of 30, amongst others detailed in Table 3.2. A random state (i.e., keeping the same train-test split for every iteration the work is being run) was set to ensure the results' reproducibility for accurate code runs. The MG-CGN results were further compared with a baseline MG-MLP with similar hyperparameters to test and demonstrate the significance of the added complexity.

For the end-to-end polymer coating formulation dataset (dataset 2), two separate ECFP featurization schemes of lengths of 1024 and 2048 bits were evaluated, both were applied with a radius of 2. We also introduced a voting regressor for all featurization tests and a convolutional layer in the CGN network (hence termed a GCN here). Furthermore, a train: test split of 80:20 was performed on all featurization tests. An overview image representation can be found in Figure 3.2. A linear regressor was further chosen as a meta learner and the base learners were a random forest regressor and an XGBoost regressor.

Table 3.2: All model hyperparameters used in this study

Dataset	Models	Hyperparameters
Dataset 1 (Train: test - 70:30)	Random forest	▪ Number of trees: 200 ▪ Criterion: mean squared error ▪ Maximum depth: 30 ▪ Minimum samples split: 2 ▪ Minimum samples leaf: 1 ▪ Bootstrap: True ▪ Maximum samples: None ▪ Minimum impurity decrease: 0 ▪ Minimum weight fraction: 0
	Neural networks	▪ Number of layers: 4 Linear layers ▪ Number of neurons for each hidden layer: 32, 64 ▪ Number of input features: 1 ▪ Number of output features: 4 ▪ Activation function: ReLU function for each layer ▪ Optimizer: Adam ▪ Learning rate: 0.001 ▪ Number of epochs: 5 ▪ Batch size: 10,000
Dataset 2 (Train: test - 80:20)	Voting regressor	
	Base learner 1: Random forest	Linear regressor (meta learner): - ▪ Weights for each base learner: 0.5 ▪ Bias: None set Random forest regressor (base learner 1): - ▪ Number of trees: 100 ▪ Criterion: mean squared error ▪ Maximum depth: None

		▪ Minimum samples split: 2 ▪ Minimum samples leaf: 1 ▪ Minimum weight fraction leaf: 0.0 ▪ Maximum features: 1.0 ▪ Maximum leaf nodes: None ▪ Minimum impurity decrease: 0.0 ▪ Bootstrap: True ▪ Maximum samples: None XGBoost (base learner 2): - ▪ Number of trees: 100 ▪ Booster: GBTree ▪ Base score: None ▪ Gamma: None ▪ Early stopping rounds: None ▪ Importance type: None ▪ Maximum delta step: None ▪ 'Enable categorical' feature: False ▪ Maximum depth and leaves: None ▪ Learning rate and maximum bin: None ▪ Minimum child weight: None
	Neural networks	GCN: - ▪ Number of layers: 7 (1 convolutional layer with padding of 2 and kernel size of 5 as the first layer, followed by 6 linear layers) ▪ Number of nodes in each hidden layer: 10 ▪ Activation function: Leaky ReLU; coming after each of the first 6 layers in the network. ▪ Optimizer: Adam ▪ Learning rate: 0.01 ▪ Number of epochs: 500 ▪ Batch size: None MLP: - ▪ Number of layers: 7 linear layers ▪ Number of nodes in each hidden layer: 10 ▪ Activation function: Leaky ReLU for each layer ▪ Optimizer: Adam ▪ Learning rate: 0.01 ▪ Number of epochs: 500 ▪ Batch size: None

3.8. Implementation

For a broad summary of this work's overall procedure, after developing each model and featurization scheme, the SMILES strings from the datasets were converted into each featurization scheme via RDKit as in the fingerprints (MACCS and ECFP) and via Pytorch as in deep network (MLP and CGN) embeddings. Each featurization representation was then fed into an ensemble model to make predictions on the target variables. A testing set (30 percent of dataset 1 and 20 percent of dataset 2) was set aside to evaluate the performance of the downstream ML models for each featurization scheme while the remaining percentages were used as the training set. For dataset 1, a random forest model was used as the downstream ML model while for dataset 2, a voting regressor that combines both random forest and XGBoost regressors via a linear regressor as the meta learner was used as the downstream ML model. Predictions were then made on the testing set to compare with the ground truth labels and evaluate model performance. Final prediction results from each featurization scheme were then compared.

3.9. Performance and Evaluation

The performance metrics, used in this study to measure the closeness in prediction between the predicted output (y_{pi}) of the ML models and the corresponding ground truth labels (y_i) for the given set of input and featurization schemes (X), are the coefficient of determination (r^2), the mean squared error (MSE) and the MAE. We had selected these three as against selecting one to account for possible individual drawbacks of each of them and for a properly accurate evaluation of our system. An r^2 value closest to one is desired, while MSE and MAE values closest to zero is also desired. The equations for our three metrics are illustrated below:

$$r^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_{a,i} - y_{p,i}\right)^2}{\sum_{i=1}^{n}\left(y_{p,i} - y_a\right)^2} \qquad (1)$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{p,i} - y_{a,i})^2 \qquad (2)$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|(y_{a,i} - y_{p,i})\right| \qquad (3)$$

Where $y_{a,i}$ is the actual y label (ground truth), $y_{p,i}$ is the prediction label made by the model and n is the number of samples in the dataset. For dataset 1, 'y' corresponds to the TPSA of the molecules, while for dataset 2, 'y' represents the viscosity of the final cured polymer coating.

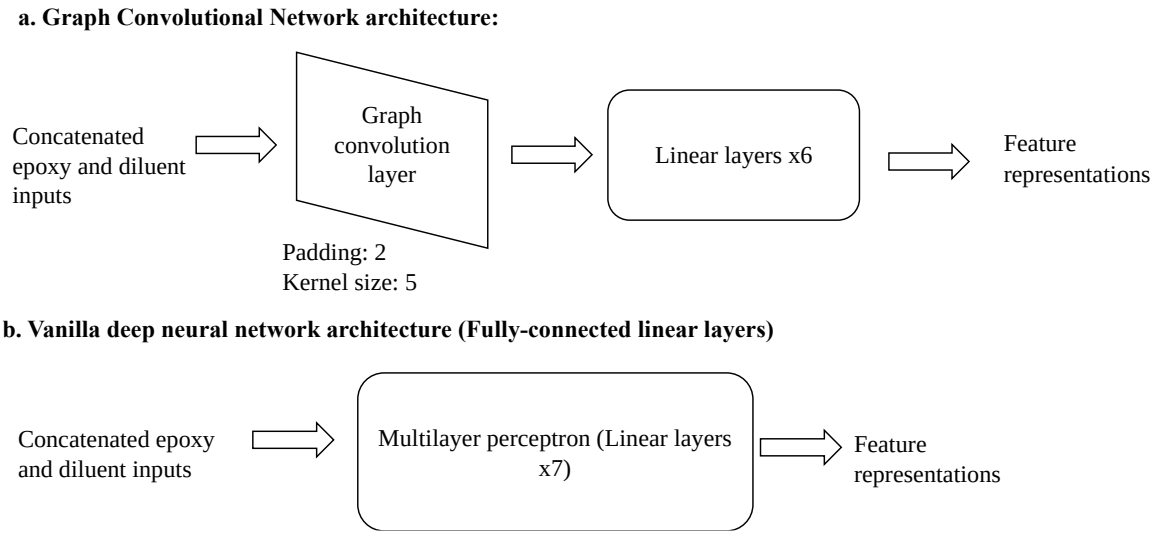


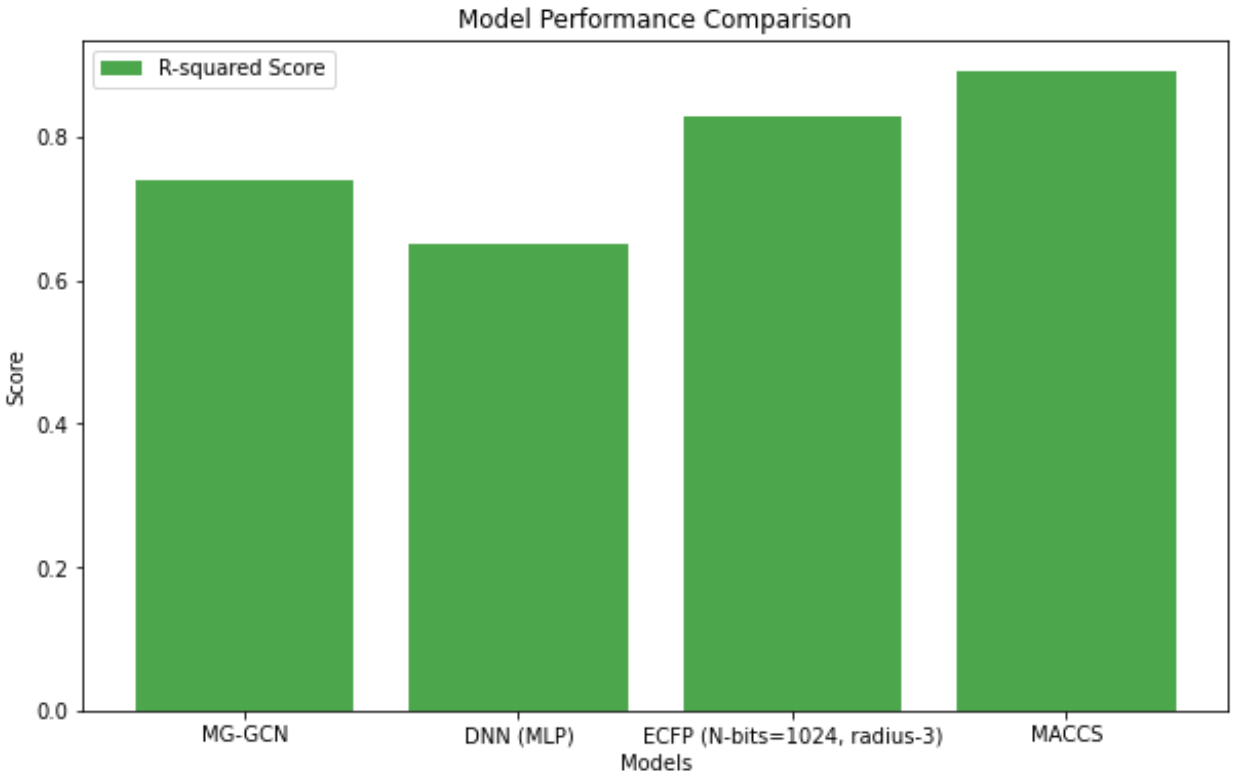
Figure 3.2: Deep neural network architectures for dataset 2 implementation (a) Graph convolutional network (b) Vanilla deep neural network (MLP). Leaky ReLU was applied on all layers in both the GCN and MLP. The neural network optimizer, number of epochs, and learning rate for both models are set to Adam, 500, and 0.001 respectively.

CHAPTER 4

RESULTS AND DISCUSSION

Early works using the polymer coating formulation dataset (dataset 2) have used the group contribution method which chemical information can potentially be lost. Although they had recorded r^2 values as high as 0.95 on their test sets, there exist the chances of overfitting and generalization problems for new polymer coating instances as the group contribution method they used only considers the presence of functional groups and no explicit chemical connectivity information. To alleviate this potential issue, our current work aims to find a better feature representation that encodes necessary chemical information and uses this as a baseline to make more reliable polymer property predictions.

Our method compares different featurization techniques and we have further shown that the MG-GCN featurization framework performs considerably well amongst other good-performing featurization schemes, as the framework has the potential to learn important hidden features by itself. Furthermore, we compared this to the baseline version of the neural network framework, which is the vanilla DNN architecture, an MLP, to demonstrate that the graph architecture indeed can learn features relatively better and that the added graph-representation feature was necessary.



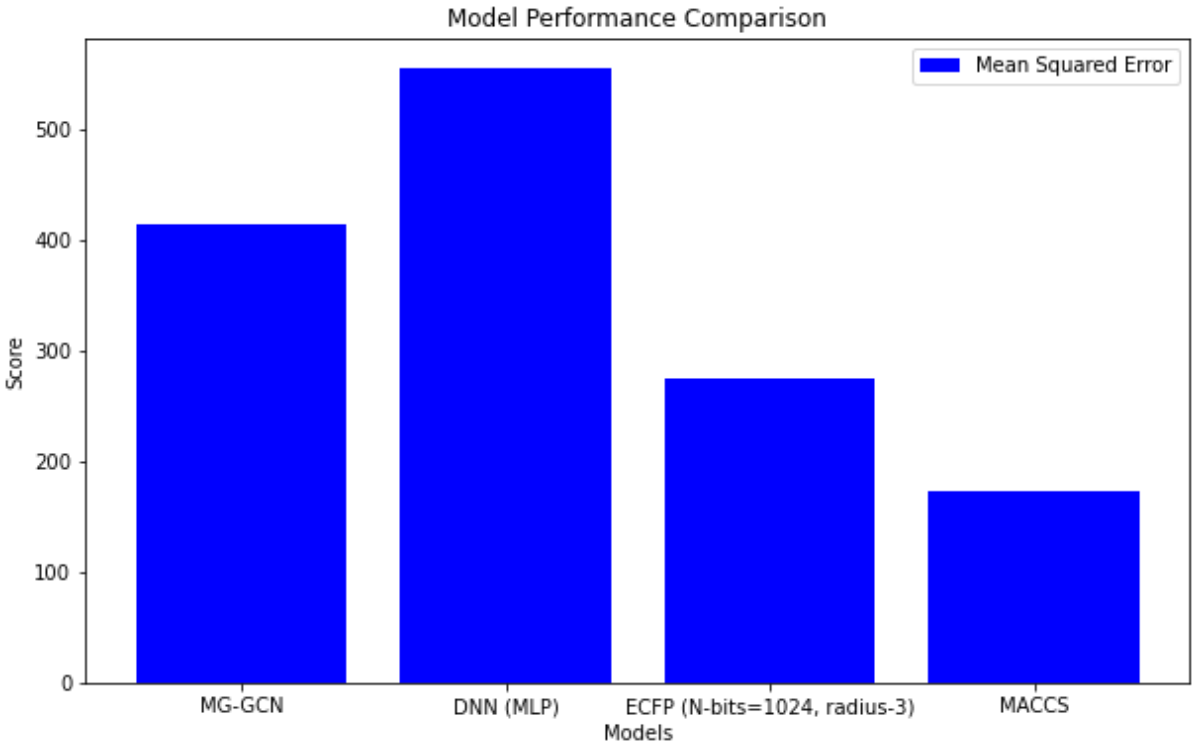


Figure 4.1: r^2 and MSE performance metric comparison for the various featurization schemes

For the graph-based implementation, the CGN class is defined as a subclass of the torch.nn.Module class. The model consisted of four fully connected (linear) layers. The first layer takes a single input feature, with 2 hidden layers and a final layer that outputs a tensor with four dimensions (final output dimension). For the forward pass, the rectified linear unit (ReLU) activation function was applied after each hidden layer, and a matrix multiplication was performed between the adjacency matrices.

4.1. Report from Dataset 1

As observed from Figure 4.1, r^2 and MSE evaluations were carried out on the 700,000 molecules in batches of 100,000s. Baseline comparisons (i.e., using the same random forest regressor for all feature representations used in this study) inferred that the MACCS fingerprint presented the best performance on the testing set. The superior performances of the MACCS and the ECFP could potentially be indicative of the close relationship of the TPSA to the atoms within the molecules. However, this may not be the case for more downstream predictions that do not have a direct relationship with the compound's molecular structure.

4.2. Report from Dataset 2

ECFP of a radius of 2 and different lengths of 1024 and 2048 bits were calculated for both the epoxy and the diluent molecules, and both fingerprints were concatenated into one feature representation. After training, as shown in Figure 4.2, the results suggest that the MG-GCN-VR framework was the model with the best generalization to an end-to-end polymer coating formulation dataset, slightly outperforming the ECFP models and the MACCS which showed comparable performance. Also, the vanilla MLP with an r^2 value of 0.781 was significantly outperformed by the MG-GCN-VR with an r^2 value of 0.835. The superiority of the MG-GCN-VR over the vanilla MLP model can be attested by its use of the permutation invariance technique

within its graph framework, which ensures that the neighborhood nodes of a central node can be in any order, and the GCN operation should then yield the same result regardless of that order. This can potentially improve the model's predictability.

The performance could also be attested to the learning of the GCN which could potentially uncover important features that are paramount to the predictability of the viscosity which was previously unknown but now revealed by the GCN model. The ECFP (2048)-VR performed closely as well. Still, it is logical to suspect overfitting as the number of features exceeds the number of sample data points (what researchers today term 'the curse of dimensionality'). The MACCS-VR also presented a good performance, however, the lack of sophistication in the fingerprint may not be sufficient to generalize for molecules or data points where chemical components and connectivity would highly influence predictability in certain and unique ways.

The authors of the dataset recorded higher performance scores with a group contribution method as their featurization technique, however, the disadvantage of this technique is the amount of information the group contribution method can gain as it is primarily based on the presence and absence of functional groups. Although our work recorded lower values than the ones presented in the paper, the basis of the thesis is not to outperform but to utilize the dataset to test state-of-the-art featurization techniques that can hold the most information for polymer coating formulation predictability.

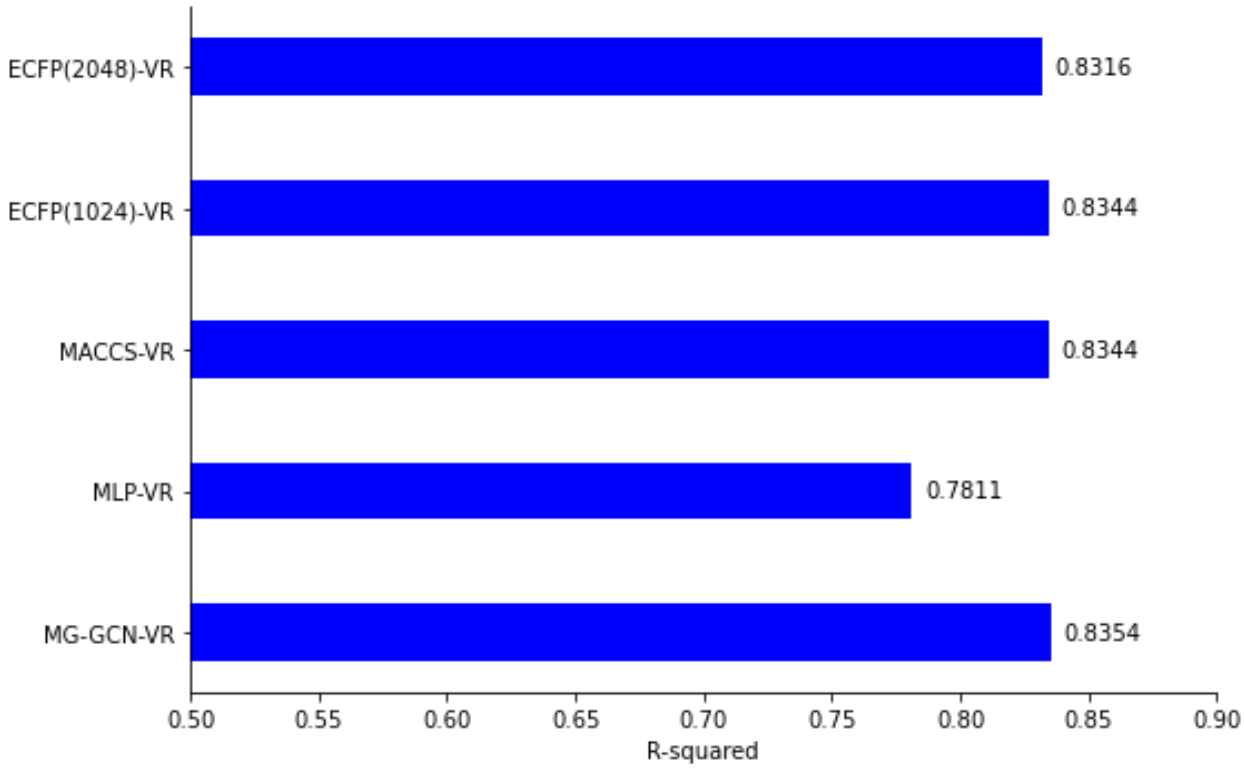


Figure 4.2: r^2 plot for the epoxy-diluent dataset in predicting the viscosity of the final cured coating. MSE and MAE plots can be found in Table A.1 in the Appendix.

To summarize our study, Figure 4.1 showed that the MACCS and the ECFP demonstrated the best r^2 and MSE performances, however, MG-CGN showed superior generalization to new instances when performed on dataset 2 (Figure 4.2) which exhibited a broader non-parametric relationship between the input and the output space. Also, for downstream operations where data is sparse and small in quantity, MACCS, and ECFP modeling may overfit (high number of features), and MG-CGN feature dimension embeddings can be adjusted to a very small number while retaining accuracy.

Lastly, we generated only 4 feature representations as output from our neural network model which yielded great performance results, hence, with increases in this output number of features for the downstream ML operation, more hidden features can be learned and hence could potentially lead to better performance. However, this may come at an added computational cost.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

Hence, our current work has demonstrated, specifically for polymer coating formulation problems, the performance of harnessing information from molecular graphs compared to fingerprints such as ECFP and MACCS, by implementing a GCN while using the resulting embeddings to train a voting regressor model to further make predictions. Furthermore, our benchmarking studies provide a solid foundation for integrating ML with physical modeling to facilitate the development of polymer coating formulations, representing the next pivotal phase of this project.

For this next phase of the project, physics-based calculations would be used to ensure ML models obey the laws of physics and a potential way to implement this is by including these laws in the loss functions of the ML models.

It is, however, important to note that GCNs (and other deep neural networks in general) depend hugely on large datasets, and hence to curb this drawback, our future implementation involves the use of active learning[29] where we strategically predict the next set of experimental combinations that provides a great level of confidence to explore uncertain areas of our design space. This new data learned by active learning will be labeled, possibly by experimental or computation techniques, and further appended to the original dataset. The training framework is then repeated till a stopping criterion is reached. A stopping criterion could potentially be the realization of a low loss from a previous training experiment. This technique enables the use of a few data points to ensure adequate predictions, further supporting the predictive performance of the ensemble model. This further lays the basics for our next steps of computational experiments.

Another interesting area that will further provide deeper insights into the properties of the molecules is utilizing the shape information of the molecules. Feeding the shape information into the GCNs can help predictability as the model tends to learn features by themselves but with more information they can perform even better. Utilizing shape information (and even conformer information) can potentially be a promising step up and then combining this with the MG–CGN can lead to more powerful predictions. The OpenEye Scientific cheminformatics toolkit,[30] can be a good tool to be utilized as a starting point to investigate this, aside from the RDKit cheminformatics toolkit.

The disadvantage of this integration so far is the computational expense. Hence future works should investigate ways to cut down on computational costs with the application of algorithmic thinking which may include avoiding re-calculating steps in the development of these models. When this is reduced, future works should incorporate convolutions into the chemical graph networks as this could potentially be a promising avenue for improved predictive performance.

APPENDICES

Appendix I: MG-GCN-RF Architecture

Pseudocode:

1. Import required libraries:
2. Define the CGN model:
    - Initialize CGN class:
        - Define **init** method:
            - Initialize parent class (nn.Module)
            - Define model layers:
                - fc1: Linear layer with input size 1 and output size 32
                - fc2: Linear layer with input size 32 and output size 64
                - fc3: Linear layer with input size 64 and output size 64
                - fc4: Linear layer with input size 64 and output size 4
        - Define forward method:
            - Apply ReLU activation on fc1 output
            - Apply ReLU activation on fc2 output
            - Apply ReLU activation on fc3 output
            - Perform matrix multiplication between adj_matrix and x
            - Apply fc4 on the resulting matrix
            - Return the output
3. Instantiate the CGN model.
4. Initialize an empty list called embeddings.
5. Process each molecule in the dataframe (df):

- For each SMILES in df['SMILES']:
    - Create an RDKit molecule from SMILES.
    - Add hydrogens to the molecule.
    - Get the number of atoms in the molecule.
    - Initialize an empty list called atom_features.
    - For each atom in the molecule:
        - Append the atomic number of the atom to atom_features.
    - Create an adjacency matrix filled with zeros.
    - For each bond in the molecule:
        - Get the indices of the bonded atoms.
        - Update the adjacency matrix by setting the corresponding positions to 1.
    - Convert atom_features and adjacency matrix to tensors.
    - Perform a forward pass on the CGN model with x and adj_matrix as inputs.
    - Calculate the average of the output embeddings over all atoms.
    - Append the averaged embedding tensor to the embeddings list.

6. Convert the embeddings list to a DataFrame called df_embeddings.
7. Split the data into training and testing sets using train_test_split with parameters:
    - Input: df_embeddings, y (target variable)
    - Test size: 0.3
    - Random state: 42
    - Assign the results to X_train, X_test, y_train, y_test.

8. Initialize a RandomForestRegressor with parameters:
    - Number of estimators: 200
    - Maximum depth: 30
    - Number of jobs: -1 (using all available processors)
9. Fit the RandomForestRegressor model on X_train and y_train.
10. Make predictions on X_test using the trained model and assign them to y_pred.
11. Evaluate the model:
    - Calculate the MSE between y_test and y_pred.
    - Calculate the r^2 between y_test and y_pred.
12. Return the MSE and r^2 score.

Appendix II: MG-MLP-RF Algorithm

Pseudocode:

1. Import required libraries:
    - Torch
    - Torch.nn
    - Torch.nn.functional
    - rdkit.Chem
    - rdkit.Chem.rdmolops
    - Pandas
2. Define the MLP model:
    - Initialize MLP class:
        - Define **init** method:
            - Initialize parent class (nn.Module)

        - Define model layers:
            - fc1: Linear layer with input size 1 and output size 32
            - fc2: Linear layer with input size 32 and output size 64
            - fc3: Linear layer with input size 64 and output size 64
            - fc4: Linear layer with input size 64 and output size 4
    - Define forward method:
        - Apply ReLU activation on fc1 output
        - Apply ReLU activation on fc2 output
        - Apply ReLU activation on fc3 output
        - Apply fc4 on the resulting matrix
        - Return the output
3. Instantiate the MLP model.
4. Initialize an empty list called embeddings.
5. Process each molecule in the dataframe (df):
    - For each SMILES in df['SMILES']:
        - Create an RDKit molecule from SMILES.
        - Add hydrogens to the molecule.
        - Get the number of atoms in the molecule.
        - Initialize an empty list called atom_features.
        - For each atom in the molecule:
            - Append the atomic number of the atom to atom_features.
        - Convert atom_features to a tensor.
        - Perform a forward pass on the MLP model with x as input.

        - Calculate the average of the output embeddings over all atoms.
        - Append the averaged embedding tensor to the embeddings list.
6. Convert the embeddings list to a DataFrame called df_embeddings.
7. Split the data into training and testing sets using train_test_split with parameters:
    - Input: df_embeddings, y (target variable)
    - Test size: 0.3
    - Random state: 42
    - Assign the results to X_train, X_test, y_train, y_test.
8. Initialize a RF Regressor with parameters:
    - Number of estimators: 200
    - Maximum depth: 30
    - Number of jobs: -1 (using all available processors)
9. Fit the RF Regressor model on X_train and y_train.
10. Make predictions on X_test using the trained model and assign them to y_pred.
11. Evaluate the model:
    - Calculate the MSE between y_test and y_pred.
    - Calculate the r^2 score between y_test and y_pred.
12. Print the MSE and r^2 score.

Appendix III: ECFP-RF Algorithm

Pseudocode:

1. Import required libraries:
    - Numpy
    - rdkit

- rdkit.Chem.AllChem
- rdkit.Chem
- Pandas

2. Create an empty NumPy array called X with dimensions (number of rows in df, 1024) and dtype as int8.
3. Process each molecule in the dataframe (df):
    - For each SMILES in df['SMILES']:
        - Create an RDKit molecule from SMILES.
        - Generate Morgan fingerprint as a bit vector for the molecule with radius 3 and 1024 bits.
        - Convert the bit vector to a NumPy array called features with dtype int8.
        - Assign the features array to the i-th row of X.
4. Create a DataFrame called X_df from the array X with column names as ['Bit_0', 'Bit_1', ..., 'Bit_1023'].
5. Split the data into training and testing sets using train_test_split with parameters:
    - Input: X_df, y (target variable)
    - Test size: 0.3
    - Random state: 42
    - Assign the results to X_train, X_test, y_train, y_test.
6. Initialize a RF Regressor with parameters:
    - Number of estimators: 200
    - Maximum depth: 30
    - Number of jobs: -1 (using all available processors)

- Random state: 42

7. Fit the RF Regressor model on X_train and y_train.
8. Make predictions on X_test using the trained model and assign them to y_pred.
9. Evaluate the model:
    - Calculate the MSE score between y_test and y_pred.
    - Calculate the r^2 score between y_test and y_pred.
10. Print the MSE and r^2 score

Appendix IV: MACCS-RF Algorithm

Pseudocode:

1. Import required libraries:
    - Numpy
    - rdkit
    - rdkit.Chem.MACCSkeys
    - rdkit.Chem
    - Pandas
2. Create an empty NumPy array called X with dimensions (number of rows in df, 167) and dtype as int8.
3. Process each molecule in the dataframe (df):
    - For each SMILES in df['SMILES']:
        - Create an RDKit molecule from SMILES.
        - Generate MACCS fingerprint for the molecule.
        - Convert the MACCS fingerprint to a NumPy array called features with dtype int8.

        - Assign the features array to the i-th row of X.

4. Create a DataFrame called X_df from the array X with column names as ['Bit_0', 'Bit_1', ..., 'Bit_166'].
5. Split the data into training and testing sets using train_test_split with parameters:
    - Input: X_df, y (target variable)
    - Test size: 0.3
    - Random state: 42
    - Assign the results to X_train, X_test, y_train, y_test.
6. Create an empty NumPy array called y with dtype float64.
7. Process each molecule in the dataframe (df) to calculate the target variable (TPSA):
    - For each SMILES in df['SMILES']:
        - Create an RDKit molecule from SMILES.
        - Calculate the TPSA for the molecule.
        - Assign the TPSA value to the i-th element of y.
8. Initialize a Random Forest Regressor with parameters:
    - Number of estimators: 200
    - Maximum depth: 30
    - Number of jobs: -1 (using all available processors)
    - Random state: 42
9. Fit the RF Regressor model on X_train and y_train.
10. Make predictions on X_test using the trained model and assign them to y_pred.
11. Evaluate the model:
    - Calculate the MSE between y_test and y_pred.

- Calculate the r^2 score between y_test and y_pred.

12. Print the MSE and r^2 score.

Table A.1: MSE, MAE, and r^2 results for epoxy-diluent formulation dataset

	MG-GCN-VR	MLP-VR	MACCS-VR	ECFP(1024)-VR	ECFP(2048)-VR
MSE	**0.1646**	0.2189	0.1656	0.17199	0.1680
MAE	**0.0605**	0.0626	0.0645	0.0631	0.0631
r^2	**0.8354**	0.7811	0.8344	0.8344	0.8316

REFERENCES

(1) Raissi, M.; Perdikaris, P.; Karniadakis, G. E. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *Journal of Computational Physics* **2019**, *378*, 686. https://doi.org/10.1016/j.jcp.2018.10.045.

(2) *What is Curing? - Definition from Corrosionpedia*. Corrosionpedia. http://www.corrosionpedia.com/definition/354/curing (accessed 2023-07-14).

(3) Zhang, J.; Tu, W.; Dai, Z. Transparent Polyester Polyol-Based Polyurethane Coatings: The Effect of Alcohols. *Journal of Coatings Technology and Research* **2013**, *10*, 887. https://doi.org/10.1007/s11998-013-9527-x.

(4) Jin, K.; Luo, H.; Wang, Z.; Wang, H.; Tao, J. Composition Optimization of a High-Performance Epoxy Resin Based on Molecular Dynamics and Machine Learning. *Materials & Design* **2020**, *194*, 108932. https://doi.org/10.1016/j.matdes.2020.108932.

(5) Rane, A.; Abitha, V.; Sabnis, A.; Kathalewar, M; Jamdar, V., Patil, S.; Jayaja, P. A Greener and Sustainable Approach for Converting Polyurethane Foam Rejects into Superior Polyurethane Coatings. *Chemistry International* **2022**, *1*, 184.

(6) Dall Agnol, L.; Dias, F. T. G.; Ornaghi, H. L.; Sangermano, M.; Bianchi, O. UV-Curable Waterborne Polyurethane Coatings: A State-of-the-Art and Recent Advances Review. *Progress in Organic Coatings* **2021**, *154*, 106156. https://doi.org/10.1016/j.porgcoat.2021.106156.

(7) Pugar, J. A.; Childs, C. M.; Huang, C.; Haider, K. W.; Washburn, N. R. Elucidating the Physicochemical Basis of the Glass Transition Temperature in Linear Polyurethane Elastomers with Machine Learning. *Journal of Physical Chemistry B* **2020**, *124*, 9722. https://doi.org/10.1021/acs.jpcb.0c06439.

(8) Hu, Y.; Zhao, W.; Wang, L.; Lin, J.; Du, L. Machine-Learning-Assisted Design of Highly Tough Thermosetting Polymers. *American Chemical Society: Applied. Materials Interfaces* **2022**, *14*, 55004. https://doi.org/10.1021/acsami.2c14290.

(9) Gribova, V.; Navalikhina, A.; Lysenko, O.; Calligaro, C.; Lebaudy, E.; Deiber, L.; Senger, B.; Lavalle, P.; Vrana, N. E. Prediction of Coating Thickness for Polyelectrolyte Multilayers via Machine Learning. *Scientific Reports* **2021**, *11*, 18702. https://doi.org/10.1038/s41598-021-98170-x.

(10) Patel, R. A.; Borca, C. H.; Webb, M. A. Featurization Strategies for Polymer Sequence or Composition Design by Machine Learning. *Molecular Systems Design and Engineering* **2022**, *7*, 661. https://doi.org/10.1039/D1ME00160D

(11) Cereto-Massagué, A.; Ojeda, M. J.; Valls, C.; Mulero, M.; Garcia-Vallvé, S.; Pujadas, G. Molecular Fingerprint Similarity Search in Virtual Screening. *Methods* **2015**, *71*, 58. https://doi.org/10.1016/j.ymeth.2014.08.005.

(12) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *Journal of Chemical Information and Modeling* **2010**, 50, 742. https://doi.org/10.1021/ci100050t.

(13) Kokabi, M.; Donnelly, M.; Xu, G. Benchmarking Small-Dataset Structure-Activity-Relationship Models for Prediction of Wnt Signaling Inhibition. *Institute of Electrical and Electronics Engineering Access* **2020**, *8*, 228831. https://doi.org/10.1109/ACCESS.2020.3046190.

(14) Ju, C.-W.; French, E. J.; Geva, N.; Kohn, A. W.; Lin, Z. Stacked Ensemble Machine Learning for Range-Separation Parameters. *Journal of Physical Chemistry Letters* **2021**, *12*, 9516. https://doi.org/10.1021/acs.jpclett.1c02506.

(15) Pattanaik, L.; Coley, C. W. Molecular Representation: Going Long on Fingerprints. *Chem* **2020**, *6*, 1204. https://doi.org/10.1016/j.chempr.2020.05.002.

(16) Deng, D.; Chen, X.; Zhang, R.; Lei, Z.; Wang, X.; Zhou, F. XGraphBoost: Extracting Graph Neural Network-Based Features for a Better Prediction of Molecular Properties. *J. Chem. Inf. Model.* **2021**, *61*, 2697. https://doi.org/10.1021/acs.jcim.0c01489.

(17) Ma, R.; Luo, T. PI1M: A Benchmark Database for Polymer Informatics. *J. Chem. Inf. Model.* **2020**, *60*, 4684. https://doi.org/10.1021/acs.jcim.0c00726.

(18) *Polymer Database (PoLyInfo) - DICE :: National Institute for Materials Science*. https://polymer.nims.go.jp/ (accessed 2023-07-14).

(19) Qiu, H.; Zhao, W.; Pei, H.; Li, J.; Sun, Z.-Y. Highly Accurate Prediction of Viscosity of Epoxy Resin and Diluent at Various Temperatures Utilizing Machine Learning. *Polymer* **2022**, *256*, 125216. https://doi.org/10.1016/j.polymer.2022.125216.

(20) Weininger, D. SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. *Journal of Chemical Information and Computer Sciences* **1988**, *28*, 31. https://doi.org/10.1021/ci00057a005.

(21) PubChem. *PubChem*. https://pubchem.ncbi.nlm.nih.gov/ (accessed 2023-07-15).

(22) *RDKit*. https://www.rdkit.org/ (accessed 2023-07-09).

(23) Dalby, A.; Nourse, J. G.; Hounshell, W. D.; Gushurst, A. K. I.; Grier, D. L.; Leland, B. A.; Laufer, J. Description of Several Chemical Structure File Formats Used by Computer Programs Developed at Molecular Design Limited. *Journal of Chemical Information and Computer Sciences* **1992**, *32*, 244. https://doi.org/10.1021/ci00007a012.

(24) *Extended-Connectivity Fingerprints | Journal of Chemical Information and Modeling*. https://pubs.acs.org/doi/pdf/10.1021/ci100050t (accessed 2022-10-25).

(25) Zhu, S.; Nguyen, B. H.; Xia, Y.; Frost, K.; Xie, S.; Viswanathan, V.; Smith, J. A. *Improved Environmental Chemistry Property Prediction of Molecules with Graph Machine Learning*; preprint; Chemistry **2023**, 1. https://doi.org/10.26434/chemrxiv-2023-8c62c.

(26) Breiman, L. Random Forests. *Machine Learning* **2001**, *45*, 5. https://doi.org/10.1023/A:1010933404324.

(27) Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd Association of Computing Machinery (ACM): Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) International Conference on Knowledge Discovery and Data Mining*; ACM: San Francisco California USA, **2016,** 785. https://doi.org/10.1145/2939672.2939785.

(28) Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. arXiv **2017**. https://doi.org/10.48550/arXiv.1609.02907.

(29) Settles, B. *Active Learning Literature Survey*; Technical Report; University of Wisconsin-Madison Department of Computer Sciences, **2009**. https://minds.wisconsin.edu/handle/1793/60660 (accessed 2023-06-05).

(30) *OEChem TK | OEChem Toolkit | Cheminformatics*. https://www.eyesopen.com/oechem-tk (accessed 2023-07-15).