Democratizing Drug Discovery: Active Learning for Accessible High Throughput Screening

Gabriel Hart Stocker Dreiman

Supervisors: Dr. Lewis Griffin & Dr. Fredrik Svensson

This report submitted in partial fulfillment

of the requirements for the degree of

Master of Science

in

Machine Learning.

Department of Computer Science

University College London

September 20, 2019

Code for this project can be found in the following public repository: `https://github.com/gdreiman1/thesis`.

Abstract

Modern Drug Discovery relies on High Throughput Screening, a process which tests a large library of chemical compounds for biological activity against a disease target. Only ~5% of tested compounds show any biological activity, so in their hunt for good starting points, major pharmaceutical companies typically screen millions of compounds. Each test costs ~£1.20, so screens of this size are fiscally infeasible for smaller academic and non-profit labs. Iterative screening is a batchwise method that subsamples and screens a compound library to generate labeled data which are used to train a predictive model. The model's predictions determine the compounds selected for the next round of screening. Repeating these iterations retrieves active compounds at a rate higher than random sampling and reduces the number of screens needed in a campaign.

This thesis compares models and sampling strategies from previously published work and adds state of the art, chemical-specific, Graph Convolutional Neural Networks. Importantly, it considers the practicalities of implementing the developed methods in a lab setting. In doing so, it provides the first comprehensive comparison of classifier performance when applied to raw HTS data under realistic methodological constraints. Using performance metrics derived in collaboration with HTS experts, I show that my proposed methods yield tangible improvements over current HTS approaches. In particular, I identify a method with a 10% start size, 5% batch size, and Random Forest classifier as the best performing, and show that it generalizes to novel HTS campaigns, offering cost savings of approximately £780,000 compared with a typical HTS campaign.

Acknowledgements

Thanks to both of my supervisors Dr. Fredrik Svensson and Dr. Lewis Griffin whose generous contributions of time and insight made this thesis possible. Thanks also to Alzheimer's Research UK for funding the Drug Discovery Institute, and by extension my thesis. Dr. Magda Bictash provided crucial insight into the HTS process, for which I am extremely grateful. I'm grateful also to the various medicinal chemists, microbiologist, and drug formulation scientists who mentored me over the years (sorry about all the broken glassware). Finally, thanks to my family who have always nurtured my passions for science and medicine.

Contents

List of Figures

List of Tables

Chapter 1

Introductory Material

Creating a new drug is hard. Most chemical compounds are not biologically active, and of those that are, only a few are medically useful and non-toxic. This thesis focuses on finding active compounds, which is a crucial step in the incredibly costly and time-consuming process of developing a new drug. Challenges in drug development contribute, in part, to the soaring cost of healthcare, one of the major public policy crises of our time. Drug development also a process that relies on luck and intuition as much as it does on scientific principles and data driven decision making. Using a combination of passed down heuristics and brute force research, chemists have made stunning advances in the past century, but the enormity of the task (there are an estimated 10^{65} possible combinations of molecules and target [1]) and the spiraling costs of development suggest that a new paradigm is needed. Machine Learning, with its ability to derive meaning from data far beyond human comprehension, is poised to make significant contributions to drug discovery. Before explaining the ways in which Machine Learning, and by extension this thesis, can contribute to the field, it is important to establish a basic understanding of the process by which a drug is developed, as the particularities of the process are at the heart of the question this thesis considers.

The drug development process can be understood by focusing on the development of statins, the family of cholesterol control drugs. This process can be broken down into five steps: Basic Research, Lead Discovery, Preclinical Development, Clinical Development, and Approval [2]. Basic Research is catchall term for a vast

array of primary biological research, but in general its focus is discovering, and understanding, basic biological processes. Ideally this results in a 'target', a specific biological unit (DNA, RNA, or protein) that contributes to a disease and can be modified to prevent the disease. In the statin example, two key events occurred. First, the Framingham heart study established a conclusive link between high cholesterol and heart disease [3]. Second, biologists discovered and elucidated the network of proteins that synthesize cholesterol in humans. One protein in this chain (HMG-CoA reductase), if inhibited, dramatically reduces the production of cholesterol without causing a build up of toxic byproducts [4]. This crucial role made HMG-CoA reductase a compelling target. The next task was to find a molecule that inhibited it, ie Lead Discovery.

The first HMG-CoA reductase inhibitor was discovered in a classic manner: by accident. Scientists in Japan looking for new antibiotics had cultured a fungus and extracted a soup of various chemical compounds the fungus had produced. After separating the compounds, each was scanned for activity against a series of proteins, including HMG-CoA reductase. One compound was a potent inhibitor of HMG-CoA reductase, and thus a Lead was discovered[4]. After Lead Discovery, medicinal chemists usually make a series of modifications to the initial molecule to tune its performance; this is known as Preclinical Development. Some drugs never make it past this step, often because they have poor physical characteristics (e.g.a drug that does not dissolve in water is very hard to dissolve into the blood). To pass the Preclinical step, drugs must proven to be effective and non-toxic in animal models. The drugs that do pass proceed to Clinical Development, and are tested on human patients. The HMG-CoA reductase inhibitor that the Japanese scientists discovered turned out to be extremely toxic, and failed in the Preclinical stage. However, now that the general structure of a potent HMG-CoA reductase inhibitor was known, Merk continued searching for similar compounds. Eventually a less toxic one was found and showed remarkable performance in clinical trials. Having pascsed the Clinical stage, it was submitted for approval to the FDA. The FDA approved the drug (completing the final stage of the development pathway, Approval),

and it immediately revolutionized cardiology. Since their discovery, statin drugs have dramatically reduced the incidence of heart disease (treated patients have a 42% lower chance of dying from a heart attack than non-patients)[5].

In recent years, advances in biology have revolutionized target discovery. Fast and cheap genetic sequencing allows scientists to identify proteins correlated with particular diseases. These proteins can be easily and precisely silenced with CRISPR-CAS9 [6], if this silencing prevents the disease, the protein is immediately established as a target. The Lead Discovery process has not progressed nearly as quickly. The current state of the art for searching for leads is High Throughput Screening (HTS). In simple terms, HTS uses lab automation to screen the members of a 'compound library' against a biological target, searching for 'hits' which indicate that the compound is biologically active against the target. Each hit discovered is a possible lead. HTS relies on conducting many miniaturized biological experiments in parallel on a 'plate'. The physical plate has hundreds of tiny wells, each of which contains a sample of the biological target. Each well has a single compound from the library dispensed into it and the entire plate is 'read' by a machine that converts a raw physical signal into an estimate of the compound's biological activity. This technique has been refined to the point that million compound screening campaigns are well within the realm of possibility [7]. However, there is a fundamental challenge: less than 5% of the library will likely be active, meaning 95% of the cost of compounds, labour, and most importantly, time has been wasted.

An intuitive solution is to perform the campaign in batches, use the collected data to train a predictive model, and then use the model to select unscreened molecules from the library for subsequent batches. This is referred to as Iterative Screening, and while previous papers have explored it using single Machine Learning techniques, and single strategies [8][9][10], there has not been a comparison of modern ML techniques and strategies. Moreover, few recent papers have considered the practicalities of implementing their findings in a laboratory setting. In recent years, advanced biological techniques have been standardized and made ubiquitous. Their decreased cost and increased power has allowed academic and nonprofit labs

to discover targets in ways previously limited to large pharmaceutical companies. The cost of HTS remains stubbornly high, limiting the labs' abilities to pursue their discoveries. A properly implemented iterative screening strategy stands capable of democratizing HTS for smaller labs, replacing expensive and time-consuming physical experiments with cheap and fast machine learning modeling. In what follows, I seek to replicate and then extend previous work using Machine Learning to perform iterative screening. Crucially, in addition to making novel applications of Graph Convolutional Neural Networks to this problem, I adhere to guidelines and goals laid out by the biological experts who would potentially use this solution in real life. As such, this thesis represents the first study that I know of to compare modern Machine Learning classifiers' performance in active learning strategies for HTS under settings that are determined by the practicalities of experimental reality and and with results measured by metrics defined by intended HTS users.

Chapter 2

Background and Literature Review

In this section, I aim to establish two points. In section 2.1 I explain what High Throughput Screening is, and explore its benefits and limitations. In section 2.2 I detail Machine Learning approaches I use in the thesis and how they relate to HTS, while also providing a survey of previous work in the field of Iterative Screening.

2.1 High Throughput Screening

Drug target interactions are often described in terms of a lock and key. In this analogy, the target (often a protein) is the lock, and the drug is the key. A key that fits a lock can change its state, opening or closing it. If a drug fits into a protein, it also can change its state, increasing or decreasing its activity. The history of drug discovery can be viewed as a process of refining techniques for testing keys against locks. Ignoring the early days of drug discovery, which were often driven by blind luck, one of the best examples of an early systematic search for a drug is Communist China's search for antimalarial drugs, Project 523. The project started by combing traditional Chinese medical texts for herbal fever remedies, which yielded a large pool of candidates. Chemical compounds were extracted from these herbs, formed into pills, and given to malaria infected mice. If the mice recovered, the compound was studied further, eventually reaching human trials with malaria patients [11]. This labour intensive search lasted five years and eventually yielded a potent anti-malaria drug. However, it relied on experimental techniques that were neither fiscally nor ethically available to private western companies. Nevertheless, the gen-

eral process of testing a large pool of candidates against a target, remained the best way to discover a new drug throughout the 1960's and 1970's. While refinements in lab techniques, and the testing of cultured microorganisms (rather than entire animals) sped up the process, Pereira and Williams note that well into the 1980's 'drug discovery methods required 1ml reactions in individual test tubes' which '... limited assay capacity for a laboratory to 20 to 50 compounds per week' [12]. The major limitation was the serial and large scale nature of the experiments. As such, a parallel miniaturized methodology was developed. The key to this method was the use of 96 well plates, essentially a thermoplastic plate with 96 individual ~100µl wells. These plates allowed 96 identical wells of bacteria to be grown under identical conditions, and handled as a single unit. Each well could then have a different test compound added to it, immediately yielding at nearly 100 fold increase in number of compounds screened [12]. Moreover, a standardized physical interface allowed for substantial increases in laboratory automation [13], and for the miniaturization of experiments, reducing reagent costs.

In the intervening years, plates expanded significantly, often holding 1536 wells. This increased capacity combined with laboratory robots, allows modern HTS campaigns to consider massive compound libraries with sizes in the millions. Digital imaging allows compound activity to be measured by recording emissions from fluorescently labeled proteins or imaging cultured cells to assess their population or viability. These modern techniques have yielded important new drugs, most recently the NS5A inhibitors. These drugs provide the first single treatment cure for Hepatitis C, but cost over $100,000 per patient in the US [7]. Despite the success of treatments developed with HTS, it has sustained criticism on several fronts. Often these criticisms can be traced to a simple fact: less than 5% of the entire library can be expected to register as a hit and even fewer of these hits will have characteristics favorable to further development. On the face of it, the HTS paradigm is a naive one. The library is searched exhaustively and inefficiently to generate a few viable hits. This inefficiency makes HTS very expensive and thus inaccessible to those without the resources required to screen massive libraries. Machine Learn-

ing promises to democratize HTS, and make it accessible to more academic and non-profit researchers.

2.2 Machine Learning

There has been a long held interest in applying Machine Learning to drug discovery. The drug discovery process relies on pattern matching (synthesizing a molecule with a similar shape to a known drug is an important source of novel drug compounds), and heuristics (Lipinski's rule of 5 is a classic rule-set to determine "drug-likeness") to narrow the search for novel compounds. In the 1960's, Hansch and Fujita introduced Quantitative Structure Activity Relationship (QSAR) modeling which attempts to link quantitative information about a compound's structure to its activity in a biological system [14]. The appeal of reducing medicinal chemists' reliance on gut instincts and passed-down heuristics with a data driven modeling is obvious. So in the following years as more sophisticated Machine Learning models were developed they were quickly applied to drug discovery. What follows is a brief description of the history of the Machine Learning models I use in this thesis and their relationship to drug discovery. Prior to any ML attempts however, one must represent the data (in this case molecules) in a machine interpretable manner.

2.2.1 Chemical Representation and Embeddings

Molecules are localized quantum fields, proteins are larger localized quantum fields, and the interactions of these quantum fields when a protein and molecule are in close proximity are what determine whether or not the molecule has a biological effect on the protein. While computational chemists do model these quantum interactions, the modeling is far too computationally expensive to be used on a large compound library. Instead classical approaches attempt to describe the properties of a molecule's quantum nature and generally fall into two camps: Molecular Fingerprints (which represent the structure of the molecule) and Molecular Characteristics (which represent its intrinsic qualities) [15]. A more modern approach, Molecular Graphs attempts to represent molecules in a manner similar to the way that chemists traditionally draw them: as an undirected graph (see Fig. 2.1). In these graphs,

nodes represent atoms and edges represent the bonds between them. Considering molecules as undirected graphs is useful for understanding all three methods.

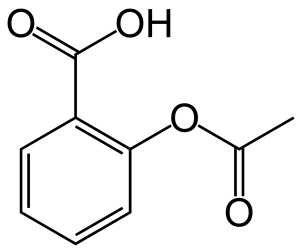

Figure 2.1: A graphical representation of Aspirin [16]

2.2.1.1 Molecular Fingerprints

Molecular Fingerprints attempt to represent these graphs by mapping sub-graphs of the molecular graph to a fixed length bit vector. The current state of the art are Extended Connectivity Fingerprints[17]. This algorithm traverses the nodes of the graph, and at each node collects information about the node itself and its neighborhood. This collected information is passed to a hash function and mapped to a 32-bit integer which is stored in a list. The number of times that the graph traversal algorithms passes over the graph is a user defined parameter, with each additional iteration expanding the size of the neighborhood considered. Duplication is prevented (i.e. if two identical sub-graphs are encountered, the corresponding hashed identifier will only appear once in the fingerprint), and the result is a variable length list of large integers. This list can be considered as representing a molecular graph, matching each sub-graph to its key in a dictionary of all possible chemical sub-graphs. Unfortunately, because these integer lists are of variable length, they are difficult for many algorithms to take as an input. To solve this problem, they are often rehashed, this time to a fixed-length bit vector. This rehashing is lossy [17], though previous work suggests that the loss is relatively small [18]. It also further obfuscates the actual chemical structure, which makes an exploration of feature importance challenging at best.

2.2.1.2 Molecular Descriptors

The other option for obtaining a fixed-length representation of a molecule is to calculate measurements of the chemical and physical properties of the entire molecule and add them to a vector that describes the molecule. These descriptors vary in their complexity, the simplest (1D descriptors) are often continuous measurements of a single molecular property (its mass, for example) [15]. Unfortunately, 1D descriptors are degenerate as distinct compounds can be mapped to identical or near identical descriptor values [15]. More complicated descriptors (termed 2D and 3D) can also be calculated, and these descriptors take the structure of the molecule into account and provide better differentiation. For example, logP is a common measure of the hydrophobicity of a molecule and helps determine if a drug can cross cell membranes (this is crucial as drugs must dissolve through the gut to reach the rest of the body). While they measure important qualities, these descriptors accumulate information across the entire molecule. This is unfortunate because the potency of a compound is often driven by a specific chemical configuration in a small region of the molecule. For example, caffeine and ethanol have similar logPs which allows both to cross the blood-brain barrier. Once in the brain, they have slightly different effects. Put another way, would you be willing to tell me the length, weight, and number of teeth on your house-key? Probably yes, because the actual location of the teeth is what determines if a key can open the lock. Asking for a picture of the key however, might generate a different response.

2.2.1.3 Molecular Graphs

Humans struggle to visualize electron probability density clouds, so chemists often represent chemicals as graphs. In these graphs, nodes are atoms and edges are the bonds between them. For a trained chemist, this graph contains a substantial amount of implicit information about bond length and strength or an atom's electronegativity. To supplement this intuitive understanding of a molecule's nature, chemists calculate a number of numerical measurements associated with the components of a molecule (atomic number, charge, bond length, bond angle, etc), which in their totality form a low dimensional and static representation of the quantum state. To

represent a molecule for a computer, each node and edge in a graph of a compound's structure has an attribute vector associated with it. While many possible attribute vectors exist, this work uses those established by DeepChem [19]. The 75 features for each atom's feature vector are derived from RDKit calculations of atomic properties (see [1] for details).

2.2.2 Machine Learning Models

2.2.2.1 SVM

The Support Vector Machine is a machine learning algorithm first proposed by Vapnik and Lerner in 1963 and formalized in its modern soft margin form in 1995 [20]. In its most simple binary classification form, the linear hard margin SVM takes a group of data-points $x \in \mathbb{R}^n$ with labels $y \in \{-1,1\}$, and attempts to establish a hyper-plane in $\mathbb{R}^n$ defined by $w \in \mathbb{R}^n$ that separates the two classes such that any point x_i can be classified by $\hat{y}_i = \text{sign}(wx_i + b)$. Normally, one seeks to maximize the margin between the hyperplane and the data-points as this minimizes the number of points needed to define the hyperplane. These remaining points are termed the support vectors [21]. A soft margin SVM is used in the case that the classes are not linearly separable in $\mathbb{R}^n$ and uses a loss function to penalize misclassified points. This loss is often hinge loss: $\textbf{hinge}(x_i, y_i) = \max(0, 1 - y_i(wx_i + b))$, and the new objective function becomes: $\min([\frac{1}{m}\sum_{i=1}^{m} \textbf{hinge}(x_i, y_i)] + \lambda ||w||^2)$. In simple terms, the soft-margin linear SVM seeks a separating hyperplane that best separates the two classes while minimizing the number and scale of misclassifications.

2.2.2.2 Random Forest

Random Forests (RF), introduced by Breiman in 2001 is an ensemble based machine learning algorithm that utilizes decision trees and bagging to construct a classifier [22]. A single decision tree is a hierarchical series of splitting rules applied to partition a set of data-points into smaller and smaller subsets. Each rule splits a set of data points at a threshold on a single feature, and splitting terminates once a set reaches a stopping criteria (including impurity, size, or depth of tree) [22]. Classi-

[1] `https://raw.githubusercontent.com/deepchem/deepchem/master/deepchem/feat/graph_features.py`

fication is achieved by applying the set of rules developed during training to a new data-point, and labeling it with the class that makes up the majority of the leaf it has been assigned to. RF uses bagging, a technique that randomly resamples (or subsamples) data to construct an uncorrelated ensemble of classifiers. The training set for each member tree is sampled randomly (with replacement) from the full training data, and each split is considered on a random subsample of features. The ensemble has lower variance than any of its constituent models, and thus by aggregating the votes of each decision tree, one can obtain better generalization performance than any single tree [23]. RF is popular because it scales well, is relatively immune to over-fitting, and can handle a combination of categorical and continuous features. These features make it useful in drug discovery. Indeed, within 2 years of Breiman's 2001 paper introducing Random Forests, Svetnik et al. published a paper describing the particular benefits of applying RF to drug discovery noting that it 'handles high-dimensional data well; has the ability to ignore irrelevant descriptors; handles multiple mechanisms of action; and is amenable to model interpretation' [24].

2.2.2.3 Gradient Boosting Machine

Gradient Boosting Machines are based on the concept of boosting. Boosting is a meta-learning algorithm in which an ensemble of weak learners is created in an additive manner, with each new learner attempting to model (and thereby correct) the errors of the previous ensemble. AdaBoost, is perhaps the best known early Boosting Machine [25]. Introduced in 1996, it re-weights data points after each iteration, causing the subsequent weak learner to focus more on correctly classifying point that were previously misclassified. In this formulation, AdaBoost does not explicitly calculate and attempt to descend a loss gradient. However, Breiman later demonstrated that AdaBoost could be reformulated with an explicit loss function and solved with gradient descent [26]. Subsequently, Friedman published a generalization that he termed Gradient Boosting Machines which can use any number of convex loss functions [27]. The key to this generalization is fitting the newest model to the negative gradient of the loss function with respect to the current model at each data point (these gradients are termed 'pseudo-residuals').

The new model is then assigned a scalar value that is calculated to minimize the loss when the new model is added to the ensemble. In a rough analogy to Newton's Method, the new weak learner approximates the loss gradient vector and meta-learning algorithm assigns the vector a weight to step down the loss function. Below is an example of a pseudo code algorithm for a GBM adapted from [27] with $F(x) =$ the ensemble model, $L(y_i, x_i) =$ loss, $h_m =$ weak learner, $\gamma_m =$ weight, and $M =$ number of weak learners.

Algorithm 1: Gradient Boosting Machine

$F_0(x) = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, x_i)$

for *m=1 to M* **do**

  $\hat{y}_i = -[\frac{\partial L(y_i, F(x_i))}{\partial (x_i)}]_{F(x)=F_{m-1(x)}}, i = 1, N$

  $h_m = \arg\min_{\mathscr{H}} \sum_{i=1}^{n} [\hat{y}_i - h(x_i)]^2$

  $\gamma_m = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$

  $F_m(x) = F_{m-1}(x) + \gamma h_m(x)$

end

GBMs often use decision trees, which can impose high computational costs due to their need to evaluate each data points value for every feature before deciding on a split point [28]. LightGBM (LGBM) is an adaption of the GBM framework that introduces sampling techniques to speed up model fitting [28]. Given the high dimensional nature of the compound embeddings, the relatively large library sizes, and the need for repeated fitting, I chose LightGBM as the representative GBM model for this thesis.

2.2.2.4 Neural Network

Neural Networks have a long history in machine learning tracing back to Rosenblatt's Perceptron [29]. Initially modeled on human neurons, perceptrons are composed of two functions. The first, the linearity, applies a linear transformation, multiplying a series of inputs by a set of weights and summing them. The second, the non-linearity, takes the weighted sum and applies a non-linear function, returning a single output. A single unit combining a linearity and a non-linearity continues

to serve as the fundamental building block of neural networks. Crucially, modern non-linearities are differentiable. This is important because units are trained by modifying their weights using gradient descent of a convex loss function. There are numerous excellent reviews on the topic, notably [30], which provides a thorough treatment.

What follows is a brief review of the structure and training of an example network. Neural networks are best imagined as layers of units, each unit h_i in a layer L_j receiving an input X_j and passing an output y_i. Units in the first layer receive a datapoint as vector X, each applies its linearity $z(X) = \sum_k w_{i,k} x_k$ and passes the result (z_i) to the nonlinearity $y_i = f(z_i)$. The resulting y_i's are concatenated into a new vector X_{j+1} which serves as an input to each of the units in the following layer. The repeated modification and aggregation of the information contained in the raw data means that each layer's output has higher information density than the last. For a simple binary classifier, the final layer is a single unit often with a sigmoidal nonlinearity which converts the transformed and aggregated input into a single label prediction, $\hat{y}$ between 0 and 1. This predicted label is passed to an error function $E(\hat{y}, y)$, log loss is a common choice, and the gradient of the error with respect to the weights of each unit are calculated. This gradient is scaled by a learning rate (γ) and used to update a unit's weights, $w_{new} = w - \gamma \nabla_w E(\hat{y}, y)$. For multilayer networks, automatic differentiation allows efficient propagation of the loss gradient across the computational graph of the network. This process, termed backpropagation, is the key to training multilayered networks [31][32].

2.2.2.5 Graph Convolutions

Graph Convolutions are machine learning algorithms that can operate directly on graphs. The core concept of Graph Convolutions (GCs) is to formulate an algorithm that can serve as a unit in a neural network and traverse the nodes of an arbitrary graph. At each node, the unit aggregates information about the node and its neighbors and yields a single output. GCs use the word convolution because they operate in a manner similar to that of traditional convolutional neural networks (typically applied to image data). A normal convolution unit has a fixed set of weights

(smaller than the input) which 'slide' across the input to aggregate local information into a single representation (see Fig. 2.2 for a graphic representation [33]). Duvenaud et al.were the first to publish a fully differentiable algorithm that allowed aggregation of local information across arbitrary molecular graphs [34]. Gilmer et al. established the best notational framework for Graph Convolutional Neural Networks (GCNNs) which they term Message Passing Neural Networks (MPNNs) [35]. Their very succinct definition follows:

> MPNNs ... operate on undirected graphs G with node features x_v and edge features e_{vw}. It is trivial to extend the formalism to directed multi-graphs. The forward pass has two phases, a message passing phase and a readout phase. The message passing phase runs for T time steps and is defined in terms of message functions M_t and vertex update functions U_t. During the message passing phase, hidden states h_v^t at each node in the graph are updated based on messages m_v^{t+1} according to
>
> $$m_v^{t+1} = \sum_{w \in N(v)} M_t(H_v^t, h_w^t, e_{vw})$$
> $$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$
>
> where in the sum, $N(v)$ denotes the neighbors of v in graph G. The readout phase computes a feature vector for the whole graph using some readout function R according to
>
> $$\hat{y} = R(\{h_v^T | v \in G\}).$$
>
> The message functions M_t, vertex update functions U_t, and readout function R are all learned differentiable functions.

There have been numerous of publications of novel MPNNs for learning on molecules, each of which modifies the general MPNN framework to claim state of the art performance on a particular data-set[34, 36, 37, 38, 39]. Unfortunately, none of previously published tasks corresponds directly to the central problem in this thesis, making it hard to choose a representative benchmark model. Fortunately, Wu et al. published a machine learning chemistry benchmark, MoleculeNet, which included a GCN example [19]. This example uses a GCNN algorithm published by Kipf and Welling shown diagrammatically in Fig. 2.3 [36]. Acknowledging the substantial number of architectures, algorithms, and embeddings[40][41][42], this

thesis adapts previous implementation of the DeepChem model PyTorch Geometric [43][44].

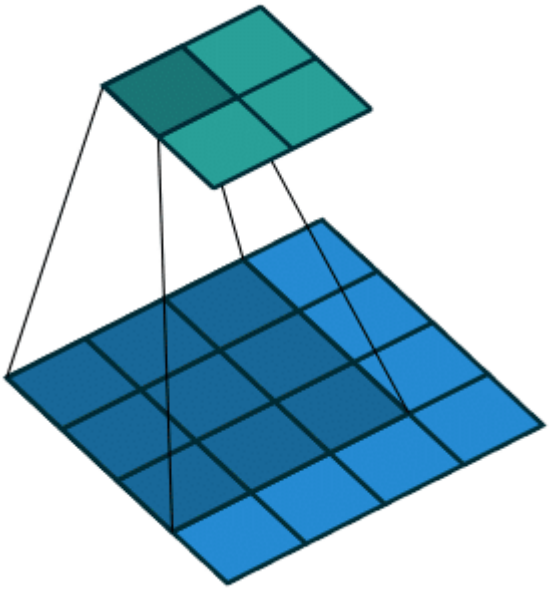

Figure 2.2: An example from Dumoulin and Visin's 'A guide to convolution arithmetic for deep learning'. A 3x3 convolution aggregates local information to represent a 4x4 pixel array with a 2x2 pixel array [33].

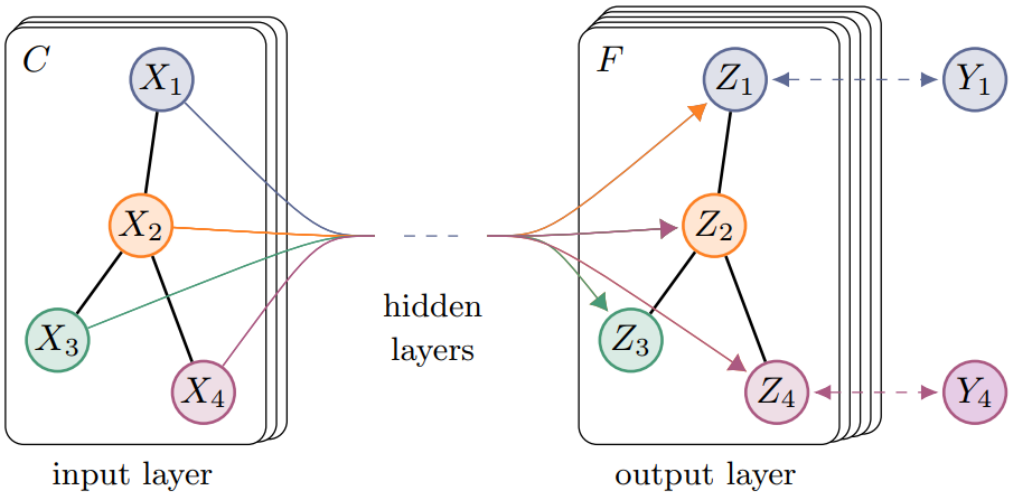

Figure 2.3: A diagram representing the GCNN algorithm used in this thesis. Reproduced from [36]

2.2.3 Machine Learning in Drug Discovery

2.2.3.1 Model application examples

All of the models listed above have previously been applied in the context of drug discovery. An early comparison of ML models was published by Doniger et al. using neural networks and kernel SVMs [45]. As with much pre-GPU research, the neural net under-performed the SVM when predicting blood-brain-barrier penetration. The following year, Svetnik et al. applied random forests for predicting binding affinity and CNS penetration. The deep learning revolution after 2010 brought deep neural networks (DNNs) to the forefront in drug discovery with a group from Hinton's lab using DNNs to win the 2013 Merck Kaggle challenge[46]. The following year, the first broad review of DNNs in the context of multi-task drug prediction was published, noting 'multitask networks afford limited transferability to tasks not in the training set' an important foreshadowing [47]. The most recent advance, Graph Convolutional networks, were introduced by Duvenaud et al., and were initially framed as an explicit drop-in differentiable replacement for the calculation of Molecular Fingerprints and Molecular Descriptors. Subsequently, GCNNs and their reformulated cousins MPNNs have been used for among other things, predicting quantum effects [35], building generative models [48] , and low-data learning [49].

2.2.3.2 Active Learning

Active learning is a learning framework in which an ML agent interacts with a pool of labeled and unlabeled data. Training on the labeled data, it's predictions on the unlabeled data are used to select unlabeled data for labeling. This paradigm comports perfectly with drug development, in which candidate molecules are selected for further screening based on chemists' intuitions, simple assay results, or similarity to already tested compounds among other things. Warmuth et al. introduced active learning to drug discovery, applying SVMs to a screening data set. This paper neatly describes the setting:

> At any stage of the process, three types of compounds can be distin-

> guished: (a) a very small fraction of compounds that already have been identified as active, (b) a much larger fraction of compounds that already have been identified as inactive, and (c) by far the largest fraction of compounds that have not yet been tested (the unlabeled compounds). [10]

The authors also make an important note that while traditional active learning may only request a single label per round, '[i]n practice it is not cost- effective to test a single example at a time'[10]. This limitation comports with the opinions of the biologists who I consulted when designing this experiment, and like Warmuth et al., this thesis uses a 5% batch size as the lower bound for requested labels in each round.

Active learning has an inherent dichotomy, often referred to as 'Exploration vs Exploitation'. For any given model and pool of unlabeled data, two options present themselves. One can use the model to 'explore' the unlabeled data, requesting labels for data-points for which the model has high uncertainty. Feeding this data back to the model will improve the model's predictive performance. On the other hand, one can exploit the model's predictions. This means retrieving labels for data-points the model predicts to be positive with high confidence. While the model's general predictive performance may not improve as substantially, the result aligns nicely with this thesis' goal of providing a pool of confirmed active compounds and a relative enrichment of active compounds. See [50] for a more thorough discussion of the implications of 'Exploration vs Exploitation' in the context of drug discovery.

In the past decade, multiple active learning for drug discovery papers have been published [9][8][51][52]. However, each selects a single data-point for labeling in each iteration. While this allows for impressive results, it is not feasible in a HTS setting for reasons noted in Sec. 3.1 . Unfortunately, this renders many of their conclusions regarding selection strategy somewhat inapplicable to our problem. Thus Warmuth et al.'s preference for a greedy strategy, which matched my own intuitions, and the desires of the biologists to get confirmatory data on predicted actives drove my selection strategy selection.

2.2.3.3 Iterative Screening

While the modern HTS paradigm is evidently highly refined, its requirement to physically interact with experiments is inherently limiting. To (painfully) rephrase Watson and Crick, it has not escaped notice that the entirely independent nature of each well's experiment immediately suggests a possible parallelized computational modeling opportunity for High Throughput Screening. While Warmuth et al. framed this as Active Learning, a parallel idea has existed in drug discovery context often terms as 'Iterative Screening' or 'Sequential screening'. Rusinko et al. describes using recursive partitioning (a t-test based decision tree) to select likely actives for screening in what they term 'Sequential Screening' [53]. Kutchukian et al. utilize a diverse selection of compounds as a starting point to train a naive Bayesian classifier to predict likely actives. These predictions are used for a single additional iteration with the goal of generating a 'focused library' [54]. Svensson et al. use conformal predictors with an RF base classifier trained on a starting set to predict further actives from the unscreened pool for a single further iteration. This two-step iteration yielded approximately 57% of actives [55]. A further study by Svensson et al. applied a similar technique to HTS datasets from PubChem, this time using a cost function along with conformal predictors to select variable sized secondary iterations [56]. Paricharak et al. describe building small 'informer sets' intended to initialize models for active learning by leveraging historical HTS data. They note that random sampling may not provide optimal coverage of chemical space [57]. Maciejewski et al. explored various active learning strategies for iterative screening on internal Novartis datasets. They conclude that selecting weak actives (i.e. those with low confidence) improves model performance. While their paper demonstrates interest in the general concept of iterative screening, their methodology uses only one or two follow-up iterations [58].

To my knowledge, no paper published under the active learning paradigm nor the iterative screening paradigm has sought to systematically compare various predictors nor have they focused on developing methods viable for laboratory application on genuine HTS datasets. This thesis seeks to correct this gap in the literature.

2.2.3.4 Imbalanced Data

As noted previously, a particularly challenging aspect of working with HTS is the extreme imbalance inherent to the data. At the extreme, fewer than 1% of compounds in a library will be active [51]. Even in less extreme cases, a hit rate around 5% can be expected. The implications for ML in this type of environment are substantial. A model predicting only negative would have a ~95% accuracy, and traditional measures like an ROC curve are likewise uninformative. Precision-Recall curves and Matthew's Correlation Coefficient are better metrics for assessing performance [59]. The issues do not stop at human interpretation of the model performance, indeed it is very difficult to make a model perform at all in this setting [60]. As noted by Chen et al., 'the most commonly used classification algorithms do not work well for such problems because they aim to minimize the overall error rate, rather than paying special attention to the positive class' [61]. There are two substantial ways to address this, cost-sensitive learning, and re-sampling. Each method attempts to increase the contribution of minority class error to totality of the error term. Cost sensitive learning re-weights the members of the minority class, essentially up-scaling their contribution to a summed error term. In contrast, re-sampling shifts the population of the classes either by increasing the number of minority members or by decreasing the number of majority members. Each has its flaws. In cost sensitive learning, an aggressive rescaling can create excessively large gradients, detrimental for learning. Likewise, over or under-sampling imply either duplicating work on repeated minority data-points or discarding potentially useful information about the majority. Some variants of oversampling attempt to address this duplication of work by creating synthetic minority points. This technique, known as SMOTE, essentially places new minority points at the midpoints of lines drawn between minority data-points in descriptor-space [62]. This technique has previously been applied to HTS data, with claimed benefits [63]. However, a recent survey found little evidence for SMOTE outperforming simple Random Over Sampling (ROS) [64], and previous work suggests that ROS helps classifiers sensitive to class imbalance and has no impact on those that are insensitive [60].

Moreover, SMOTE does not make sense for bit-array molecular fingerprints and is inapplicable to molecular graphs. As detailed below, I decided to mix both cost sensitive learning with re-sampling to mitigate the drawbacks of each.

Chapter 3

Initial Experiments

3.1 Problem Formulation

The goal of this thesis is to evaluate various ML based iterative screening strategies, and there is merit to explicitly formulating the problem at hand. As previously noted, large HTS campaigns are extremely expensive. This puts them out of reach of smaller academic laboratories, and limits their use even at larger non-profit laboratories such as the Alzheimer's Research UK Drug Discovery Institute (DDI). A typical compound library at the DDI can range from ~50,000 to ~120,000 compounds, costing ~£1.20 per compound. The primary goal of this thesis is to investigate and refine methods for minimizing the number of compounds screened in a campaign typical for the DDI while maximizing the number of actives found from the library. Central to this goal is a desire to constrain the parameters of these methods in such a manner than they can be implemented for future DDI campaigns.

Iterative Screening, which can be considered Active Learning, poses a solution to this problem by breaking a campaign into batches in which labeled data are used to train a classifier, and this classifier selects unlabeled compounds from the library for the subsequent batch. Two experimental factors make the traditional Active Learning approach of requesting labels one at a time infeasible. The first is time. Running a single experiment and working up the data may take several days, a 100,000 compound library would not be screenable in a human lifetime. The second is repeatability. Modern HTS methods store compounds in the same size plates as

those used in screening. Chemical samples degrade when handled repeatedly, so repeatedly accessing a plate to screen a single compound damages the others stored on the plate. Perhaps more importantly, the targets are often cultured cell lines. These cells are living populations and can easily die. Even worse, they may begin to undergo biological changes, subtly diverging from each other as experiments progress. Losing a cell line can prematurely end a campaign and a diverged line might yield misleading data.

Cognisant of these limits, I asked the DDI's head of biology (an expert in HTS) to detail the experimental limits for an Iterative Screening method at the DDI. She identified two crucial considerations. First, any solution could not require screening more than 50% of the library. At that point the benefits of reduced screen count are superseded by the additional work required to batch the campaign. She sought a solution that performed best when screening between 30% and 35% of the library. Second, in order to reduce the potential degradation and divergence mentioned above, batch-sizes could not fall below 5% of the library. She considered it acceptable to reduce the number of actives recovered in order to yeild a substantial reduction in iterations (e.g. moving from 5% iterations to 10% iterations).

The parameters above help define the question which could be phrased "Can active learning reduce the cost of a typical DDI HTS campaign subject to constraints on batch size no smaller than 5% and scanning no more than 50% of the library? If so, what percentage of actives can be expected when screening 35% of the library and what parameters of the method maximize that?" As mentioned in the introduction, I am unaware of any previous work that has taken experimental concerns into account in designing an Iterative Screening technique. Similarly, this thesis is the first to compare multiple Machine Learning methods in an Iterative Screening framework.

3.2 Datasets

I downloaded nine representative HTS datasets from PubChem Bio-Assay to use for developing the Iterative Screening methods (see Table 3.1). These datasets av-

eraged ~85,000 entries, with between 1 and 7 % active compound (4.1% average). Using RDKit, two molecular embeddings were calculated. First, 1024 bit Morgan Fingerprints with r=2 were calculated for each molecule in each of the campaigns. Morgan Fingerprints are RDKit's implementation of ECFP, with r=2 roughly equal to ECFP4. Additionally, I used RDKit to calculate 97 1D and 2D molecular descriptor for each of molecules in the campaigns. More details about these implementation of these embeddings are available in Appendix C.4. These two descriptor sets, and their concatenation, served as the initial set of options for embeddings.

The raw signal from an HTS screen can be anything from a light intensity to a measure of cell viability. To unify these disparate signals, HTS data on PubChem are provided as a corrected and normalized PubChem Activity Score scale (0-100) [65]. Because the data are derived from single replicate biological experiments, there is a high degree of noise associated with the scores [66]. Additionally, each compound is provided with an activity label either binary (Active, Inactive) or triplet(Inactive, Inconclusive, Active). While there may be merit in attempting regression on the Activity Scores, the objective of screening is primarily to discover starting points for further modifications. Thus, establishing relative potency among the active compounds is of little benefit. In typical laboratory practice at the DDI (an in HTS in general), compounds are binned into either binary (Active,Inactive) or ternary (Inactive,Inconclusive,Active) labels. While I initially experimented with maintaining Inconclusive labels, they were not universally present in the nine HTS datasets, and the DDI's HTS experts indicated that they would consider Inconclusives to be Inactives in most settings. My initial experiments with the three class labels also provided no clear improvement in classifier performance relative to the binary labels. For these reasons, in each case I formulated the problem as binary classification: Active vs Inactive. In cases where Inconclusive labels were present, I relabeled them as Inactive.

AID	Active	Total	Active Percentage
AID_1345083	6153	93211	6.6
AID_624255	4582	76537	6.0
AID_449739	4230	104728	4.0
AID_995	707	70898	1.0
AID_938	1794	72026	2.5
AID_628	2179	63656	3.4
AID_596	1391	69668	2.0
AID_893	5649	73912	7.6
AID_894	6428	148481	4.3

Table 3.1: HTS Datasets from PubChem, AID is the record identifier for each dataset. Note the distributions of Active Percentage and Library Size

3.3 Classifier Evaluation

In preparation for evaluating iterative screening, I set out to benchmark various classifiers and embedding techniques. The immediate goal was to understand the behavior of models trained either with very little data or with a substantial portion of a particular data-set. This was intended to simulate the beginning and end of iterative experiments (10% and 80% of the library, respectively) without the computational costs. At these training pool sizes, I explored the performance of various classifiers from previous literature citations while varying metaparameters of the problem formulation including embedding type, and data balancing (I refer to the parameters of a strategy as metaparameters to avoid confusion with a model's hyperparameters). Initially, I considered four classifiers: Random Forests, Gradient Boosting Machines, Support Vector Machines, and Deep Neural Networks. Embeddings were Molecular Fingerprints, Molecular Descriptors, or a concatenation of both. Additionally, I considered keeping the natural active/inactive ratio or either using random over-sampling (ROS) or random under-sampling (RUS) to re-balance the ratios to 3:1 Inactive to Active. In each case, the problem was framed as binary classification of compounds into Active and Inactive classes.

For the given problem, the goal is different from the goals of traditional binary classification. Chemists care primarily about identifying active compounds in the set of unscanned molecules, any improvement in this hit rate above random represents a reduction in cost and a gain in speed. Moreover, because the goal is

to iteratively improve the model, I put considered False Negatives and Positives to have different values. False Positives serve only to help refine the decision boundary in the classifier for future rounds, while False Negatives represent a missed active that in some settings could literally be a cure for cancer. With this in mind, I chose Recall for the active class and Matthews Correlation Coefficient as the primary metrics of success. I intended Recall to measure classifier's ability to find actives while Matthews Correlation Coefficient would provide a measure of overall classifier performance given the significant class imbalance and also confirm that classifiers were not simply achieving high recall by classifying all compounds as active. Additionally f1 was recorded along with and precision and recall for both classes. These measure were less aligned with the thesis goals for reasons stated above.

Model details for these for all experiments are available in the appendix (Apx C.4). In summary, hyper-parameters were set close to defaults, while enabling weight balancing for cost sensitive learning. An important note: given the high dimensionality and sample size for the datasets, the SVM model was scikit-learn's SGDClassifier which, with hinge loss, approximates a traditional SVM. Platt Scaling via scikit-learn's CalibratedClassifierCV method was use to generate probabilistic predictions for the SVM. Other classifiers all returned probabilistic predictions either by default (GCNNs and DNNs) or via a built in `predict_proba` method. The DNN model was built in `tensorflow.keras`.

Prior to this experiment, I explored several design components of the DNN. I initially experimented with Focal Loss [67], which up-weights loss gradients for hard to classify examples. While it was developed for use in imbalanced classification scenarios, it assumes that the majority class will be be easy to classify. This assumption may not hold true in this problem setting, and the re-weighting led to unstable gradients and poor learning performance, so I reverted to categorical cross-entropy (which allowed experiments with both binary and ternary labels).

3.3.1 Experimental Details and Classifier performance

The nine HTS datasets were processed with RDKit to extract Morgan FingerPrints (radius=2) and 97 Molecular Descriptors (referred to as MFP and MolChars respectively in plots), further details of which are available in the appendix (Apx C.4). Using stratified 5-fold cross validation, five 80% / 20% splits were created. Each 80% split was used to generated 8 further splits, each 10% of the size of the total library. Each classifier was trained on the splits, varying the compound embeddings and re-balancing techniques and tested on the corresponding 20% test split. The results are shown in Fig 3.1. The complexity of this graph shows the impossibility of using these initial experiments to select a classifier. It is interesting to note that, retrospectively, it seems that precision is more predictive of performance than recall as RF and LGBM tended to perform best in the iterative setting. At the time it was unclear how the metrics would relate to performance in the iterative setting. However, it was possible to compare performance across embedding type and imbalance correction. I used MCC to make these comparisons as it provided the most holistic measure of classifier performance (score of 1.0 is perfect labeling while 0 indicates labeling no better than random prediction).

3.3.2 Over vs Under Sampling

As noted previously, HTS datasets are significantly imbalanced, and a popular way to address this imbalance in re-sampling [68][64]. I used the imbalanced learn library [69] to re-sample the minority majority ration to 1:3. Box plots showing the results from this experiment are displayed in Figure 3.2. The clearest conclusion is that under-sampling caused a decrease in performance for small datasets, most evident in the MCC scores for the DNN trained on a 10% split (see bottom right of Fig 3.2). Over-sampling improved 10% performance for all but the DNN, surprising given the literature suggesting otherwise. Loss curves for the DNN suggested rapid over-fitting, logical given the substantial repetition implied by oversampling a 6% class to 25%. In other classifiers, oversampling seemed to have a normalizing effect, reducing extreme low scores. In an iterative setting, I considered it to be more detrimental to start with a very poor classifier than beneficial to start with a very

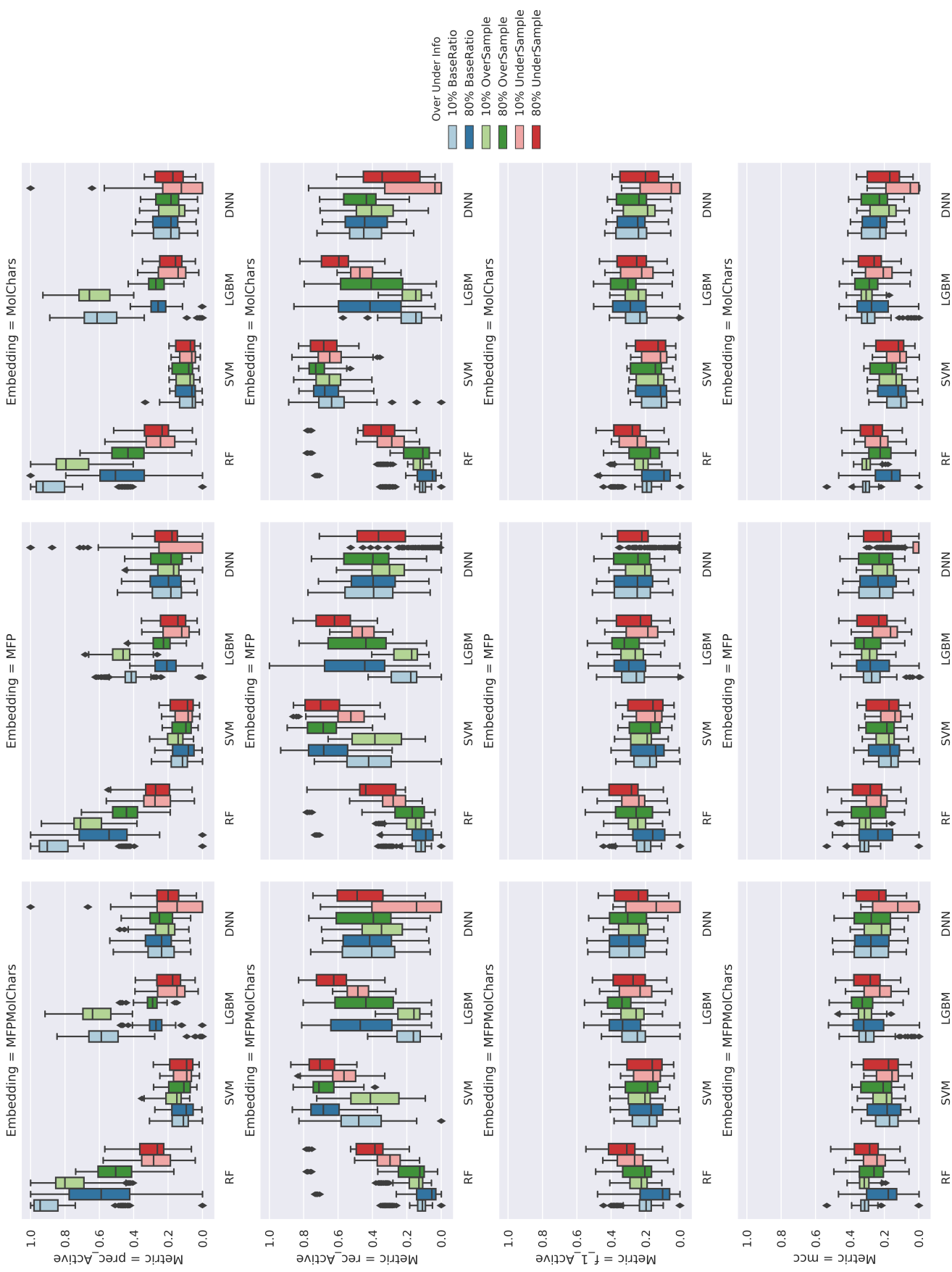

Figure 3.1: Precision,Recall, f1, and MCC for initial benchmarking. RF and LGBM had noticeably better precision at the expense of recall. SVM reversed this ratio while the DNN showed less of a disparity. With the benefit of hindsight, the classifiers with higher precision performed better than those with high recall. At the time, it was not clear how any of these metrics would translate to performance in an iterative setting.

strong one. Moreover, as the iterative process progresses, active composition of the labeled pool will climb reducing the required oversampling. This, combined with the possibility of hyper-parameter tuning to reduce DNN over-fitting and a desire to maintain identical experimental conditions across classifiers drove a decision to continue with oversampling in future experiments.

3.3.3 Embedding choice

With the decision to proceed with oversampling, it became important to investigate the influence of embedding on the performance of the classifiers trained on over-sampled 10% training splits. Plotting MCC for each embedding (see Fig. 3.3) revealed that the concatenation of Molecular Fingerprints and Molecular Characteristics provided a small performance boost over either embedding by itself. Therefore, I selected the combination as the primary embedding going forward.

3.4 Initial Iterative Testing

For the given problem, the goal is different from the goals of traditional binary classification. I care primarily about identifying active compounds in the set of unscanned molecules, any improvement in this hit rate above random represents a reduction in cost and a gain in speed. Moreover, because I intend to use an iterative selection strategy that prioritizes scanning expected Active compounds, False Positives serve only to help refine the decision boundary in the classifier. As mentioned previously, prior to beginning the design of the iterative strategy, I spoke with HTS experts to determine the parameter bounds for the strategy and also understand desired performance. There were two significant takeaways. The first was they preferred to maximize performance achieved when 30% of the library was scanned, and felt that any solution scanning over 50% of the library would not be worthwhile unless it could discover 100% of active compounds. The second was that they preferred an iteration size equal to 10% of the library and that it would be impractical to use batches of less than 5% of the total library size. These bounds defined the experimental design, experiments would be run until 50% of the library was consumed, in batch sizes between 5 and 10%. In addition to considering the

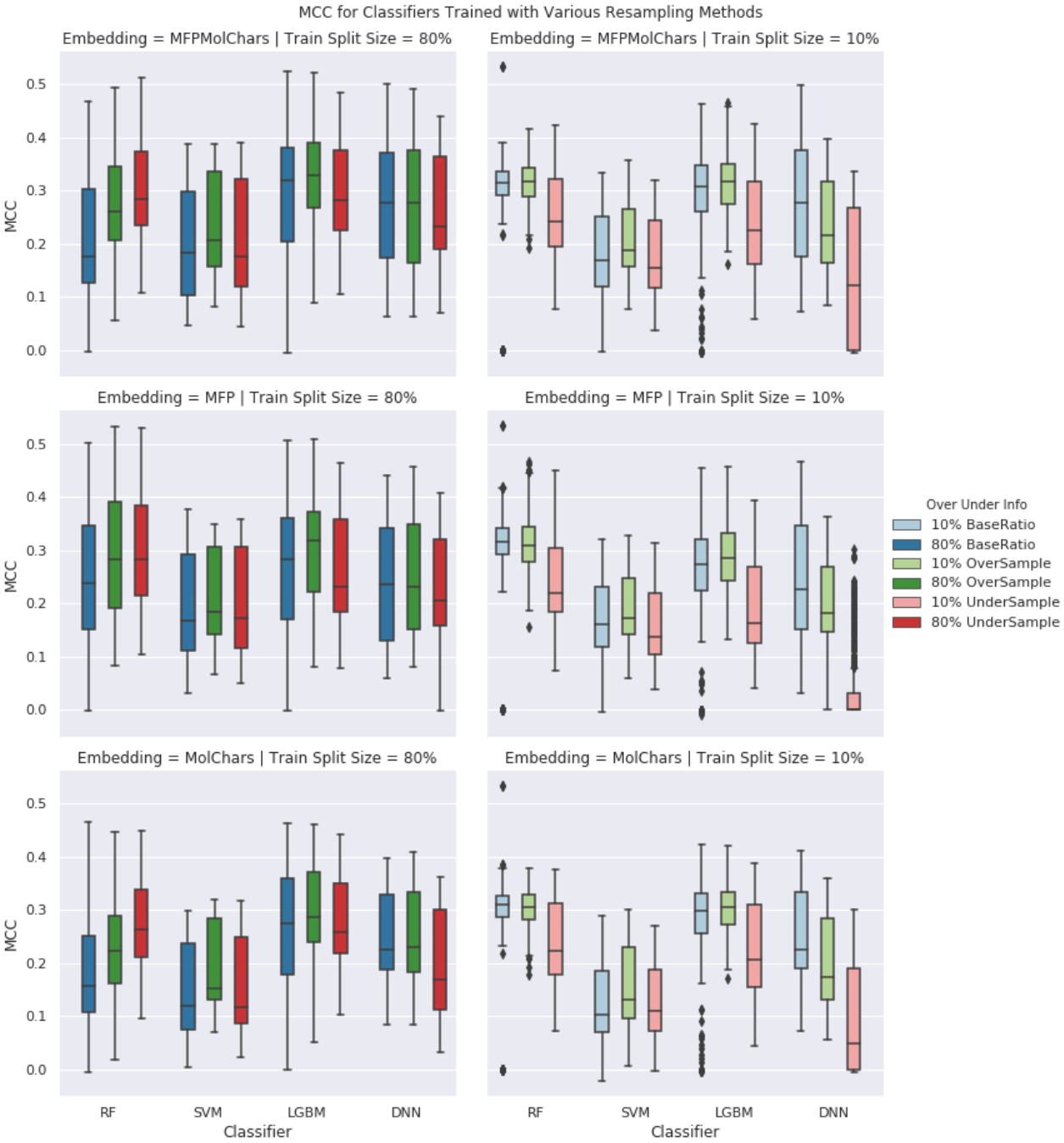

Figure 3.2: Matthew's Correlation Coefficient for initial benchmarking experiments. Note that over sampling improves classifier performance and/or reduces low scoring outliers on 10% splits except for DNNs. At 80%, RF benefit more from under-sampling than over-sampling

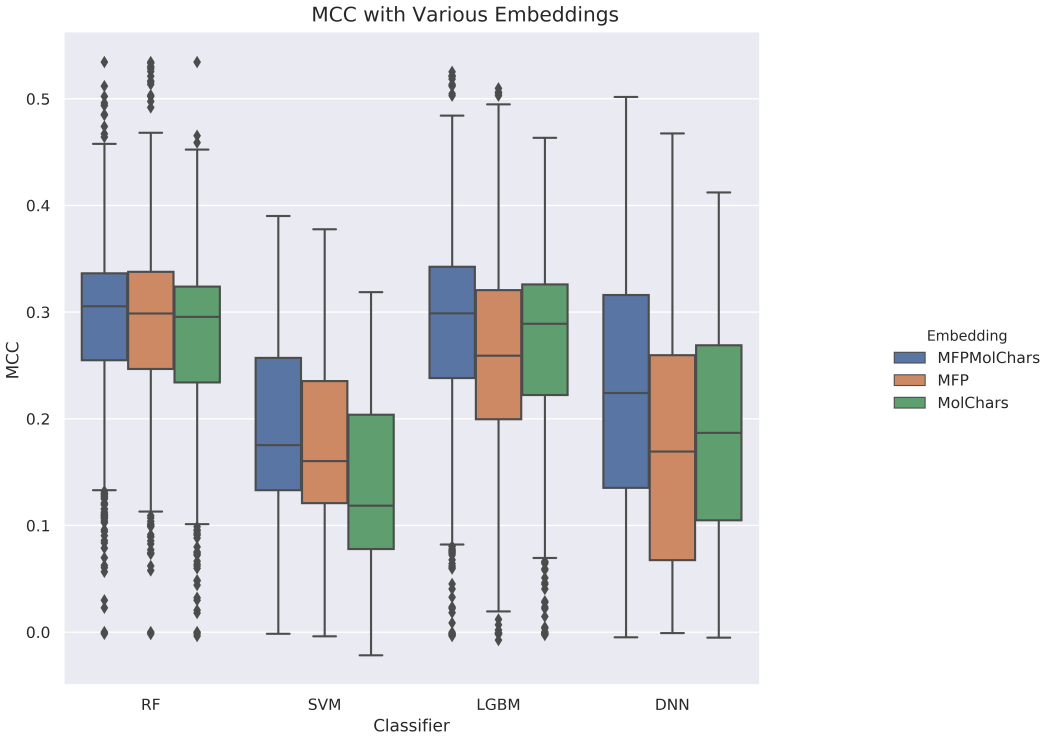

Figure 3.3: Matthew's Correlation Coefficient for Classifiers Trained with Over-sampled 10% Training Splits and Various Molecular Embeddings. Note that in all cases, the combination of Morgan Fingerprints and Molecular Descriptors outperforms either embedding by itself.

choice of classifier, I set out to explore a limited number of metaparameters for the active learning experiment. These included, initialization, selection strategy, initial training size, and iteration size.

3.4.1 Initial Start Selection

Active learning suffers from a cold start problem. One must select a subset of the library to screen in order to generate data to train on. This raises two questions: how much should be screened and how should it be selected? Cortés-Ciriano et al. found that a more structurally diverse starting set led to increased performance for iterative screening [70]. RDKit [71] provides an implementation of the MaxMin diversity selection algorithm [72], which seeks to maximize the distance between selected compounds Morgan Fingerprints. Crucially, it does not require computation of a full distance matrix (allowing its use on large datasets). This implementation was used to select all starting pools, which were set to 10% of library size.

3.4.2 Strategy for Requesting New Labels

Selection strategy proved complicated to define. In talking with the DDI's HTS expert, it was made clear that there was merit in screening predicted actives beyond simply retrieving data for further learning. In particular, because the target performance was at 30-35% there was an incentive to screen possible hits early, supporting an exploitative strategy. Adding more actives in early rounds also promised to increase the pool of active to be re-sampled from. However, [58] had found that selectively screening high uncertainty compounds improved model performance. As such, I decided to choose a partially greedy strategy that would take mostly highly predicted compounds along with a small percentage of highly uncertain compounds to refine decision boundaries. An additional concern was over-fitting to initial actives. Given the extreme class imbalance, it was likely that only a few actives would be present in the first training batch. With the small number of actives examples available to the classifier, it might struggle to recognize the remaining unlabeled actives if they were in 'pockets' of embedding space that had either not been sampled, or were surrounded by inactive labels. I believed that exploring chemical space away from the currently labeled compounds could prevent this. Therefore I decided to either add a random selection of compounds, or use the diversity selection algorithm to explicitly retrieve compounds dissimilar to those previously scanned. Either option made the selection strategy 'ε-greedy'.

3.4.2.1 First Iterative Experiment

Initially, I ran a single replicate experiment with the nine datasets and four classifiers (SVM, RF, LGBM, DNN). I used a complicated selection strategy to split predictions into active and inactive at 0.5 and sampled the top and bottom of the actives, the top of the in-actives, and added a set of 'exploration' molecules. If insufficient actives were predicted, all predicted actives were selected with the remaining space in the batch split evenly between weak negatives and exploration. This strategy is depicted in Fig. 3.4, and the details of the experiment are available in C.4. Two variants of this experiment used either the MaxMin picker or random selection to fill in the set of 'exploration' molecules. Starting size was 10% of the

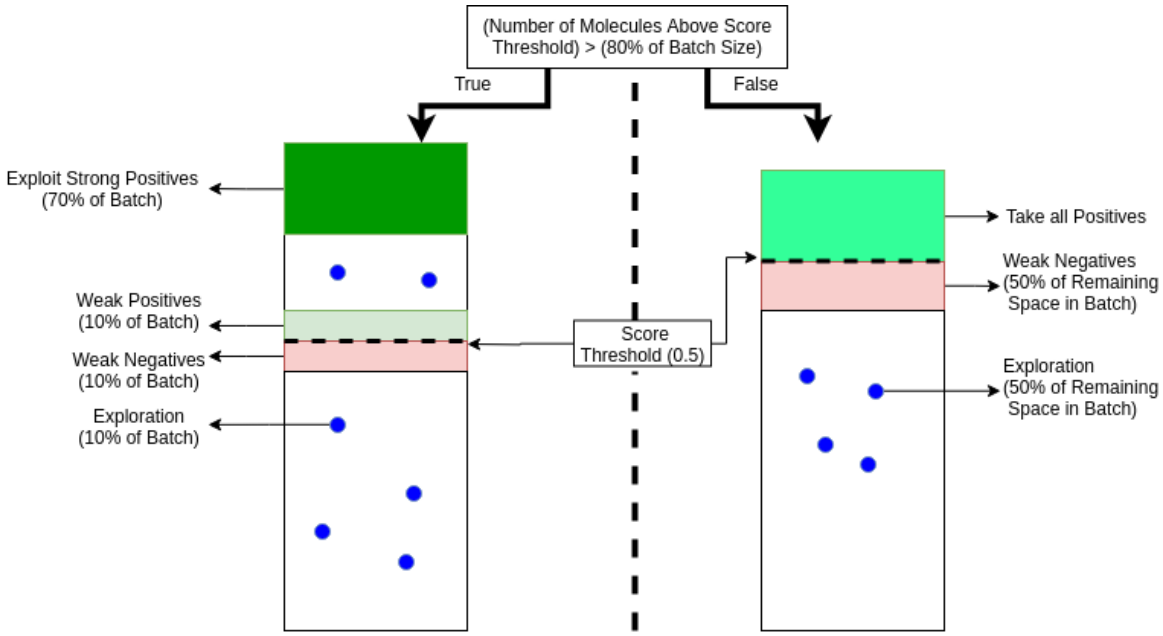

Figure 3.4: Initial selection strategy combined uncertainty sampling, exploitation, and exploration of the library. Later experiments demonstrated that this complicated strategy was unneeded.

library and iterations were 5%. This experiment confirmed that a first attempt could still retrieve at-least 68% of the actives at 35% of the library and ~85% at 50% of the library (see Fig 3.5). All classifiers showed similar average performance across the datasets, with SVM lagging slightly behind the others as seen in Fig 3.6. Additionally, the use of diversity selection decreased performance relative to random exploration as seen in Fig3.5. It is difficult to draw too many conclusions from this because the next experiment demonstrated that thresholding the predictions had the effects of quickly limiting the number of predicted actives, and thus had an undue influence on the strategy.

3.4.2.2 Graph Convolutions

Having validated that the initial experimental strategy worked, I was interested in exploring the performance of Graph Convolution Neural Networks (GCNNs) which represent a different method of classification, and also use an alternative embedding technique. As mentioned in the literature review, this method uses a series of units to aggregate information across an undirected graph that represents the molecular structure. The edges of the graph represent bonds and the nodes atoms. Nodes (and

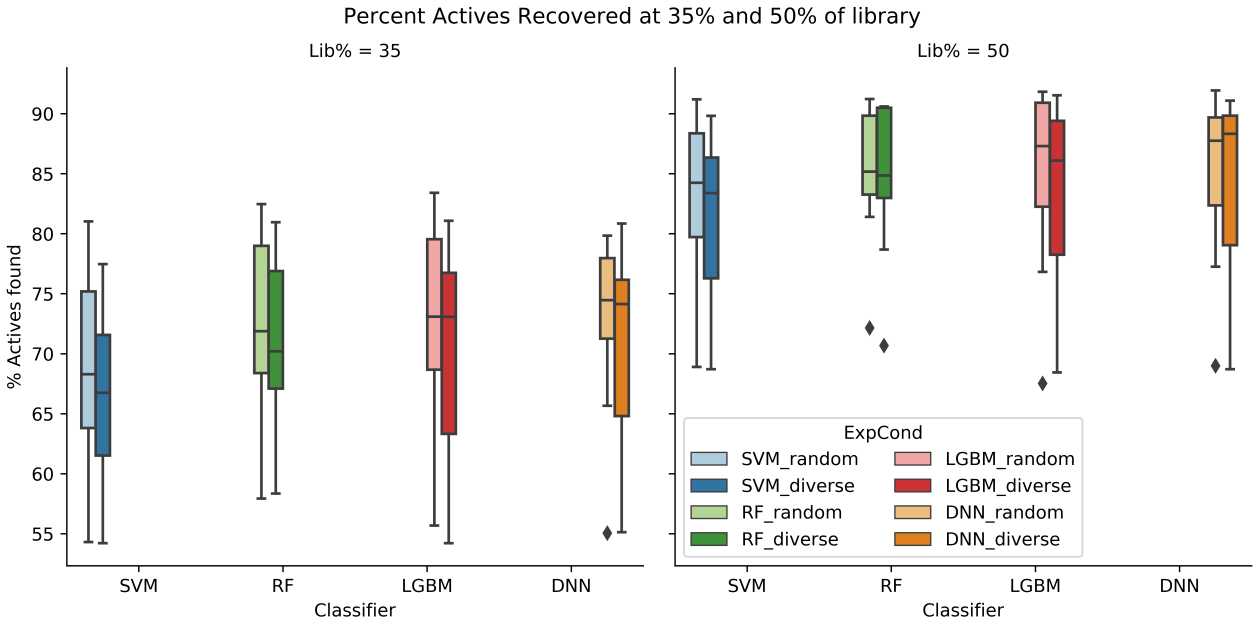

Figure 3.5: Percent of actives recovered at two checkpoints (35% and 50% of library screen). Again, SVM underperforms other classifiers.

in some implementations, edges) have descriptor vectors associated with them.

The first experiment to introduce GCNNs also simplified the previous selection algorithm. If there were not sufficient predicted actives to satisfy the top and bottom sampling, all predicted actives were selected and the iteration padded with random samples from the library, discarding the uncertainty selection. Results are seen in Fig 3.7. While the GCNN performed well, approximately matching previous classifier performance (~80% recovery at 50% of the library) in the initial experiment (see Fig 3.6), the other classifiers suffered significant performance declines. The only experimental change for these classifiers was the removal of weak inactive sampling, suggesting that this sampling was important. To confirm, I implemented and ran a purely greedy strategy.

3.4.2.3 Pure Greed

The purely greedy strategy was intended to identify classifiers that would benefit from the weak negative sampling, as they would likely perform poorly when only given labels for predicted positives. The selection strategy ran for a maximum of 10 iterations or until 50% of the library had been scanned (whichever came first).

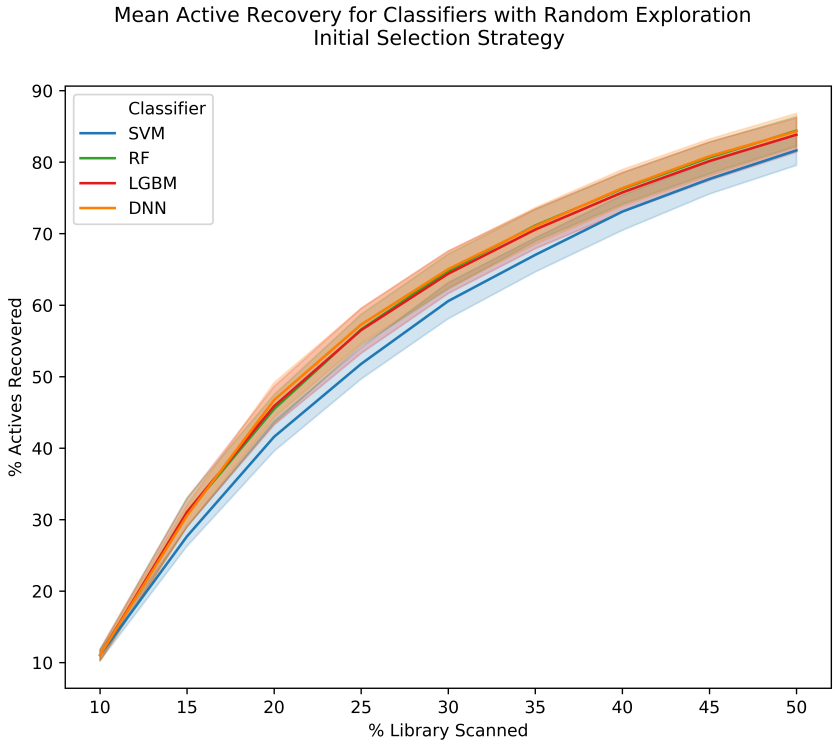

Figure 3.6: Classifier performance with initial selection strategy. SVM which had high Recall performs the worst, suggesting that more holistic metrics like MCC may be more predictive. Note that this experiment used random rather than diverse exploration. Shaded intervals represent 68% CI.

Within each iteration all molecule predicted active were provided as labeled training data for the subsequent round. The results, shown in Fig 3.8, indicated that some classifiers were simply more optimistic (i.e. generated more predictions above 0.5) than others. In particular, the GCNN requested substantially more labels than any other method, and in doing so demonstrated the best absolute performance. However, the slopes of lines for the GCNN (which represent the efficiency of each classifier as $\frac{\text{actives recovered}}{\text{samples requested}}$) were lower than all but those of the SVM. This implied that the GCNNs previous out-performance of other classifiers (seen in Fig 3.7) may have simply been due to a mismatch between strategy and classifier output. If classifiers made fewer predictions over 0.5, then they would more quickly be forced into the heavy random sampling of the second part of the modified initial strategy. To counter this mismatch, I simplified the strategy and refocused it on ranking compounds rather than establishing a threshold by which to classify them. This

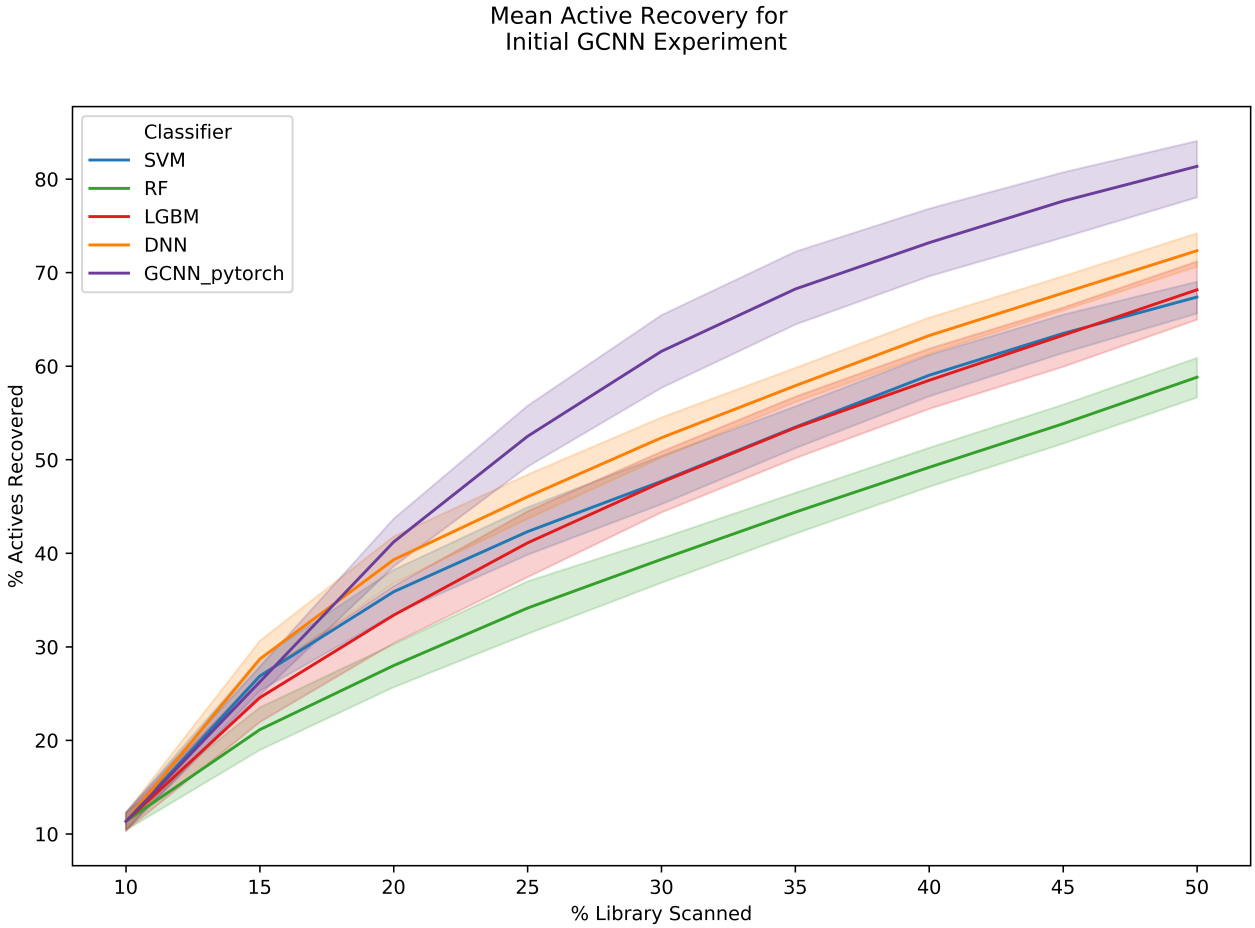

Figure 3.7: Classifier (including GCNN) performance with modified selection strategy. The change in strategy caused a decline in the non-GCNN classifiers relative to their previous recovery rates. The GCNN performed similarly to the other classifiers' previous performance. This disparity spurred further investigations into the impact of strategy on active recovery. Shaded intervals represent 68% CI.

experiment exposed another interesting point: the slopes of the classifiers implied that they could perform substantially better if allowed to use smaller iteration sizes and higher iteration counts than those requested by the HTS experts. For instance, LGBM often retrieved around 30% of actives using 10 iterations that scanned less than 20% of the library in total, suggesting it could perform well with batches far below 5%. While I did not consider exploring this due to my interest in meeting those requests, these results align well with [52].

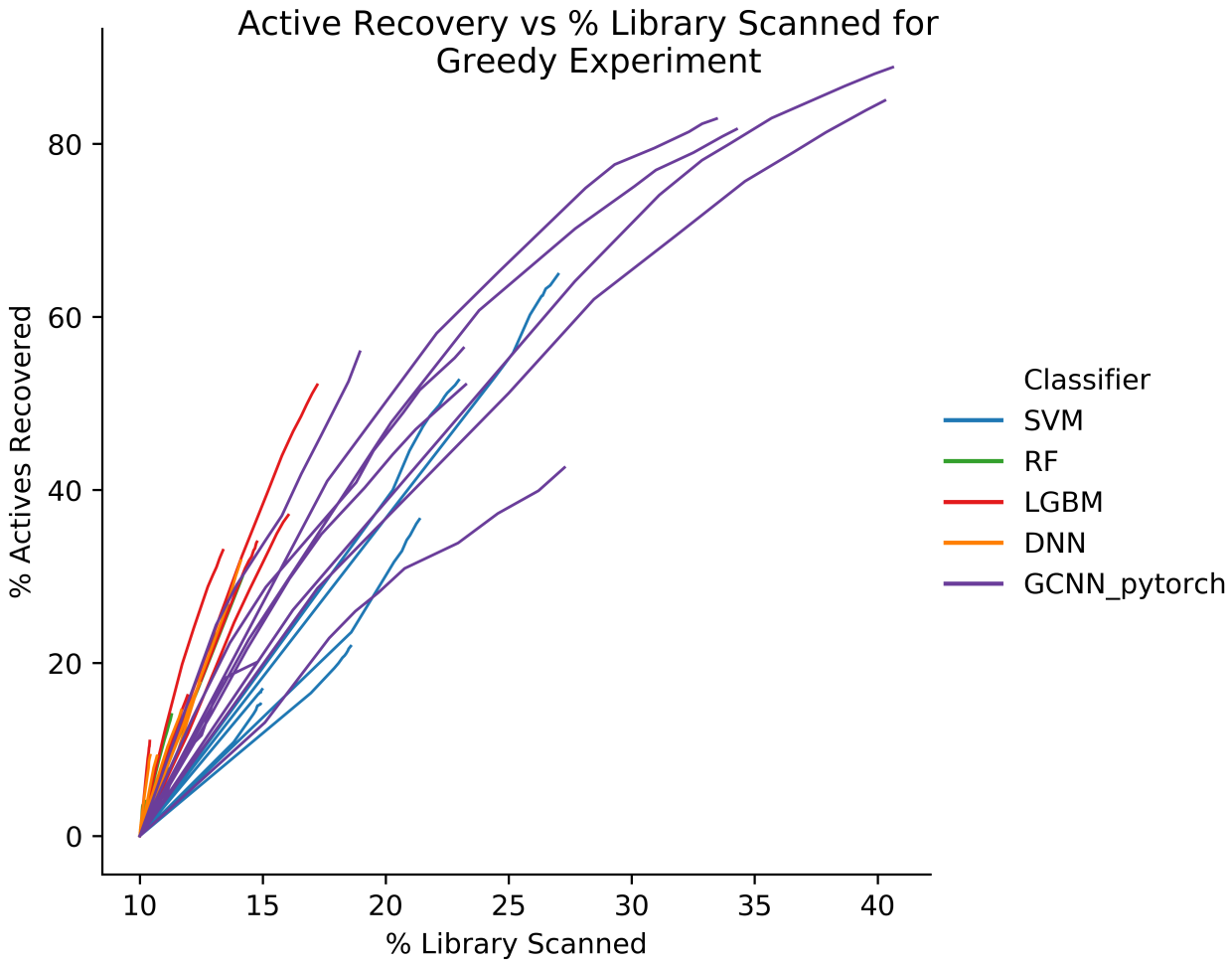

Figure 3.8: Actives recovered vs % library scanned with greedy strategy (selecting all compounds above threshold for scanning). Each line represents a different AID and Classifier combination. Slopes are a rough approximation of efficiency, with RF and LGBM appearing the most efficient. This plot demonstrates that the GCNN is much more 'optimistic', with many more super-threshold predictions than the other classifiers.

3.4.2.4 Ranked ε-Greedy

The previous experiments demonstrated that a complicated strategy that relied on thresholding predictions was obfuscating the true potential of several of the classifiers. As such, a simple strategy (ranked ε-greedy) was established. Given a fixed batch-size, a fixed portion of the batch (of size $x = (1-\varepsilon) * \text{batch size}$) would be the top x predicted compounds (ranked by predicted probability), while the remaining portion of the batch (of size $y = \varepsilon * \text{batch size}$) would be sampled from the pool of unlabeled compounds to encourage exploration of the chemical space. Concerned with ensuring appropriate exploration, ε was set to 0.2. This strategy removed the need to explicitly balance exploiting strong actives vs refining the decision bound with weak actives or in-actives. I predicted that initial iterations would 'discover' new actives, and in the process enrich the ratio of actives in the training pool. Having exhausted the supply of active similar to those discovered in the initial screen, later iterations would select lower confidence molecules, and thus would help to refine the decision boundary and also exploring neighborhoods around actives discovered via the exploration sampling process. Additionally, because the strategy relied on a continuous ranking rather than thresholding, I began measuring Area Under the Curve for the Precision Recall Curve (AUC-PRC) as a classifier performance metric. In essence, the ranking method chose a threshold that corresponded to the score of final compound in the top 4%. Because AUC-PRC summarizes a classifier's performance at all possible thresholds and the precision recall curve measures classifier performance well for imbalanced data [59] it now aligned well with the study's objectives (replacing MCC).

The new strategy reversed the loss in performance for the non-GCNN classifiers observed under both the greedy and modified initial strategies. As seen in Fig 3.9, RF, LGBM, and the simple DNN all now outperformed the GCNN which, in turn, outperformed the SVM. It is important to note that this single experiment, three replicates over nine data sets, required 36 hours of computational time. This was primarily driven by the GCNN, for which a single epoch took longer than the fitting of an entire RF model. Despite consuming substantially more computation

resources, the GCNN under-performed relative to RF, LGBM, and DNN. With these three models recovering ~65% of actives with 30% of the library and nearly 85% at 50% scanned, I had substantially reduced the costs of screening for a hypothetical campaign. Indeed, given the biological noise inherent to HTS, I believed that achieving 100% active recovery would have inherently implied over-fitting to the noise that generates False Positives in the HTS datasets. While not directly applicable to this problem formulation, a previous study of experimental uncertainty of publicly available binding affinity data (the signal which is binned to Active and Inactive labels for HTS is an indirect measure of binding affinity) supported this belief. In this study, the authors concluded that, for regression models, the highest possible R^2 value was 0.81 [66]. While I could not conclude that ~85% represented the maximum performance for the models, I considered the primary question of the study to be answered: yes, applying Active Learning to HTS could substantially reduce resource use while maintaining experimental utility. That answer implied a further question, could the parameters of the solution be tuned to improve the performance? I set out to explore this question.

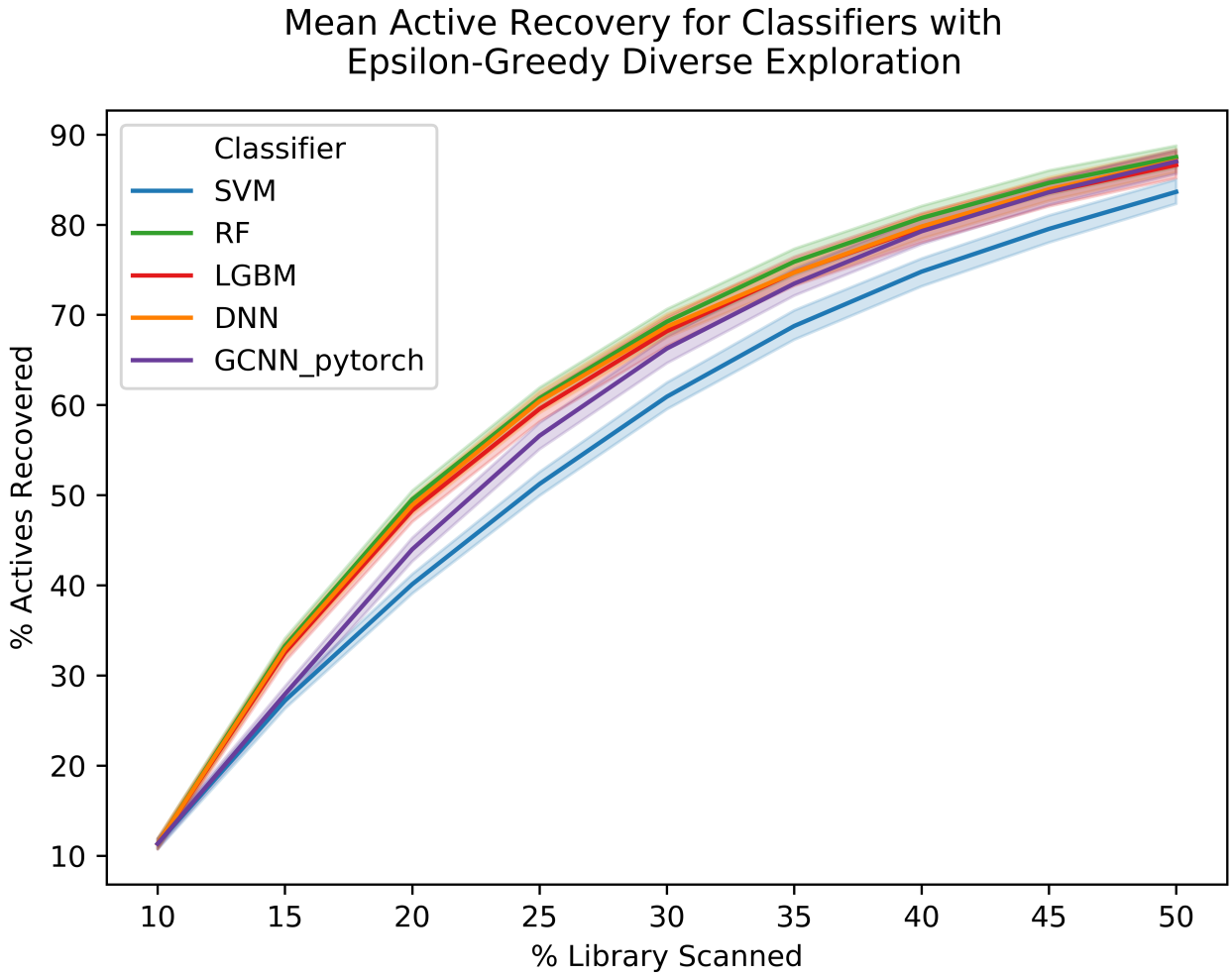

Figure 3.9: The ranked ε-greedy strategy increased active recovery for all but the GCNN which remained essentially unchanged.

Chapter 4

Model and Strategy Tuning

4.1 Hyperparameter Tuning

The previous experiments had already demonstrated significant potential to reduce the number of compounds screened in an HTS campaign. This occurred despite each classifier running with a single set of hyperparameters, essentially the defaults used in the packages from which they originated. Given the substantial run-times of a single three replicate experiment with all nine datasets and five classifiers, it would be impossible to run any significant hyperparameter tuning on the full experimental setup. Moreover, the realities of the use case make it difficult to justify serious hyper parameter tuning. If one wanted to conduct a 5-fold cross validation to tune a model for a certain data set, one would need all of the labels for said data-set. Acquiring the labels for the data-set would in turn require scanning the entire library, making the tuned model useless. One could use previously completed HTS campaigns to tune the model, but it was unclear if the tuned parameters would transfer well from one library-target pairing to another. I set out to make a limited investigation of that question.

Actively tuning the DNN and GCNN was unrealistic, their training times were long, and the parameter space was massive if architecture, activation, epoch number, regularization, batch-size, loss function, etc were included. The only modification I made was to reduce the number of epochs used by the GCNN from the DeepChem default of 100 to 20. Based on loss curves obtained in the ranked ϵ-greedy ex-

periment, I believed that 20 epochs would reduce over-fitting in early iterations while not adversely impacting performance at later points (see Fig A.4). For RF, SVM, and LGBM, a simple Bayesian hyperparameter tuning was completed using scikit-optimize [73] (see Appendix B for details). To avoid the long runtimes of full experiments, the tuning was completed attempting to maximize AUC-PRC for 10% samples of the library for each dataset. Optimization was completed independently for each dataset, as I wished to understand if the results were aligned for different experiments. If they diverged substantially, it would not be useful to attempt to transfer tuned parameters to new HTS experiments in the future.

The results for this hyperparameter tuning were hard to interpret conclusively. In some-cases, tuned parameters aligned (tables with tuning results are available in C.4). For instance, RF's `max_features` was 'log2' for seven of nine datasets. However, its `n_estimators` results were distributed evenly across the range allowed by the hyperparameter tuning. Similar results were observed for other classifiers. To test the impact of using these tuning results, I updated each model's hyperparameters to the mean of the tuned results for continuous variables and to the mode for categorical ones. Once the tuning was complete, I explored the strategy space.

4.2 Strategy Tuning

Having investigated the hyperparameters of the model, I sought to optimize the metaparameters of the strategy itself. Specifically, I examined how much of the library was scanned for initial training, how big the iterations were, and which type of exploration was best suited to the problem. Additionally, I added a 'Random' classifier to compare all classifiers to the result expected from simple random selection from the pool of unlabeled data (which is how an HTS campaign would typically proceed). This experimental setting provided an opportunity to return to the question of using a diversity selection algorithm vs random sampling to achieve this exploration. While initial experiments (see Fig 3.5) had suggested that the MaxMin selection slightly under-performed random sampling, I sought to

re-validate this conclusion with the new selection strategy and to run multiple replicates of this experiment. As the MaxMin algorithm scales poorly with increased library size to scan against, and because the biologists were most concerned about performance at 35% of the library, I decided to run experiments solely to 35% of the library size. First however, I performed a full run out to 50% to validate the new hyperparameters. This run (Exp_1 in Tab 4.1), demonstrated that the optimized hyperparameters for the SVM yielded a classifier that performed worse than random selection (see Fig A.2 for plot). Reverting the loss function from the tuning selection of `squared_hinge` to the original `hinge` corrected this performance decrease. Comparing results from this experiment (tuned hyperparameters with random selection) with those from the previous (untuned hyperparameters with diverse selection) demonstrated improvements in the three tuned classifiers at 35% of library scanned and improvements for the tuned RF and SVM classifiers at 50% library scanned. Fig 3.2 also shows that the GCNN's performance declined as the epoch count was reduced from 100 to 20. This is interesting, as plots of loss curves often showed over-fitting well before 20 epochs and rarely showed validation loss improvement after 20 epochs (Fig A.4). It may be the case that loss curves are less informative with highly imbalanced data. The DNN's performance was relatively unchanged between the two experiments. Because the DNN's hyperparameters were unchanged between experiments, this suggests that the change in hyperparameters, not the change in exploration strategy, drove the differences in relative performance between experiments for other classifiers.

To follow up this experiment, five further experiments were run to 35% of library scanned. These experiments (Exp_2 - Exp_5 in Table 4.1) varied start size, iteration size, and exploration type in a manner to enable pairwise comparisons ((detailed in Table 4.1).

The results (shown in Fig 4.2) allow several conclusions. The first is that the RF in Exp_1 had the absolute best performance of all classifiers and strategies. The second is that when comparing performance between exploration strategies for each classifier (i.e. Exp_2 vs Exp_3 and Exp_4 vs Exp_5), diverse exploration performed

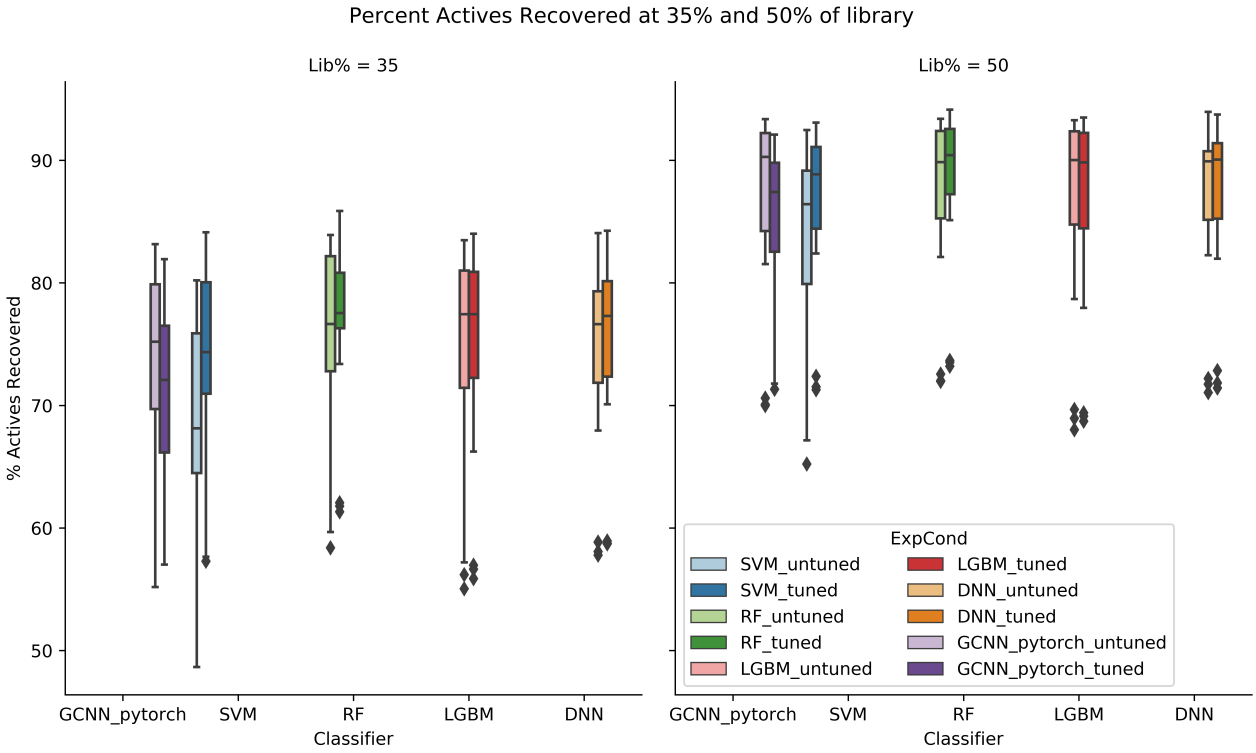

Figure 4.1: Classifier performance with ranked ε-greedy strategy, experiment shown with dark shade had tuned hyperparameters and a random exploration strategy, the light shaded experiment had un-tuned hyperparameters and greedy exploration strategy. Tuning improved SVM, RF, and LGBM performance at 35% raising the mediam for SVM and RF while shrinking LGBM's interquartile range. GCNN performance decreased with the reduction in epochs

Name	Start Size	Iter Num	Exploration
Exp_1	10%	5%	Random
Exp_2	15%	5%	Diverse
Exp_3	15%	5%	Random
Exp_4	15%	10%	Diverse
Exp_5	15%	10%	Random

Table 4.1: Experimental conditions with tuned hyperparameters

slightly worse than random exploration. The third, and possibly most practical, conclusion resulted from comparing RF performance between Exp_1 and Exp_5 with median active recoveries of ~0.77 and ~0.71 respectively. Exp_5's result took only 3 iterations, less than half of those required by Exp_1. For a research laboratory, it may be worthwhile to accept a 6% loss in performance in order to decrease workload, experimental time, and the degradation inherent in repeated handling of

chemical compounds.

The most significant result was the substantial increases in actives recovered with any of the classifiers when compared to the results expected from the current state of the art HTS (shown as random in Fig 4.2). Even in the worst cases, RF nearly doubles the expected recovery with the current screening strategy. Fig 4.3 shows boxplots of classifier performance calculated relative to that of RF at 35% of library (this was done for each experimental replicate on each dataset). It further demonstrates that RF outperformed all other classifiers across each experiment.

4.3 Confirmation

The very nature of the iterative screening problem makes it difficult to confirm a solution's performance. One cannot simply withhold a portion of a library to test upon later. Instead, to validate performance of the final strategy presented above (tuned hyper parameters, 10% start size, 5% batch size, and random exploration), three novel datasets were acquired from PubChem [65] (detailed in Table 4.2 below). These datasets varied in active percent and screening type and represented a variety of possible HTS scenarios. Using an identical pipeline to the previous nine training datasets, I ran simulated iterative screening campaigns with the finalized strategy in triplicate for each dataset. The results, shown below in 4.4, demonstrated that the proposed solution generalized to HTS campaigns independent from those it was developed on. The datasets used differing biological assays; two were phenotypic (for example, counting the number of cancer cells killed by a compound) while the third used laser excitation to measure how well compounds bound to a purified protein. Additionally, each dataset used a compound library and target not present in the nine development datasets. Despite these differences, the performance is similar. RF again provide the best mean performance, and GCNNs the worst. At the critical 35% checkpoint, ~71% of the actives have been recovered, and ~81% when half of the library was scanned. This performance demonstrated the generalizability of the method developed above.

AID	Active	Total	Description	Active Percentage
AID_1259354	1804	75924	protein	2.37
AID_598	5142	85200	cell based	6.01
AID_488969	2166	105151	cell based	2.06

Table 4.2: Details of HTS datasets selected for testing.

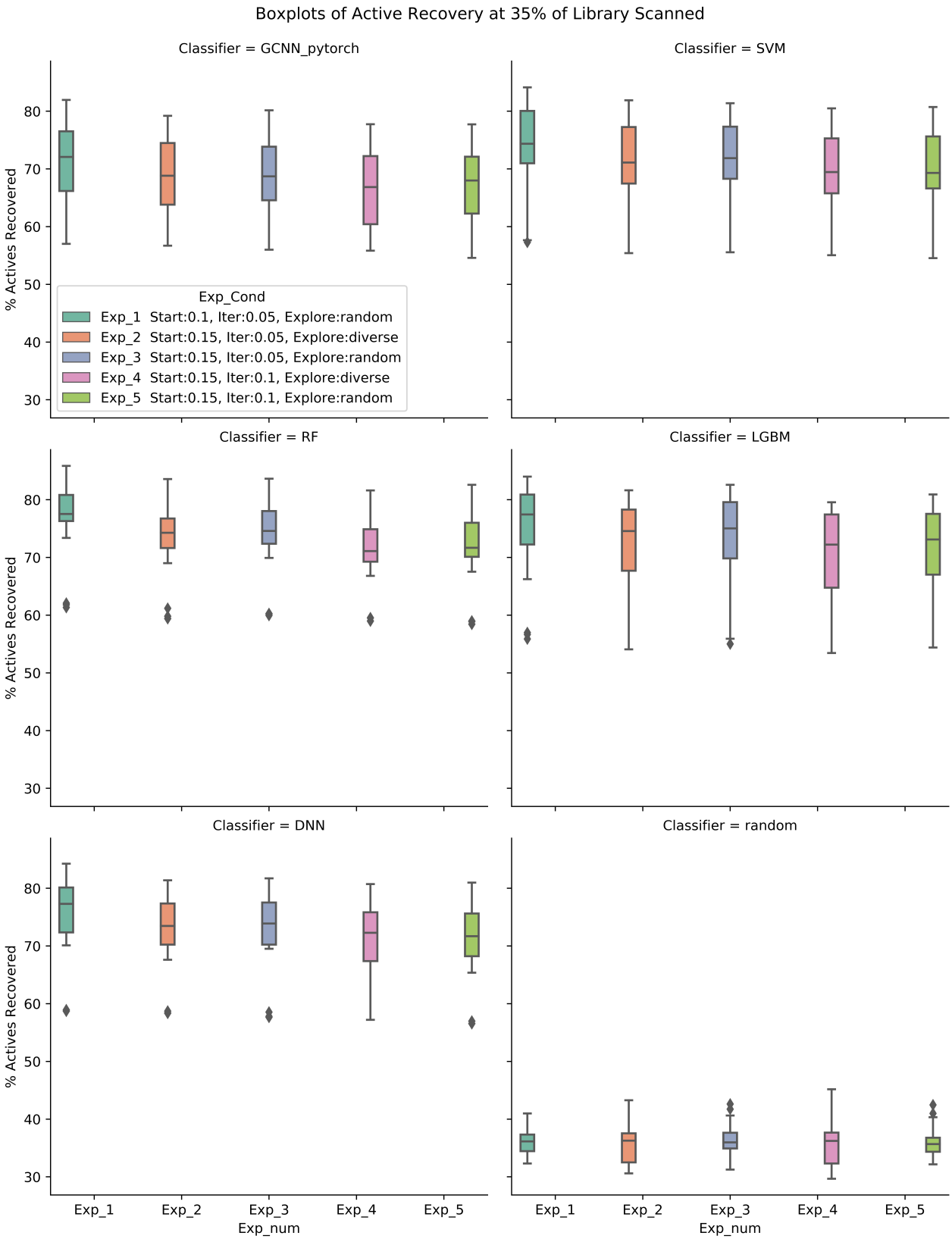

Figure 4.2: Classifier performance for multiple strategies. RF in Exp_1 has the highest median active recovery percentage and also the tightest interquartile range. The worst performance of any classifier under any experimental condition still nearly doubles that which would be expected with current HTS approaches (shown by the random classifier).

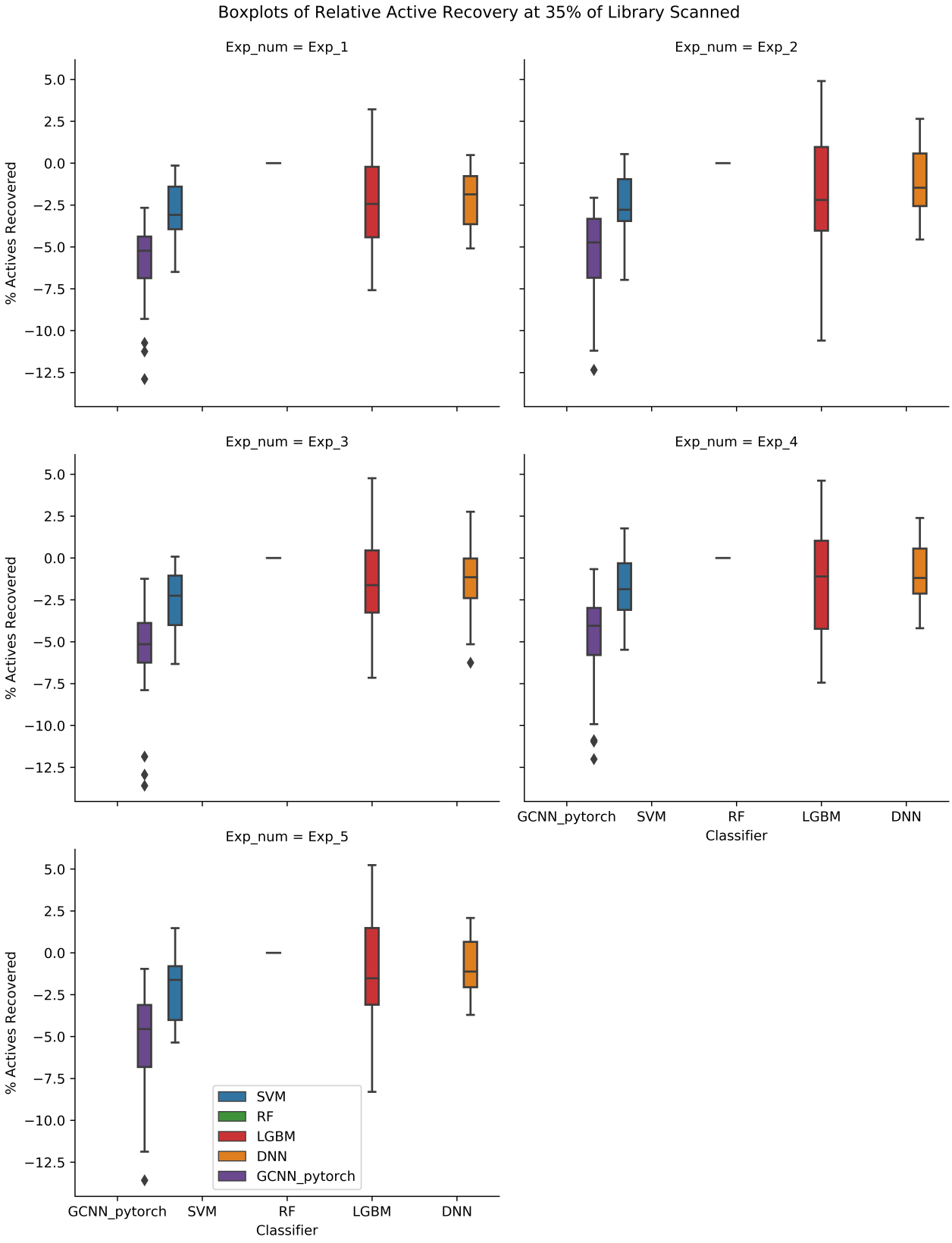

Figure 4.3: Classifier performance relative to RF at 35% of library scanned. Median performance is lower than that of RF for all other classifiers, confirming that RF produces the best performance.

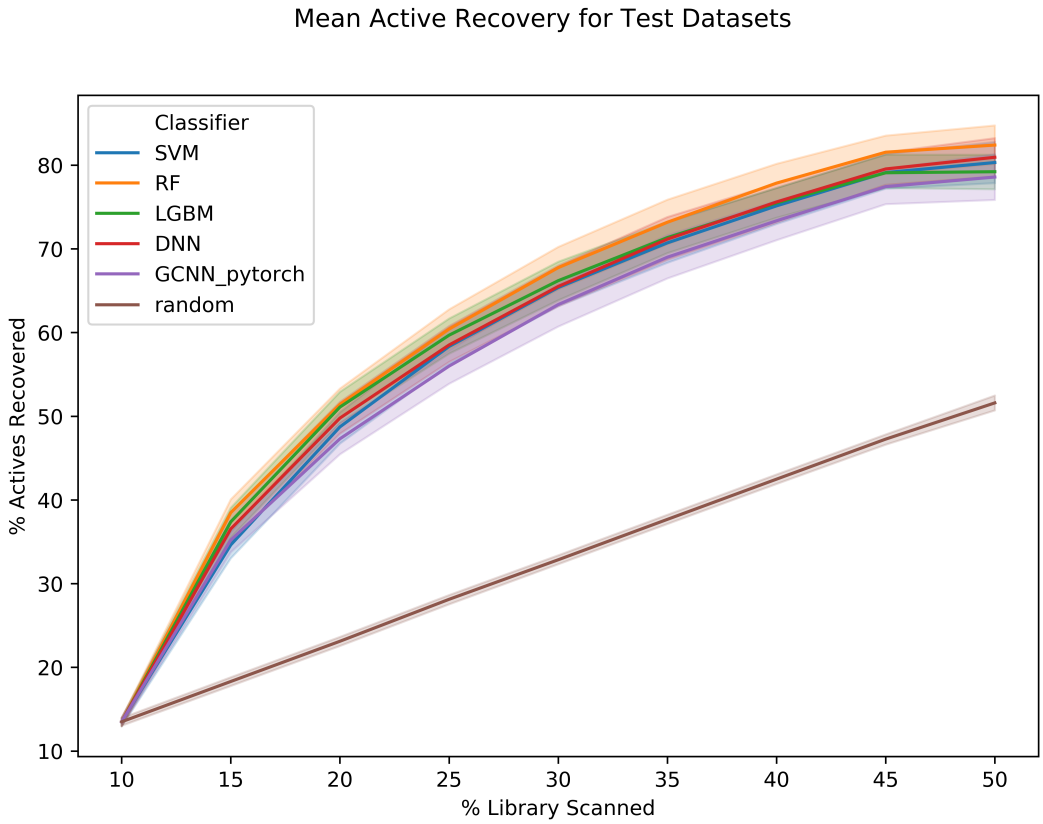

Figure 4.4: Classifier performance for new test datasets with Exp_1 experimental setup. RF again shows best mean performance at 35% and 50% of the library scanned (71% and 81% actives recovered respectively). Trends in classifier performance hold, with GCNNs under-performing relative to other classifiers. Bands represent 68% CIs.

Chapter 5

General Conclusions

So, what can be concluded from this research? First, I present a brief recap. To develop a new drug, one must find compounds that are active against a disease target. The current experimental approach is to take a large library of compounds and scan them against a biological model of the disease. While highly automated, this HTS process is an inherently inefficient way to identify the ~5% of the library that is active against the disease and makes no use of the data it produces as the campaign progresses. I sought to identify a Machine Learning classifier and an Active Learning strategy to reduce the number of molecules screened subject to logistical constraints imposed by the nature of bio-chemical experimental techniques.

To do so, I first benchmarked the classifiers on representative amounts of data meant to roughly represent the beginnings and ends of an HTS campaign (Sec 3.4). I used these experiments to establish how I would represent molecules (a combination of Molecular Fingerprints and Molecular Descriptors) and how I would handle the extreme class imbalance inherent to the problem (Random Over Sampling). In Sec 3.4, I describe the development of an initial Active Learning strategy which merged ε-greedy strategy with uncertainty sampling [58] and refined it to a ranked ε-greedy similar to that used in [10]. With this initial strategy in-place, I tuned the hyperparameters of the Non-Neural Network based classifiers to maximize Area Under the Precision Recall Curve in Sec 4.1. The Graph Convolutional Neural Network was roughly hand adjusted to prevent over-fitting. Finally, as discussed in Sec 4.2, I explored the metaparameters of the Active Learning strategy itself.

In the following paragraph, I summarize and integrate conclusions from each section above. The findings in Sec 3.3 confirmed that giving classifiers more data (the combination of Molecular Fingerprints and Molecular Descriptors) improved their performance. Additionally, for smaller splits of the data, removing data (under-sampling) worsened the performance and over-sampling had a slight benefit in some cases. More interestingly, with large amounts of data, several classifiers performed better with under-sampling than with over-sampling (see Fig A.3), the implications of which are discussed below. Sec 3.4 demonstrated the futility of handcrafting complicated selection strategies, especially the use of thresholding when classifiers had significantly different distributions over the prediction range. The final strategy I adopted was to simply take a fixed percentage of the highest confidence predictions and then sample the remaining library to explore the space. Possible alternatives to these hand selected strategies are discussed below. Sec 4.1 explored tuning model hyperparameters. Initially I was unsure whether tuning would transfer well from one HTS campaign to another. However, results showed that tuning only on 10% splits of the datasets, and averaging the results yielded performance gains for the classifiers. This implies that more advanced tuning approaches may yield further gains. It also became clear that GCNNs were more complicated to optimize in this setting than anticipated, reasons for this are explored below.

Finally, in Sec 4.2, I performed an initial exploration of the parameters of the Active Learning strategy. The initial conclusion from this section was that, of the methods for exploring chemical space, random sampling yielded higher active recovery than diversity selection. Additionally, it became clear that increasing start size decreases the strategies overall performance. Finally, the results showed that increasing iteration size only moderately decreased classifier performance. As noted previously, using larger iterations has substantial real-world benefits in reducing labour, screening time, and chemical degradation or cell line divergence. The results of this final set of experiments suggest that RF is the best classifier for these datasets and this experimental strategy.

5.1 Future Work

5.1.1 Model Tuning

In the following section, I discuss the previously highlighted results and their implications for future work. First I consider the tuning of models. As noted, for 80% splits of datasets, under-sampling yielded increased performance relative to oversampling for several models. Additionally, hyperparameter tuning improved model performance for the scikit-learn models, while naive tuning hurt GCNN performance. The fact that limited hyperparameter tuning improved performance even when averaged across nine campaigns with varying libraries and targets suggests that further development of this method would benefit from thorough hyperparameter tuning for any classifier type. Moreover, the performance differential between over and under-sampling at different split sizes suggests that there is merit in tuning the models at each iteration step. RF's performance increase with under-sampling on large training pool sizes indicates that even simple strategies like increasing the number of estimators as the training pool grows could yield improved performance. One could also consider using warm-starts to build on the previously trained classifiers. However, these previous models would have been trained on a smaller pool of over-sampled actives and therefore might maintain a bias towards them. For GCNNs, it may be attractive to carry over the convolutional layers, as these in theory act mostly as unbiased feature extractors. The classifier layers of the networks that take the input of the convolutional head could be reset at each iteration to avoid biases.

It is somewhat surprising that the GCNNs performed poorly, as they are thought to use much better representations of chemistry (molecular graphs) and to have more representative power than other classifiers. However, the MoleculeNet benchmark [19] also found that RF perform best (ahead of GCNNs) on tasks similar to that considered in this thesis. That being said, the GCNN implementation used by MoleculeNet, DeepChem, and in this has been criticised for the way it handles edge information [35], and several recent papers claim that novel architectures and MPNN algorithms outperform it [74] [39]. In more traditional CNNs, once a suc-

cessful architecture is established, and its upper convolutional layers trained, transfer learning with high performance is possible by retraining only the bottom few layers (see the use of Alexnet and Resnet50). Recently Hu et al. published work demonstrating that pretraining their GCNN architecture significantly improved performance on molecular property prediction, and importantly that it improved generalization over 'out-of-distribution' graphs [41]. Researchers have also considered low-data learning with GCNNs. These may prove particularly powerful given the paucity of labeled actives in any HTS scenario [75] [49]. These architectures may provide a more elegant solution to the class imbalance problem, allowing for instance, an ensemble of low-data models trained on smaller balanced subsets of the labeled data. While classifiers pretrained on large chemical libraries may be able to immediately extract relevant features rather than relearning them for every new campaign. This is a rapidly growing field, and it is likely that in the next few years a consensus will develop around ideal architectures, algorithms, and training strategies. Once this has occurred it may make sense to invest the substantial computational resources required to tune any deep learning model into tuning a GCNN model for use in Iterative Screening particular problem. I had neither the resources, nor the confidence in any single GCNN model to consider this tuning. Moreover, RF offers better performance than the tested GCNN model with substantially less computational resource use, and a much more constrained hyperparameter space. Eventually, more complicated GCNNs backed by more compute may outperform them, but for the moment RFs perform well and can be applied easily by a lab without a resident deep learning expert.

5.1.2 Active Learning Strategies

Iterative Screening is complicated by the fact that model performance on a single task is not the goal. Rather one seeks to define a strategy and match a model to it in order to best discover active compounds under the constraints imposed by the physical realities of HTS. Different strategies may benefit from different models. Sec 3.4 which describes how I modified the selection strategy by hand, demonstrated the difficulty in balancing various aspects of exploration vs exploitation. In the end,

I settled on a simple ranking system. Exploring the metaparameters of this strategy in Sec 4.2 suggested that there are performance gains to be made by tuning the strategy to the problem, although I also note that end-users may wish to consider the balance between outright performance and ease of implementation (for instance in deciding iteration size). One surprising result described in this section is that intentionally sampling molecules far from currently scanned ones in chemical space did not seem to contribute to discovery of isolated actives and hindered rather than helped performance. This may be because actives are more tightly clustered than we expected, or it may mean simply that choosing points far away from a set that contains most of a library's actives is more likely to yield inactives than randomly sampling the remaining unscanned library members. Obviously, spending more time exploring the possible selection strategies and the meta-parameters of the Active Learning strategy itself may yield increased performance. While this could be done via traditional hyper-parameter tuning, there are more interesting alternatives.

In particular, reinforcement learning and meta learning offer intriguing possibilities of training a model to learn its own selection strategy. It is simple to view an HTS campaign as a long episode of actions to either predict or request a label for each compound, with a cost associated with each label request and a reward (positive or negative) for each prediction. Woodward et al. published an elegant implementation of a similar concept, training a Reinforcement Learning agent to learn when to request labels for data and then to incorporate the supplied label into a classifier for the same data all in a low data setting [76]. Training an agent to develop its own selection strategy, informed by its own knowledge about the system, may prove more adaptable than a static human designed selection strategy. Assuming that this strategy about *how* to request labels generalizes from one campaign to another, this agent could be pre-trained using the thousands of publicly available campaigns prior to being applied to a new library-target pairing of interest. In an ideal fully automated HTS environment, this agent could act essentially online, requesting labels and learning on them without the constraints of fixed batch size (limited only to plate size) and thus could yield even more substantial reductions in

screens required.

5.2 Final Conclusions

The headline conclusion is this: iterative screening works. The method developed in this paper substantially reduced the number of compounds screened in representative HTS campaigns, while still recovering the majority of active compounds. Using an RF initially trained on 10% of a library, followed by 5% iterations divided 4:1 between top ranked compounds and random exploration; it was possible, on average, to recover 79% of the actives in a library while only screening 35% of the compound library. Even in the worst case, the hit ratio was nearly double that which a traditional screening campaign would discover. This performance held not only over the datasets on which the method was developed but generalized well to novel HTS datasets with characteristics that varied from the development data. This suggests that this method can be successfully implemented in the laboratory contexts it was designed for. Given the approximate cost of £1.20 per compound tested, and typical campaigns of 1,000,000 compounds; a single application of the method stands to save a lab over £780,000. For small academic and non-profit laboratories, the implications are substantial. As this method is intended for real-world applications in laboratories that may not have substantial computational resources, it is fortuitous that RF proved to be the best performing classifier. RF requires minimal hyperparameter tuning, is easy to understand, and provides methods to assess feature importance which can contribute to further drug discovery. Moreover, it can be implemented with the typical computational hardware available in any laboratory. The nascent techniques around pre-training GCNNs [41] may trigger a similar explosion in performance as seen with the family of ImageNet CNN models. If this occurs, GCNNs are likely to begin to outperform RF in iterative screening contexts. As it stands today, however, this work proposes a powerful and practical method for dramatically reducing the cost of an HTS campaign and demonstrates its general applicability.

Appendix A

Additional Plots

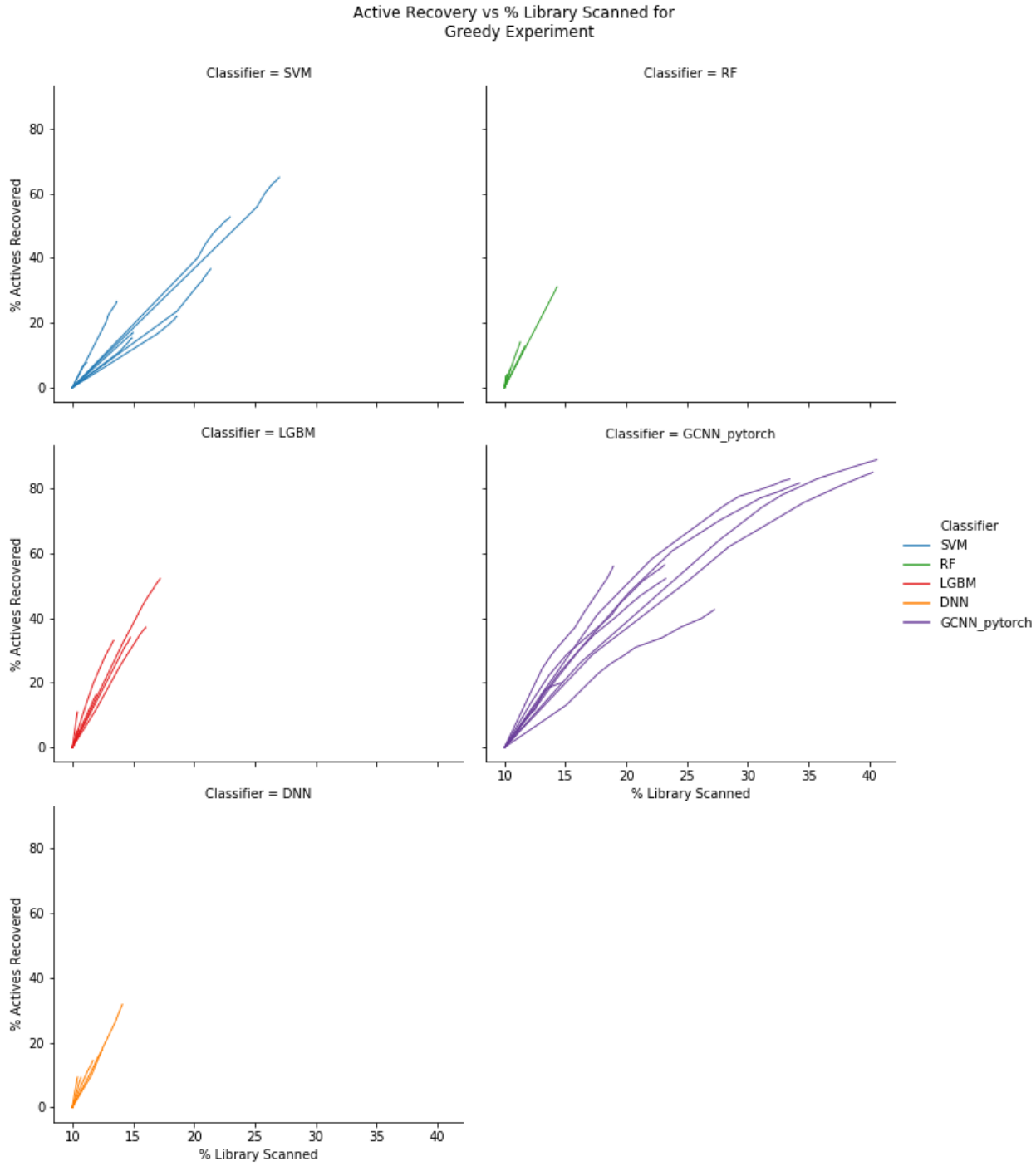

Figure A.1: Separated plots for each classifier under the greedy strategy showing a clear (and intuitive) relationship between number of labels requested, and actives found. Slopes are a rough approximation of efficiency, with RF and LGBM appearing the most efficient.

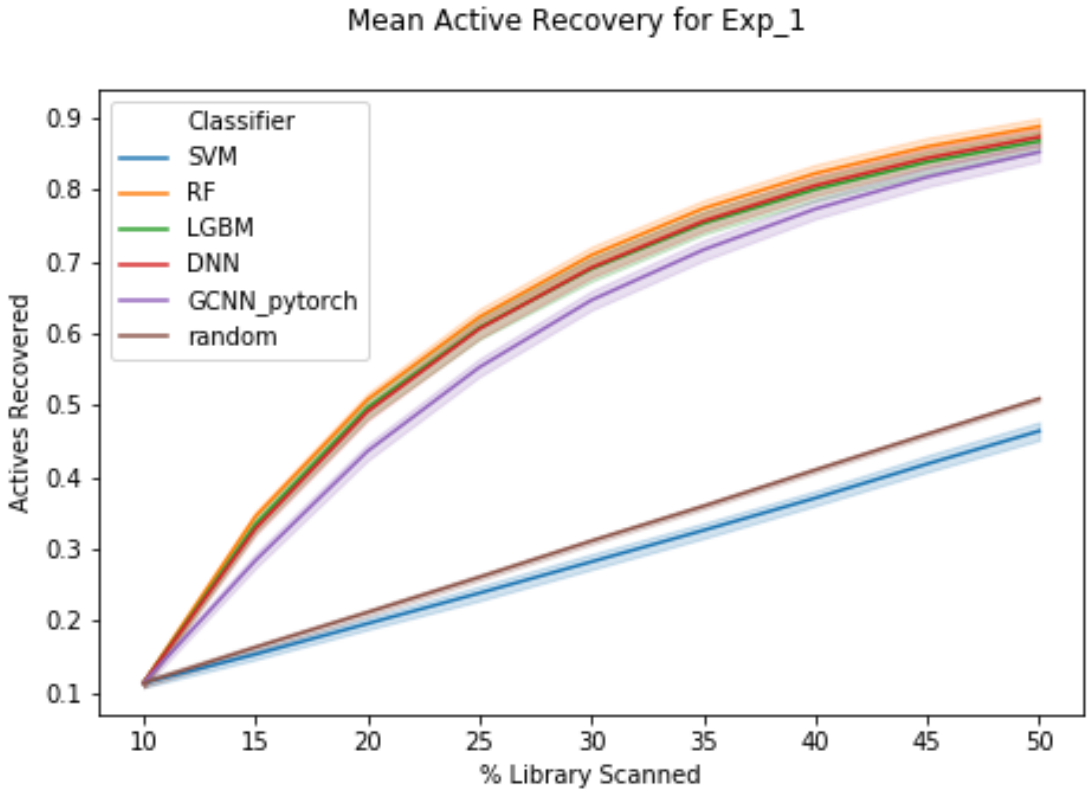

Figure A.2: Initial hyperparameters for SVM caused a substantial decrease in performance (below random). Reverting the loss from 'squared_hinge' to 'hinge' corrected this problem

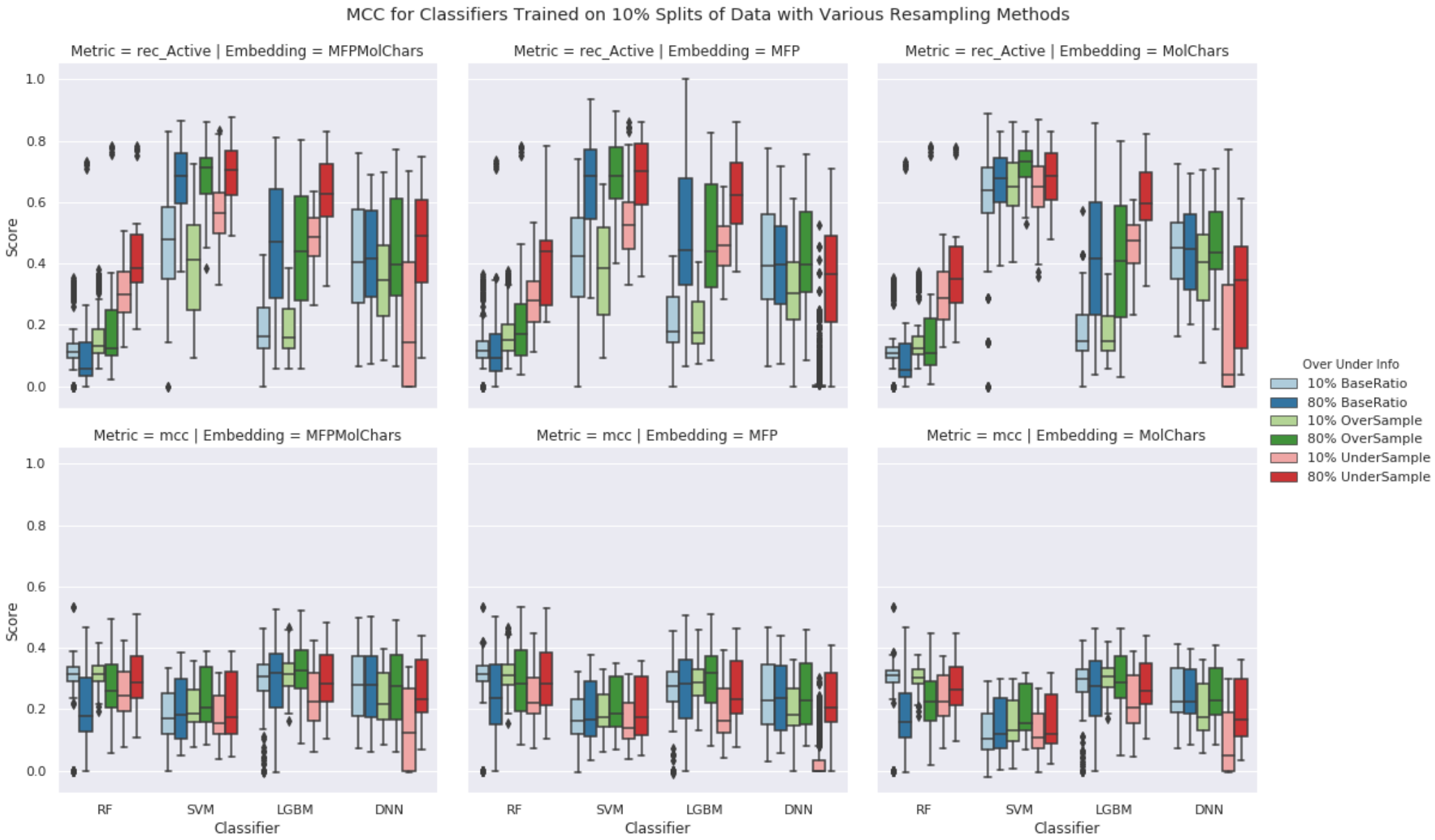

Figure A.3: Recall and MCC for initial benchmarking. Note that the left most plots show improvements in Recall and MCC for under-sampling vs over-sampling.

Figure A.4: Loss curves for each HTS dataset at each iteration. Note that test loss rarely decreases after 20 epochs in later iterations, while it often increases after 20 epochs in early iterations.

Appendix B

Hyperparameter results

The script for the hyperparameter tuning experiment is available at `https://github.com/gdreiman1/thesis/blob/master/Aug/hyper_param_tune.py`

AID	max_depth	class_weight	n_estimators	max_features	bootstrap	min_samples_split	min_samples_leaf
AID_1345083	NaN	balanced_subsample	3200	log2	False	10	5
AID_624255	10.0	balanced_subsample	3200	auto	False	10	5
AID_449739	NaN	balanced	1600	log2	False	5	4
AID_995	20.0	balanced	200	log2	True	10	1
AID_938	NaN	balanced	800	log2	True	10	1
AID_628	NaN	balanced	200	log2	True	2	5
AID_596	20.0	balanced_subsample	1600	auto	True	2	5
AID_893	NaN	balanced	800	log2	False	2	1
AID_894	NaN	balanced_subsample	400	log2	True	10	1

Table B.1: Hyperparameters with maximal AUC-PRC for RF. Note how some parameters agree for than others. For instance max_features aligns around log2, while min_sample_leaf is more variable.

AID	learning_rate	max_iter	loss	penalty	eta0	alpha	class_weight	average
AID_1345083	optimal	500	squared_hinge	l2	0.001000	0.000335	None	5
AID_624255	optimal	4048	squared_hinge	l2	0.000775	0.000436	balanced	True
AID_449739	optimal	3167	squared_hinge	elasticnet	0.000593	0.000333	None	5
AID_995	optimal	10000	squared_hinge	elasticnet	0.000010	0.000810	None	5
AID_938	optimal	7833	squared_hinge	l2	0.000172	0.000657	balanced	10
AID_628	optimal	10000	squared_hinge	elasticnet	0.000010	0.000010	None	False
AID_596	optimal	500	squared_hinge	l2	0.000010	0.000010	balanced	False
AID_893	optimal	5506	squared_hinge	l2	0.000555	0.000010	balanced	20
AID_894	optimal	1714	squared_hinge	elasticnet	0.000970	0.000729	None	True

Table B.2: Hyperparameters with maximal AUC-PRC for SVM. Note that the optimal loss is squared_hinge. Actually implementing this caused the SVM to under-perform even random selection. Reverting to hinge loss reversed this performance decline. This further highlights the difficulty of hyperparameter tuning in this context.

AID	max_depth	num_leaves	boosting_type	is_unbalance	n_estimators	learning_rate	max_bin	min_data_in_leaf
AID_1345083	15	31	dart	False	400	0.065531	183	18
AID_624255	20	18	dart	False	400	0.047602	25	10
AID_449739	-1	45	gbdt	True	3200	0.341069	144	33
AID_995	-1	35	dart	True	1600	0.121826	83	48
AID_938	10	44	dart	False	800	0.408183	25	10
AID_628	5	5	dart	False	3200	0.010000	500	10
AID_596	-1	50	gbdt	False	400	0.183745	25	50
AID_893	10	50	dart	False	800	0.500000	500	35
AID_894	15	50	gbdt	True	3200	0.067970	500	50

Table B.3: Hyperparameters with maximal AUC-PRC for LGBM. LGBM showed the least consistently tuned hyperparameters.

Appendix C

Model and Experimental Details

C.1 Initial Classifier Evaluation

Details of descriptor generation are available in https://github.com/gdreiman1/thesis/blob/master/test_demo/Data_ProcessandAlign.py For Classifier Evaluation experiments in Sec. 3.3, the following model settings were used. The script containing the full experiment is available at https://github.com/gdreiman1/thesis/blob/master/git_June_10/Comet_Fold_Crick_onebigdf.py. They are reproduced here for convenience. SVM:

```
from sklearn.linear_model import SGDClassifier
SGDClassifier(loss='hinge', penalty='l2',
alpha=0.0001, l1_ratio=0.15, fit_intercept=True,
max_iter=500000,tol=0.001, shuffle=True, verbose=0,
epsilon=0.1, n_jobs=-1, random_state=None,
learning_rate='optimal', eta0=0.0, power_t=0.5,
early_stopping=False,validation_fraction=0.1,
n_iter_no_change=5, class_weight='balanced',
warm_start=False, average=False)
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
RandomForestClassifier(n_estimators=100, random_state=2562,
```

```
    class_weight="balanced_subsample", n_jobs = -1)
```

LGBM

```
import lightgbm as lgb
lgb.LGBMClassifier(boosting_type='gbdt', num_leaves=31,
max_depth=-1, learning_rate=0.1, n_estimators=500,
subsample_for_bin=200000, objective='binary',
is_unbalance=True, min_split_gain=0.0,
min_child_weight=0.001, min_child_samples=20,
subsample=1.0, subsample_freq=0, colsample_bytree=1.0,
reg_alpha=0.0, reg_lambda=0.0, random_state=None,
n_jobs=-1, silent=True, importance_type='split')
```

DNN:

5 Dense layers

Units: [512,128,64,16,2]

Activation: [Sigmoid,ReLu,ReLu,Relu,SoftMax]

Loss: 'Categorical Crossentropy'

Optimizer: adam

Batch-size: 500

Epochs: 5

Class_weight: `balanced` with

`sklearn.utils.class_weight.compute_class_weight()`

```
import tensorflow as tf


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
class_weights =
class_weight.compute_class_weight('balanced',
np.unique(y_train), y_train)
  fast_NN = Sequential(name = 'quick')
```

```
fast_NN.add(Dense(512, activation = 'sigmoid',
name = 'input'))
fast_NN.add(Dense(128, activation = 'relu',
name = 'first',bias_initializer =
tf.keras.initializers.Constant(value=0.1)))
fast_NN.add(Dense(64, activation = 'relu',
name = 'second',bias_initializer =
tf.keras.initializers.Constant(value=0.1)))
fast_NN.add(Dense(16, activation = 'relu',
name = 'third',bias_initializer =
tf.keras.initializers.Constant(value=0.1)))
fast_NN.add(Dense(num_labels,
activation = 'softmax', name = 'predict',
bias_initializer =
tf.keras.initializers.Constant(value=final_bias)))
fast_NN.compile(loss = 'categorical_crossentropy',
optimizer='adam',metrics=['categorical_accuracy',
tf.keras.metrics.Recall(),tf.keras.metrics.Precision()])
fast_NN_model = fast_NN.fit(X_train,
to_categorical(y_train),
 validation_data=(X_test,to_categorical(y_test)),
 epochs=5,
 batch_size=500,class_weight = class_weights,
 shuffle=True,
 verbose=0)
```

C.2 Initial Iterative Experiments

Model Details for the initial iterative experiments are available at `https://github.com/gdreiman1/thesis/blob/master/git_july/Iterative_help_funcs.py`.

Section	Description	File Name
3.4.2.1	initial selection strategy	iterative_step_rand_diverse.py
3.4.2.2	updated strategy with GCNN	iterative_step_rand_diverse_gcnn.py
3.4.2.3	greedy strategy	iterative_ranked_innerrep.py
3.4.2.4	ranked ε-greedy	iterative_greedy.py

Table C.1: Links for experiments from initial iterative screening section. Files provided in tables should be can be found in `https://github.com/gdreiman1/thesis/tree/master/git_july`

The following table provides links to representative scripts for each experiment:

C.3 Tuned Iterative Experiments

Model Details for the iterative experiments after hyperparameter tuning are available at `https://github.com/gdreiman1/thesis/blob/master/git_july/Iterative_help_funcs_tuned.py`.

The following table provides links to representative scripts for each experiment:

Name	Start Size	Iter Num	Exploration	File Name
Exp_1	10%	5%	Random	iterative_ranked_innerrep_tuned1.py
Exp_2	15%	5%	Diverse	iterative_ranked_innerrep_tuned2.py
Exp_3	15%	5%	Random	iterative_ranked_innerrep_tuned3.py
Exp_4	15%	10%	Diverse	iterative_ranked_innerrep_tuned4.py
Exp_5	15%	10%	Random	iterative_ranked_innerrep_tuned5.py

Table C.2: Links for experiments from tuned screening section. Files provided in tables should be can be found in `https://github.com/gdreiman1/thesis/tree/master/git_july`

C.4 Novel HTS Test

The scripts needed to convert the raw HTS data for the three test sets into embeddings and then run the test example are located in `https://github.com/gdreiman1/thesis/tree/master/test_demo` and should be run in the order specified in the table below

Order	Description	Name
1	make compound embeddings and match with labels	Process_3Test_AID_Files.py
2	adds MFP bit vectors needed for MaxMin Diversity	Adding_bit_mfp_3test.py
3	add RDKit Mol objects needed to add graph embeddings	matchMol2Active_3Test.py
4	adds molecular graphs in PyTorch tensor format	add_GraphReps_3_test.py
5	runs experiment	iterative_ranked_innerrep_tuned1_test.py

Table C.3: Order of scripts to be run to replicate final test experiments.

Appendix D

Environment

This thesis was conducted in Python 3.7, a conda environment file is availible at `https://github.com/gdreiman1/thesis/blob/master/RD_environment.yml`

Bibliography

[1] Regine S. Bohacek, Colin McMartin, and Wayne C. Guida. The art and practice of structure-based drug design: A molecular modeling perspective, 1 1996. ISSN 01986325. URL http://doi.wiley.com/10.1002/%28SICI%291098-1128%28199601%2916%3A1%3C3%3A%3AAID-MED1%3E3.0.CO%3B2-6.

[2] J P Hughes, S Rees, S B Kalindjian, and K L Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–49, 3 2011. ISSN 1476-5381. doi: 10.1111/j.1476-5381.2010.01127.x. URL http://www.ncbi.nlm.nih.gov/pubmed/21091654http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3058157.

[3] William P. Castelli, Robert J. Garrison, Peter W. F. Wilson, Robert D. Abbott, Sona Kalousdian, and William B. Kannel. Incidence of Coronary Heart Disease and Lipoprotein Cholesterol Levels. *JAMA*, 256(20):2835, 11 1986. ISSN 0098-7484. doi: 10.1001/jama.1986.03380200073024. URL http://jama.jamanetwork.com/article.aspx?doi=10.1001/jama.1986.03380200073024.

[4] Akira Endo. A historical perspective on the discovery of statins. *Proceedings of the Japan Academy Series B: Physical and Biological Sciences*, 86 (5):484–493, 2010. ISSN 03862208. doi: 10.2183/pjab.86.484. URL http://www.ncbi.nlm.nih.gov/pubmed/20467214http:

//www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3108295.

[5] Scandinavian Simvastatin Survival Study Group. Randomised trial of cholesterol lowering in 4444 patients with coronary heart disease: the Scandinavian Simvastatin Survival Study (4S). *The Lancet*, 344(8934): 1383–1389, 11 1994. ISSN 0140-6736. doi: 10.1016/S0140-6736(94)90566-5. URL https://www.sciencedirect.com/science/article/pii/S0140673694905665?via%3Dihub.

[6] Patrick D Hsu, Eric S Lander, and Feng Zhang. Development and applications of CRISPR-Cas9 for genome engineering. *Cell*, 157(6): 1262–78, 6 2014. ISSN 1097-4172. doi: 10.1016/j.cell.2014.05.010. URL http://www.ncbi.nlm.nih.gov/pubmed/24906146http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4343198.

[7] Ricardo Macarron, Martyn N. Banks, Dejan Bojanic, David J. Burns, Dragan A. Cirovic, Tina Garyantes, Darren V. S. Green, Robert P. Hertzberg, William P. Janzen, Jeff W. Paslay, Ulrich Schopfer, and G. Sitta Sittampalam. Impact of high-throughput screening in biomedical research. *Nature Reviews Drug Discovery*, 10(3):188–195, 3 2011. ISSN 1474-1776. doi: 10.1038/nrd3368. URL http://www.nature.com/articles/nrd3368.

[8] Joshua D Kangas, Armaghan W Naik, and Robert F Murphy. Efficient discovery of responses of proteins to compounds using active learning. *BMC Bioinformatics*, 15(1):143, 5 2014. ISSN 1471-2105. doi: 10.1186/1471-2105-15-143. URL http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-15-143.

[9] Yukiko Fujiwara, Yoshiko Yamashita, Tsutomu Osoda, Minoru Asogawa, Chiaki Fukushima, Masaaki Asao, Hideshi Shimadzu, Kazuya Nakao, and Ryo Shimizu. Virtual Screening System for Finding Structurally Diverse

Hits by Active Learning. *Journal of Chemical Information and Modeling*, 48(4):930–940, 4 2008. ISSN 15499596. doi: 10.1021/ci700085q. URL https://pubs.acs.org/doi/10.1021/ci700085qhttps://pubs.acs.org/sharingguidelines.

[10] Manfred K. Warmuth, Jun Liao, Gunnar Rätsch, Michael Mathieson, Santosh Putta, and Christian Lemmen. Active Learning with Support Vector Machines in the Drug Discovery Process. *Journal of Chemical Information and Computer Sciences*, 43(2):667–673, 3 2003. ISSN 0095-2338. doi: 10.1021/ci025620t. URL https://pubs.acs.org/doi/full/10.1021/ci025620thttps://pubs.acs.org/doi/10.1021/ci025620t.

[11] Liwang Cui and Xin-zhuan Su. Discovery, mechanisms of action and combination therapy of artemisinin. *Expert review of anti-infective therapy*, 7(8):999–1013, 10 2009. ISSN 1744-8336. doi: 10.1586/eri.09.68. URL http://www.ncbi.nlm.nih.gov/pubmed/19803708http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2778258.

[12] D A Pereira and J A Williams. Origin and evolution of high throughput screening. *British journal of pharmacology*, 152(1):53–61, 9 2007. ISSN 0007-1188. doi: 10.1038/sj.bjp.0707373. URL http://www.ncbi.nlm.nih.gov/pubmed/17603542http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1978279.

[13] Pawe Szymański, Magdalena Markowicz, and Elbieta Mikiciuk-Olasik. Adaptation of high-throughput screening in drug discovery-toxicological screening tests. *International journal of molecular sciences*, 13(1):427–52, 2012. ISSN 1422-0067. doi: 10.3390/ijms13010427. URL http://www.ncbi.nlm.nih.gov/pubmed/22312262http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3269696.

[14] Corwin Hansch and Toshio Fujita. ρ-σ-π Analysis. A Method for the Correlation of Biological Activity and Chemical Structure. Technical Report 8, 1964. URL https://pubs.acs.org/sharingguidelines.

[15] Yu Chen Lo, Stefano E. Rensi, Wen Torng, and Russ B. Altman. Machine learning in chemoinformatics and drug discovery, 8 2018. ISSN 18785832. URL https://www.sciencedirect.com/science/article/pii/S1359644617304695.

[16] Benjah-bmm27. Aspirin-skeletal, 2007. URL https://commons.wikimedia.org/wiki/File:Aspirin-skeletal.png.

[17] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010. ISSN 15499596. doi: 10.1021/ci100050t. URL https://pubs.acs.org/doi/10.1021/ci100050t.

[18] * Joseph L. Durant, Burton A. Leland, Douglas R. Henry, , and James G. Nourse. Reoptimization of MDL Keys for Use in Drug Discovery. 2002. doi: 10.1021/CI010132R. URL https://pubs.acs.org/doi/10.1021/ci010132r.

[19] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 3 2018. ISSN 20416539. doi: 10.1039/c7sc02664a. URL http://arxiv.org/abs/1703.00564.

[20] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 9 1995. ISSN 0885-6125. doi: 10.1007/BF00994018. URL http://link.springer.com/10.1007/BF00994018https://doi.org/10.1007/BF00994018.

[21] Christopher M Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition,

2007. ISBN 0387310738. URL http://www.amazon.com/Pattern-Recognition-Learning-Information-Statistics/dp/0387310738%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0387310738.

[22] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[23] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[24] Vladimir Svetnik, Andy Liaw, Christopher Tong, J. Christopher Culberson, Robert P. Sheridan, and Bradley P. Feuston. Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling. *Journal of Chemical Information and Computer Sciences*, 43(6):1947–1958, 2003. ISSN 00952338. doi: 10.1021/ci034160g. URL https://pubs.acs.org/doi/10.1021/ci034160g.

[25] Yoav Freund and Robert E Schapire. Experiments with a New Boosting Algorithm. Technical report, 1996. URL http://www.research.att.com/.

[26] Leo Breiman. ARCING CLASSIFIERS 1. Technical Report 3, 1998. URL https://projecteuclid.org/download/pdf_1/euclid.aos/1024691079.

[27] Jerome H Friedman. Greedy Function Approximation: A Gradient Boosting Machine. Technical Report 5, 2001. URL https://www.jstor.org/stable/pdf/2699986.pdf?refreqid=excelsior%3A8f7e2458cc0575b1609f74fe38ec6a14.

[28] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Technical report. URL https://github.com/Microsoft/LightGBM.

[29] F Rosenblatt. THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN. *Psychological Review*, 65(6):1–23, 1958. ISSN 1939-1471(Electronic);0033-295X(Print). doi: 10.1098/rspb.1976.0087. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.335.3398&rep=rep1&type=pdfpapers://d471b97a-e92c-44c2-8562-4efc271c8c1b/Paper/p322.

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015. ISSN 0028-0836. doi: 10.1038/nature14539. URL http://www.nature.com/articles/nature14539.

[31] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 10 1986. ISSN 0028-0836. doi: 10.1038/323533a0. URL http://www.nature.com/articles/323533a0.

[32] Paul J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1 1988. ISSN 0893-6080. doi: 10.1016/0893-6080(88)90007-X. URL https://www.sciencedirect.com/science/article/pii/089360808890007X.

[33] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. 3 2016. URL http://arxiv.org/abs/1603.07285.

[34] David K. Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alan Aln Aspuru-Guzik, Ryan P. Adams, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aln Aspuru-Guzik, and Ryan P. Adams. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2224–2232, Cambridge, MA,

USA, 2015. MIT Press. URL https://papers.nips.cc/paper/5954-convolutional-networks-on-graphs-for-learning-molecular-fin //dl.acm.org/citation.cfm?id=2969442.2969488.

[35] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. 4 2017. URL http://arxiv.org/abs/1704.01212.

[36] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. 9 2016. URL http://arxiv.org/abs/1609.02907.

[37] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608, 2016. ISSN 15734951. doi: 10.1007/s10822-016-9938-8. URL https://arxiv.org/pdf/1603.00856.pdf.

[38] Ke Liu, Xiangyan Sun, Lei Jia, Jun Ma, Haoming Xing, Junqiu Wu, Hua Gao, Yax Sun, Florian Boulnois, and Jie Fan. Chemi-Net: A Molecular Graph Convolutional Network for Accurate Drug Property Prediction. *International Journal of Molecular Sciences*, 20(14):3389, 2019. doi: 10.3390/ijms20143389. URL https://arxiv.org/ftp/arxiv/papers/1803/1803.06236.pdf.

[39] Filippo Maria Bianchi, Daniele Grattarola, Cesare Alippi, and Lorenzo Livi. Graph Neural Networks with convolutional ARMA filters. 2019. URL https://arxiv.org/pdf/1901.01343.pdfhttp://arxiv.org/abs/1901.01343.

[40] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-Ichi Kawarabayashi, and Stefanie Jegelka. Representation Learning on Graphs with Jumping Knowledge Networks. Technical report, 2018. URL https://arxiv.org/pdf/1806.03536.pdf.

[41] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Pre-training Graph Neural Networks. 2019. URL https://arxiv.org/pdf/1905.12265.pdfhttp://arxiv.org/abs/1905.12265.

[42] Junying Li, Deng Cai, and Xiaofei He. Learning Graph-Level Representation for Drug Discovery, 9 2017. URL http://arxiv.org/abs/1709.03741https://arxiv.org/pdf/1709.03741.pdf.

[43] iwatobipen. gcn. \url{https://github.com/iwatobipen/playground/blob/master/gcn.ipynb}, 4 2019.

[44] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. 3 2019. URL http://arxiv.org/abs/1903.02428.

[45] Scott Doniger, Thomas Hofmann, and Joanne Yeh. Predicting CNS permeability of drug molecules: Comparison of neural network and support vector machine algorithms. *Journal of Computational Biology*, 9(6):849–864, 2002. ISSN 10665277. doi: 10.1089/10665270260518317. URL www.liebertpub.com.

[46] George E. Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task Neural Networks for QSAR Predictions. 6 2014. URL http://arxiv.org/abs/1406.1231.

[47] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively Multitask Networks for Drug Discovery. 2 2015. URL https://tripod.http://arxiv.org/abs/1502.02072.

[48] Steven Kearnes, Li Li, and Patrick Riley. Decoding Molecular Graph Embeddings with Reinforcement Learning. Technical report, 2019. URL https://graphreason.github.io/papers/6.pdfhttps://arxiv.org/pdf/1904.08915.pdf.

[49] Han Altae-Tran, Bharath Ramsundar, Aneesh S. Pappu, and Vijay Pande. Low Data Drug Discovery with One-Shot Learning. *ACS Central Science*, 3(4): 283–293, 4 2017. ISSN 2374-7943. doi: 10.1021/acscentsci.6b00367. URL http://pubs.acs.org/doi/10.1021/acscentsci.6b00367.

[50] Daniel Reker and Gisbert Schneider. Active-learning strategies in computer-assisted drug discovery. *Drug Discovery Today*, 20(4):458–465, 4 2015. ISSN 13596446. doi: 10.1016/j.drudis.2014.12.004. URL https://www.sciencedirect.com/science/article/pii/S1359644614004735#tbl0005https://linkinghub.elsevier.com/retrieve/pii/S1359644614004735.

[51] Christin Rakers, Daniel Reker, and J.B. Brown. Small Random Forest Models for Effective Chemogenomic Active Learning. *Journal of Computer Aided Chemistry*, 18(0):124–142, 2017. doi: 10.2751/jcac.18.124.

[52] Daniel Reker, Petra Schneider, Gisbert Schneider, and JB Brown. Active learning for computational chemogenomics. *Future Medicinal Chemistry*, 9(4):381–402, 3 2017. ISSN 1756-8919. doi: 10.4155/fmc-2016-0197. URL http://www.future-science.com/doi/10.4155/fmc-2016-0197.

[53] Andrew Rusinko, Stanley Young, David Drewry, and Sam Gerritz. Optimization of Focused Chemical Libraries Using Recursive Partitioning. *Combinatorial Chemistry & High Throughput Screening*, 5 (2), 2002. ISSN 13862073. doi: 10.2174/1386207024607383. URL http://www.ingentaconnect.com/content/ben/cchts/2002/00000005/00000002/art00004.

[54] Peter S. Kutchukian, Lee Warren, Brian C. Magliaro, Adam Amoss, Jason A. Cassaday, Gregory ODonnell, Brian Squadroni, Paul Zuck, Danette Pascarella, J. Chris Culberson, Andrew J. Cooke, Danielle Hurzy, Kelly-Ann Sondra Schlegel, Fiona Thomson, Eric N. Johnson, Victor N. Uebele,

Jeffrey D. Hermes, Sophie Parmentier-Batteur, and Michael Finley. Iterative Focused Screening with Biological Fingerprints Identifies Selective Asc-1 Inhibitors Distinct from Traditional High Throughput Screening. *ACS Chemical Biology*, 12(2):519–527, 2 2017. ISSN 1554-8929. doi: 10.1021/acschembio.6b00913. URL https://pubs.acs.org/doi/10.1021/acschembio.6b00913.

[55] Fredrik Svensson, Ulf Norinder, and Andreas Bender. Improving Screening Efficiency through Iterative Screening Using Docking and Conformal Prediction. *Journal of Chemical Information and Modeling*, 57(3):439–444, 3 2017. ISSN 1549-9596. doi: 10.1021/acs.jcim.6b00532. URL http://pubs.acs.org/doi/10.1021/acs.jcim.6b00532.

[56] Fredrik Svensson, Avid M. Afzal, Ulf Norinder, and Andreas Bender. Maximizing gain in high-throughput screening using conformal prediction. *Journal of Cheminformatics*, 10(1):7, 12 2018. ISSN 1758-2946. doi: 10.1186/s13321-018-0260-4. URL https://jcheminf.biomedcentral.com/articles/10.1186/s13321-018-0260-4.

[57] Shardul Paricharak, Adriaan P. IJzerman, Jeremy L. Jenkins, Andreas Bender, and Florian Nigsch. Data-Driven Derivation of an Informer Compound Set for Improved Selection of Active Compounds in High-Throughput Screening. *Journal of Chemical Information and Modeling*, 56(9):1622–1630, 9 2016. ISSN 1549-9596. doi: 10.1021/acs.jcim.6b00244. URL http://pubs.acs.org/doi/10.1021/acs.jcim.6b00244.

[58] Mateusz Maciejewski, Anne Mai Wassermann, Meir Glick, and Eugen Lounkine. Experimental Design Strategy: Weak Reinforcement Leads to Increased Hit Rates and Enhanced Chemical Diversity. 2015. doi: 10.1021/acs.jcim.5b00054.

[59] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbal-

anced datasets. *PloS one*, 10(3):e0118432, 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0118432. URL http://www.ncbi.nlm.nih.gov/pubmed/25738806http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4349800.

[60] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449, 11 2002. ISSN 15714128. doi: 10.3233/IDA-2002-6504. URL http://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/IDA-2002-6504.

[61] Chao Chen, Andy Liaw, and Leo Breiman. Using Random Forest to Learn Imbalanced Data. *Discovery*, (1999):1–12, 2004. URL https://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf.

[62] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. Technical report, 2002. URL https://arxiv.org/pdf/1106.1813.pdf.

[63] Ming Hao, Yanli Wang, and Stephen H Bryant. An efficient algorithm coupled with synthetic minority over-sampling technique to classify imbalanced PubChem BioAssay data. *Analytica chimica acta*, 806:117–27, 1 2014. ISSN 1873-4324. doi: 10.1016/j.aca.2013.10.050. URL http://www.ncbi.nlm.nih.gov/pubmed/24331047http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3884825.

[64] Justin M. Johnson and Taghi M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27, 12 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0192-5. URL https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0192-5.

[65] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian

He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton. PubChem 2019 update: improved access to chemical data. *Nucleic Acids Research*, 47(D1): D1102–D1109, 1 2019. ISSN 0305-1048. doi: 10.1093/nar/gky1033. URL http://www.ncbi.nlm.nih.gov/pubmed/30371825http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC6324075https://academic.oup.com/nar/article/47/D1/D1102/5146201.

[66] Christian Kramer, Tuomo Kalliokoski, Peter Gedeck, and Anna Vulpetti. The experimental uncertainty of heterogeneous public K i data. *Journal of Medicinal Chemistry*, 55(11):5165–5173, 2012. ISSN 00222623. doi: 10.1021/jm300131x. URL https://pubs.acs.org/sharingguidelines.

[67] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. Technical report. URL https://github.com/facebookresearch/Detectron.

[68] Qingliang Li, Yanli Wang, and Stephen H. Bryant. A novel method for mining highly imbalanced high-throughput screening data in PubChem. *Bioinformatics*, 25(24):3310–3316, 12 2009. ISSN 13674803. doi: 10.1093/bioinformatics/btp589. URL http://www.ncbi.nlm.nih.gov/pubmed/19825798http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2788930.

[69] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. URL http://jmlr.org/papers/v18/16-365.html.

[70] Isidro Cortés-Ciriano, Nicholas C. Firth, Andreas Bender, and Oliver Watson. Discovering Highly Potent Molecules from an Initial Set of Inactives Using

Iterative Screening. *Journal of Chemical Information and Modeling*, 58(9): 2000–2014, 9 2018. ISSN 1549-9596. doi: 10.1021/acs.jcim.8b00376. URL http://pubs.acs.org/doi/10.1021/acs.jcim.8b00376.

[71] Greg Landrum and others. RDKit: Open-source cheminformatics, 2006.

[72] Mark Ashton, John Barnard, Florence Casset, Michael Charlton, Geoffrey Downs, Dominique Gorse, John Holliday, Roger Lahana, and Peter Willett. Identification of diverse database subsets using property-based and fragment-based molecular descriptions. *Quantitative Structure-Activity Relationships*, 21(6):598–604, 12 2002. ISSN 09318771. doi: 10.1002/qsar.200290002. URL http://doi.wiley.com/10.1002/qsar.200290002.

[73] Tim Head, MechCoder, Gilles Louppe, Iaroslav Shcherbatyi, fcharras, Z Vinícius, cmmalone, Christopher Schröder, nel215, Nuno Campos, Todd Young, Stefano Cereda, Thomas Fan, rene-rex, Kejia (KJ) Shi, Justus Schwabedal, carlosdanielcsantos, Hvass-Labs, Mikhail Pak, SoManyUsernamesTaken, Fred Callaway, Loc Estève, Lilian Besson, Mehdi Cherti, Karlson Pfannschmidt, Fabian Linzberger, Christophe Cauet, Anna Gut, Andreas Mueller, and Alexander Fabisch. scikit-optimize/scikit-optimize: v0.5.2. 3 2018. doi: 10.5281/ZENODO.1207017. URL https://doi.org/10.5281/zenodo.1207017#.XXUKGeIHMK4.mendeley.

[74] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? Technical report, 2018. URL http://arxiv.org/abs/1810.00826.

[75] Victor Garcia and Joan Bruna. Few-Shot Learning with Graph Neural Networks. 2017. URL https://arxiv.org/pdf/1711.04043.pdfhttp://arxiv.org/abs/1711.04043.

[76] Mark Woodward, Chelsea Finn, and Berkeley Ai Research. Active One-shot Learning. Technical report, 2017. URL https://arxiv.org/pdf/1702.06559.pdf.