Predicting Enzyme Inhibition

## Problem Statement

There are many more drugs than could evere clinically tested. To focus on only testing the most promising drugs, machine learning is used to predict a molecule's properties from its chemical structure. The [Therapeutic Data Commons](https://tdcommons.ai/) (TDC) provided a dataset of 12,665 drugs with boolean labels representing if inhibit the CYP P450 2C19 gene. [According to TDC:](https://tdcommons.ai/single_pred_tasks/adme/#cyp-p450-2c19-inhibition-veith-et-al)

“The CYP P450 genes are essential in the breakdown (metabolism) of various molecules and chemicals within cells. A drug that can inhibit these enzymes would mean poor metabolism to this drug and other drugs, which could lead to drug-drug interactions and adverse effects. Specifically, the CYP2C19 gene provides instructions for making an enzyme called the endoplasmic reticulum, which is involved in protein processing and transport.”

| Drug | Y |
| --- | --- |
| Clc1ccccc1-c1nc(-c2ccccc2)n[nH]1 | 1 |
| COc1ccccc1C(c1nnnn1C(C)(C)C)N1CCN(Cc2ccncc2)CC1 | 1 |
| CCC(c1nnnn1CC1CCCO1)N(CCN1CCOCC1)Cc1cc2cc(C)cc... | 0 |
| Br.N=c1n(CCN2CCOCC2)c2ccccc2n1CC(=O)c1ccc(Cl)c... | 1 |
| COc1ccc(/C(O)=C2/C(=O)C(=O)N(CCCC(=O)O)C2c2ccc... | 0 |

A subsample of the dataset.

The “Drug” column is the [“SMILE” string representing the chemical structure molecule](https://en.wikipedia.org/wiki/Simplified_molecular-input_line-entry_system) and the “Y” column is the label of if that drug will inhibit the CYP2C19 enzyme.

There were 12,665 labeled drugs where ~1/3 were positive samples. The SMILES have a variable size with an average of 46.4 (+- 20.8) characters, a minimum size of 2 and a maximum of 340 characters.

The first part of the problem was to extract the relevant features from the molecules through a graph-to-vector algorithm. This is non-trivial because molecules are variable sized and interact chemically based on variable sized subgraphs. The specific atoms and their topology determine if the molecule will inhibit the enzyme. To extract the relevant features the embedding algorithm would need to keep both node attributes and topological information.

## Graph Embedding With Color Refinement

The approach used was a variation of a well known graph embedding algorithm named Color Refinement. The classical Color Refinement algorithm initializes each node color as a constant and updates colors as a hash of the current color and an order insensitive combination of neighbor colors. Because there was no open source [Python implementation of Color Refinement, I wrote it here.](https://github.com/parkerburchett/TDC-Binary-Classification/blob/main/ColorRefinement/ColorRefinement.py)

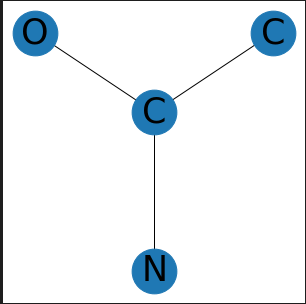
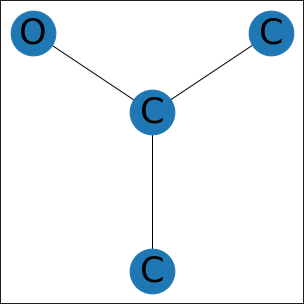
The variation created for this project was to initialize node colors as a HASH of node attributes. This preserves both node attributes and topological information. This matters because two isomorphic subgraphs will often have very different chemical properties.

Where C is the current node, H is the hop number, K is the number of colors, and % is the modulo operator. For any value of H, each graph is embedded into a length K integer vector representing the count of each node color with a given color .

Where G(B,H) is the sparse “bag of colors” vector at hop H for the graph G.

The ColorRefinement algorithm takes two user-defined parameters, K, the number of possible colors and H, the number of hops defining the size of each subgraph. Larger values of K cause a lower false positive rate at the cost of greater sparsity. Larger values of H capture information about larger subgraphs at the cost of increasing the false positive rate. K was selected based on compute limits, and and H by empirically measuring the AUPRC lost as H increases.

This property of near 0 false positive rates at lower H value and near 1 at higher H values is due to the exponential increase in the number of possible isomorphic subgraphs as then number of nodes increases. For example, when K=1 then each subgraph consists of a single node and its immediate neighbors. For the molecule dataset that typically would mean a subgraph with a number of vertices between 2 and 4, or a total lower bound of 9 isomorphic graphs.



Two isomorphic subgraphs that are chemically distinct.

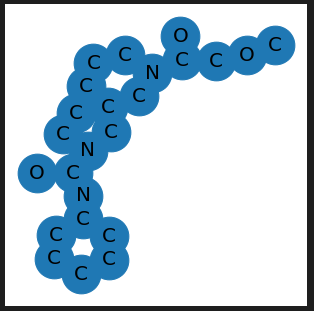
| Number of Vertices | Number of non-isomorphic undirected subgraphs |
| --- | --- |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 6 |
| 5 | 21 |
| 6 | 112 |
| 7 | 853 |
| 8 | 11,117 |
| 9 | 261,080 |
| 10 | 11,716,571 |

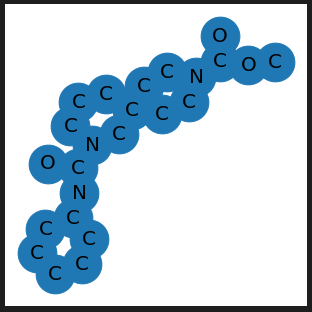
[Source](https://www.researchgate.net/figure/Number-of-non-isomorphic-subgraphs-for-undirected-and-directed-graphs-with-up-to-10_tbl1_320424167)

The false positive rate increases explonenitally can be understood by looking at the number of non-isomorphic subgraphs increases with the number of vertices. Each subgraph is hashed into one of K buckets at larger H values; there are orders of magnitude more possible non-isomorphic subgraphs than K. In practice this is typically not much of an issue because the embedding vectors are typically sparse. The number of non zero values in G(B,H) has a strict upper bound defined by:

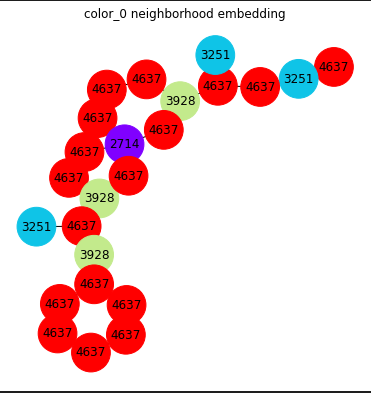
Each node is only assigned a single color at each hop level so when embedding a graph with 46 nodes into a bag of colors vector of size 2,000 the vector is between (2000-46)/2000 or 97.70% and (2000-1)/2000 or 99.95% zeros. So even while there might be 1,000 chemically distinct subgraphs that would be hashed into the same bucket the probability that two distinct subgraphs at a given value of H would be hashed into the same bucket is 1/K. Because the hash function does not create false negatives, the false positive rate will only increase as H increases. This is mitigated because the probability of a true positive decreases as H increases again due to the exponential increase in possible non-isomorphic subgraphs.

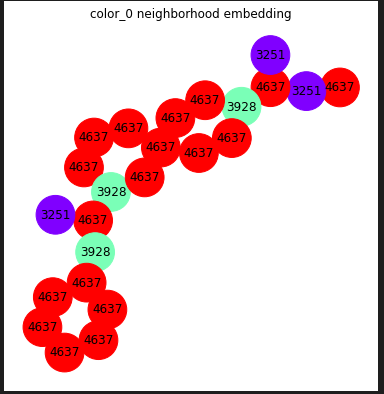
#



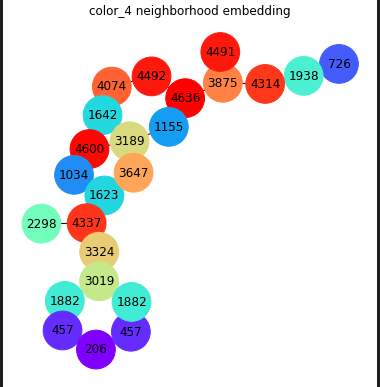


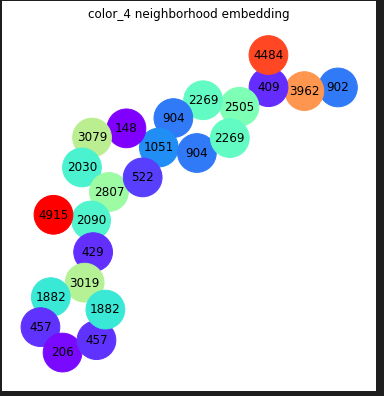
Similar molecules in the TDC dataset.





The H=0 K=5000 embeddings of some similar molecules.



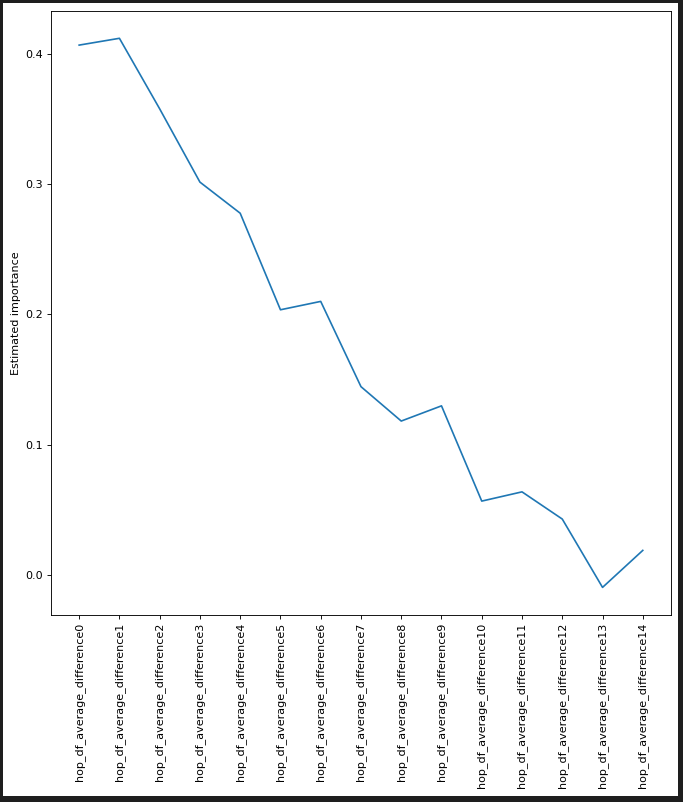


The embedding at the H=4 and K=5000. The left bottom part of each graph is identical but the top right has diverged because they are not identical.

## Selecting Hyper Parameters

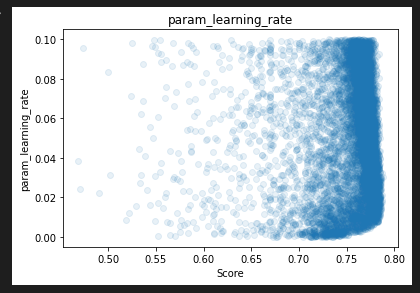
Calculating the false positive rate at each H is difficult and depends on the degree of each node in each graph and the number of distinct nodes. Instead of trying to calculate the false positive rate a value for H was selected empirically by measuring information lost as H increases and the limits on computation.

At each hop H a default LGBMClassifier() was trained on the graph embedding at that H, then compared to the Area Under the Precision Recall Curve (AUPRC) when the same model was retrained on a permuted set of targets. The difference between the unpermuted targets and permuted targets can be thought of as the lower bound on the amount of information lost when there is no relationship between the features and targets.

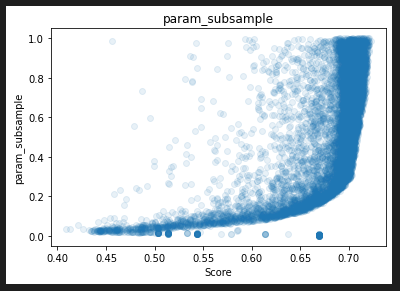


There is a linear negative relationship between the amount of information lost as H increases after H=1. Because of this the graphs were chosen to be embedded into 4 vectors at H=[0,3]. This cutoff was selected by considering the computational resources required for building and tuning models. An opportunity for improvement would be to create an ensemble of models each trained on the embedding at h=[0,~13] when the information lost from permuting targets is close to zero.

K was arbitrarily chosen to be 2,000 due to resource limits. It is probable that there would be improvements to be made by building an ensemble model with both a larger value of K and of H. The final model was a weighted ensemble of Light Gradient Boost Machines. Each model had 5000 random combinations of hyperparameters computed withSklearn’s RandomizedSearchCV(). The relationship between average 5-fold CV AUPRC and the given hyperparameter is viszulaied below.



Impact of the learning rate parameter on the model trained on the embedding at H=0.



The impact of the subsample parameter the model trained on the embedding at H=3.

## Computing Ensemble Weights

Once a tuned model for each H=[0,3] weights were selected by randomly selecting % weights for each model and computing the average 5-fold cross validation AUPRC of that weight combination.

| **index** | **model\_0** | **model\_1** | **model\_2** | **model\_3** | **target** |
| --- | --- | --- | --- | --- | --- |
| 1935 | 0.312784 | 0.208339 | 0.295249 | 0.286069 | False |
| 1936 | 0.33413 | 0.34766 | 0.449999 | 0.583438 | True |
| 1937 | 0.460933 | 0.445365 | 0.691044 | 0.297753 | True |
| 1938 | 0.426971 | 0.588611 | 0.686233 | 0.507866 | True |
| 1939 | 0.859865 | 0.984376 | 1.068336 | 0.679388 | True |

Each model’s prediction on a CV-fold of the training data.

| **index** | **weight\_1** | **weight\_2** | **weight\_3** | **weight\_4** | **AUPRC** |
| --- | --- | --- | --- | --- | --- |
| 31079 | 0.228070 | 0.508772 | 0.157895 | 0.105263 | 0.796028 |
| 4834 | 0.232394 | 0.507042 | 0.183099 | 0.077465 | 0.796014 |
| 93031 | 0.237113 | 0.505155 | 0.159794 | 0.097938 | 0.796001 |
| 62021 | 0.236994 | 0.502890 | 0.179191 | 0.080925 | 0.795998 |
| 11659 | 0.236842 | 0.506579 | 0.157895 | 0.098684 | 0.795997 |

Because the weights converged, the final ensemble was defined as the average of the best 20 of 100,000 random weights.

The best models and best weights were hard coded into Modeling/Final\_weighted\_model.ipynb and submitted to Therapeutic Data Commons. The final weighted ensemble had an AUPRC of .767 +-.003 and currently [has the highest AUPRC of those submitted](https://tdcommons.ai/benchmark/admet_group/10cyp2c9i/).

## Next Steps

The Color Refinement algorithm should be refactored and pushed to Pip so that other researchers can easily use it. There are other datasets provided by TDC that a weighted ensemble of models trained on different values for H could be well suited for.

[The pysmiles library](https://pypi.org/project/pysmiles/), a open source tool for converting the SMILES into [NetworkX Graph objects](https://networkx.org/) calculated the ‘charge’, ‘hcount’, ‘aromatic’, and ‘element’ attributes for every atom. The modeling could be improved by excluding some of these attributes from initial node coloring.

There is likely significant improvement to be made in hyperparameter tuning and model selection. For example, Pycaret was used to initially test a variety of models and Logistic Regression was found to be the best model when H=0. The decision to use Light Gradient Boost regression for each component model was made to simplify hyper parameter tuning rather than through cross validation.

Optimal values for H and K should be chosen through cross validation rather than arbitrarily selected.There can also be improvement by using more models and a more sophisticated technique for creating an ensemble such as Bayesian Model Averaging or Particle Swarm Optimization.