2021

# Information Architecture for a Chemical Modeling Knowledge Graph

Adam R. Luxon
*Virginia Commonwealth University*

# INFORMATION ARCHITECTURE FOR A CHEMICAL MODELING KNOWLEDGE GRAPH

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Engineering at Virginia Commonwealth University.

by

ADAM RILEY LUXON

Bachelors of Science, University of Richmond, 2017

Advisor: Dr. James K. Ferri, PhD,

Professor, Department of Chemical and Life Science Engineering

Virginia Commonwealth University

Richmond, Virginia

May, 2021

# TABLE OF CONTENTS

Chapter Page

Table of Contents . . . . . i

List of Tables . . . . . ii

List of Figures . . . . . iii

Abstract . . . . . viii

1 Introduction . . . . . 1

1.1 Design-Build-Test Cycle . . . . . 1
1.2 Machine Learning in Molecular Disciplines . . . . . 7
1.2.1 Machine Learning Predictions of Chemical Properties . . . . . 7
1.2.2 Strengths and Limitations . . . . . 8
1.2.3 Machine Learning Workflow . . . . . 9
1.2.4 Molecular Representations and Featurization . . . . . 13
1.3 Navigating Design Space (Motivation) . . . . . 17
1.4 Graph Databases for Chemistry . . . . . 19
1.4.1 Databases . . . . . 19
1.4.2 Chemical Networks . . . . . 20
1.4.3 Statement of Work . . . . . 21

2 Methodology . . . . . 23

2.1 Machine Learning Pipeline . . . . . 23
2.1.1 Datasets . . . . . 23
2.1.2 Model Generation . . . . . 23
2.1.3 Storage & Export . . . . . 26
2.2 Graph Database . . . . . 26
2.2.1 Software . . . . . 26
2.2.2 Architecture . . . . . 26
2.2.3 Queries . . . . . 28

3 Results . . . . . 30

3.1 Python Pipeline . . . . . 30

3.2 Graph Database . . . . . . . . . . . . . . . . . . . . . . . . . . 30
3.2.1 Model Evaluation and Comparison . . . . . . . . . . . . . . . . 31
3.2.2 Graph Queries for Molecular Error Analysis . . . . . . . . . . 36
3.3 Lipophilicity Case Study . . . . . . . . . . . . . . . . . . . . . . 42
3.3.1 Query Setup . . . . . . . . . . . . . . . . . . . . . . . . . . 42
3.3.2 Query Results . . . . . . . . . . . . . . . . . . . . . . . . . 44
3.3.3 Query Follow-up . . . . . . . . . . . . . . . . . . . . . . . . 47
3.3.4 Case Study Conclusion . . . . . . . . . . . . . . . . . . . . 52

4 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

Appendix A Abbreviations . . . . . . . . . . . . . . . . . . . . . . 54

Appendix B Cypher Graph Queries . . . . . . . . . . . . . . . . . 55

B.1 General Graph Queries . . . . . . . . . . . . . . . . . . . . . . 55
B.2 Queries for Figures . . . . . . . . . . . . . . . . . . . . . . . . 56
B.3 Queries For Fragment Error Analysis . . . . . . . . . . . . . . . 57

References . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61

## LIST OF TABLES

Table Page

1 Major free energy contributions considered in thermodynamic models of surfactants. . . . . . . . . . 5

2 Datasets used to train models. Size is number of unique molecular observations after processing. . . . . . . . . . 23

3 Node labels. For each node label, the count of such nodes and the number of properties stored on the node are included. . . . . . . . . . 32

4 Relationship types. For each relationship type, the count of such relationships and the number of properties stored on that type of relationship are included. . . . . . . . . . 33

5 Fragment error analysis results for the lipophilicity dataset. Fragments are sorted by average error of parent molecules. Query parameters: *cutoff*=0.9; *easy limit*=0.003; *hard limit*=0.03. . . . . . . . . . 41

6 Case study fragment analysis results sorted by number of fragment parent molecules. . . . . . . . . . 44

7 Case study fragment analysis results sorted by average difficulty. . . . . 46

8 Case study results for high error molecules sorted by difficulty. . . . . . 47

## LIST OF FIGURES

Figure Page

1 General relationship between prediction cost and accuracy for chemical modelling. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2 General machine learning process flow. . . . . . . . . . . . . . . . . . . . . 10

3 Depiction of 3-fold cross validation of model tuning. The training data set is broken up multiple times and used to train and validate the model. The model performance is based on the composite of the cross validation iterations. Figure adapted from [39]. . . . . . . . . . . . . . . . 13

4 Images from Picasso's *The Bull* shows how an animal can be drawn with various levels of detail and complexity. Running parallel is the analogous progression of molecular representations. Each step along the progression adds more detail, allowing for a more complete and unique representation. . . . . . . . . . . . . . . . . . . . . . . . . . . 15

5 A: Current practice of model evaluation considers models in isolation and limits model comparison to aggregated performance metrics. B: The proposed approach connects model inputs and outputs as a descriptive graph. Database queries access the full model information and context for richer model analyses. . . . . . . . . . . . . . . . . . . . 19

6 Data process flow diagram. . . . . . . . . . . . . . . . . . . . . . . . . . . 25

7 The database schema. The *ML Model* node is the backbone of the presented information architecture. Nodes represent entities in a machine learning workflow and relationships describe how those entities interact. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27

8 Example path relating a model node (A) to its test set (B) and the test set to a predicted molecule (C) via their respective prediction relationships. In the property graph model, properties can be stored on graph nodes and relationships to further describe and quantify the structures. Properties can be used for formatting the color and size of nodes and relationships. . . . . . . . . . . . . . . . . . . . . . . . . . . 28

9 Snapshot of the graph database containing approximately 11,000 nodes and 20,000 relationships. All node labels and relationship types included. 32

10 Graph structures which enable inter- (A) and intra-model prediction evaluations (B). The architecture of the graph is such that shifting from inter-model comparisons to intra-model analysis is just one additional edge traversal. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

11 Direct comparisons of two models' parameters and performance. (A) Two models share a dataset (light blue node) and molecule featurization (yellow node). (B & C) The models use different learning algorithms. (E & F) The models' prediction performances (colored arrows). (F) Common test set molecules. . . . . . . . . . . . . . . . . . . 35

12 Molecule overlap between ten models' test sets and the individual molecular predictions. The graph connects models to their predicted molecules via test sets, enabling analysis of molecular property prediction difficulty. Prediction relationships are formatted using prediction error (color) and uncertainty (thickness). . . . . . . . . . . . . . . . . . 36

13 Fragment error analysis sub-graph and high error sulfonamides. (A, C, & D) clusters of molecules connected to a set of common fragments, indicating the fragments are the error origin. (B) Terminal fragments connected only to a single molecule, suggesting that those fragments are not the error origin. Molecule nodes are sized based upon their average prediction error – larger molecules have higher error. . . . . . . 38

14 Parameter response surface for the fragment error analysis. The fragment error analysis parameters can be adjusted to locate regions containing fragments of interest. Star denotes region examined in previous figure and table. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

15 High error molecule from Lipophilicity fragment analysis. . . . . . . . . . 48

16 Bloom query panel. Custom query for searching nodes by SMILES shown. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49

17 Expansion of macrocycle molecule node and the ML models that use neural networks to predict it's lipophilicity. . . . . . . . . . . . . . . . . . 50

18 Full expansion of ML models predicting the macrocycle molecule. . . . . . 51

## LIST OF SOURCE CODES

1 Code used to run a ML model in the pipeline. . . . . . . . . . . . . . 31
2 Cypher commands to prepare the graph for the fragment analysis. . . 43
3 Cypher commands to run the fragment analysis. . . . . . . . . . . . . 45
4 Cypher commands to delete analysis weights to return graph to default state. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
5 Cypher commands to return high difficulty molecules. . . . . . . . . . 48

## Abstract

INFORMATION ARCHITECTURE FOR A CHEMICAL MODELING KNOWLEDGE GRAPH

By Adam Riley Luxon

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Engineering at Virginia Commonwealth University.

Virginia Commonwealth University, 2021.

Advisor: Dr. James K. Ferri, PhD,

Professor, Department of Chemical and Life Science Engineering

Machine learning models for chemical property predictions are high dimension design challenges spanning multiple disciplines. Free and open-source software libraries have streamlined the model implementation process, but the design complexity remains. In order better navigate and understand the machine learning design space, model information needs to be organized and contextualized. In this work, instances of chemical property models and their associated parameters were stored in a Neo4j property graph database. Machine learning model instances were created with permutations of dataset, learning algorithm, molecular featurization, data scaling, data splitting, hyperparameters, and hyperparameter optimization techniques. The resulting graph contains over 83,000 nodes and 4 million edges and can be explored with interactive visualization software. The structure of the property graph is centered around models and molecules which enables efficient and intuitive inter- and intra-model evaluation. We use a curated lipophilicity dataset to demonstrate graph use cases. Difficult to predict molecules were identified across multiple models simultaneously. Powerful and expressive graph queries were implemented to identify molecular fragments that were both prevalent and associated with high lipophilicity prediction error.

# CHAPTER 1

# INTRODUCTION

## 1.1 Design-Build-Test Cycle

In chemical and material engineering, the objective is to create a molecular system to meet the needs of an application. In some cases, such as petroleum refinement, the system is designed around the physical chemistry of the molecules. In other applications, the molecule or material is designed to fulfill a specific role, such as a drug or light absorber. Regardless of the specifics or scale of the to-be-engineered system, the Design-Test-Build paradigm is an iterative means to a solution. This work is concerned with the rational design of molecules.

Prior to the design cycle, system requirements and criteria must be established to define the desired performance of the system and provide a target for the optimization process. During the design phase, engineers use the information and knowledge at their disposal to construct an initial molecular system and predict how it will perform. The system is then built to the design specifications, or in the case of molecules, synthesized. The performance of the molecule is evaluated against the system requirements. Performance information is used to inform the next iteration of design, sometimes referred to as a fourth learning step in the cycle. The design-test-build sequence is iterated upon until the system performance requirements are satisfied. Unfortunately, the chemical design space is very large, diverse, and costly to explore. The rest of this section will use interfacial surfactant systems to illustrate the challenges associated with the rational design of molecules and techniques used to improve the efficiency of the prediction process.

Given the wealth of chemical knowledge accrued, it is should be possible to predict new chemical phenomena *a priori*. There are two major approaches to chemical predictions. One approach to chemical predictions is the use of quantum mechanics (QM). Quantum mechanics is a collection of theories and postulates that allows for the prediction of chemical properties based on first principles. The fundamental postulate of quantum mechanics is that microscopic systems can be *completely* described by a wave function, often given the symbol $\Psi$ [1]. To get a physical observable, such as energy, one applies a mathematical operator to the wave function.

$$\hat{H}\Psi = E\Psi \tag{1.1}$$

The operator that returns the energy of a system is called the Hamiltonian operator. Similar operators exist for observables such as kinetic energy, momentum, angular momentum, spin, and dipole. Theoretically, quantum mechanics is extremely general, in that it will accurately describe systems ranging from a single atom to a single cell. However, the mathematical and physical complexity associated with QM makes it intractable for all but the smallest systems. Through the use of reducing assumptions and approximations, QM can be applied to systems ranging from a half dozen to several hundred heavy atoms. In order to get experimentally accurate results from a quantum mechanics calculation, rigorous and computationally intensive algorithms, such as coupled cluster methods, must be implemented [1]. Since systems of chemical interest often have multiple molecules with dozens of heavy atoms in the condensed phase, the most rigorous computational methods are prohibitively expensive. Less computationally intensive methods, such as density functional theory (DFT), describe the quantum mechanics well but often lack experimental accuracy [2]. DFT is frequently employed to make qualitative prediction for systems of interest.

The second prediction approach considers prior experience and data to identify patterns. If the patterns are expressed using mathematical equations, the predictor is an empirical regression. If the patterns are not framed in mathematics, they are referred to as heuristics or rules, e.g the Baldwin rules of ring closure or hydrophilic lipophilic balance (HLB).[3, 4, 5]

Surfactant design exemplifies the challenges of predicting complex chemical phenomena. Formulation chemists desire a way to predict emulsion stability and experimental observables. Predicting emulsion stability is a difficult task due the complex physics and multiple mechanisms in which emulsions can destabilize. Predictive models for surfactants often focus on simpler observables than emulsion stability such as critical micelle concentration (CMC), interfacial tension (IFT) and partition coefficients (LogP). These simpler surfactant characteristics are not predictive of emulsion stability. HLB is used industrially as a heuristic tool to predict how surfactants will behave in complex systems like emulsions. It is especially useful for systems containing multiple surfactant molecules.

HLB was first introduced by Griffin and Atlas Powder Company. [4, 5]. HLB is a numerical value assigned to a molecule in order to quantify the balance between hyrophilic (water-loving) and lipophilic (oil-loving) groups within the same *non-ionic* surfactant. This is generally done on a weight basis, i.e $HLB = f(W_h, W_l)$ where $W_h$ is the total weight of hydrophilic groups and and $W_l$ is the total weight of lipophilic groups.

For non-ionic surfactants where ethylene oxide units are the hydrophilic groups, the HLB value is defined by Griffin[5] to be:

$$HLB = \frac{E}{5} \tag{1.2}$$

where $E$ is the weight percentage (non-decimal) of oxyethylene groups. For polyhydric alcohol fatty acid esters, the HLB value is defined as:

$$HLB = 20\ (1 - \frac{S}{A}) \tag{1.3}$$

where $S$ is the saponification value of the ester and A is the acid value of the acid group. For surfactants that contain both polyoxyethylene chains and polyhydric alcohols, Griffin used the following equation:

$$HLB = \frac{E + P}{5} \tag{1.4}$$

where $P$ is the weight percent of the polyhydric alcohol. These equations place limitations of the value of HLB, that is $0 \leq HLB \leq 20$. HLB is not rigorously defined and not based on empirical measurements. Instead, HLB, as defined by Griffin, is a relative scale used as a heuristic to estimate how a non-ionic surfactant or mixtures of non-ionic surfactants will behave as an emulsifying agent. All of the above definitions are not explicitly rooted in physics associated with hyrdophilicity or lipophilicity. Even without a firm foundation in physics, HLB has proven useful as a predictive system which suggests that HLB is correlated with some molecular physics that determine a surfactants behavior in multi-phase systems.

Several studies have developed a rigorous thermodynamic approach to include more physics in the prediction of surfactant behavior.[6, 7, 8, 9, 10, 11, 12, 13, 14, 15] These studies provide a mathematical regression framework based on Gibbs free energy contributions for the prediction of bulk, interfacial and disperse phase properties (Table 1). Many of these studies focus on micelles and use either the mass-action or phase separation approach to the thermodynamic modelling of micelles. The mass-action approach treats micelles of different sizes as independent chemical species which are all in equilibrium with each other and the free monomers in solution. Most of the research cited above fall under this category. The phase separation approach treats the micelles as separate phases from the bulk solution. Using this approach, if one assumes homogenous distribution of surfactant molecules in the micelle phase, the CMC can be predicted using statistical associating fluid theory (SAFT), which is based on first order perturbation theory. [16, 17] These frameworks are based in physics, but like all models, make assumptions and concessions to model cost.

Table 1: Major free energy contributions considered in thermodynamic models of surfactants.

| Energy Contribution | Description | Theory |
|---|---|---|
| Transfer | Energy associated with transferring head and tail from bulk phase to dispersed phase. | Empirical regression |
| Deformation | Energy associated with loss of degrees of freedom in dispersed phase. | Flory lattice approach |
| Sterics | Steric interactions of head groups at interface. | van der Waals |
| Interface | Creation of high energy interfacial area. | Bulk phase surface tension |
| Dipole | Repulsive energy of aligned dipoles in close proximity. Applies mostly to Zwitterions. | Capacitor model |
| Ionic Interactions | Interaction energy of charged groups in close proximity. Applies only to ionic surfactants. | Corrected Debye-Huckle |

Both regression and heuristics, which are not necessarily based on underlying physical chemistry, are useful within the range of systems from which they were developed. However, there are frequently exceptions to the “rules” and extrapolation to new systems is often inaccurate. The prediction techniques discussed so far, heuristics, thermodynamics, density functionals, and quantum mechanical wave functions, follow a trend of increasing accuracy with an increase in cost (Figure 1). Machine learning has showed potential to deviate from this trend by producing accurate predictions with low computational cost [18, 19, 20]. Machine learning for chemical

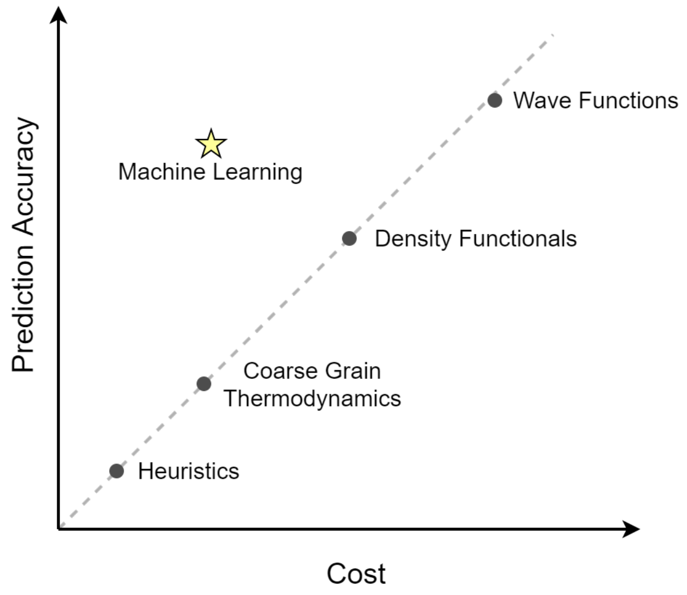


Figure 1: General relationship between prediction cost and accuracy for chemical modelling.

property prediction is a regression approach. Machine learning models correlate large sets of observation data and descriptions of each observed system. The process is "data driven", in the sense that all information is contained either in the observations or the descriptors – there are no mathematical frameworks predefined by the modeler. Another defining feature of machine learning is that the model is only evaluated on the predictions it makes, not the underlying framework. In other words, as long as the predictions are correct, the *how* is unimportant.

Surfactant properties, such as CMC, IFT and HLB, have been the target of machine learning studies [21, 22, 23, 24, 25]. Surfactant machine learning models frequently suffer from small data sets and a limited number of descriptors. For example, Wang et al. used four descriptors which were selected heuristically, to train a neural network model to predict the HLB values of 73 anionic surfactants. Due to these limitations, the majority of quantitative structure property relationships for surfactants have not employed machine learning algorithms, but instead have used simple regression models.[21] While machine learning has yet to revolutionize surfactant design, the technique has demonstrated great potential for aiding in the design of molecular systems for which more information is available.

## 1.2 Machine Learning in Molecular Disciplines

### 1.2.1 Machine Learning Predictions of Chemical Properties

Computational resources and statistical learning algorithms are advanced enough that computer models can effectively identify patterns given a set of observations. Machine learning is the use of statistical models to better understand complex data. There are generally two major classifications of machine learning techniques: unsupervised and supervised. Unsupervised machine learning aims to identify structure and relationships within a set of inputs but is not guided by a desired output. In supervised learning, the statistical models are used to predict an output given a set of inputs. When the target output of supervised learning is a discrete variable it is referred to as classification; when the output variable is continuous it is known as regression. Regardless of the type of machine learning, the success and accuracy of the resulting model depends on the quality of the inputs and known outputs.

For molecular and material engineering applications, the input is frequently a molecular representation and the output is a property of interest, such as solubility, reaction yield, or absorbance. The predictive power of machine learned models has the potential to transform molecular design and engineering by allowing engineers to efficiently design molecules of value. In the fields of chemistry and materials engineering, machine learning algorithms have successfully been implemented to predict a variety of properties of interest.[26] Machine learning has been utilized to discover new materials with desired properties. [27, 28, 29, 30]

Chemical reactivity is a popular topic of machine learning studies, two of such studies will be described here. [31, 32, 33, 34, 35, 36] Ahneman et al. used high throughput experimentation and DFT calculations to predict the yield of Buchwald-Hartwig cross-coupling reactions.[32] They performed gas phase B3LYP/6-31G* optimizations and frequency calculations on all reagents, additives, bases, and catalyst ligands. They extracted descriptors from their DFT calculations and used them as input for supervised machine learning where the target output was reaction yield. Coley et al. represented molecules as graphs and used machine learning to predict which bonds were most likely to break or be formed.[33] Complex atom and bond level descriptors, such as partial charges and surface area, were excluded from their molecular description. They used a neural network algorithm to create their model for chemical reactivity and benchmarked it against expert chemists. The models developed by Ahneman et al. and Coley et al. were both successful in predicting chemical reactivity. However, the manner in which they built their model, specifically how they chose to represent molecules are quite different. Ahneman et al. chose to use fine details extracted from electronic structure calculations while Coley et al. used a coarser graph representation that excluded fine details. Neither molecular representation is a complete description of the systems being studied. For example, neither included solvent or temperature in their model – two parameters that all chemists would consider important. This leads to the challenging question: What is the optimal level of molecular detail to include in a model to answer a specific question?

### 1.2.2 Strengths and Limitations

One of the most compelling strengths of machine learning is the large amount of complexity (variables) that can be analyzed. Since machine learning uses statistical modeling rather than rigorous first principle systems of equations, the complexity of a system is not limited to mathematical tractability. This allows for a modeler to make fewer assumptions about the relationship between inputs and outputs. In contrast to the thermodynamic frameworks previously described, machine learning modeling is a top-down approach. A highly complex set of inputs is used to uncover the relationship between the input variables and the output behavior of the system. Such an approach should not only capture the leading term, but also the next several significant terms in a governing relationship.

Of course, the machine learning approach has draw backs as well. Probably the largest one is the reliance on a large and high quality data set. The second is interpretability of the model, which depends on the machine learning algorithms used. For instance, a linear model is much more interpretable than a neural network model, but also less predictive. It will be more difficult to gain mechanistic insights from a machine learning model than from a ground-up thermodynamic framework. In other words, machine learning models can suffer from the proverbial "black box" issue.

### 1.2.3 Machine Learning Workflow

A machine learning workflow begins after the identification of the challenge. For this example, consider a researcher interested in creating a supervised machine learning model capable of predicting lipophilicity as a proxy for drug candidate bioavailablity [37, 38]. The first step in a machine learning workflow is gathering labelled data relevant to the prediction target, small molecule lipophilicity (Figure 2). Gathering data is the most important, difficult, and expensive step in machine learning. Data can be gathered from published sources, simulation, or directly from experimental observation. Currently, quality chemical datasets are rare because of the high cost of collecting trustworthy chemical observations on a large number of molecules.

Once collected, the data must be processed to ensure its quality. Duplicate

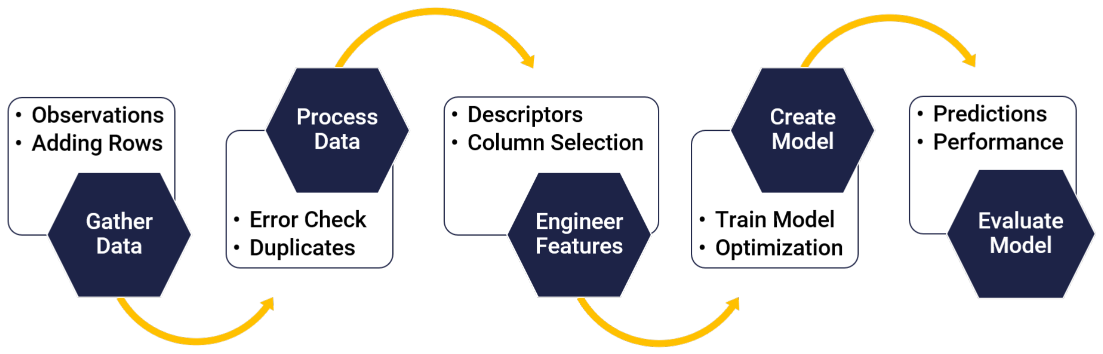


Figure 2: General machine learning process flow.

entries and incorrect chemical structure must be removed and target property units must be reconciled. It is also useful to explore and visualize the data before creating a model. The model will be limited to information and patterns contained in the dataset, so it is important to understand the contents and boundaries of the dataset. To continue the drug design example, if the application requires a molecule with a negative lipophilicity, then the dataset should contain a significant population of molecules with negative lipophilicity.

Once the observations are curated, the feature vectors are constructed. Feature engineering is the process of designing the way in which the molecules are presented to the machine learning algorithm. This can involve enhancing, transforming, or removing information. The process can be thought of as deciding what columns to include from your data table. The objective is to populate the feature vector with chemical information that is relevant to the target property. It is important to note that relevant in this context just means that the feature can be positively or negatively correlated with the target property – it does not have to be related through causality. For example, the number of rings in a molecule can be correlated with water solubility, but the number of rings is not physicochemically causing water solubility. Domain experts would recognize that the number of rings is likely a proxy for the amount of non-polar surface area. However, the number of rings is much easier to calculate than the polar surface area, so the feature engineer is faced with the cost-accuracy trade off presented in Figure 1. There is also a cost of including too many features or features with low relevance. Too many features causes the model to be hyper-flexible and respond too much to noise in the dataset. This is known as overfitting. Overfitting can be identified during model training when the training set error continues to decrease but the validation set error increases. For molecular systems, there are many vetted and standardized ways to represent a molecule which helps alleviate the feature design complexity. Molecular representations are further discussed in section 1.2.4.

After featurization, the data is ready to be split into training, test, and validation data sets. These sets are used during various stages of model development and it is crucial that no observations are duplicated across sets – a data point should only be in either the training, test, or validation set. The training set is typically the largest portion of data and is provided to the statistical learning algorithm for pattern identification. Training data is what the model "learns" from. Validation data is typically a small portion of the overall data and is used during training to measure training progress. The validation data is not explicitly learned from, but instead is used to make predictions during training or tuning to measure model performance for a given set of algorithm parameters.

The training data and validation data are passed to the statistical learning algorithm for model training. Generally, the algorithm is approximating the function which can predict the target properties given the feature vectors as input.

$$\mathbf{y} = f(\mathbf{X}) \tag{1.5}$$

Here, $\mathbf{y}$ is the vector of target properties and $\mathbf{X}$ is a matrix of the feature vectors. The target vector $\mathbf{y}$ consists of property observations, where $y_i$ represents the measurement for the $i$th observation. Correspondingly, $\mathbf{X}$ is composed of feature vectors for each observation, $\mathbf{x}_i$. The feature vector $\mathbf{x}_i$ has $p$ components, where $p$ is the number of features selected. The objective of training is to approximate $f$ to minimize a prediction error metric such as mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 \tag{1.6}$$

During training the algorithm adjusts learnable model weights until a completion criteria, which varies by algorithm, has been met.

There are also non-learnable algorithm parameters that can be adjusted by the modeler to increase model performance. These non-learnable parameters are called hyperparameters and the optimization of them is important to maximizing performance, but is a costly operation. Hyperparameter tuning begins with specifying which alogrithm parameters to adjust and over what range. This is referred to as the parameter grid. The objective is to find which point in the parameter grid provides the best model performance. This is a costly operation because the model must be retrained for each iteration of algorithm parameters. Additionally, this process requires a validation dataset for assessing the performance of each model iteration. Setting aside additional data for validation is unfavorable because the data is typically taken from the training set, reducing the amount of data the algorithm can learn from. The cross validation (CV) is a clever method for validation for hyperparameter optimization without reducing the total observations available for learning. Figure 3

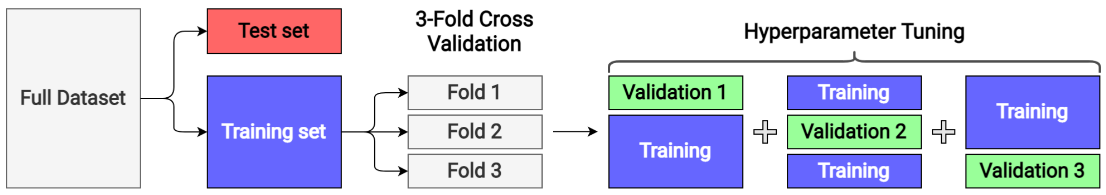


Figure 3: Depiction of 3-fold cross validation of model tuning. The training data set is broken up multiple times and used to train and validate the model. The model performance is based on the composite of the cross validation iterations. Figure adapted from [39].

shows how a dataset is split for 3-fold CV. The training set is split into three equal folds. During model tuning, for a given set of algorithm parameters, the model will be trained thrice – each iteration using a two folds for training and the third fold for predictions (validation). This process is repeated until all folds have been used as validation folds and the performance of the algorithm is evaluated on the composite performance of all three iterations.

Trained models are evaluated on how well they can predict previously unseen observations. During evaluation, the feature vectors from the test set data are passed to the model. The model makes a prediction for the observable and the predicted value is compared against the true observed value. The model performance is based on the error in the test set predictions in the same way as shown in equation 1.6.

### 1.2.4 Molecular Representations and Featurization

A central challenge of machine learning for chemical property predictions is how to represent molecules and their physics to a computer. The general challenge of how to represent molecules has been evolving for the last two centuries and a brief history of this evolution is provided here for context.

Dalton first published his theory of the atom in 1808[40] and the first molecular formula soon followed.[41] At the time, molecules were defined by the ratio of elements of which they are composed. Nearly 50 years later, theories were established regarding the connectivity of atoms within a molecule. In 1852, Frankland proposed the ground work for what would become valance bond theory.[42] The first two dimensional representations of molecules were produced by Alex Crum Brown in 1864.[43, 44] Brown represented molecules as circles (atoms) connected by lines (bonds), much the same way as molecules are drawn today. The reasoning for how and why atoms form bonds was not well understood until Pauling proposed hybridization in 1931.[45]

The 1950's ushered in major breakthroughs in molecular representation. Corey and Pauling developed the first space-filling representations of molecules. [46] They also used x-ray crystallography to discern the 3D arrangement of atoms in peptide sequences and identified α-helices and β-sheets. [47, 48] Two years later, Watson, Crick, and Franklin solved the 3D structure of DNA. [49] These advances brought molecular representations off of a piece of paper and into 3-dimensional space. 3D representations convey not only the connectivity of each atom, but also how each atom is positioned relative to other atoms. This is crucial for large molecules where atoms may not be bonded to each other, but still interact because they are near in space.

Concurrent with the development of 3D crystal structures was the development of nuclear magnetic resonance (NMR) spectroscopy, a powerful tool for probing the chemical environment of atoms within molecules. [50] NMR, along with mass and infrared spectroscopy, allows a researcher to determine the atomic composition of a molecule, the nature of the bonds between atoms, and the electronic environment around certain elements within the molecule. For the last 60 years, a full molecular representation has consisted of a 2D drawing, 3D structure, and a handful of spectra. These molecular puzzle pieces are often requirements for publication in order to confirm the identities of the chemicals of interest. As a result, the chemical literature contains millions of molecular drawings, structures, and spectra.

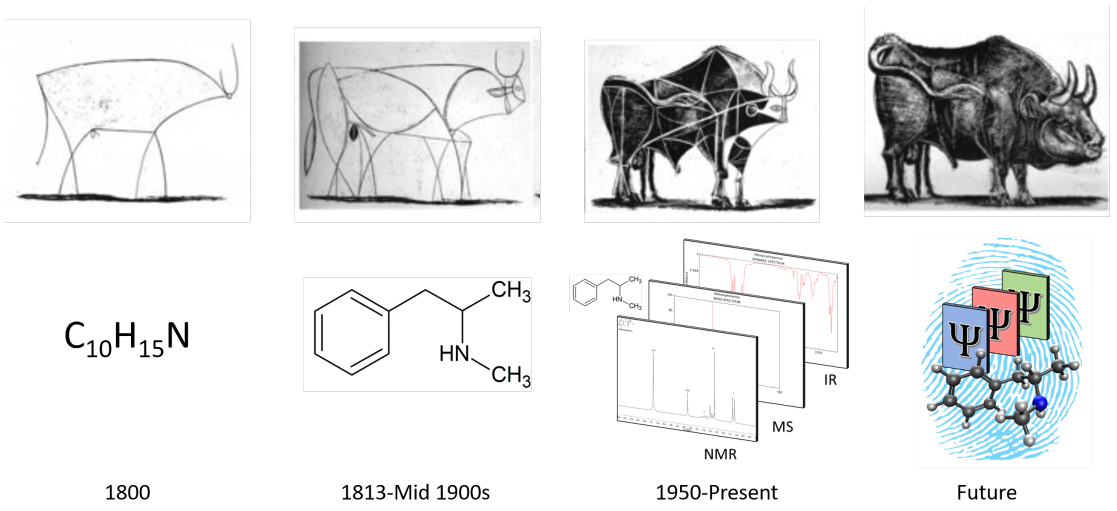


Figure 4: Images from Picasso's *The Bull* shows how an animal can be drawn with various levels of detail and complexity. Running parallel is the analogous progression of molecular representations. Each step along the progression adds more detail, allowing for a more complete and unique representation.

The manner in which molecular information is presented to machine learning algorithms is an ongoing research topic.[28, 51, 52, 53, 54] A molecular representation should be applicable to all of chemical space and should not be affected by symmetric operations such as rotation, reflection, translation, etc. Current molecular representations fall into three broad categories: discrete (e.g., text), continuous (e.g., vectors and tensors), and weighted graphs.[54]

SMILES is a 1-dimensional text representation of a molecule that is frequenty used in cheminformatics databases. It contains the atomic composition and connectivity of a molecule. Single, double and triple bonds can be encoded as well as formal charges. SMILES representation lacks specific electronic or 3D information. SMILES, which is a descrete representation, can be converted to a continuous representation for optimization of molecular properties. [28] There are multiple different formats for SMILES, not all of which produce a unique string for a given molecule, i.e a single molecule can be accurately represented by two different SMILES strings.

Graphical representations, where an atom is node and a bond is an edge, are used along side artificial neural networks (ANN).[33] Graphical representations can have weight vectors associated with each atom and bond. These weight vectors can encode the environment around each piece of the molecule by incorporating the types of atoms or bonds nearby.

Coulomb matricies have also been used as inputs to machine learning algorithms.[51] The are not affected by symmetry operations and require no bond information; only atomic coordinates and nuclear charges are required to generate a Coulomb matrix. Coulomb matrices run into issues when molecules with different numbers of atoms are considered. The dimensionality of the matrix will change based on the number of atoms in the molecule. Additionally, the are $N!$ different ways to express a molecule as a Coulomb matrix, where N is the number of atoms in the molecule.

A 3D geometry and charge is all that must be specified in order to perform first principle calculations on a molecule. With that in mind, one might expect a 3D geometry to peform well as a molecular representation for machine learning. Given the 3D coordinates and atomic charge of each atom, a machine learning algorithm could, in principle, learn the fundamentals of quantum mechanics and produce highly predictive models. Many groups are researching the possibility of using machine learning to predict the results of first princicple calculations.[55, 52, 19, 56] However, if the goal of the machine learning study is the prediction of complex properties such as reactivity, training machine learning to solve the Schrödinger equation is inefficient.

Although expensive to obtain, the information available once the Schrödinger equation is approximated is extremely valuable. As mentioned in section 1.1, the wave function contains all information about the system. Anheman et al. used molecular descriptors calculated using gas phase DFT as input to machine learning algorithms to predict the yield of a cross-coupling reaction.[32] They used an ensemble of energetic, electronic, and vibrational descriptors to describe the reagents, additives, and catalyst ligands in their reactive system. The resulting model was highly predictive because it incorporated much of the underlying physics involved in the phenomena of interest.

## 1.3 Navigating Design Space (Motivation)

Information is contextualized raw data or facts and knowledge is understanding how pieces of information are connected. Machine learning is being used to accelerate chemical knowledge generation by creating models based on perceived patterns in molecular property datasets. Datasets are collections of information as they contain a series of measurements contextualized with the measurement label, e.g solubility, and system label, e.g molecular identity. Model-generated chemical knowledge is leveraged for designing functional molecules [57, 28, 58, 59, 60], planning chemical syntheses [61, 62, 34, 63, 64], and predicting material properties [65, 66, 67, 68, 69, 19]. In addition to the model itself, there is information generated in the process of designing and applying machine learning models. Model design and meta information is often neglected, but information architecture can be leveraged to better utilize neglected information for the exploration chemical property modeling. Machine learning models for chemical property predictions are high dimension design challenges with many independent variables and dependent variables[26, 70]. Software libraries, such as DeepChem[71], RDKit[72], Scikit-Learn[73], and Tensorflow[74] streamline the model design and implementation process. However, the design space complexity remains.

Model design variables are chemical, mathematical, or data science based and they all have an impact on the performance of the model. Additionally, many independent variables depend upon another. For example, the choice of learning algorithm impacts the data pre-processing and the composition of the dataset influences how it is split into training, validation, and test sets [71]. Further dependencies exist between featurization technique, the molecules in the dataset, and their molecular fragments. These relationships need to be understood in order to gain knowledge about how design choices and molecular moieties affect model quality. Such modeling subtleties may be known to model developers, but are likely unknown to the model user and are lost when models are aggregated from literature for comparison.

Collecting more information about the choices made during model development can enable richer comparison hypotheses. Comparing the performance of new models against existing models is common practice in literature and necessary for measuring progress [75, 71, 76]. Currently, models are examined in isolation (Figure 5A). Workflows are distilled to a few headline parameters and a single prediction metric, such as mean absolute error (MAE) or root mean squared error (RMSE) for regression and area under the curve (AUC) for classification. The model performances are then aggregated for comparison, usually in the form of a bar chart. This evaluation approach reduces a model's complexity to a single metric, which is practical for quick, surface level assessment of model performance. However, condensing models discards rich details that can be used to connect models into a knowledge framework which can be queried for hypothesis testing. Figure 5B illustrates three machine learning models and their relationships as a graph. The graph format enables quick identification of shared model parameters, such as featurization technique for models 1 and 2, and different parameters, such as learning algorithm. Performance metrics, such as RMSE, are also included to enable quantitative evaluations. Information architecture can improve the utility of chemical modeling information, but requires information to be aggregated, standardized, organized, and contextualized.

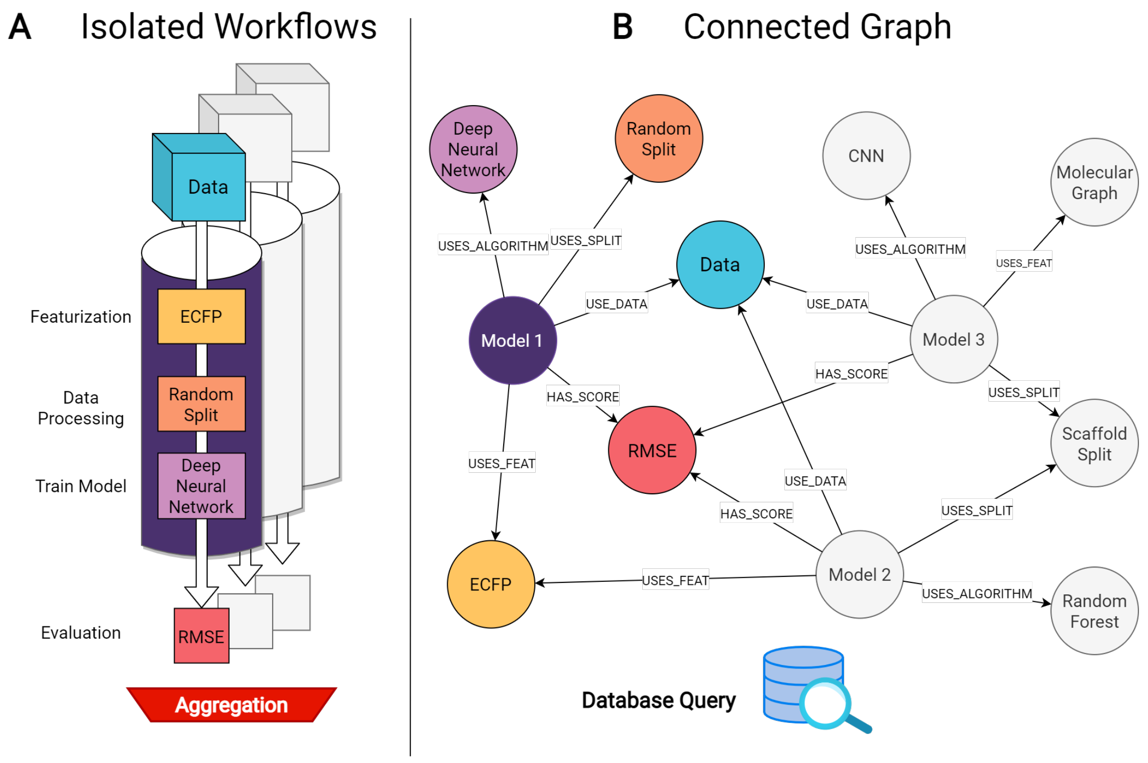


Figure 5: A: Current practice of model evaluation considers models in isolation and limits model comparison to aggregated performance metrics. B: The proposed approach connects model inputs and outputs as a descriptive graph. Database queries access the full model information and context for richer model analyses.

## 1.4 Graph Databases for Chemistry

### 1.4.1 Databases

Significant progress has been made towards the aggregation and standardization of chemical information. Public chemical databases, such as PubChem [77, 78], ZINC[79], ChEMBI [80, 81], GDB[82, 83], the Materials Project [84], and MoleculeNet.ai [71] provide the community with clean and trustworthy chemical datasets. Hastings et al. developed an standardized ontology for calculated properties (descriptors) of chemical entities on the semantic web. Ruusmann, Sild, and Maran created a database for quantitative structure activity relationship (QSAR) models called QSAR-DB [86]. QSAR-DB is a standardized collection of models and their meta-information accessible via the web (http://qsardb.org/). The database is searchable and provides functionality for visualizing model metrics. Understandably, QSAR-DB, like many model-based applications, is focused on individual model performance and does not readily facilitate connecting and comparing models [87].

### 1.4.2 Chemical Networks

Graphs are a popular and versatile structure for organizing and contextualizing chemical information. Molecules have a natural graph structure with atoms as nodes and bonds as edges, which is leveraged in cheminformatics and molecular featurization algorithms as a way to represent molecules to computers. [88, 89, 75] The framing of chemical reaction pathways as networks has been fruitful [90, 91]. Chemical reaction networks have been used to observe the evolution of organic chemistry[92, 93], identify the most influential synthons [94], and intelligently design synthetic routes [95, 96, 97, 98, 99]. The use of graphs to represent chemical transformations and material flows between equipment has enabled autonomous execution of lab-scale chemical syntheses[100, 101]. Macro-scale chemical networks, such as those involved in industrial parks and environmental chemistry, have also been modelled using graph databases [102, 103, 104, 105]. PubChem translated their curated information into the Resource Description Framework (RDF) format, resulting in the largest biochemical knowledge network which provides important biological context by connecting molecules, substances, proteins, assays, and diseases [77, 106, 107].

Functionally, graph databases can be instantiated as RDF or property graphs. Both graph technologies have been used for chemical and biological applications and their merits have been compared [108, 109, 110, 111, 112]. As such, we will briefly present our reasons for choosing the property graph model implemented via Neo4j and refer readers to more comprehensive comparisons [108, 109, 110, 111]. In property graph databases (PGDB), entities are represented as graph nodes and relationships between entities are represented as graph edges. Both nodes and edges have internal structure which enables properties to be stored to them. PGDB do not require a rigid schema or ontology and are therefore very flexible and extensible. The flexibility of the data structure and the simplicity of the Neo4j graph query language, Cypher,[113] provide an easy to use and intuitive graph database experience. Additionally, Neo4j software includes built-in visualization tools and graph data science algorithms. The property graph model implemented with Neo4j is a ready-to-use solution for rapid prototyping.

### 1.4.3 Statement of Work

In this paper, instances of machine learned chemical property models and their associated parameters are stored in a property graph database. The input molecules, molecular features, learning algorithm, algorithm parameters, and other inputs are related to a model instance. Likewise, the outputs of the model (time, predictions, uncertainty, feature importance, etc.) are also related to a model instance. Model instances are connected to each other through shared parameters. For example, two neural network models that use the same dataset would have two similar relationships. One relationship to the dataset node and the other relationship to the neural network learning algorithm node. By modeling chemical models as a graph, we can quickly identify overlap between models or datasets, evaluate models at multiple levels, and investigate complex relationships such as molecular fragments that are associated with high prediction error. Similar to how PubChem connects molecules to the biological processes in which they participate, the aim of this work is to connect molecules to the machine learning processes in which they participate.

# CHAPTER 2

# METHODOLOGY

## 2.1 Machine Learning Pipeline

### 2.1.1 Datasets

Six physicochemical regression datasets containing small molecules were collected from the literature (Table 2). All the datasets contain simplified molecular-input line-entry system (SMILES) molecular identifiers, which were validated with the RDKit Python package [72]. Molecules that failed to be resolved by RDKit were dropped from the datasets. This was rare since all datasets had already been curated by others [71, 114].

Table 2: Datasets used to train models. Size is number of unique molecular observations after processing.

| Dataset | Property | Task | Size | Source |
|---|---|---|---|---|
| Lipophilicity | logP | Regression | 4200 | moleculenet.ai |
| logP14k | logP | Regression | 14176 | Github |
| ESOL | logS | Regression | 1128 | moleculenet.ai |
| FreeSolv | Hydration free energy | Regression | 642 | moleculenet.ai |
| JAK2 | $IC_{50}$ | Regression | 1911 | Github |
| Flashpoint | Flashpoint | Regression | 9198 | Github |

### 2.1.2 Model Generation

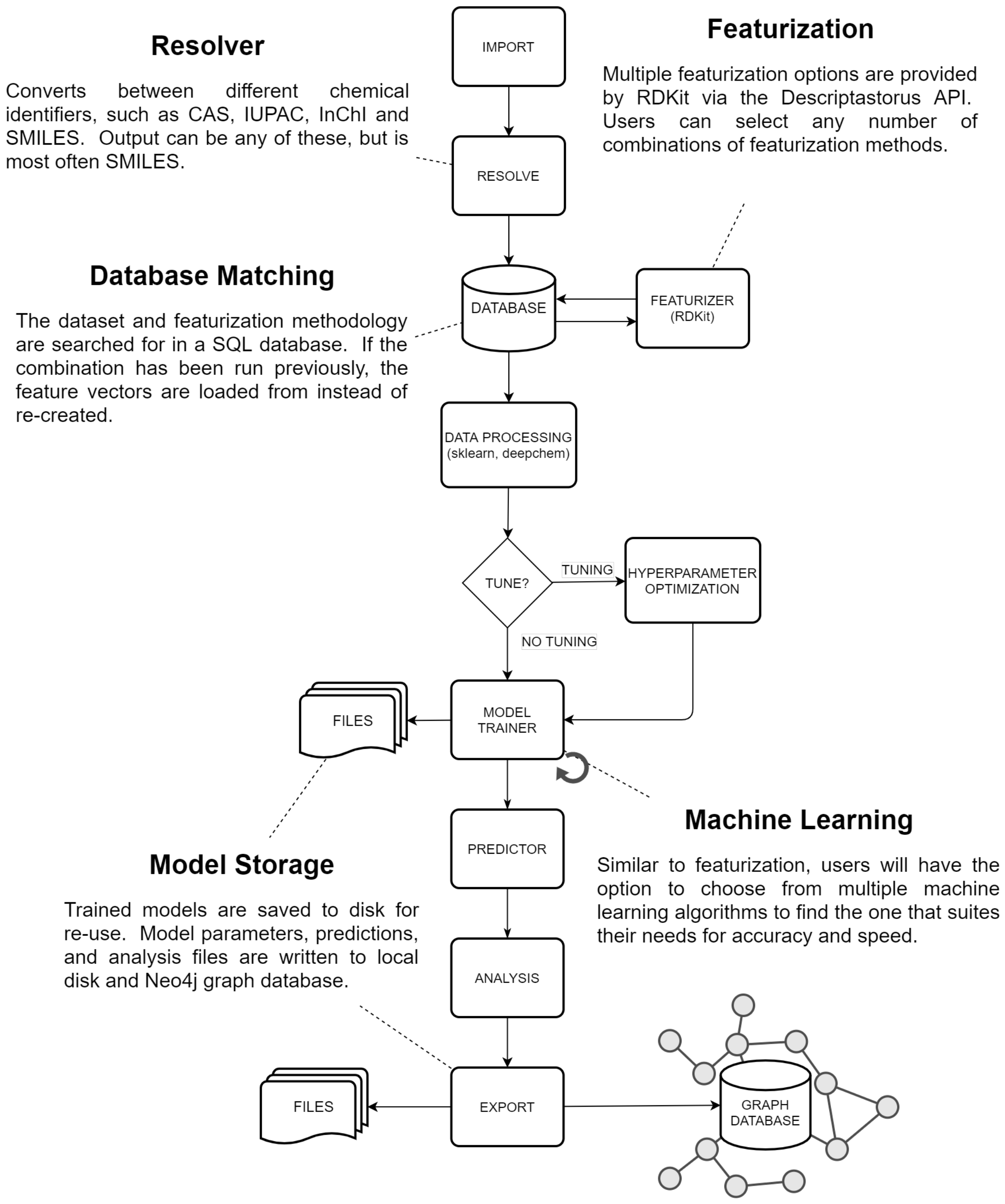
The Python pipeline used for model generation is depicted in Figure 6. Machine learning model instances were created with permutations of dataset, learning algorithm, molecular featurization, data scaling, data splitting, hyperparameters, and hyperparameter optimization techniques. Permutations expected to produce poor models, such us under-optimized hyperparameters, as were intentionally included for comparison purposes. In total, 215 machine learning models were generated and stored in the property graph.

For speed and simplicity, off-the-shelf Python learning algorithms were used. Random forest, gradient descent boost, k-nearest neightbor, Adaboost, and support vector machine were implemented with Scikit-Learn [73]. Dense neural networks were implemented with Tensorflow using the Keras application programming interfaces (API) [74, 115].

Molecules were featurized using Descriptastorus, an API for generating RDKit molecular descriptors and fingerprints [116, 72]. We implemented all descriptors supported by Descriptastorus except for the normalized RDKit 2D descriptors. Global molecular features, e.g RDKit molecular descriptors, and local molecular features, e.g fingerprints, were applied separately and in combination. The scaling of the featurized datasets was varied by different Scikit-Learn built-in scaling functions. The DeepChem python package was used for implementing random, scaffold, and stratified data splitting techniques. The portion of the dataset used for training, validation, and testing was also varied with the default being 70:10:20.

Cross validation (CV) hyperparameter tuning was performed using exhaustive grid and random search from the Scikit-Learn package[73] and Bayesian optimization from the skopt package [117]. The number of CV folds was set between 2 and 10. For a given learning algorithm, the same parameter search space was used across all optimization methods and the number of optimization iterations ranged from 5 to 100.

An ensemble approach was implemented for model predictions. All models were trained five times and the property predictions for individual molecules were averaged. Prediction uncertainty was calculated as the standard deviation of the five predictions.



**Resolver**

Converts between different chemical identifiers, such as CAS, IUPAC, InChI and SMILES. Output can be any of these, but is most often SMILES.

**Featurization**

Multiple featurization options are provided by RDKit via the Descriptastorus API. Users can select any number of combinations of featurization methods.

**Database Matching**

The dataset and featurization methodology are searched for in a SQL database. If the combination has been run previously, the feature vectors are loaded from instead of re-created.

**Machine Learning**

Similar to featurization, users will have the option to choose from multiple machine learning algorithms to find the one that suites their needs for accuracy and speed.

**Model Storage**

Trained models are saved to disk for re-use. Model parameters, predictions, and analysis files are written to local disk and Neo4j graph database.

Figure 6: Data process flow diagram.

All models were evaluated using RMSE, but MSE and $R^2$ were also calculated and included in the graph.

### 2.1.3 Storage & Export

All trained models were saved to local files, enabling them to be re-used without having to be retrained. Additionally, extensive model parameter and performance data was exported to Neo4j using the `py2neo` Python library. The information exported to Neo4j was also written to local JSON files. These JSON act as schema agnostic backup files which can be used to import models into the Neo4j database.

## 2.2 Graph Database

### 2.2.1 Software

The Neo4j software platform (version 4.1.0 Community edition) was used for creating the property graph database. Graph queries and visualization were carried out using Neo4j Browser and Neo4j Bloom, respectively. Model instance data was directly imported into the PGDB using the py2neo Python package. During the import process, each molecule was processed by RDKit to generate molecular fragments. These fragments were solely used in the graph and were not used in any property prediction model. Molecular fragments were included as nodes in the graph with a relationship to the parent molecule node.

### 2.2.2 Architecture

The graph schema was designed to preserve the uniqueness of model instances and to facilitate potential end-user queries (Figure 7). Each model instance is represented as a *MLModel* node and is connected to its major input and output entities. When a new model was run and imported to the graph, the necessary nodes were created

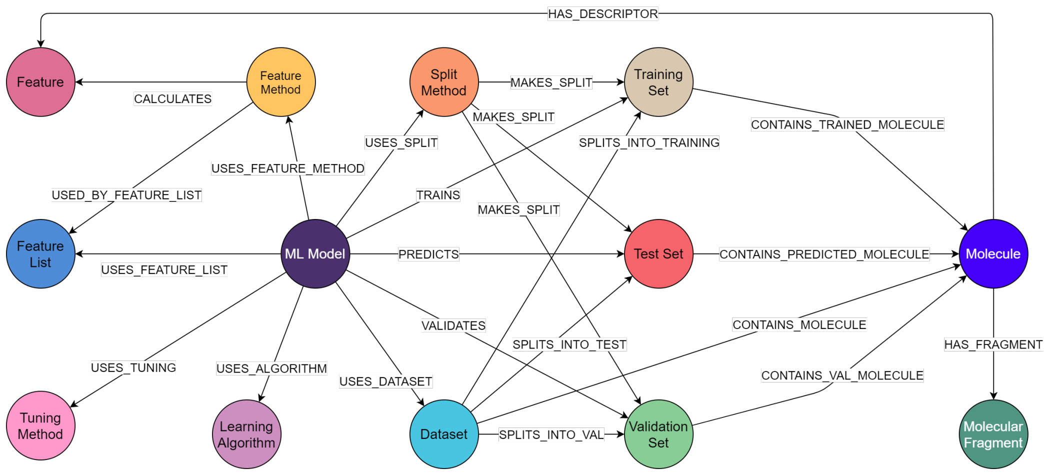
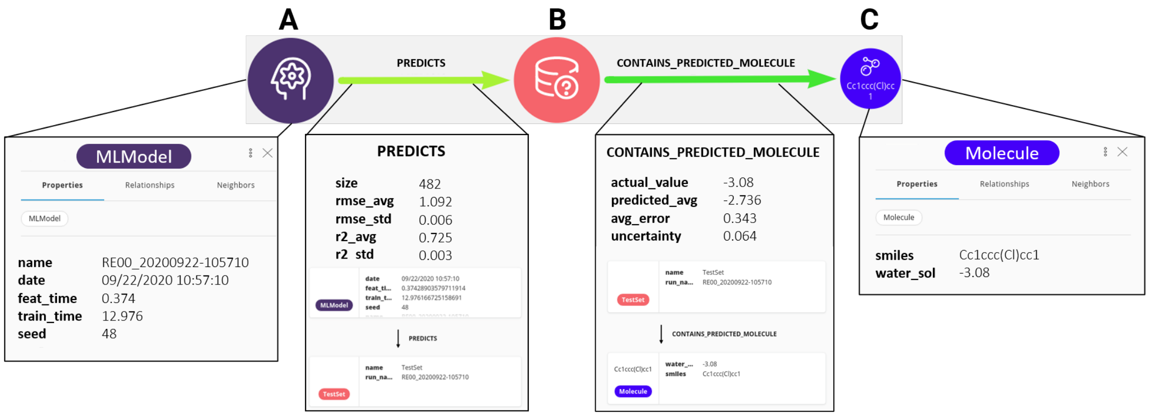


Figure 7: The database schema. The *ML Model* node is the backbone of the presented information architecture. Nodes represent entities in a machine learning workflow and relationships describe how those entities interact.

if they did already exist. A new model always created a unique *MLModel*, training set, test set, and if applicable, validation set nodes. Entities such as dataset, learning algorithm, or molecules only needed to be created once, and new models created new relationships to such nodes. Figure 7 shows multiple relationships between the molecule nodes and set nodes, but a molecule will only be related to either a training, test, or validation set node for a single given model.

The ability to store properties on both nodes and relationships was extensively used (Figure 8). Storing properties on relationships enabled tracability of model instance's specific values to global entities. For example, two unique model instances may predict the same molecule and property but with different predicted values. If the predictions were stored on the molecule node, two molecule nodes would be needed to capture the difference between the two model instances. However, this approach violated our design principle that a unique entity, the molecule, should appear only once in the graph. Instead, we stored the property predictions on the

*CONTAINS_PREDICTED_MOLECULE* relationship between the unique model's test set and a single molecule node. Further details on the Neo4j graph design and import process are discussed in the supplemental information.



MLModel

| | |
|---|---|
| name | RE00_20200922-105710 |
| date | 09/22/2020 10:57:10 |
| feat_time | 0.374 |
| train_time | 12.976 |
| seed | 48 |

PREDICTS

| | |
|---|---|
| size | 482 |
| rmse_avg | 1.092 |
| rmse_std | 0.006 |
| r2_avg | 0.725 |
| r2 std | 0.003 |

MLModel: date 09/22/2020 10:57:10; feat_ti... 0.37428903579711914; train_t... 12.976166725158691; seed 48

PREDICTS

TestSet: name TestSet; run_na... RE00_20200922-105710

CONTAINS_PREDICTED_MOLECULE

| | |
|---|---|
| actual_value | -3.08 |
| predicted_avg | -2.736 |
| avg_error | 0.343 |
| uncertainty | 0.064 |

TestSet: name TestSet; run_na... RE00_20200922-105710

CONTAINS_PREDICTED_MOLECULE

Molecule Cc1ccc(Cl)cc1: water_... -3.08; smiles Cc1ccc(Cl)cc1

Molecule

| | |
|---|---|
| smiles | Cc1ccc(Cl)cc1 |
| water_sol | -3.08 |

Figure 8: Example path relating a model node (A) to its test set (B) and the test set to a predicted molecule (C) via their respective prediction relationships. In the property graph model, properties can be stored on graph nodes and relationships to further describe and quantify the structures. Properties can be used for formatting the color and size of nodes and relationships.

### 2.2.3 Queries

Neo4j's Cypher language is used to execute all graph queries in Neo4j Desktop Browser. All figures containing sub-graphs are accompanied by Cypher commands for gathering the respective quantitative results.

For a given dataset, each molecule node's property prediction errors were averaged across incoming *CONTAINS_PREDICTED_MOLECULE* relationships and stored to the molecule node as a new *difficulty* property. The *difficulty* property was also propagated to outgoing *HAS_FRAGMENT* relationships between the molecule and its fragment nodes. The average of the *difficulty* property of the incoming relationships was used as a measure of a fragment's impact on error.

Fragment prevalence within the dataset can be determined by counting the number of distinct incoming relationships. Certain molecular motifs, such as aromatic rings, are extremely common and appear in both difficult and easy to predict molecules. To identify fragments that play a significant role in prediction errors, we first distinguish high error molecules and low error molecules. Using Cypher queries, we categorized molecules above a *cutoff* percentile of property prediction error as high error molecules and the lower percentile group as low error molecules. We then collected the fragments associated with each group of molecules and subtracted the most frequent fragments found in low error molecules from the most frequent fragments found in high error molecules.

This analysis has three parameters: *cutoff*, *hard limit*, and *easy limit*. The *cutoff* is the percentile threshold separating easy and hard molecules. *Hard limit* is a percentile value that adjusts the prevalence threshold for fragments associated with high error molecules. Increasing *hard limit* will include more uncommon fragments in the analysis results. *Easy limit* is a percentile value that adjusts prevalence threshold for fragments associated with low error molecules. Increasing *easy limit* will increase the number of fragments considered to be part of the background population.

# CHAPTER 3

# RESULTS

## 3.1 Python Pipeline

The developed machine learning pipeline automates the process of parameterizing, training, optimizing, and evaluating off-the-shelf machine learning models for chemical property prediction. The pipeline is efficient and easy to implement. Listing 3 shows example python code for running a gradient decent boost model with hyperparameter tuning. In only 8 lines of code and a few keyword arguments, a user can create a machine learning model and export it to a running Neo4j database. The feature vectors for each combination of dataset and featurization are stored in an SQL database. When required, the feature vectors are loaded from disk instead of generating them on the fly. The pipeline supports parallel computing across multiple CPU threads for Scikit-Learn algorithms and GPU computing for neural network models. The model inputs and outputs are automatically stored in a Neo4j graph database.

## 3.2 Graph Database

The graph database contains 83,000 nodes with 14 different node labels (Table 3) and 4 million relationships with 22 different relationship types (Table 4). There are 56 properties stored on the nodes and 55 properties stored on the relationships. Figure 9 shows a representative sample of the graph database; the full graph is too large to be visualized in Neo4j Bloom. The bulk of the nodes are molecules and fragments. The bulk of the relationships are between molecules and their fragments and between molecules and their calculated descriptors. Adding a single machine learning model to the graph will typically create three or four new unique nodes and thousands of

```
# initialize model with major parameters
model = MlModel(algorithm='gdb', dataset='Lipophilicity-ID.csv',
                target='exp', feat_meth=[0],
                tune=True, cv=10, opt_iter=25)
model.featurize()
model.data_split(split="random", test=0.1, scaler="standard")

with cd('output'):  # Have files write to output/ dir
    model.reg()
    model.run(tuner="bayes")  # train and tune model
    model.store()  # gather and write outputs
    model.org_files(zip_only=True)  # compress outputs

# export to Neo4j
model.to_neo4j(port="bolt://localhost:7687",
                username="neo4j", password="password")
```

Listing 1: Code used to run a ML model in the pipeline.

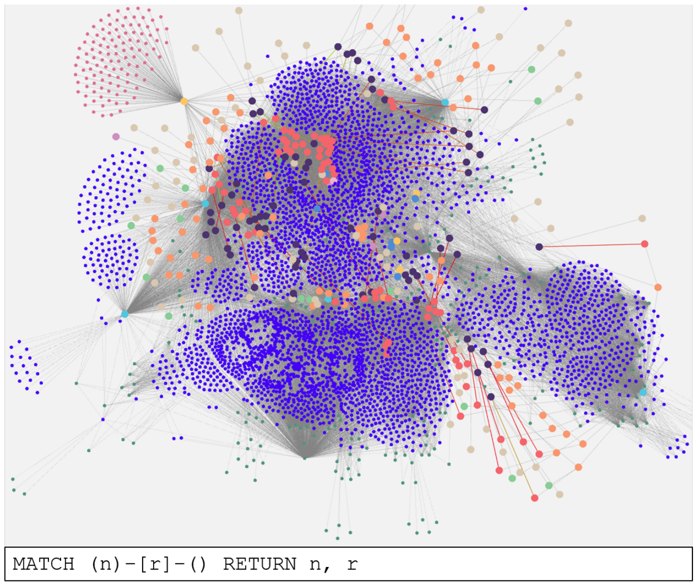
new relationships.

### 3.2.1 Model Evaluation and Comparison

The structure of the machine learning knowledge graph enables efficient and simple inter- and intra-model evaluation. Inter-model comparison is fundamental to identifying improvements in model development. Figure 10 shows a sub-graph of models which used the MoleculeNet lipophilicity dataset and their test sets [71]. This sub-graph provides a visual, semi-quantitative comparison of models through relationship color and weight. One can quickly identify top performing models by thick green arrows (A). A model's test set node can be expanded to display any number of the predicted molecules contained within (B). This intra-model evaluation is visually represented with a predicted versus actual parity plot. The graph architecture presented here contains the information typically used by researchers to evaluate their models and determine how they perform compared to others. The property graph

Table 3: Node labels. For each node label, the count of such nodes and the number of properties stored on the node are included.

| Node Label | Count | Properties |
|---|---:|---:|
| Fragment | 56,083 | 1 |
| Molecule | 25,940 | 6 |
| MLModel | 215 | 9 |
| TestSet | 215 | 3 |
| TrainSet | 215 | 3 |
| Feature | 110 | 1 |
| SplitMethod | 110 | 6 |
| RandomSplit | 105 | 6 |
| ValSet | 35 | 3 |
| FeatureList | 7 | 3 |
| DataSet | 6 | 6 |
| FeatureMethod | 6 | 1 |
| Algorithm | 6 | 2 |
| Tuning | 3 | 1 |



```
MATCH (n)-[r]-() RETURN n, r
```

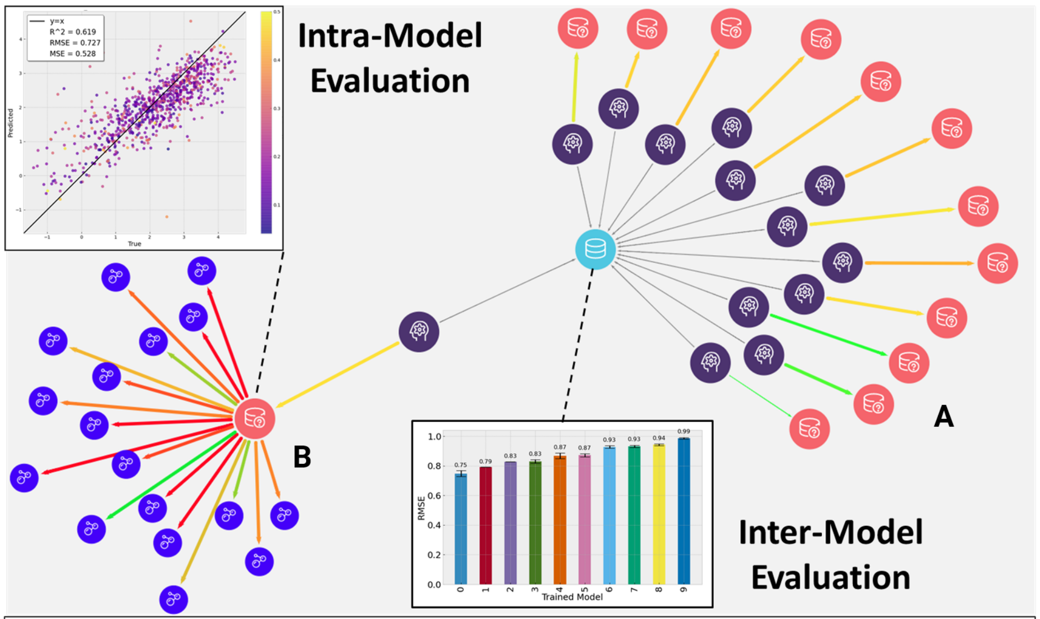
Figure 9: Snapshot of the graph database containing approximately 11,000 nodes and 20,000 relationships. All node labels and relationship types included.

Table 4: Relationship types. For each relationship type, the count of such relationships and the number of properties stored on that type of relationship are included.

| Relationship Type | Count | Properties |
|---|---:|---:|
| HAS_DESCRIPTOR | 1,888,897 | 1 |
| HAS_FRAGMENT | 1,374,420 | 0 |
| CONTAINS_TRAINED_MOLECULE | 521,981 | 1 |
| CONTAINS_PREDICTED_MOLECULE | 162,053 | 7 |
| CONTAINS_MOLECULE | 29,802 | 0 |
| CONTAINS_VALIDATED_MOLECULE | 15,170 | 1 |
| MAKES_SPLIT | 465 | 0 |
| USES_FEATURE_METHOD | 236 | 0 |
| TRAINS | 215 | 1 |
| PREDICTS | 215 | 13 |
| USES_FEATURE_LIST | 215 | 0 |
| USES_ALGORITHM | 215 | 6 |
| USES_DATASET | 215 | 0 |
| USES_SPLIT | 215 | 0 |
| SPLITS_DATASET | 215 | 0 |
| SPLITS_INTO_TEST | 215 | 0 |
| SPLITS_INTO_TRAIN | 215 | 0 |
| CALCULATES | 110 | 0 |
| USES_TUNING | 52 | 24 |
| SPLITS_INTO_VAL | 35 | 0 |
| VALIDATES | 35 | 1 |
| USED_BY_FEATURE_LIST | 8 | 0 |

provides the capability to query all models that have predicted a certain dataset, evaluate their performances, and determine the state of the art. Further information about interesting models and their predictions are just one graph edge away, which is easily accessed via the interactive graph interface or additional Cypher statements.

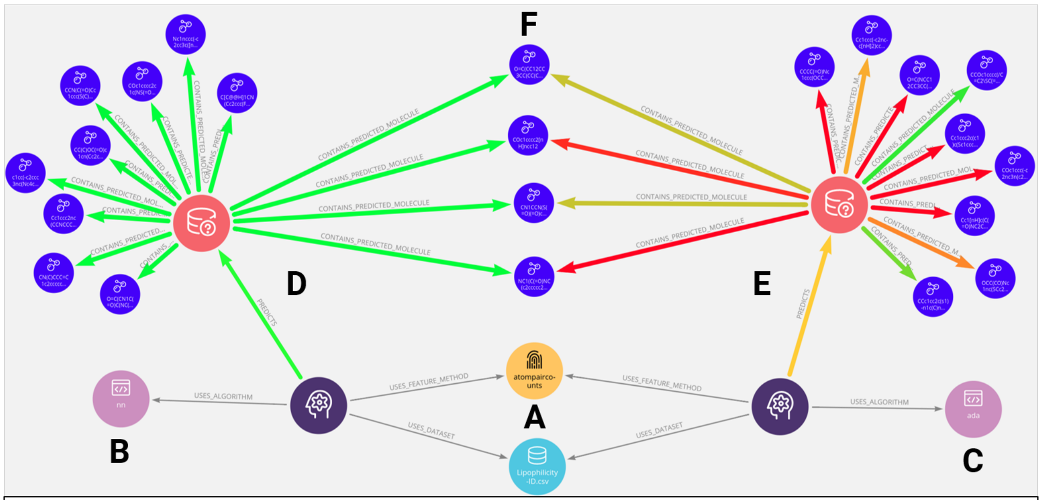
Models of interest can be further analyzed and compared in detail by expanding the *MLModel* node. This expansion produces the dataset, featurization method, learning algorithm, and data splits nodes connected to the model. Figure 11 demonstrates a head-to-head comparison enabled by the expansion of two model nodes. Both models use the same dataset and molecular featurization method (A), but differ



```
MATCH path =(:Molecule)<-[:CONTAINS_PREDICTED_MOLECULE]-(:TestSet)<-[:PREDICTS]-
(:MLModel)-[:USES_DATASET]->(:DataSet {data:"Lipophilicity-ID.csv"}) RETURN path
```

Figure 10: Graph structures which enable inter- (A) and intra-model prediction evaluations (B). The architecture of the graph is such that shifting from inter-model comparisons to intra-model analysis is just one additional edge traversal.

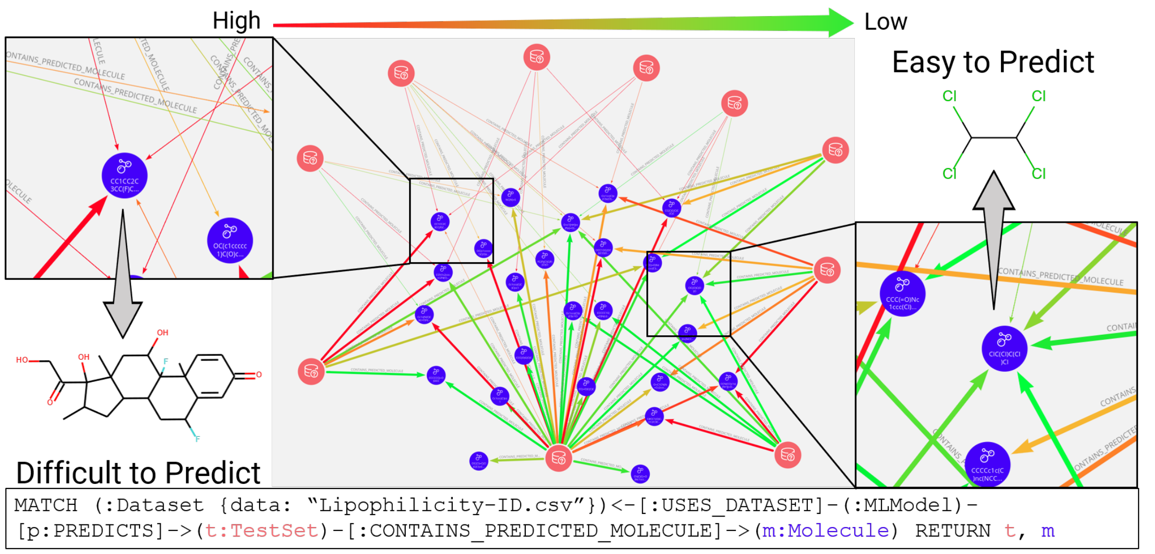
in their learning algorithm (B & C) and predictive performance (D & E). In this specific case, the model that uses a neural network (B) predicts its test set well, as shown by the green arrow (D). It also consistently performs well on individual molecules. In contrast, the model that uses Adaboost (C) has mediocre prediction performance on its test set and scattered performance on individual molecules, as shown by the variations in arrow colors (E). There are molecules in both test sets and the visualization shows specific prediction comparison at the molecular level (F). Direct model comparisons are enabled by the centrality of the *MLModel* node in the graph schema. A model's most important parameters and prediction results are connected directly to the *MLModel* node and molecular details are just two edges away from the central *MLModel* node.



```
MATCH (d:Dataset)<-[:USES_DATASET]-(m:MLModel)-[:USES_FEATURE_METHOD]->
(f: FeatureMethod) WITH d, m, f
MATCH (a:Algorithm)<-[:USES_ALGORITHM]-(m)-[:PREDICTS]->(t:TestSet)-
[:CONTAINS_PREDICTED_MOLECULE]->(mol:Molecule)RETURN d, m, f, t, mol, a
```

Figure 11: Direct comparisons of two models' parameters and performance. (A) Two models share a dataset (light blue node) and molecule featurization (yellow node). (B & C) The models use different learning algorithms. (E & F) The models' prediction performances (colored arrows). (F) Common test set molecules.

The knowledge graph can be used to investigate molecular property prediction difficulty and causality. Figure 12 shows a sub-graph containing 10 models' test set nodes and a subset of the overlapping predicted molecules. The visualization can be used to identify troublesome molecules, both by prediction error, shown by the arrow color, and by prediction uncertainty, shown by arrow thickness. In Figure 12 the inset on the left identifies a difficult to predict molecule based on all incoming arrows being red and the majority being thin, indicating high error and high deviations in the predictions. The opposite pattern is illustrated in the right inset for an easy to predict molecule. Once nodes of interest are identified, their chemical structures can be access by means of the SMILES stored as a property on the molecule nodes. The natural follow-up to identifying which molecules are difficult to predict is investigating



```
MATCH (:Dataset {data: "Lipophilicity-ID.csv"})<-[:USES_DATASET]-(:MLModel)-
[p:PREDICTS]->(t:TestSet)-[:CONTAINS_PREDICTED_MOLECULE]->(m:Molecule) RETURN t, m
```

Figure 12: Molecule overlap between ten models' test sets and the individual molecular predictions. The graph connects models to their predicted molecules via test sets, enabling analysis of molecular property prediction difficulty. Prediction relationships are formatted using prediction error (color) and uncertainty (thickness).

why molecules are difficult to predict. The graph database schema was designed with this query as a priority. The graph structure connects models' property predictions to molecules and molecules to their sub-molecular fragments. The prediction values and errors are stored on the prediction relationships, which enables queries to target specific portions of the graph based upon not only holistic model metrics, such as RMSE, but also molecular predictions.

### 3.2.2 Graph Queries for Molecular Error Analysis

The analyses thus far demonstrate that the graph database developed in this work is capable of facilitating visual evaluations of chemical property models at both the model level and the molecular level. Graph queries are a powerful and expressive means filter the graph to the user's specific interests and retrieve quantitative values. Figures 9-12 are accompanied by Cypher query statements for each respective analysis. The Cypher queries can be manually executed in Neo4j Browser or programmatically executed via one of Neo4j's APIs.

Cypher queries were developed which identify high error molecules for a given dataset (see Supplemental Info). The fragments associated with these high error molecules are collected and ubiquitous "background" fragments are removed from this population. The *cutoff* parameter was set to 0.9, designating the molecules in the top 10 percentile of prediction error as hard. Both the *hard limit* and *easy limit* were set to use the 1000 most frequent fragments from each population. Executing the analysis for the lipophilicty dataset produces the sub-graph shown in Figure 13. The sub-graph contains the difficult fragments and their parent molecules, which are sized based upon their average lipophilicity prediction errors. The graph visualization can help explore the causality of prediction error. That is to ask, is the molecule present in the analysis results because it contains difficult fragments or because the molecule presents challenges that are greater than the sum of its parts? By visual inspection, one can identify clusters within the graph where many molecules are connected to a group of common fragments (Figure 13 A, C, D). These clusters suggest that the central fragments are the source of the prediction challenge for the connected molecules. On the other hand, there are fragments that are only connected to one or a few molecules (Figure 13 B). This moiety indicates that the terminal fragments are not the source of the error and the molecule as a whole is difficult or contains other highly influential fragments. It is interesting to observe that there are no isolated populations of molecules and fragments in the results – there is a path that can connect any node to any other node.

Table 5 shows the top ten fragments returned by the Cypher queries sorted by the fragment's average parent molecules' error. The analysis reveals that sulfon-

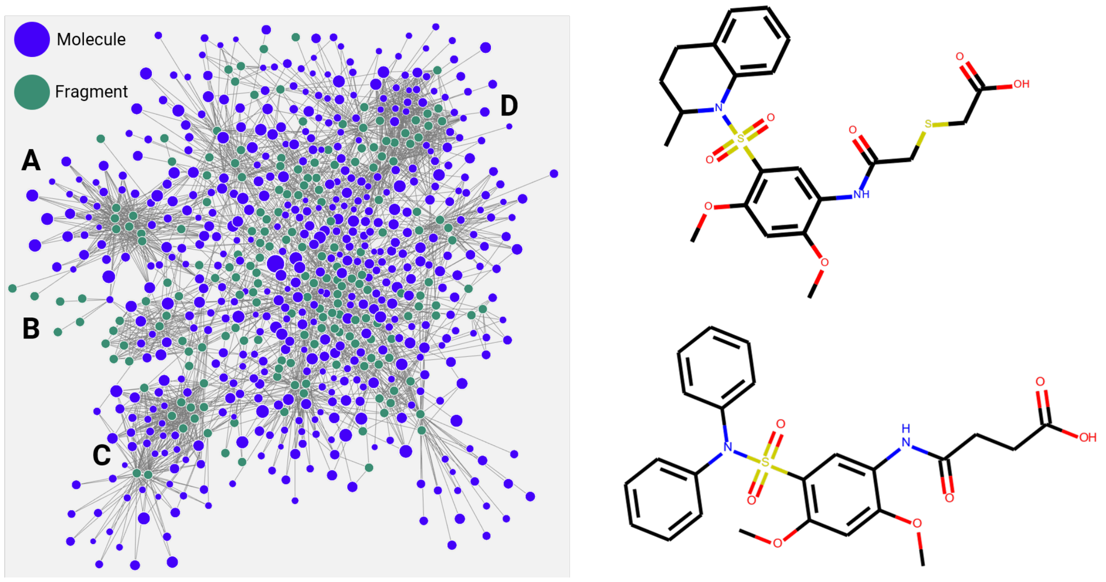


Figure 13: Fragment error analysis sub-graph and high error sulfonamides. (A, C, & D) clusters of molecules connected to a set of common fragments, indicating the fragments are the error origin. (B) Terminal fragments connected only to a single molecule, suggesting that those fragments are not the error origin. Molecule nodes are sized based upon their average prediction error – larger molecules have higher error.

amide groups are troublesome fragments. Sulfonamide containing compounds and their properties have been the subject of theoretical studies due to their use in anti-glaucoma therapies, an application in which balanced aqueous and lipid solubility is critical [118, 119, 120]. The $pK_a$ of sulfanomides typically ranges from 7-10 [119, 121] and the experimental values for the lipophilicity dataset were collected at a pH of 7.4 [71]. At experimental conditions, there would be a mixture of protonation states for a low $pK_a$ sulfonamide while a high $pK_a$ molecule would have a homogeneous population. The variation in protonation states would affect the partitioning of the molecule and is not communicated by the SMILES. This chemical challenge captured by the knowledge graph is inherent to the dataset and the way in which molecules are digitally represented.

The fragment analysis queries were automated via Python and the py2neo API, enabling the query parameter space to be explored (Figure 14). The objective of the analysis was to find fragments that were both prevalent and difficult. The large plateau in Figure 14 is caused by fragments related to a single high error molecule. In this region, the *hard limit* parameter is too loose – the analysis is including infrequent fragments. Decreasing the value of the *hard limit* parameter removes the infrequent fragments and returns fragments that are both moderately prevalent and related to difficult molecules. However, if the *hard limit* and *cutoff* parameters are too restrictive, no results will be returned. The *easy limit* parameter, which dictates how prevalent fragment must be to be considered background, must be non-zero to exclude extremely common fragments but has little affect the analysis except for at very high values.

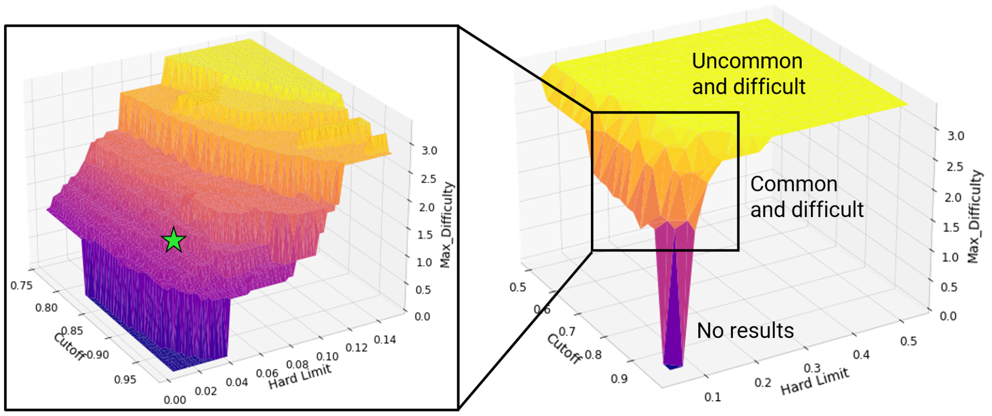
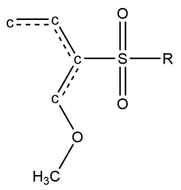
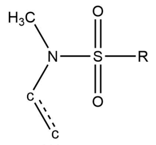
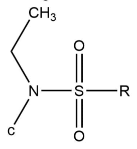
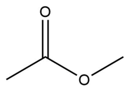
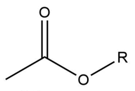
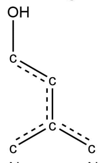
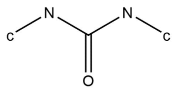
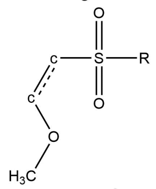
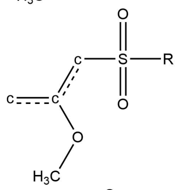
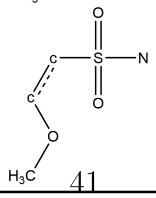


Figure 14: Parameter response surface for the fragment error analysis. The fragment error analysis parameters can be adjusted to locate regions containing fragments of interest. Star denotes region examined in previous figure and table.

The queries for analyzing molecular fragment impact on chemical property prediction error demonstrate the capabilities of the property graph for investigating complex hypotheses. The queries traverse four relationship types between four node labels. To apply filters and gather quantitative values, the queries access three node properties and two relationship properties. Despite the complexity of the hypothesis, the query execution is fast (96 ms), and the query commands are intuitive to a human reader (see Supplemental Info). The query parameters are easily tuned and additional filters can be applied to target specific molecular featurization methods or learning algorithms.

Table 5: Fragment error analysis results for the lipophilicity dataset. Fragments are sorted by average error of parent molecules. Query parameters: *cutoff*=0.9; *easy limit*=0.003; *hard limit*=0.03.

| Fragment | | Parent Molecules | Mean Error |
|---|---|---|---|
| ccc(c<-OMe>)S(<=O>)<=O> |  | 15 | 1.275 |
| ccN(C)S(<=O>)<=O> |  | 21 | 1.230 |
| CCN(c)S(<=O>)<=O> |  | 16 | 1.225 |
| CC<=O>OC |  | 27 | 1.212 |
| CC<=O>O |  | 30 | 1.205 |
| c<-O>cc(c)c |  | 21 | 1.205 |
| cNC<=O>Nc |  | 16 | 1.159 |
| c<-OMe>cS(<=O>)<=O> |  | 20 | 1.131 |
| cc<-OMe>cS(<=O>)<=O> |  | 20 | 1.131 |
| c<-OMe>cS(<=O>)<=O>N |  | 20 | 1.131 |

## 3.3 Lipophilicity Case Study

This case study is meant to demonstrate and further explicate how the property graph database is actually used for an fragment analysis. Recall that it is desirable to avoid contributions from ubiquitous fragments such as `ccc`. Because the investigation is only for a single molecular property the unscaled property and predictions values can be used in the analysis. However, care must be taken not to have any other molecular properties leak into the analysis, which can happen if a molecule has predictions for both lipophilicity and other properties. The methods used in this study is the same approach described in section 3.2.2 and is summarized by the following steps.

1. Remove all `difficulty` weights
2. Make new `difficulty` weights for the property of interest
3. Run the fragment analysis
4. Remove the `difficulty` weights

### 3.3.1 Query Setup

Cypher commands can be run in Batch using ';' to separate the commands, but the outputs will be suppressed. So the command that returns your results should be run by itself. With this in mind, the fragment analysis was broken into three sets of Cypher commands.

The first step in the analysis is to create a `difficulty` property based on molecule predictions. This will be used to categorize molecules as hard to predict. It is important to clear any prior `difficulty` properties from the nodes so that there is no contamination from other datasets. In Listing 2, the first code block removes any existing `difficulty` properties. The second code block creates a Cypher parameter for storing the dataset name. This parameter is similar to object oriented programming languages' variables in that it is globally accessible, which enables the dataset of interest to be changed in one place and have it mirrored throughout the rest of the analysis. The third code block in Listing 2 creates new `difficulty` weights based on the predicted values for the property of interest. This `difficulty` value is stored on the molecule nodes and the CONTAINS_PREDICTED_MOLECULE relationships.

```
// Delete old weights
MATCH (M:Molecule)-[f:HAS_FRAGMENT]->(F:Fragment)
REMOVE M.difficulty, f.difficulty
RETURN M, F, f;

// Set a parameter for the  Dataset you are interested in
// so you change it in one place only.
:param data => "Lipophilicity-ID.csv";


// Make new weights for Dataset
MATCH (D:DataSet{data: $data})-[:SPLITS_INTO_TEST]->
(T:TestSet)-[p:CONTAINS_PREDICTED_MOLECULE]->
(M:Molecule)-[f:HAS_FRAGMENT]->(F:Fragment)
WITH avg(p.average_error) as difficulty, f, M, F
SET M.difficulty = difficulty
SET f.difficulty = difficulty
RETURN M, F, f;
```

Listing 2: Cypher commands to prepare the graph for the fragment analysis.

Once the weights have been created, the analysis is ready to be executed. Listing 3 contains the Cypher commands for executing the anlaysis. The first analysis step finds the molecules in the dataset above the 90th percentile for `difficulty`. In other words, the 10% of molecules with the highest average error. The second code block in Listing 3 collects the "easy" molecules and their fragments. It then orders the fragments based upon their relationship count, which is a proxy for prevalence in the dataset. The third block performs the same actions but for the "hard" molecules. The fourth block uses the APOC library to subtract the easy fragments from the hard fragments. Block five takes the fragment results of the subtraction and finds the parent molecules in the dataset. The final code block in Listing 3 calculates statistics for the resulting fragments and parent molecules for reporting in a results table.

The final set of Cypher commands, shown in Listing 4, should be run after the analysis to remove the `difficulty` weights used in the analysis. This procedure is redundant with the deletion in the first set of commands, but a necessary precaution.

### 3.3.2 Query Results

Upon running the fragment analysis, the Cypher results were saved as a CSV. The Cypher query sorts the fragments first by the number of incoming relationships, i.e how many molecules have that fragment and then by the average prediction difficulty. The results are shown in Table 6. The results look pretty normal except for a single fragment that has huge `difficulty` values.

Table 6: Case study fragment analysis results sorted by number of fragment parent molecules.

| | fragment | number_of_rel | sum_difficulty | avg_difficulty |
|---|---|---|---|---|
| 0 | cc<-X>cNC | 68 | 5.230488e+01 | 0.769189 |
| 1 | ccnc[nH] | 67 | 5.157731e+01 | 0.769811 |
| 2 | c<=O>nCc | 67 | 4.938120e+01 | 0.737033 |
| 3 | ccC<=O>C | 65 | 4.125765e+06 | 63473.300050 |
| 4 | CCCC<-O> | 65 | 4.937961e+01 | 0.759686 |
| 5 | c<-X>cC<=O>NC | 65 | 4.860527e+01 | 0.747773 |
| 6 | ccCnc<=O> | 65 | 4.744537e+01 | 0.729929 |
| 7 | cc-c(c)s | 65 | 4.620461e+01 | 0.710840 |
| 8 | c<-O>c | 64 | 6.609076e+01 | 1.032668 |
| 9 | ncc[nH] | 64 | 5.058815e+01 | 0.790440 |

Table 7 shows the same results sorted by average difficulty as the primary sort

```
// Remove Common Fragments
MATCH (D:DataSet{data: $data})-[c:CONTAINS_MOLECULE]->(M:Molecule)
WITH  percentileCont(M.difficulty, 0.90) as cutoff

MATCH (D:DataSet{data: $data})-[c:CONTAINS_MOLECULE]->
(eM:Molecule)-[ef:HAS_FRAGMENT]->(eF:Fragment)
// easy molecules
WHERE eM.difficulty < cutoff
// gath frags and frequency
WITH eF, count(ef) as efreq, cutoff
//  limit to top n
ORDER BY efreq DESC LIMIT 1000
WITH  collect(eF) as easyFrags, cutoff

MATCH (D:DataSet{data: $data})-[c:CONTAINS_MOLECULE]->
(hM:Molecule)-[hf:HAS_FRAGMENT]->(hF:Fragment)
// hard molecules
WHERE hM.difficulty > cutoff
WITH hF, count(hf) as hfreq, easyFrags
ORDER BY hfreq DESC LIMIT 1000
WITH collect(hF) as hardFrags, easyFrags

// use APOC to do list intersect & subtraction
WITH apoc.coll.intersection(easyFrags, hardFrags) as overlap,
apoc.coll.subtract(hardFrags, easyFrags) as remain

// Find Mol-Frag pairs that have remaining fragments and in dataset
UNWIND remain as rFrags
MATCH (D:DataSet{data: $data})-[c:CONTAINS_MOLECULE]->
(M:Molecule)-[f:HAS_FRAGMENT]->(rFrags)
WITH M, rFrags
MATCH (M)-[f:HAS_FRAGMENT]->(rFrags)

// Get Difficulty Stats for Remaining Fragments
WITH rFrags.name as fragment, count(f) as number_of_rel,
sum(f.difficulty) as sum_difficulty,
sum(f.difficulty)/count(f) as avg_difficulty
RETURN fragment, number_of_rel, sum_difficulty, avg_difficulty
ORDER BY number_of_rel DESC, avg_difficulty DESC
```

Listing 3: Cypher commands to run the fragment analysis.

```
// Delete weights again
MATCH (D:DataSet{data: $data})-[:SPLITS_INTO_TEST]->
(T:TestSet)-[p:CONTAINS_PREDICTED_MOLECULE]->
(M:Molecule)-[f:HAS_FRAGMENT]->(F:Fragment)
WITH avg(p.average_error) as difficulty, f, M, F
REMOVE M.difficulty = difficulty
REMOVE f.difficulty = difficulty
RETURN M, F, f LIMIT 20;
```

Listing 4: Cypher commands to delete analysis weights to return graph to default state.

key. This view shows astronomical difficulty values and then sharp drop off. This indicates that a small number of molecule(s) were the culprit and likely the large errors were caused by some models that were totally haywire. The next step is to track the errors to the source.

Table 7: Case study fragment analysis results sorted by average difficulty.

| | fragment | number_of_rel | sum_difficulty | avg_difficulty |
|---|---|---|---|---|
| 176 | CC<=O>OC | 27 | 4.125751e+06 | 152805.587743 |
| 165 | CC<=O>O | 30 | 4.125754e+06 | 137525.143215 |
| 139 | C<=O>NC(C)C<=O> | 37 | 4.125748e+06 | 111506.693103 |
| 128 | CC[C@@H](C<=O>)N | 39 | 4.125749e+06 | 105788.431380 |
| 114 | CCCCC<=O> | 41 | 4.125754e+06 | 100628.138366 |
| 106 | CCC<-N> | 43 | 4.125756e+06 | 95947.815838 |
| 107 | CCC<-C(=O)O> | 43 | 4.125746e+06 | 95947.589936 |
| 62 | c<-N>cC<=O> | 54 | 4.125763e+06 | 76403.011438 |
| 48 | CC<-N> | 57 | 4.125768e+06 | 72381.886109 |
| 12 | cccC<=O>C | 63 | 4.125763e+06 | 65488.298270 |
| 3 | ccC<=O>C | 65 | 4.125765e+06 | 63473.300050 |
| 199 | ccc(c<-OMe>)S(<=O>)<=O> | 15 | 1.913052e+01 | 1.275368 |
| 186 | ccN(C)S(<=O>)<=O> | 21 | 2.582229e+01 | 1.229633 |
| 196 | CCN(c)S(<=O>)<=O> | 16 | 1.959603e+01 | 1.224752 |
| 187 | c<-O>cc(c)c | 21 | 2.530094e+01 | 1.204807 |

### 3.3.3 Query Follow-up

The previous tables show results for fragments, but fragments inherit their `difficulty` from the parent molecules. Listing 5 contains a Cypher query to rank molecules by their `difficulty`.

The results are displayed in Table 8. The results show that a single molecule is responsible for the huge fragment `difficulty`. With the SMILES and Node ID in hand, it is easy to visualize the molecule and find it in the graph to explore its connections.

Table 8: Case study results for high error molecules sorted by difficulty.

| | SMILES | Difficulty | Node_ID |
|---|---|---|---|
| 0 | CCCCCCCCCC(=O)N[C@@H](Cc1c[nH]c2ccccc12)C(=O)N... | 4.125720e+06 | 11205 |
| 1 | CC(N)(COP(=O)(O)O)C(=O)O | 3.334416e+00 | 9995 |
| 2 | Cc1oc(CN2CCNCC2)cc1C(=O)NCC12CC3CC(CC(C3)C1)C2 | 3.242000e+00 | 11514 |
| 3 | CN(C)c1ccc2nc3ccc(=[N+](C)C)cc-3sc2c1 | 3.177085e+00 | 10507 |
| 4 | N#Cc1ccc(-c2csc(Nc3ccc(O)cc3)n2)cc1 | 3.148000e+00 | 9108 |
| 5 | NCc1ccc(NC(=O)c2cc(Nc3ncccn3)c3cc(/C(N)=N/O)cc... | 3.143136e+00 | 9616 |
| 6 | COCCCOc1ccnc(C[S+]([O-])c2nc3ccccc3[nH]2)c1C | 3.141442e+00 | 11022 |
| 7 | COCCNCc1ccc(CCNC[C@H](O)c2ccc(O)c3[nH]c(=O)sc2... | 3.049231e+00 | 8433 |
| 8 | COc1ccc(-c2nc3c(NCCCNC(=O)c4ccccc4)c(Br)cnc3[n... | 2.943623e+00 | 10397 |
| 9 | O=C(O)c1ccc2cccc(O)c2n1 | 2.890912e+00 | 9533 |

Figure 15 show the molecule that causes massive errors. Chemical intuition is satisfied that this complex molecule might cause prediction challenges. Next is to find it in the graph and explore its context. This could be done with either Bloom or the Browser, but I prefer exploring in Bloom when I can.

Figure 16 shows a query for Bloom to execute which will find molecules by their SMILES. This search enables simple searching in Neo4j Bloom by simply typing `Molecule has "<your_SMILES>"`. The search yielded the singular molecule node, which is then expanded to show all related `TestSet` nodes, which enables one to see

```
MATCH (D:DataSet{data: $data})-[:CONTAINS_MOLECULE]->(M:Molecule)
WHERE EXISTS (M.difficulty)
WITH M.smiles as SMILES, M.difficulty as Difficulty, id(M) as Node_ID
RETURN DISTINCT SMILES, Difficulty, Node_ID
ORDER BY Difficulty DESC LIMIT 100
```

Listing 5: Cypher commands to return high difficulty molecules.

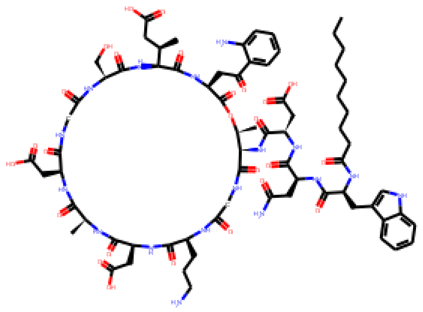
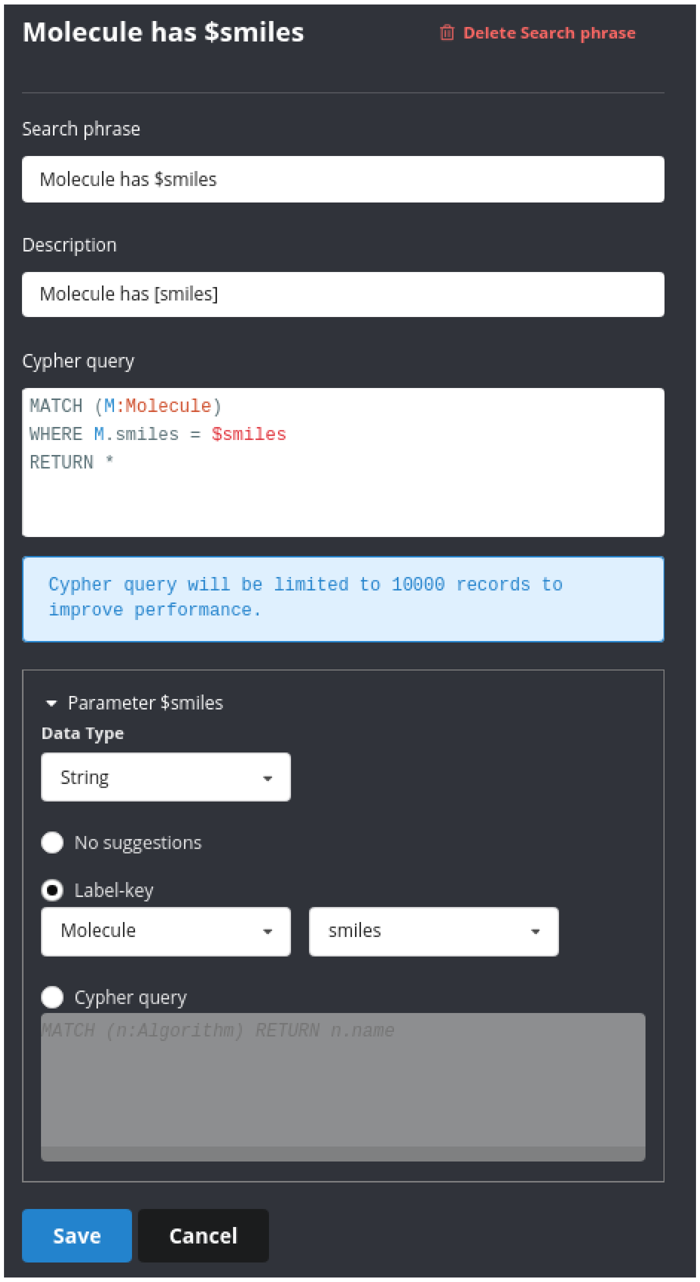


Figure 15: High error molecule from Lipophilicity fragment analysis.

the `CONTAINS_PREDICTED_MOLECULE` relationships, where the prediction values and errors are stored (Figure 17).


**Molecule has $smiles**

Search phrase: Molecule has $smiles

Description: Molecule has [smiles]

Cypher query:

```
MATCH (M:Molecule)
WHERE M.smiles = $smiles
RETURN *
```

```
Cypher query will be limited to 10000 records to
improve performance.
```

Parameter $smiles

Data Type: String

No suggestions

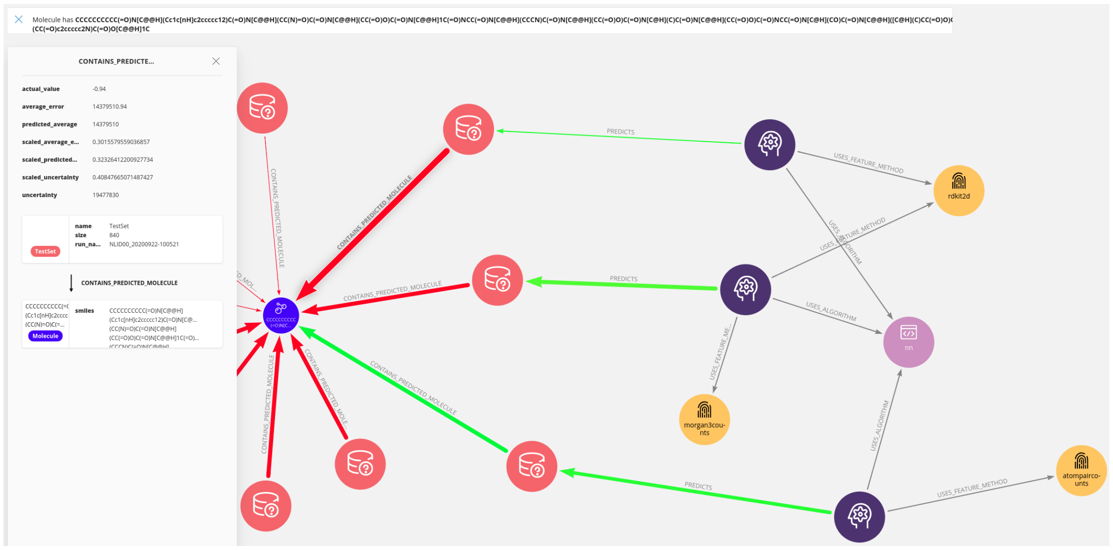
[✓ Label-key]: Molecule, smiles

Cypher query

```
MATCH (n:Algorithm) RETURN n.name
```

Figure 16: Bloom query panel. Custom query for searching nodes by SMILES shown.

This view allows one to see how many models have predicted lipophilicity for this



Molecule has **CCCCCCCCCC(=O)N[C@@H](Cc1c[nH]c2ccccc12)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H]1C(=O)NCC(=O)N[C@@H](CCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@H](CO)C(=O)N[C@@H]([C@H](C)CC(=O)O)C(CC(=O)c2ccccc2N)C(=O)O[C@@H]1C**

CONTAINS_PREDICTE...

| | |
|---|---|
| actual_value | -0.94 |
| average_error | 14379510.94 |
| predicted_average | 14379510 |
| scaled_average_e... | 0.3015579559036857 |
| scaled_predicted... | 0.32326412200927734 |
| scaled_uncertainty | 0.40847665071487427 |
| uncertainty | 19477830 |

TestSet

| | |
|---|---|
| name | TestSet |
| size | 840 |
| run_na... | NLID00_20200922-100521 |

CONTAINS_PREDICTED_MOLECULE

CCCCCCCCCC(=( (Cc1c[nH]c2cccc (CC(N)=O)C(=...
Molecule

| | |
|---|---|
| smiles | CCCCCCCCCC(=O)N[C@@H] (Cc1c[nH]c2ccccc12)C(=O)N[C@... (CC(N)=O)C(=O)N[C@@H] (CC(=O)O)C(=O)N[C@@H]1C(=O)... |

Figure 17: Expansion of macrocycle molecule node and the ML models that use neural networks to predict it's lipophilicity.

molecule (10) and how well each model performed. It turns out that two models had the astronomical values, had bad but relatively reasonable predictions, and a single model did okay. I was interested in the two terrible models and the alright model, so I expanded the those nodes to show what `Algorithm` and `FeaturizationMethod` they used. As can be seen in Figure 17, all three models of interest were neural networks. The two horrid models used `rdkit2d` featurization and one of them also used `morgan3counts`. The good model used `atompaircounts` featurization method.

Figure 18 shows further expansion of the models that predict the troublesome macro-cycle. This view shows that the one model that predicts it well is not only unique in its performance but also in its combination of algorithm and featurization method. The performant model is the only model to use a neural network and `atompaircounts`. Models that use a neural network but a different featurization method do not perform as well. Similarly, models that use `atompaircounts` and other learning algorithms do not perform well.

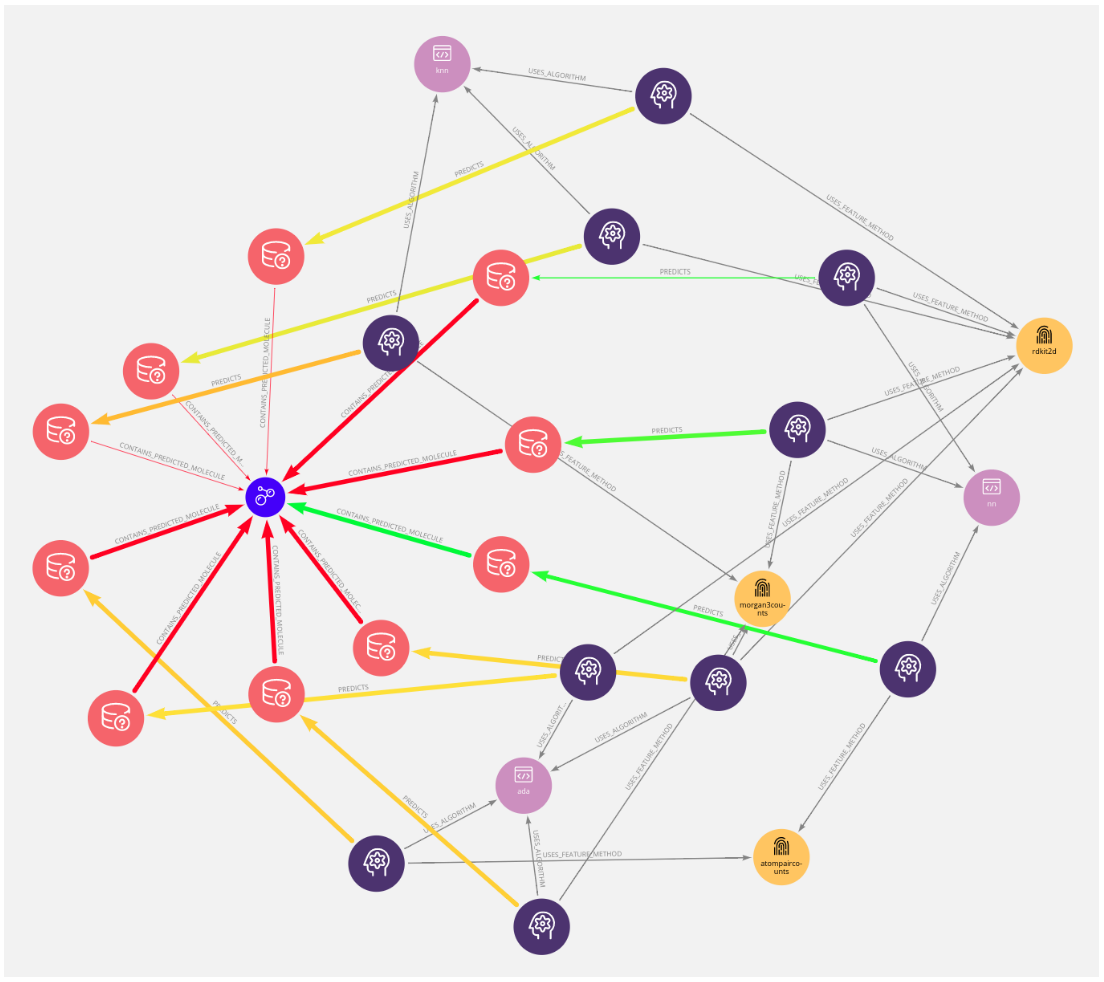


Figure 18: Full expansion of ML models predicting the macrocycle molecule.

### 3.3.4 Case Study Conclusion

In this case study, the graph is queried to investigate what molecular fragments cause error when predicting lipophilicity. The graph database is leveraged to design and execute a rather complex query to rank molecular fragments based upon their frequency in tough-to-predict molecules. The query returned anomalous results. The graph was then queried to find the source of the anomaly. Once the source, a structurally complex molecule, was identified, a simple graph query was used to locate the molecule in the graph visualization tool (Bloom). Bloom was then used to further explore the context around the anomaly which revealed potentially promising model parameters for improving prediction performance.

This whole process, from recognizing the issue in the fragment analysis to exploring the graph around the problematic molecule was quick – approximately 25 minutes. Moreover, only Python and Neo4j were used for the entire process. The efficiency is possible because of the connected structure of the graph database. The same analysis is possible with other relational databases structures, but would lack the speed, ease, visualizations possible by Neo4j's property graph database.

# CHAPTER 4

# CONCLUSION

The property graph database presented in this work organizes and contextualizes machine learning models for chemical property predictions. A portion of the chemical modeling design space is interrogated by implementing models with permutations in design parameters and storing the model inputs and outputs to the Neo4j property graph. Entities such as models, learning algorithms, molecular featurization methods, molecules, and fragments are represented as nodes in the graph and connected by relationships. Detailed information about the entities and their connections are stored as properties on the nodes and relationships, creating a rich information architecture. Explicit inclusion of molecules and model predictions of their chemical properties extends analytical capabilities to the molecular level. The structured information facilitates visual and quantitative evaluation of model performance. The interactive user interface enables intuitive exploration and manipulation of the graph.

Graph queries are implemented to test the hypothesis that the property graph architecture can be used to investigate which parts of a molecule are associated with high prediction error. The approach is validated by query results indicating that sulfonamide groups are associated with high lipophilicity prediction error. The graph queries, written in the Cypher language, are performant and human accessible. The current property graph uses off-the-shelf models to demonstrate utility. Moving forward, the flexible database schema can be extended to include state-of-the art learning and featurization approaches.

# Appendix A

## ABBREVIATIONS

| | |
|---|---|
| ML | Machine Learning |
| API | Application Programming Interface |
| SMILES | Simplified Molecular Input Line Entry System |
| MSE | Mean Squared Error |
| RMSE | Root Mean Squared Error |
| PGDB | Property Graph DataBase |
| CV | Cross Validation |
| RDF | Resource Description Framework |
| QSAR | Quantitative Structure Activity Relationship |
| MAE | Mean Absolute Error |
| AUC | Area Under the Curve |
| DFT | Density Functional Theory |
| QM | Quantum Mechanics |
| ANN | Artificial Neural Network |
| NMR | Nuclear Magnetic Resonance |
| HLB | Hydrophilic Lipophilic Balance |
| CMC | Critical Micelle Concentration |
| IFT | Interfacial Tension |
| SAFT | Statistical Associating Fluid Theory |
| SQL | Structured Query Language |
| JSON | JavaScript Object Notation |

# Appendix B

# CYPHER GRAPH QUERIES

## B.1 General Graph Queries

### Display all nodes and relationships within limit

```
MATCH (n)-[r]-()

RETURN n, r LIMIT 1000
```

### Search by node label

```
MATCH (model:MLModel)

RETURN model
```

### Search by label and property

```
MATCH (algo:Algorithm)<-[:USES_ALGORITHM]-(model:MLModel)

WHERE algo.name = "nn"

RETURN model
```

### Find molecules in multiple datasets

```
MATCH (m:Molecule)<-[r:CONTAINS_MOLECULE]-(d:DataSet)

WITH m, count(r) as rel_count, collect(d.data) as datasets

WHERE rel_count > 1

RETURN m.smiles, rel_count, datasets

ORDER BY rel_count DESC
```

**Find molecules in specific datasets**

```
MATCH (d1:DataSet)-[:CONTAINS_MOLECULE]->(mol:Molecule)<-[:CONTAINS_MOLECULE]-
(d2:DataSet)
WHERE d1.data = "water-energy.csv" AND d2.data = "ESOL.csv"
RETURN mol.smiles, d1.data, d2.data
```

**Sort molecules by prediction error**

```
MATCH (testset:TestSet)-[error:CONTAINS_PREDICTED_MOLECULE]->(molecule:Molecule)
RETURN molecule.smiles, error.scaled_average_error
ORDER BY error.scaled_average_error DESC
```

## B.2 Queries for Figures

The images used in the figures were created with Neo4j Bloom and were not the direct result of Cypher Queries. Below, we provide Cypher queries for accessing analogous results. The Cypher statements will provide more results than shown in the figures. Additionally, the underlying quantitative data is better accessed via Cypher than Bloom.

**Full Graph**

```
MATCH (n)-[r]-()
RETURN n, r LIMIT 1000
```

**Inter- Intra Model Comparisons**

```
MATCH path =(:Molecule)<-[:CONTAINS_PREDICTED_MOLECULE]-(:TestSet)
<-[:PREDICTS]-(:MLModel)-[:USES_DATASET]->(d:DataSet)
WHERE d.data = "Lipophilicity-ID.csv"
RETURN path
```

#### Comprehensive Model Compares

```
MATCH (d:Dataset)<-[:USES_DATASET]-(m:MLModel)-[:USES_FEATURE_METHOD]->
(f:FeatureMethod)
WITH d, m, f
MATCH (a:Algorithm)<-[:USES_ALGORITHM]-(m)-[:PREDICTS]->
(t:TestSet)-[:CONTAINS_PREDICTED_MOLECULE]->(mol:Molecule)
RETURN d, m, f, t, mol, a
```

#### Molecule Difficulty

```
MATCH (d:Dataset)<-[:USES_DATASET]-(:MLModel)-[p:PREDICTS]->
(t:TestSet)-[:CONTAINS_PREDICTED_MOLECULE]->(m:Molecule)
WHERE d.data = "Lipophilicity-ID.csv"
RETURN t, m
```

## B.3 Queries For Fragment Error Analysis

These queries make use of Cypher parameters, which are like storing variables for later use. This makes it easy to change the variable by only having to set it on place instead of many. In this case, the dataset name is stored as a parameter and later recalled using *$data*.

#### Prepare the Graph

```
// Delete old weights
MATCH (M:Molecule)-[f:HAS_FRAGMENT]->(F:Fragment)
REMOVE M.difficulty, f.difficulty
RETURN M, F, f;
```

```
// Set a parameter for the  Dataset you are interested in
// so you change it in one place only.
:param data => "Lipophilicity-ID.csv";

// Make new weights for Dataset
MATCH (D:DataSet{data: $data})-[:SPLITS_INTO_TEST]->
    (T:TestSet)-[p:CONTAINS_PREDICTED_MOLECULE]->
    (M:Molecule)-[f:HAS_FRAGMENT]->(F:Fragment)
WITH avg(p.average_error) as difficulty, f, M, F
SET M.difficulty = difficulty
SET f.difficulty = difficulty
RETURN M, F, f;
```

**Run Fragment Analysis**

The command below produces the fragment analysis and returns the number of relationships, the sum of their errors and the average error.

```
// Remove Common Fragments
MATCH (D:DataSet{data: $data})-[c:CONTAINS_MOLECULE]->(M:Molecule)
WITH  percentileCont(M.difficulty, $cutoff) as cutoff

MATCH (D:DataSet{data: $data})-[c:CONTAINS_MOLECULE]->
(eM:Molecule)-[ef:HAS_FRAGMENT]->(eF:Fragment)
WHERE eM.difficulty < cutoff
WITH eF, count(ef) as efreq, cutoff
ORDER BY efreq DESC LIMIT $final_easy_frag
WITH  collect(eF) as easyFrags, cutoff
```

```
MATCH (D:DataSet{data: $data})-[c:CONTAINS_MOLECULE]->
(hM:Molecule)-[hf:HAS_FRAGMENT]->(hF:Fragment)
WHERE hM.difficulty > cutoff
WITH hF, count(hf) as hfreq, easyFrags
ORDER BY hfreq DESC LIMIT $final_hard_frag
WITH collect(hF) as hardFrags, easyFrags

// use APOC to do list intersect & subtraction
WITH apoc.coll.intersection(easyFrags, hardFrags) as overlap,
apoc.coll.subtract(hardFrags, easyFrags) as remain

// Find Molecule-Fragment pairs that have the remaining fragments
UNWIND remain as rFrags
MATCH (D:DataSet{data: $data})-[c:CONTAINS_MOLECULE]->
(M:Molecule)-[f:HAS_FRAGMENT]->(rFrags)
WITH M, rFrags
MATCH (M)-[f:HAS_FRAGMENT]->(rFrags)

// Get Difficulty Stats for Remaining Fragments
WITH rFrags.name as fragment, count(f) as number_of_rel,
sum(f.difficulty) as sum_difficulty,sum(f.difficulty)/count(f) as avg_difficulty
RETURN fragment, number_of_rel, sum_difficulty, avg_difficulty
ORDER BY number_of_rel DESC, avg_difficulty DESC
```

### Clean Up Graph

**Run this at the end to clean up after yourself!**

```
// Delete weights again
MATCH (D:DataSet{data: $data})-[:SPLITS_INTO_TEST]->
(T:TestSet)-[p:CONTAINS_PREDICTED_MOLECULE]->
(M:Molecule)-[f:HAS_FRAGMENT]->(F:Fragment)
WITH avg(p.average_error) as difficulty, f, M, F
REMOVE M.difficulty = difficulty
REMOVE f.difficulty = difficulty
RETURN M, F, f LIMIT 20;
```

### Fragment Analysis Graph

Figure 8 is the results of the above analysis, but visualized in Bloom. The Cypher commands above were slightly modified to return the node IDs of all the fragments and molecules found in the analysis. Then Bloom was used to search for nodes by those IDs.

# REFERENCES

[1] Christopher J Cramer. *Essentials of computational chemistry: theories and models*. John Wiley & Sons, 2013.

[2] Wolfram Koch and Max C Holthausen. *A chemist's guide to density functional theory*. John Wiley & Sons, 2015.

[3] Jack E Baldwin. "Rules for ring closure". In: *Journal of the Chemical Society, Chemical Communications* 18 (1976), pp. 734–736.

[4] William G Griffin. "Classification of surface-active agents by" HLB"". In: *J. Soc. Cosmet. Chem.* (1949).

[5] William C Griffin. "Calculation of HLB values of non-ionic surfactants". In: *J. Soc. Cosmet. Chem.* 5 (1954), pp. 249–256.

[6] R Nagarajan and E Ruckenstein. "Theory of surfactant self-assembly: a predictive molecular thermodynamic approach". In: *Langmuir* (1991). ISSN: 0743-7463. DOI: 10.1021/la00060a012.

[7] R Nagarajan. "Micellization, mixed micellization and solubilization: the role of interfacial interactions". In: 26 (1986), pp. 205–264.

[8] R Nagarajan and E Ruckenstein. "Critical micelle concentration: A transition point for micellar size distribution A statistical thermodynamical approach". In: *J Colloid Interf Sci* 60.2 (1977), pp. 221–231. ISSN: 0021-9797. DOI: 10.1016/0021-9797(77)90282-X.

[9] R Nagarajan and E Ruckenstein. "Aggregation of amphiphiles as micelles or vesicles in aqueous media". In: *Journal of Colloid and Interface Science* 71.3 (1979), pp. 580–604. ISSN: 0021-9797. DOI: 10.1016/0021-9797(79)90331-X.

[10] E Ruckenstein and R Nagarajan. "Aggregation of amphiphiles in nonaqueous media". In: *The Journal of Physical Chemistry* (1980). DOI: 10.1021/j100448a013.

[11] C Tanford. "The hydrophobic effect and the organization of living matter". In: *Science* (1978). DOI: 10.1126/science.653353.

[12] C Tanford. "Theory of micelle formation in aqueous solutions". In: *The Journal of Physical Chemistry* 78.24 (1974), pp. 2469–2479. ISSN: 0022-3654. DOI: 10.1021/j100617a012.

[13] John W Cahn and John E Hilliard. "Free Energy of a Nonuniform System. I. Interfacial Free Energy". In: 28.2 (1958), pp. 258–267. ISSN: 0021-9606. DOI: 10.1063/1.1744102.

[14] Sudhakar Puvvada and Daniel Blankschtein. "Molecular-thermodynamic approach to predict micellization, phase behavior and phase separation of micellar solutions. I. Application to nonionic surfactants". In: 92.6 (1990), pp. 3710–3724. ISSN: 0021-9606. DOI: 10.1063/1.457829.

[15] Sudhakar Puvvada and Daniel Blankschtein. "Theoretical and experimental investigations of micellar properties of aqueous solutions containing binary mixtures of nonionic surfactants". In: 96.13 (1992), pp. 5579–5592. DOI: 10.1021/j100192a071.

[16] Le Wang et al. "Modeling micelle formation and interfacial properties with iSAFT classical density functional theory". In: 146.12 (2017), p. 124705. ISSN: 0021-9606. DOI: 10.1063/1.4978503.

[17] Jianzhong Wu. “Density functional theory for chemical engineering: From capillarity to soft materials”. In: *Aiche J* 52.3 (2006), pp. 1169–1193. ISSN: 1547-5905. DOI: 10.1002/aic.10713.

[18] Justin S Smith et al. “Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning”. In: *Nature communications* 10.1 (2019), pp. 1–8.

[19] Justin S Smith, Olexandr Isayev, and Adrian E Roitberg. “ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost”. In: *Chemical science* 8.4 (2017), pp. 3192–3203.

[20] Abe Stern. *AI in Chemistry*. URL: https://ssl.lvl3.on24.com/event/20/69/14/3/rt/1/documents/resourceList1568691609246/hpcngcaiinquantumchemistryde pdf.

[21] B Creton and Nieto-Draghi C. “Prediction of surfactants’ properties using multiscale molecular modeling tools: A review”. In: *Oil & Gas Science ...* (2012). ISSN: 1294-4475. DOI: 10.2516/ogst/2012040.

[22] Jiwei Hu, Xiaoyi Zhang, and Zhengwu Wang. “A review on progress in QSPR studies for surfactants.” In: *Int J Mol Sci* 11.3 (2010), pp. 1020–47. ISSN: 1422-0067. DOI: 10.3390/ijms11031020.

[23] ZW Wang, DY Huang, and SP Gong. “Prediction on Critical Micelle Concentration of Nonionic Surfactants in Aqueous Solution: Quantitative Structure-Property Relationship Approach”. In: *Chinese Journal of ...* (2003).

[24] Zhengwu Wang et al. “A quantitative structure-property relationship study for the prediction of critical micelle concentration of nonionic surfactants”. In:

197.1-3 (2002), pp. 37–45. ISSN: 0927-7757. DOI: 10.1016/S0927-7757(01)00812-3.

[25] Zheng-Wu Wang et al. “Prediction on critical micelle concentration of anionic surfactants in aqueous solution: quantitative structure-property relationship approach”. In: *ACTA CHIMICA SINICA-CHINESE EDITION-* 60.9 (2002), pp. 1548–1552.

[26] Keith T Butler et al. “Machine learning for molecular and materials science”. In: *Nature* 559.7715 (2018), pp. 547–555.

[27] Seyed Mohamad Moosavi et al. “Capturing chemical intuition in synthesis of metal-organic frameworks”. In: *Nature communications* 10.1 (2019), p. 539.

[28] Rafael Gómez-Bombarelli et al. “Automatic chemical design using a data-driven continuous representation of molecules”. In: *ACS central science* 4.2 (2018), pp. 268–276.

[29] Kyoungmin Min et al. “Machine learning assisted optimization of electrochemical properties for Ni-rich cathode materials”. In: *Scientific reports* 8.1 (2018), p. 15778.

[30] Paul Raccuglia et al. “Machine-learning-assisted materials discovery using failed experiments”. In: *Nature* 533.7601 (2016), p. 73.

[31] Wiktor Beker et al. “Prediction of Major Regio-, Site-, and Diastereoisomers in Diels–Alder Reactions by Using Machine-Learning: The Importance of Physically Meaningful Descriptors”. In: *Angewandte Chemie International Edition* 58.14 (2019), pp. 4515–4519.

[32] Derek T Ahneman et al. “Predicting reaction performance in C–N cross-coupling using machine learning”. In: *Science* 360.6385 (2018), pp. 186–190.

[33] Connor W Coley et al. “A graph-convolutional neural network model for the prediction of chemical reactivity”. In: *Chemical science* 10.2 (2019), pp. 370–377.

[34] Connor W Coley et al. “Prediction of organic reaction outcomes using machine learning”. In: *ACS central science* 3.5 (2017), pp. 434–443.

[35] Anna Tomberg, Magnus J Johansson, and Per-Ola Norrby. “A Predictive Tool for Electrophilic Aromatic Substitutions Using Machine Learning”. In: *The Journal of organic chemistry* (2018).

[36] Sara Szymkuć et al. “Computer-Assisted Synthetic Planning: The End of the Beginning”. In: *Angewandte Chemie International Edition* 55.20 (2016), pp. 5904–5937.

[37] Paul D Leeson and Brian Springthorpe. “The influence of drug-like concepts on decision-making in medicinal chemistry”. In: *Nature reviews Drug discovery* 6.11 (2007), pp. 881–890.

[38] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. “Deep reinforcement learning for de novo drug design”. In: *Science advances* 4.7 (2018), eaap7885.

[39] Laurent Gatto. *An Introduction to Machine Learning with R*. URL: `https://lgatto.github.io/IntroMachineLearningWithR/`.

[40] John Dalton. *A New System of Chemical Philosophy: Pt. 1/2*. Dawson, 1808.

[41] Jöns Jacob Berzelius. “On the Nature of Muriatic Acid”. In: *Annals of Philosophy* 2.10 (1813), pp. 254–259.

[42] Edward Frankland. “XIX. On a new series of organic bodies containing metals”. In: *Philosophical Transactions of the Royal Society of London* 142 (1852), pp. 417–444.

[43] A. Crum Brown. “XLIV.—On the Theory of Isomeric Compounds”. In: *Transactions of the Royal Society of Edinburgh* 23.3 (1864), pp. 707–719. DOI: 10.1017/S0080456800020007.

[44] Alexander Crum Brown. “2. On the Classification of Chemical Substances by Means of Generic Radicals”. In: *Proceedings of the Royal Society of Edinburgh* 5 (1866), pp. 561–562.

[45] Linus Pauling. “The nature of the chemical bond. Application of results obtained from the quantum mechanics and from a theory of paramagnetic susceptibility to the structure of molecules”. In: *Journal of the American Chemical Society* 53.4 (1931), pp. 1367–1400.

[46] Robert B Corey and Linus Pauling. “Molecular models of amino acids, peptides, and proteins”. In: *Review of Scientific Instruments* 24.8 (1953), pp. 621–627.

[47] Linus Pauling, Robert B Corey, and Herman R Branson. “The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain”. In: *Proceedings of the National Academy of Sciences* 37.4 (1951), pp. 205–211.

[48] Linus Pauling and Robert B Corey. “The pleated sheet, a new layer configuration of polypeptide chains”. In: *Proceedings of the National Academy of Sciences of the United States of America* 37.5 (1951), p. 251.

[49] James D Watson and Francis HC Crick. “Molecular structure of nucleic acids”. In: *Nature* 171.4356 (1953), pp. 737–738.

[50] JT Arnold and ME Packard. “Variations in absolute chemical shift of nuclear induction signals of hydroxyl groups of methyl and ethyl alcohol”. In: *The Journal of Chemical Physics* 19.12 (1951), pp. 1608–1609.

[51] Katja Hansen et al. “Assessment and validation of machine learning methods for predicting molecular atomization energies”. In: *Journal of Chemical Theory and Computation* 9.8 (2013), pp. 3404–3419.

[52] Felix A Faber et al. “Prediction errors of molecular machine learning models lower than hybrid DFT error”. In: *Journal of chemical theory and computation* 13.11 (2017), pp. 5255–5264.

[53] G Skoraczyński et al. “Predicting the outcomes of organic reactions via machine learning: are current descriptors sufficient?” In: *Scientific reports* 7.1 (2017), p. 3582.

[54] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. “Inverse molecular design using machine learning: Generative models for matter engineering”. In: *Science* 361.6400 (2018), pp. 360–365.

[55] O Anatole Von Lilienfeld. “First principles view on chemical compound space: Gaining rigorous atomistic control of molecular properties”. In: *International Journal of Quantum Chemistry* 113.12 (2013), pp. 1676–1689.

[56] Chenru Duan et al. “Learning from Failure: Predicting Electronic Structure Calculation Outcomes with Machine Learning Models”. In: *Journal of chemical theory and computation* (2019).

[57] Gisbert Schneider and David E Clark. "Automated de novo drug design: are we nearly there yet?" In: *Angewandte Chemie International Edition* 58.32 (2019), pp. 10792–10803.

[58] Marcus Olivecrona et al. "Molecular de-novo design through deep reinforcement learning". In: *Journal of cheminformatics* 9.1 (2017), p. 48.

[59] Marwin HS Segler et al. "Generating focused molecule libraries for drug discovery with recurrent neural networks". In: *ACS central science* 4.1 (2018), pp. 120–131.

[60] Niek van Hilten, Florent Chevillard, and Peter Kolb. "Virtual compound libraries in computer-assisted drug discovery". In: *Journal of chemical information and modeling* 59.2 (2019), pp. 644–651.

[61] Connor W Coley et al. "A robotic platform for flow synthesis of organic compounds informed by AI planning". In: *Science* 365.6453 (2019), eaax1566.

[62] Connor W Coley et al. "Computer-assisted retrosynthesis based on molecular similarity". In: *ACS central science* 3.12 (2017), pp. 1237–1245.

[63] Thomas J Struble et al. "Current and Future Roles of Artificial Intelligence in Medicinal Chemistry Synthesis". In: *Journal of Medicinal Chemistry* (2020).

[64] Hanyu Gao et al. "Combining retrosynthesis and mixed-integer optimization for minimizing the chemical inventory needed to realize a WHO essential medicines list". In: *Reaction Chemistry & Engineering* 5.2 (2020), pp. 367–376.

[65] Zach Jensen et al. "A machine learning approach to zeolite synthesis enabled by automatic literature data extraction". In: *ACS central science* 5.5 (2019), pp. 892–899.

[66] Olexandr Isayev et al. “Universal fragment descriptors for predicting properties of inorganic crystals”. In: *Nature communications* 8.1 (2017), pp. 1–12.

[67] Kirstin Alberi et al. “The 2019 materials by design roadmap”. In: *Journal of Physics D: Applied Physics* 52.1 (2018), p. 013001.

[68] Rafał Roszak et al. “Rapid and Accurate Prediction of p K a Values of C–H Acids Using Graph Convolutional Neural Networks”. In: *Journal of the American Chemical Society* 141.43 (2019), pp. 17142–17149.

[69] Rafael Gómez-Bombarelli et al. “Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach”. In: *Nature materials* 15.10 (2016), pp. 1120–1127.

[70] Artem Cherkasov et al. “QSAR modeling: where have you been? Where are you going to?” In: *Journal of medicinal chemistry* 57.12 (2014), pp. 4977–5010.

[71] Zhenqin Wu et al. “MoleculeNet: a benchmark for molecular machine learning”. In: *Chemical science* 9.2 (2018), pp. 513–530.

[72] *RDKit: Open-source cheminformatics.* URL: `http://www.rdkit.org` (visited on 05/26/2019).

[73] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[74] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[75] Connor W Coley et al. "Convolutional embedding of attributed molecular graphs for physical property prediction". In: *Journal of chemical information and modeling* 57.8 (2017), pp. 1757–1772.

[76] Kevin Yang et al. "Analyzing learned molecular representations for property prediction". In: *Journal of chemical information and modeling* 59.8 (2019), pp. 3370–3388.

[77] Sunghwan Kim et al. "PubChem substance and compound databases". In: *Nucleic acids research* 44.D1 (2016), pp. D1202–D1213.

[78] Maho Nakata et al. "PubChemQC PM6: A dataset of 221 million molecules with optimized molecular geometries and electronic properties". In: *arXiv preprint arXiv:1904.06046* (2019).

[79] John J Irwin et al. "ZINC: a free tool to discover chemistry for biology". In: *Journal of chemical information and modeling* 52.7 (2012), pp. 1757–1768.

[80] Janna Hastings et al. "The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013". In: *Nucleic acids research* 41.D1 (2012), pp. D456–D463.

[81] Janna Hastings et al. "ChEBI in 2016: Improved services and an expanding collection of metabolites". In: *Nucleic acids research* 44.D1 (2016), pp. D1214–D1219.

[82] Lorenz C Blum and Jean-Louis Reymond. "970 million druglike small molecules for virtual screening in the chemical universe database GDB-13". In: *Journal of the American Chemical Society* 131.25 (2009), pp. 8732–8733.

[83] Lars Ruddigkeit et al. "Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17". In: *Journal of chemical information and modeling* 52.11 (2012), pp. 2864–2875.

[84] Anubhav Jain et al. "Commentary: The Materials Project: A materials genome approach to accelerating materials innovation". In: *Apl Materials* 1.1 (2013), p. 011002.

[85] Janna Hastings et al. "The chemical information ontology: provenance and disambiguation for chemical data on the biological semantic web". In: *PloS one* 6.10 (2011), e25513.

[86] Villu Ruusmann, Sulev Sild, and Uko Maran. "QSAR DataBank repository: open and linked qualitative and quantitative structure–activity relationship models". In: *Journal of Cheminformatics* 7.1 (2015), pp. 1–11.

[87] Igor V Tetko, Uko Maran, and Alexander Tropsha. "Public (Q) SAR services, integrated modeling environments, and model repositories on the web: state of the art and perspectives for future development". In: *Molecular Informatics* 36.3 (2017), p. 1600082.

[88] Steven Kearnes et al. "Molecular graph convolutions: moving beyond fingerprints". In: *Journal of computer-aided molecular design* 30.8 (2016), pp. 595–608.

[89] Matthias Rupp and Gisbert Schneider. "Graph kernels for molecular similarity". In: *Molecular Informatics* 29.4 (2010), pp. 266–273.

[90] R. Frederick Ludlow and Sijbren Otto. "Systems chemistry". In: *Chem. Soc. Rev.* 37 (1 2008), pp. 101–108. DOI: 10.1039/B611921M. URL: http://dx.doi.org/10.1039/B611921M.

[91] Bartosz A Grzybowski et al. “The’wired’universe of organic chemistry”. In: *Nature Chemistry* 1.1 (2009), pp. 31–36.

[92] Marcin Fialkowski et al. “Architecture and evolution of organic chemistry”. In: *Angewandte Chemie* 117.44 (2005), pp. 7429–7435.

[93] Agnieszka Wołos et al. “Synthetic connectivity, emergence, and self-regeneration in the network of prebiotic chemistry”. In: *Science* 369.6511 (2020). ISSN: 0036-8075. DOI: 10.1126/science.aaw1955. eprint: https://science.sciencemag.org/content/369/6511/eaaw1955.full.pdf. URL: https://science.sciencemag.org/content/369/6511/eaaw1955.

[94] Kyle JM Bishop, Rafal Klajn, and Bartosz A Grzybowski. “The core and most useful molecules in organic chemistry”. In: *Angewandte Chemie International Edition* 45.32 (2006), pp. 5348–5354.

[95] Sara Szymkuć et al. “Computer-Assisted Synthetic Planning: The End of the Beginning”. In: *Angewandte Chemie International Edition* 55.20 (2016), pp. 5904–5937.

[96] Tomasz Badowski et al. “Synergy Between Expert and Machine-Learning Approaches Allows for Improved Retrosynthetic Planning”. In: *Angewandte Chemie International Edition* 59.2 (2020), pp. 725–730.

[97] Mikołaj Kowalik et al. “Parallel optimization of synthetic pathways within the network of organic chemistry”. In: *Angewandte Chemie International Edition* 51.32 (2012), pp. 7928–7932.

[98] Chris M Gothard et al. “Rewiring chemistry: Algorithmic discovery and experimental validation of one-pot reactions in the network of organic chemistry”. In: *Angewandte Chemie International Edition* 51.32 (2012), pp. 7922–7927.

[99] Karol Molga, Piotr Dittwald, and Bartosz A Grzybowski. “Navigating around patented routes by preserving specific motifs along computer-planned retrosynthetic pathways”. In: *Chem* 5.2 (2019), pp. 460–473.

[100] Sebastian Steiner et al. “Organic synthesis in a modular robotic system driven by a chemical programming language”. In: *Science* 363.6423 (2019).

[101] Luzian Porwol et al. “An Autonomous Chemical Robot Discovers the Rules of Inorganic Coordination Chemistry without Prior Knowledge”. In: *Angewandte Chemie International Edition* (2020).

[102] Ming Pan et al. “Design technologies for eco-industrial parks: From unit operations to processes, plants and industrial networks”. In: *Applied Energy* 175 (2016), pp. 305–323.

[103] Li Zhou et al. “An ontology framework towards decentralized information management for eco-industrial parks”. In: *Computers & Chemical Engineering* 118 (2018), pp. 49–63.

[104] Feroz Farazi et al. “OntoKin: An ontology for chemical kinetic reaction mechanisms”. In: *Journal of Chemical Information and Modeling* 60.1 (2019), pp. 108–120.

[105] Andreas Eibeck, Mei Qi Lim, and Markus Kraft. “J-Park Simulator: An ontology-based platform for cross-domain scenarios in process industry”. In: *Computers & Chemical Engineering* 131 (2019), p. 106586.

[106] Evan E Bolton et al. “PubChem: integrated platform of small molecules and biological activities”. In: *Annual reports in computational chemistry.* Vol. 4. Elsevier, 2008, pp. 217–241.

[107] Sunghwan Kim et al. "PubChem 2019 update: improved access to chemical data". In: *Nucleic acids research* 47.D1 (2019), pp. D1102–D1109.

[108] Davide Alocci et al. "Property graph vs RDF triple store: A comparison on glycan substructure search". In: *PloS one* 10.12 (2015), e0144578.

[109] Neil Swainston et al. "biochem4j: Integrated and extensible biochemical knowledge through graph databases". In: *PloS one* 12.7 (2017), e0179130.

[110] Faming Gong et al. "Neo4j graph database realizes efficient storage performance of oilfield ontology". In: *PloS one* 13.11 (2018), e0207595.

[111] Vasundra Touré et al. "STON: exploring biological pathways using the SBGN standard and graph databases". In: *BMC bioinformatics* 17.1 (2016), pp. 1–9.

[112] Angiras Menon, Nenad B Krdzavac, and Markus Kraft. "From database to knowledge graph—using data in chemistry". In: *Current Opinion in Chemical Engineering* 26 (2019), pp. 33–37.

[113] Nadime Francis et al. "Cypher: An evolving query language for property graphs". In: *Proceedings of the 2018 International Conference on Management of Data*. 2018, pp. 1433–1445.

[114] Xiaoyu Sun et al. "Assessing Graph-based Deep Learning Models for Predicting Flash Point". In: *Molecular informatics* 39.6 (2020), p. 1900101.

[115] François Chollet et al. *Keras*. `https://keras.io`. 2020.

[116] *Descriptor computation(chemistry) and (optional) storage for machine learning*. Dec. 11, 2020. URL: `https://github.com/bp-kelley/descriptastorus` (visited on 12/11/2020).

[117] *scikit-optimize: Sequential model-based optimization with a 'scipy.optimize' interface.* Dec. 11, 2020. URL: `https://github.com/scikit-optimize/scikit-optimize` (visited on 12/11/2020).

[118] Milan Remko. "Theoretical study of molecular structure and gas-phase acidity of some biologically active sulfonamides". In: *The Journal of Physical Chemistry A* 107.5 (2003), pp. 720–725.

[119] Milan Remko and Claus-Wilhelm von der Lieth. "Theoretical study of gas-phase acidity, pKa, lipophilicity, and solubility of some biologically active sulfonamides". In: *Bioorganic & medicinal chemistry* 12.20 (2004), pp. 5395–5403.

[120] Michael F Sugrue. "New approaches to antiglaucoma therapy". In: *Journal of medicinal chemistry* 40.18 (1997), pp. 2793–2809.

[121] Nicholas J Baxter et al. "Reactivity and mechanism in the hydrolysis of $\beta$-sultams". In: *Journal of the American Chemical Society* 122.14 (2000), pp. 3375–3385.