Semi-Supervised Learning for Predicting Biochemical Properties

Travis Persson
Abstract

Semi-Supervised Learning for Predicting Biochemical Properties

Travis Persson

The predictive performance of supervised learning methods relies on large amounts of labeled data. Data sets used in Quantitative Structure Activity Relationship modeling often contain a limited amount of labeled data, while unlabeled data is abundant. Semi-supervised learning can improve the performance of supervised methods by incorporating a larger set of unlabeled samples with fewer labeled instances.

A semi-supervised learning method known as Label Spreading was compared to a Random Forest in its effectiveness for correctly classifying the binding properties of molecules on ten different sets of compounds. Label Spreading using a k-Nearest Neighbors (LS-KNN) kernel was found to, on average, outperform the Random Forest. Using a randomly sampled labeled data set of sizes 50 and 100, LS-KNN achieved a mean accuracy of 4.03% and 1.97% higher than that of the Random Forest. The outcome was similar for the mean area under the Receiver Operating Characteristic curve (AUC). For large sets of labeled data, the performances between the methods were indistinguishable. It was also found that sampling labeled data from generated clusters using a k-Means clustering algorithm, as opposed to random sampling, increased the performance of all applied methods. For a labeled data set of size 50, Label Spreading using a Radial Basis Function kernel increased its mean accuracy and AUC by 7.52% and 3.08%, respectively, when sampling from clusters.

In conclusion, semi-supervised learning could be beneficial when applied to similar modeling scenarios. However, the improvements heavily depend on the underlying data, suggesting that there is no one-size-fits-all method.
Acknowledgements

I want to express my deepest gratitude to the thesis supervisors Dr. Andreina Fransisco and Dr. Jonathan Alvarsson, and the subject reviewer Olle Gällmo for their immense dedication towards my thesis. Everything would have been considerably less manageable without their continuous support. I would also like to thank my family and friends, especially my partner, for encouraging me and cheering me on through all the highs and lows.
Contents

1 Introduction 7

2 Background: Pharmaceutical Bioinformatics 7
   2.1 Describing Molecules ............................. 7
   2.2 QSAR ........................................ 8

3 Background: Machine Learning 8
   3.1 Supervised Learning ............................. 9
   3.2 Unsupervised Learning ........................... 9
   3.3 Semi-Supervised Learning ................. 10
      3.3.1 The Low-Density Assumption 11
      3.3.2 The Manifold Assumption 12
      3.3.3 The Cluster Assumption 12
      3.3.4 Inductive Methods 12
      3.3.5 Self-Training 13
      3.3.6 Transductive Methods 13

4 Theory 13
   4.1 The Expected Performance of a Model .......... 13
      4.1.1 The Bias-Variance Tradeoff 14
      4.1.2 Evaluating Models 14
      4.1.3 The Confusion Matrix 15
   4.2 Random Forest ................................. 16
   4.3 k-Means Clustering ............................... 17
   4.4 Graph-Based Transductive Methods ......... 17
      4.4.1 Constructing the Graph 18
      4.4.2 Label Inference 18
      4.4.3 A Label Propagation Method: Label Spreading 19

5 Method 19
   5.1 Experimental Setup .............................. 19
      5.1.1 Input Features 21
      5.1.2 Sampling the Labeled Data Sets 22
   5.2 Model Evaluation ................................. 22
   5.3 Label Spreading .................................. 23
      5.3.1 Tuning Hyperparameters 23
      5.3.2 Implementing the Label Spreading Method 24
   5.4 Implementing the Random Forest Model .... 25

6 Results 25
   6.1 Label Spreading: Scenario 1 .................. 26
      6.1.1 Hyperparameters 26
      6.1.2 Performance: Transductive Setting 27
      6.1.3 Performance: Inductive Setting 28
1 Introduction

Quantitative Structure Activity Relationships (QSAR) [23] is a procedure often used in ligand-based predictive modeling. In ligand-based modeling, the objective is to generate predictive models that can assist in deciding the binding activity of chemical compounds to a biological target (ligand/binding molecule). Machine learning (ML) plays an integral part in QSAR modeling, and algorithms such as Random Forests and Support Vector Machines have been shown to be efficient in finding correlations between chemical structures and their biochemical activities and properties [3, 19].

The ligand-binding activities of chemical compounds are often obtained through laboratory assays conducted by researchers or robotized laboratories. Conducting these experiments can be expensive and time-consuming, making the amount of labeled data available in QSAR modeling experiments quite limited. At the same time, unlabeled data is often abundantly available and easily accessible through large databases containing chemical compounds [12]. Having access to only a limited amount of labeled data can be a problem since supervised ML algorithms rely heavily on large amounts of labeled data to improve their predictive abilities [23, 13]. This makes QSAR modeling an attractive domain for the application of semi-supervised learning (SSL). SSL incorporates unlabeled data in the training process to either infer their labels or pseudo-label them [11].

This thesis makes the following contributions:

- An evaluation of the effectiveness for correctly classifying the binding activities of molecules using the Label Spreading method in a transductive and inductive setting.
- A comparison between the use of a randomly sampled labeled data set versus a sampling-from-clusters approach.
- A comparison between the performance of a Random Forest model and a Label Spreading method when the availability of labeled data is scarce.

2 Background: Pharmaceutical Bioinformatics

This section provides the necessary background of using different methods for describing molecules mathematically, and ends with a brief summary of how machine learning can be applied to the field of pharmaceutical bioinformatics.

2.1 Describing Molecules

The chemical structures of molecules can be used to predict their biochemical properties. Making accurate predictions of molecular properties is akin to how well the molecules are described. A way of describing the structures of molecules is by using molecular descriptors. Mathematical representations of molecule
structures are constructed with molecular descriptors, making them usable as input features for machine learning models. There are many different types of descriptors, and their usage depends on the type of property or activity that is being predicted.

Structural keys and molecular fingerprints can also be used to describe molecules and assess the similarity of molecules. They are both represented as bit vectors. For structural keys, each bit corresponds to whether or not (1 or 0 respectively) a particular pre-defined substructure is present in a molecule. For fingerprints, however, the bits in the bit vectors are not pre-defined, and each bit might represent a set of multiple molecular features. When calculating the similarity between bit vectors, the Tanimoto similarity equation is most often used, defined as:

$$\text{Tanimoto}(A, B) = \frac{A \cap B}{A \cup B},$$

where $A \cap B$ denotes the number of 1s in both bit vectors, and $A \cup B$ the amount of 1s in at least one of them.

2.2 QSAR

Quantitative Structure Activity (or Property) Relationships (QSAR/QSPR) is a technique that is concerned with predicting different types of activities or chemical properties of molecules and compounds based on their biological structures. In recent years, this is typically done using machine learning.

There are four primary stages in conducting QSAR/QSPR modeling. The first stage consists of selecting a chemical domain and gathering sets of data (compounds and molecules) within it. The biological activities or properties of these compounds are obtained through experiments. In the second stage, mathematical representations of the acquired data are generated by computing appropriate molecular descriptors. A mathematical model (machine learning) is applied to the data in the third stage, learning the relationship between the descriptors and their corresponding activities or properties. Ultimately the performance of the model is assessed. If satisfactory, it can be used for predicting the activities or properties of molecules and compounds in the specific chemical domain.

3 Background: Machine Learning

In this section, information about different areas of machine learning and their applied use in typical scenarios is introduced. What follows is an introductory explanation of supervised learning, unsupervised learning, and semi-supervised learning.
3.1 Supervised Learning

The idea behind supervised learning (SL) is about learning a relationship (mapping) between input variables $x$, also called input features or attributes, to some outputs $y$. The goal is to find a good predictive function $\hat{y}(x)$ (an estimate) that can approximate the real output of $y(x)$. This is done by adapting some mathematical model or system (a learning algorithm) on training data, with the goal of predicting the outputs of new and unseen data points (test data), for which only the input variables are known. The notion of creating a model that can predict the output of previously unseen data is called generalization.

The term supervised in SL denotes that the data used in the learning process is labeled, meaning that both the inputs and outputs are known, and thus supervised machine learning is a way of learning by example. This branch of machine learning contains many different algorithms, and they are typically applied to two types of problems. When the output is categorical (also called qualitative), meaning that the output belongs to some class in a set, the problem is a classification problem. When the output is numerical, which can either be quantitative or discrete with a natural ordering, predicting $y(x)$ is a regression problem. The input features can also be classified as either quantitative or qualitative, and it is usually the case that the inputs to a model are a mixture of both.

An example of a categorical output would be $y \in \{\text{dog, cat}\}$. This particular example describes a binary classification problem since there are only two possible classes to predict. An example of a numerical output would be a vehicle’s braking distance given a set of input features such as weight, max speed, and manufacturer.

3.2 Unsupervised Learning

In unsupervised learning (UL), the main goal is to discover underlying structures and patterns in some data set to separate the data into distinctive groups eventually. In contrast to supervised learning, the training data in UL contains no supervisory information, meaning that the data is typically not labeled, hence the term unsupervised.

Clustering is the creation of different groupings (clusters) of the original feature space that contain data points that are closely related to one another, more so than to points assigned to other clusters. Some notion of similarity is used in the clustering process, and it could be different depending on the clustering algorithm used.

In supervised learning, the goal is to build a model of the form $p(y|x; \theta)$, where $\theta$ denotes a set of parameters to the model. In UL, however, the model is of the form $p(x)$, meaning that the model is estimating unconditional distributions. Another use of UL is performing dimensionality reduction with the end goal of finding a lower-dimensional representation of the data.
3.3 Semi-Supervised Learning

Semi-supervised learning (SSL) is concerned with using both labeled and unlabeled data to perform specific learning tasks. The set of unlabeled data is typically much larger than the labeled ones. SSL uses and combines methods from both supervised and unsupervised learning paradigms. When SSL is applied to a supervised problem, it may use unlabeled data to improve the efficiency of the underlying model. When the problem is an unsupervised one, a small set of labeled data points can be beneficial. In other words, SSL utilizes methods usually associated with the other paradigm of the underlying problem that it is trying to solve [11].

For semi-supervised learning to work, the learning algorithm that is used needs to extract useful information from the unlabeled data, which is only possible if the data itself contains useful information. It can thus be understood that the underlying marginal distribution $p(x)$ of the input space has to have some connection to the conditional distribution $p(y|x)$. Only then can unlabeled data be of use to improve the performance of classifiers [11, 27]. The way that the distributions $p(x)$ and $p(y|x)$ interact with each other is not always the same, which has given rise to certain different assumptions that describe the interaction between them [27].

The Smoothness Assumption

Assume that two data points $x_a$ and $x_b$ are extracted from the same data set. The smoothness assumption states that if $x_a$ and $x_b$ are close in the input space, their outputs should be similar. The transitive property can also be applied to the smoothness assumption. If we introduce another unlabeled data point $x_c$ that is not close to $x_a$, but is close to $x_b$, then the label of $x_a$ is propagated through $x_b$ to $x_c$ [11].

A supervised learning method that makes use of the smoothness assumption is the k-Nearest Neighbors (KNN) [13] algorithm. The $k$ indicates the amount of neighboring data points to consider when making a prediction. The queried point will assume to have the same output as the represented majority in the $k$ closest in distance neighboring nodes for classification problems.

An example of this assumption is visualized in Figure 1. The example depicts a binary classification problem where points are labeled as two classes represented as a cross and a circle. Assume that two new points, visualized as a square and a triangle in the figure, are queried. Following the smoothness assumption, the square should belong to the class represented by a cross, and the triangle should belong to the class represented as a circle.
3.3.1 The Low-Density Assumption

Assume a classification problem where an output $y$ can be any class of $N$ classes. The input space can thus be segmented into $N$ different regions, and a data point inside a particular region will be classified with the label belonging to that corresponding region [13]. The low-density assumption states that the decision boundaries that make up the regions will likely pass through low-density areas in the input space and not cut through points in high-density areas. A decision boundary that is placed in low-density areas will not violate the smoothness assumption either [11].

An example of the low-density assumption can be seen in Figure 2. As stated in this assumption, the line that separates the two classes in the figure passes through a low-density area of the feature space visible in the figure.
3.3.2 The Manifold Assumption

A manifold is a lower-dimensional substructure of some higher-dimensional input space that is locally Euclidean. The manifold assumption states that the entire input space comprises multiple lower-dimensional manifolds and that all existing data points lie on these manifolds. It also states that data points that lie on the same manifold will share a similar output [11].

3.3.3 The Cluster Assumption

The cluster assumption states that data points that share the same cluster will have a similar label. Dividing and grouping data into clusters requires some notion of similarity between the data points that needs to define the link between the input data distribution $p(x)$ and $p(y|x)$. This concept of similarity can be defined using the smoothness, low-density, and the manifold assumptions [11].

3.3.4 Inductive Methods

Semi-supervised learning can be categorized into inductive and transductive SSL. With inductive machine learning methods, the aim is to construct classifiers that are expected to generate predictions for any data point in the input space, meaning that they should also predict with great confidence on previously unseen data. Inductive SSL methods can again be further categorized into different subcategories of methods, with the difference being how they incorporate unlabeled data in the learning process [27].
3.3.5 Self-Training

The self-training algorithm is one of the simplest inductive SSL methods available and belongs to a set of inductive methods called wrapper methods. The Self-Training algorithm consists of a single supervised model that is trained on both labeled and unlabeled data points. The simplicity stems from its modularity, as it can be attached to any base supervised learner to incorporate unlabeled data into the learning process [11].

At the start of a Self-Training procedure, the supervised model will initially be trained on the labeled set of the feature space. The resulting model is then used on a subset of the unlabeled data points to obtain predictions of these data points. The predictions made with the most confidence (by some definition of confidence) are then included with the labeled data set, and the classifier is retrained. This is known as pseudo-labeling. This process is iterative and continues until no more unlabeled data remain or a stopping criterion has been reached [11].

3.3.6 Transductive Methods

Transductive methods are only defined on the feature space (the inputs from a given set of data) [27]. The objective of transductive methods is to predict the labels of the unlabeled subset of data points in the training set. Since no model of the input space exists, information will thus have to propagate through some connection between the existing labeled data points to the unlabeled ones. All transductive methods are thus considered graph-based, where information can flow between nodes (data points) through connected edges. A graph-based transductive algorithm generally consists of three steps, the construction of the graph, graph weighting, and the process of inferring the labels of the unlabeled portion of the data [11].

4 Theory

This section begins with an explanatory section of concepts such as bias, variance and how they relate to model complexity, and the tools used to evaluate and compare the performance of different models. It then delves deeper into the machine learning algorithms used in this thesis.

4.1 The Expected Performance of a Model

The performance of a generalized machine learning model relates to how well it can predict on independent testing/validation data. Being able to assess the performance of a model is essential since it can be used in the data fitting process of a model in order to create a more accurate assessment of its performance or used to compare between different types of models [10].

Two distinctions in terms of error estimation need to be made. The first one is the training error $E_{\text{train}}$ that describes how well the model performs on the
training sample in the learning process [13], and the generalization error $E_{\text{new}}$, which describes how well the model will perform on future data [10]. The goal in SL and inductive SSL is to minimize $E_{\text{new}}$.

4.1.1 The Bias-Variance Tradeoff

The generalization error can be decomposed into a bias squared term, a variance term, and an irreducible error term [13]. The error from bias comes from the difference between the expected predictions made by some model and their actual values. The error due to variance comes from how much the prediction of a particular data point varies each time the model is trained on a different training set. Both the bias and the variance of a model need to be considered to keep the generalization error small. The model is overfitting to the training data (especially to the noise in the data) when the variance is high and bias is low. When the model has high bias and low variance, it is instead underfitting the data [10, 13].

A model would have to become more flexible to reduce the bias and thus lessen the difference between the predictions and their actual values. The model would have to be less sensitive to the specific data points present in the training data to reduce the variance. In terms of model complexity, a simple model has a high bias but a low variance, and a complex model has a low bias but high variance [13]. The objective of generating a good model is thus about finding the suitable model complexity where the reduction in variance meets the increase in bias (or vice versa) [10]. The selection of a suitable model complexity level is known as the bias-variance tradeoff [13].

4.1.2 Evaluating Models

Computing the generalization error is not straightforward and making the assumption that $E_{\text{train}} \approx E_{\text{new}}$ is not possible either. One approach is using a hold-out validation set that consists of a random sample of the original data set $T$ such that $T$ is divided into a set $T_{\text{hold-out}}$ and $T_{\text{train}}$. A model would then be trained on $T_{\text{train}}$, and its performance is evaluated using the hold-out set. The generalization error can then be estimated to be $E_{\text{new}} \approx E_{\text{hold-out}}$, and the accuracy of a model is obtained by calculating $1 - E_{\text{new}}$ [10, 13].

Using a hold-out set in order to evaluate a model will decrease the amount of data available for training, and the larger the hold-out set, the lower the variance of $E_{\text{hold-out}}$, meaning a better estimate of $E_{\text{new}}$. The problem is that the smaller the training data, the higher the generalization error. Thus, the hold-out set approach should only be used when the amount of available training data is not limited. A better option would be to use k-fold cross-validation (KFCV) [13], which allows $E_{\text{new}}$ to be estimated using all the available data.

The idea behind KFCV is to first split the data set into $k$ batches of similar sizes. Iteratively, a machine learning model is applied to the data using one batch as the hold-out set and the remaining batches as the training set. This is repeated $k$ times until each batch has been used as the hold-out set at most
once. The estimate of $E_{\text{new}} \approx E_{k\text{-fold}}$ would then be the average of all calculated generalization errors. Using the average decreases the variance of $E_{k\text{-fold}}$ and is thus a better estimate of the generalization error than using a single hold-out set \[16, 13\].

KFCV is a popular approach in model selection and hyperparameter tuning \[13\], however using them in these settings invalidates the use of $E_{k\text{-fold}}$ as an estimator of the generalization error, with the reason being that the model is overfitting to the data.

### 4.1.3 The Confusion Matrix

The confusion matrix can be used to obtain the true positives (TP), false negatives (FN), false positives (FP), and true negatives (TN) predictions of a model. An example of a confusion matrix can be seen in Table 1. These can be further used to calculate a variety of performance metrics such as:

\[
\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{FP} + \text{FN} + \text{TP} + \text{TN}},
\]

\[
\text{Recall/True positive rate (TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}},
\]

\[
\text{False positive rate (FPR)} = \frac{\text{FP}}{\text{TN} + \text{FP}},
\]

where recall/TPR describes the proportion among the positive data points that are correctly predicted as positive and FPR the proportion among the negative data points that are incorrectly predicted as positive \[13, 6\].

Classifiers use a decision threshold to determine the prediction of a queried data point. In a binary classification problem where a data point $x$ belongs to a class $m \in \{0, 1\}$, a commonly used decision threshold of 0.5 would label $x$ as 0 if the probability $p(m = 0|x) \geq 0.5$. Plotting the values of TPR and FPR over a varying threshold $\tau \in [0, 1]$ can be used to obtain the Receiver Operating Characteristic (ROC) curve as shown in Figure 3 which determines the performance of a model for all decision thresholds $\tau$. The area under the ROC curve (AUC) is a compact summary of the ROC curve that depicts the probability that a randomly chosen positive sample is correctly ranked with greater suspicion than a randomly chosen negative sample \[6\]. An AUC score of 1 depicts a perfect classifier \[13\].

<table>
<thead>
<tr>
<th>$\hat{y}(x) = 1$</th>
<th>$\hat{y}(x) = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y(x) = 1$</td>
<td>TP</td>
</tr>
<tr>
<td>$y(x) = 0$</td>
<td>FP</td>
</tr>
</tbody>
</table>

Table 1: Depicting a confusion matrix where the positive class is 1 and the negative class is 0. $y(x)$ is the true label of $x$ whilst $\hat{y}(x)$ is the predicted label.
Figure 3: Visualizes the ROC curve and corresponding AUC of a KNN classifier applied to some data. The dashed straight line depicts a classifier which would only assign random guessed predictions.

4.2 Random Forest

A Random Forest model (RF) [23] is based on the construction of multiple decorrelated decision trees [13] operating on different bootstrapped data sets of the original training data. When querying a new data point, the final prediction is based on the majority vote (or the average in regression problems) of all decision trees that make up the forest [13].

Assume a set of training data $D_T$. The RF algorithm will initially draw $N$ random samples with replacement from $D_T$, creating a bootstrapped data set $\tilde{D}_T$. This sampling procedure is known as bootstrapping (or bagging) and is done $B$ times. A decision tree is then constructed and trained on each bootstrapped data set $\tilde{D}_T^1, \ldots, \tilde{D}_T^B$. With bagging, the variance of the model can be reduced without increasing the bias by much. The reduction is possible due to the final prediction being dependent on the average (or majority vote) of all models instead of only the one prediction [13].

The decision tree algorithm splits the input space, based on some splitting criterion, into multiple disjoint regions. The splitting criterion differs between regression and classification problems. The final prediction depends on the region (or leaf node) that the queried point ends up in. The depth of a decision tree is an important parameter, mainly because of how it changes the decision boundary of the predictor and thus greatly impacts the final prediction [13].

What separates Random Forests from using bagging alone on top of any other supervised learner is that, whenever a split of the input space is being done, not all possible input variables are considered. Instead, a randomly chosen subset of the variables is used in each splitting decision and is repeated at every other splitting decision as well [13]. The default size for these random subsets are usually $\sqrt{p}$ for classification problems and $\frac{p}{2}$ for regression problems where $p \leq q$ and $q$ is the amount of features in $D_T$ [10]. This is done to de-correlate.
the trees as the bootstrapped data sets are highly correlated from the start, which in turn decreases variance more than using bagging alone [23].

4.3 k-Means Clustering

Clustering is an example of unsupervised learning where the goal is to group similar (by some notion of similarity) data points together. k-Means is a clustering method that can create $k$ clusters in a data set $T$ where all data points in $T$ end up belonging to some cluster. The clusters are chosen so that the Euclidean distances (or other measurements of distance) to the cluster centroids (centers), summed over all data points, are minimized [13]. Figure 4 visualizes an example of the clusters that could be obtained by using k-Means on some data.

Figure 4: Visualizes a possible example of clusters that could be generated by the k-Means algorithm with $k = 4$.

4.4 Graph-Based Transductive Methods

As previously stated, transductive methods are not used to create a predictor that works on the entire input space but only on the unlabeled data points of the feature space, namely the training data provided to the model.

Transductive algorithms usually operate on graph-like structures defined over all data points in the training set, both the labeled and unlabeled ones, and encode pairwise similarities between them [11]. Different label propagation algorithms can be used in graph-based machine learning models, and as the name suggests, are used to propagate the labels of nodes that represent the labeled data points to their neighboring nodes [11]. Since graph-based methods rely on the local similarity between data points in the input space, they have a connection to the smoothness, cluster, and manifold assumptions [11].
4.4.1 Constructing the Graph

The basis of a graph-based SSL method is the construction of the graph. The graph \( G = (V, E) \) consists of nodes \( V \) represented by data points \( x \in T \), where \( T \) is the set of labeled and unlabeled training data. The edges \( E \) represent the similarities between the nodes based on some similarity measure. The similarities are given by weighting the edges in \( E \), which results in a weight matrix \( W \) that contains the edge weights of all pairs of nodes. The zero-edge weight indicates that no edge exists between the two nodes [7].

Graph construction is fundamentally a problem of estimating \( W \) [18]. The edges are weighted with the use of kernels. There exist different kinds of kernels such as:

**k-Nearest Neighbors**

A KNN kernel with the following definition can be used for calculating each \( w \in W \):

\[
W_{ij} = \begin{cases} 
\text{dist}(x_i, x_j) & \text{if } i \text{ is the nearest neighbor of } j \text{ or vice versa} \\
0 & \text{otherwise}
\end{cases}
\]

(5)

where the decision of data point \( x_i \) being close to \( x_j \) is based on some distance function used in the KNN algorithm [18]. To summarize, the algorithm takes the \( k \) closest in distance (similarity) data points of \( x_i \), where \( k \in \mathbb{Z}^+ \), and if \( x_j \) is a part of that neighborhood, an edge is created between them. The result of \( \text{dist}(x_i, x_j) \) can either be a constant 1 (the graph is thus unweighted) or some other numerical value [27].

**Gaussian Radial Basis Function**

The Gaussian Radial Basis function kernel (RBF) is defined as:

\[
W_{ij} = \exp\left(-\gamma \|x_i - x_j\|^2\right),
\]

(6)

where \( \gamma > 0 \) and \( \|x_i - x_j\|^2 \) is the Euclidean distance between data points (vertices) \( x_i \) and \( x_j \) squared [24].

4.4.2 Label Inference

The inference process consist of forming the predictions of the unlabeled data points in \( T \). Transductive algorithms normally contain a component that is used to penalize predicted labels that do not match the true label of a prediction, and another one that is used to penalize the differences in predictions done for connected nodes. More formally, transductive graph-based methods wants to find the prediction \( \hat{y} \) of an unlabeled node with true label \( y \) that minimizes:

\[
\lambda \sum_{i=1}^{l} \ell(\hat{y}_i, y_i) + \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} \ell_U(\hat{y}_i, \hat{y}_j),
\]

(7)
where $\lambda$ controls the relative importance of the supervised term $\mathbb{I}$. Methods used in the inference process differ mainly by their choice of loss functions $\ell_U$ and $\ell$ used in Equation (7). Some methods add another single regularization term on the unlabeled predictions $\mathbb{I}$.  

### 4.4.3 A Label Propagation Method: Label Spreading

Label propagation algorithms are iterative and start with the nodes representing the labeled data points to propagate their corresponding labels to their neighbors, repeating the process until convergence.

The Label Spreading algorithm (also commonly known as Learning with Local and Global Consistency) initializes the graph construction by computing the weight matrix $W$ as described in Section 4.4.1 using a kernel. The diagonal degree matrix $D$ and the normalized graph Laplacian matrix $L$ is then also computed, defined as:

$$D_{ii} = \sum_j W_{ij} \quad (8)$$

and

$$L = D^{-1/2} W D^{-1/2}. \quad (9)$$

If the set of unlabeled and labeled data points $T$ is of size $n$, then let $Y$ denote a label distribution matrix of size $n \times c$ where $c$ is the amount of possible classes in $T$. The $i$th row and $j$th column in $Y$ represents the label probabilities of node $x_i$ to class $c_j$. $\hat{Y}^t$ defines the matrix $\hat{Y}$ at the iterative step $t$.

After initializing $\hat{Y}^0$, the algorithm proceeds to choose a parameter $\alpha \in [0, 1]$, and then in an iterative manner computes $\hat{Y}^{t+1}$ until convergence to $\hat{Y}^\infty$ as such:

$$\hat{Y}^{t+1} = \alpha L \hat{Y}^t + (1 - \alpha) \hat{Y}^0. \quad (10)$$

The parameter $\alpha$ is used to specify the relative amount of information that a specific point receives at each iterative step from its neighboring nodes (the first term in Equation (10)) and from the initial label distribution matrix (the second term in Equation (10)). Finally, the algorithm labels each point $x \in T$ with the label from which it has received the most information from $\hat{Y}$.

## 5 Method

This section details the data preparation, the implementation of the applied methods, and the environments in which these methods are being evaluated.

### 5.1 Experimental Setup

The data sets used in this thesis are shown in Table 2 and were provided by the Pharmaceutical Bioinformatics research group (PharmBio) at Uppsala University. Data sets A, B, and C are reserved for tuning the hyperparameters of the applied models, while sets D, E, F, G, H, I, and J are used for evaluation.
The contents in each data set are divided into two columns. The first column contains strings that represent the structures of the molecules using a line notation format for molecular structures called Simplified Molecular Input Line Entry Specification (SMILES) \[23\]. The second column contains an integer, 1 or 0, that indicates a molecule’s ability to bind to some specified target or not, respectively. An example of what some molecules look like and their corresponding SMILES representations are visualized in Figure \[4\].

The data sets are artificially balanced by the research group, meaning that they are manipulated so that each class is represented equally. In data sets where the negative class is overrepresented, molecules belonging to the negative class are removed to balance the data. When the positive class is overrepresented, molecules are added under the assumption that they are negative, meaning that no experiment has been conducted to prove that the binding activity is nonexistent in these molecules. The duplicate SMILES entries are removed from all sets as not to skew any results. The removal of duplicates in each data set can lead to a slightly imbalanced representation of the classes as depicted in Table \[2\].

In the experiments, the semi-supervised and supervised learning methods are applied to a data set \(T\) consisting of a smaller set of labeled data points \(L\) of some fixed size \(|L| \in \{50, 100, 200, 500, 1000\}\), and a more extensive set of unlabeled data points \(U\), which consists of the remaining data points. The fixed sizes of \(L\) are chosen to appropriately represent a realistic scenario where the set of labeled data points is scarce and the amount of unlabeled data is abundant. The labeled data sets of different sizes will be defined as \(L_{50}, L_{100},\) and so forth.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Size</th>
<th>Positive(%)</th>
<th>Negative(%)</th>
<th>Binding Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>15002</td>
<td>43</td>
<td>57</td>
<td>Beta-secretase 1</td>
</tr>
<tr>
<td>B</td>
<td>10643</td>
<td>47</td>
<td>53</td>
<td>CB2 receptor</td>
</tr>
<tr>
<td>C</td>
<td>10032</td>
<td>49</td>
<td>51</td>
<td>HERG</td>
</tr>
<tr>
<td>D</td>
<td>13092</td>
<td>44</td>
<td>56</td>
<td>D2 receptor</td>
</tr>
<tr>
<td>E</td>
<td>10571</td>
<td>46</td>
<td>54</td>
<td>Coagulation factor X</td>
</tr>
<tr>
<td>F</td>
<td>10173</td>
<td>49</td>
<td>51</td>
<td>JAK2</td>
</tr>
<tr>
<td>G</td>
<td>15797</td>
<td>47</td>
<td>53</td>
<td>VEGF receptor 2</td>
</tr>
<tr>
<td>H</td>
<td>11686</td>
<td>44</td>
<td>56</td>
<td>EGF receptor</td>
</tr>
<tr>
<td>I</td>
<td>11209</td>
<td>47</td>
<td>53</td>
<td>SCN9A</td>
</tr>
<tr>
<td>J</td>
<td>10919</td>
<td>48</td>
<td>52</td>
<td>Carbonic anhydrase II</td>
</tr>
</tbody>
</table>

Table 2: The table contains the different data sets used in the learning/classification processes of the implemented methods. The sizes excludes all duplicated molecules as they are removed in the implementation. The positive and negative columns indicate the ratio of molecules in the corresponding set that posses the binding activity or not respectively. The last column refers to the binding target (ligand).
5.1.1 Input Features

Molecular fingerprints, more specifically the Morgan fingerprints (MFP), also known as Extended-connectivity fingerprints (ECFP) [23] are used exclusively as input features in all applied methods. The MFP is a circular fingerprint specifically designed for similarity searching and QSAR modeling [2] and represents molecules by defining their subparts. The process of creating the MFP starts by assigning an integer identifier to each non-hydrogen atom. Then, by extending each atom’s connectivity to its neighboring atoms, the identifiers are updated based on the identifiers of each neighboring atom. Each bit in the MFP represents the hash value of information of some atom and its neighboring atoms within some radius [3, 23].

The use of molecular fingerprints in similar QSAR modeling scenarios is shown to give robust results and is thus considered to be of good use as input features [2]. Experiments [2] show that ECFP (MFP) in particular is one of the top-performing ones among 2D fingerprints.

The MFPs are implemented using the Python framework RDKit[1]. The RDKit module allows for selecting the bit size and the radius of the fingerprint. The bit size refers to the length of the bit vector. The radius refers to the number of bond lengths away that are taken into account for each neighbor when calculating the identifiers [3]. A bit size of 1024 and a radius of 2 was

---


---

Figure 5: Visualizes the structure of a molecule and its corresponding SMILES representation. The molecule is obtained from data set A referenced in Table 2.
eventually used.

5.1.2 Sampling the Labeled Data Sets

The extraction of labeled data from the data sets is done in two different scenarios. In the first scenario, the experiment is carried out using labeled data randomly sampled from the training set. The larger set of labeled data will always contain the smaller set, i.e., the samples in $L_{50}$ from data set A will be a subset of $L_{100}$ extracted from the same data set.

In the second scenario, $L$ is sampled from clusters. The k-Means clustering algorithm is used to divide the entire data set into $k$ different clusters. The data point closest to each centroid will be extracted from the data set and used in $L$. It might not be certain that any data point in the smaller labeled data set will exist in the larger one in this scenario. The different scenarios of choosing the labeled portions will be referred to as scenario 1 and scenario 2, respectively.

5.2 Model Evaluation

All the data points in each set are initially labeled so that the potential of SSL can be correctly evaluated. The unlabeled data points are then simulated by removing the labels of the portion used as the unlabeled set. The evaluation metrics of interest are the classification accuracy and the area under the ROC curve.

AUC is a useful metric when evaluating models that are applied to binary classification problems that are balanced and, in most cases, a better metric than accuracy since it provides a broader view by considering all possible decision thresholds \[6\]. The use of predictive accuracy in this experiment is still justified due to the data sets almost being perfectly balanced as visualized in Table 2.

Evaluating a Transductive Method

Comparing the performance of a transductive method to an inductive (or supervised learning) one is not trivial. This is due to the end goal of transductive and inductive methods being inherently different [27]. The performance of SSL methods used in a transductive setting is of more interest to the research group. However, since a transductive method is compared to a supervised learning model, it was deemed fair to examine their performance in an inductive setting as well. Thus two different settings for evaluating transductive methods are proposed.

In the transductive setting, each data set $T$ is split into a set of labeled data points $L$ and a set of unlabeled data points $U$. After applying the transductive method to $T$, the inferred labels of $U$ and the labeled data points from $L$ are compared to the ground truth. Since all points from $T$ is a part of the evaluation, the positive/negative class ratio of each data set depicted in Table 2 is retained. For scenario 1, a technique known as k-fold random sub-sampling [28] is used to reduce overfitting and thus achieve a better estimate of all performance metrics.
In the inductive setting, the inferred labels of the transductive methods are used as training data for a supervised learning model, which is reminiscent of a Self-Training algorithm. This would make it possible to compare a supervised model using only the initially labeled data \( L \) versus using \( L \) together with the inferred labels of \( U \) to improve the model’s performance. The evaluation is then performed on a test set. These test sets were also ensured to contain the exact class ratio representation of the whole data set. In other words, if some data set \( T \) has 40% molecules belonging to class \( A \) and 60% to class \( B \), the test set would contain the same ratio. The evaluation is performed using k-fold cross-validation to reduce overfitting and achieve a better estimate of the performance \([5, 13]\).

5.3 Label Spreading

The Label Spreading method is implemented using the Scikit-Learn\(^2\) semi-supervised learning module. For reproducibility purposes all methods that use randomness are initiated using a predefined seed \( r = 1 \).

5.3.1 Tuning Hyperparameters

The LS method in Scikit-Learn allows for choosing between a RBF and KNN kernel, which are defined in Equations (5) and (6), or a custom implemented one. Both kernels were tested using different values for the hyperparameters associated with each kernel (\( \gamma \) for RBF and the number of neighbors for KNN) and the \( \alpha \) parameter as mentioned in Section 4.4.3. The maximum allowed iterations parameter was set to 1000 to ensure that the inference process will converge. Each combination of the parametric values seen in Table 3 was used to train the method in both scenarios described in Section 5.1.2. The values were chosen by randomly applying LS to data sets A, B, and C using different values of \( \gamma, \alpha, \) and neighbors to determine appropriate ranges for these values. An exhaustive parameter search was not conducted.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>neighbors</td>
<td>7, 21, 41, 81, 121, 181, 241</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>0.1, 1, 5, 15, 30, 50</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>0.15, 0.35, 0.55, 0.75, 0.95</td>
</tr>
</tbody>
</table>

Table 3: The values used when tuning the hyperparameters of LS.

The tuning of hyperparameters was performed on data sets A, B, and C to indicate what parametric values may perform equally well on any similar data. These were tuned to maximize the accuracy of the method. In scenario 1, k-fold random sub-sampling with \( k = 5 \) was used to sample the labeled data set \( L \) for each hyperparameter combination. In scenario 2, \( L \) was sampled using the

clustering approach mentioned in Section 5.1.2 with k-Means initialized with only one random seed.

5.3.2 Implementing the Label Spreading Method

The implemented LS methods were applied to data sets D, E, F, G, H, I, and J with the obtained hyperparametric values. The methods were then evaluated in the two separate settings mentioned in Section 5.2, and each performance metric was calculated using the obtained confusion matrices for each applied method.

In the transductive setting for scenario 1, k-fold random sub-sampling with \( k = 10 \) was used to sample the labeled data set \( L \) from each data set. k-fold random sub-sampling was achieved using the `ShuffleSplit` function. The function allowed each data set \( T \) to be divided into a randomly sampled labeled set \( L \) of some size, and an unlabeled set \( U \), repeating the process ten times each time using a new random seed. The LS methods and the RF model were then applied to each split, and the intermediate results were stored away. An estimate of all performance metrics was obtained by calculating the means of all intermediate results.

For scenario 2, the k-Means clustering method was applied to \( T \) creating \( k \) clusters. The data points in \( L \) were then sampled by extracting the data point closest to the centroid in each \( k \)th cluster (as described in Section 5.1.2). The k-Means method was also obtained from the Scikit-Learn library. All parameters were set to their default values as described in the documentation except for the \( k \), which was chosen to be equal to \(|L|\), meaning \( k \in \{50, 100, 200, 500, 1000\} \). This way of choosing \( k \) was deemed most appropriate after discussing it with the thesis supervisors. The k-Means method was performed ten times, each time using a new random seed \( r \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \) for the centroid initialization part of the algorithm.

K-fold cross-validation with \( k = 10 \) was used in the inductive setting, splitting the data sets into a training and testing set for each \( k \)th fold. The training sets were then used in the same way as in the transductive setting. However, in the inductive setting for scenario 1, the random sampling of \( L \) was only performed five times per fold. For scenario 2, k-Means was also executed using five different random seeds as opposed to ten. The inferred labels and the labeled set were then used in the fitting process of a RF model. The RF model was evaluated using only the labeled set and combined with the inferred labels to see if an increase in its performance could be made.

---

5.4 Implementing the Random Forest Model

The implemented RF classifier was provided by the Scikit-Learn library. The use of RF in QSAR modeling is known to work particularly well “off-the-shelf” with no hyperparameter tuning \[19\]. Thus they were all set to their default values as specified in the Scikit-Learn documentation. The default splitting criterion is the Gini index defined as:

\[
\sum_{m=1}^{M} \hat{\pi}_m (1 - \hat{\pi}_m),
\]

where \(\hat{\pi}_m\) is the proportion of observations in the \(\ell\)th region of the training sample that belongs to class \(m\) \[13\]. The maximum amount of features to consider when looking for the best split was set to \(\sqrt{1024}\) with 1024 being the bit vector size of each MFP. The number of trees to be constructed was set to 100.

6 Results

In this section, we present the experimental results corresponding to the scenarios defined in Section 5.1.2. For each scenario, we visualize the performance of the methods using diagrams and tables, followed by a discussion. The performance is measured by taking the mean accuracy and AUC of the applied methods for all data sets. The mean performance is used because the research group is interested in how well the applied methods can perform regardless of the underlying data. A broader discussion of the methods and their limitations is presented in Section 7.

\[\text{Scikit-Learn version 0.24.1, documentation at } \text{https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html} \]
6.1 Label Spreading: Scenario 1

6.1.1 Hyperparameters

(a) Kernel: KNN

(b) Kernel: RBF

Figure 6: Heatmaps depicting the average accuracies achieved when applying the LS model on data sets A, B and C using different combinations of hyperparameters. The performance on each data set can be seen in Figures 13, 14, 15, 16 and 17. The labeled data sets $L_{50}$, $L_{100}$, $L_{200}$, $L_{500}$ and $L_{1000}$ where sampled from A, B and C by using 5-fold random sampling.

Table 4: Hyperparameters selected based on results in Figure 6

<table>
<thead>
<tr>
<th>Kernel</th>
<th>neighbors</th>
<th>$\gamma$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBF</td>
<td>-</td>
<td>1</td>
<td>0.95</td>
</tr>
<tr>
<td>KNN</td>
<td>81</td>
<td>-</td>
<td>0.75</td>
</tr>
</tbody>
</table>

A closer look at how the methods using the KNN (LS-KNN) and RBF (LS-RBF) kernel behaved in the hyperparameter tuning process for each size of $L$ can be seen in Appendix A.1. An interesting find when tuning the parameters of LS-KNN was that for smaller $||L||$, higher accuracy was achieved when using a larger value of the neighbors’ parameter. For larger $||L||$, a smaller amount of neighbors sufficed. LS-KNN consistently provided the best accuracy for any size of $L$ with $\alpha \in \{0.55, 0.75, 0.95\}$. When using LS-RBF, the value of $\alpha$ did not seem to matter. With $\gamma \in \{1, 5, 15\}$, it performed equally well in terms of accuracy for all $L$. 

26
6.1.2 Performance: Transductive Setting

(a) Mean accuracy.

Figure 7: Mean accuracy and AUC obtained by applying RF, LS-RBF, and LS-KNN on to data sets D, E, F, G, H, I, and J using 10-fold random sub-sampling.

<table>
<thead>
<tr>
<th>Size of $L$</th>
<th>RF</th>
<th>LS-RBF</th>
<th>LS-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>80.24 ($\pm$4.91)</td>
<td>79.98 ($\pm$3.97)</td>
<td>84.27 ($\pm$3.36)</td>
</tr>
<tr>
<td>100</td>
<td>85.11 ($\pm$3.45)</td>
<td>84.45 ($\pm$2.85)</td>
<td>87.08 ($\pm$2.62)</td>
</tr>
<tr>
<td>200</td>
<td>88.82 ($\pm$2.15)</td>
<td>88.39 ($\pm$2.34)</td>
<td>90.07 ($\pm$2.01)</td>
</tr>
<tr>
<td>500</td>
<td>91.70 ($\pm$2.03)</td>
<td>92.11 ($\pm$1.97)</td>
<td>92.58 ($\pm$1.89)</td>
</tr>
<tr>
<td>1000</td>
<td>93.49 ($\pm$1.77)</td>
<td>94.29 ($\pm$1.71)</td>
<td>93.82 ($\pm$1.54)</td>
</tr>
</tbody>
</table>

Table 5: Mean accuracy ($\pm$1std) over data sets D, E, F, G, H, I, and J for all $L$. The highest scores are highlighted.

<table>
<thead>
<tr>
<th>Size of $L$</th>
<th>RF</th>
<th>LS-RBF</th>
<th>LS-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>91.21 ($\pm$3.24)</td>
<td>89.23 ($\pm$3.38)</td>
<td>93.08 ($\pm$2.64)</td>
</tr>
<tr>
<td>100</td>
<td>93.71 ($\pm$2.15)</td>
<td>91.72 ($\pm$2.35)</td>
<td>94.91 ($\pm$1.88)</td>
</tr>
<tr>
<td>200</td>
<td>95.48 ($\pm$1.53)</td>
<td>94.01 ($\pm$1.62)</td>
<td>96.28 ($\pm$1.36)</td>
</tr>
<tr>
<td>500</td>
<td>96.98 ($\pm$1.10)</td>
<td>96.31 ($\pm$1.20)</td>
<td>97.56 ($\pm$0.97)</td>
</tr>
<tr>
<td>1000</td>
<td>97.88 ($\pm$0.81)</td>
<td>97.58 ($\pm$0.84)</td>
<td>98.21 ($\pm$0.72)</td>
</tr>
</tbody>
</table>

Table 6: Mean AUC ($\pm$1std) over data sets D, E, F, G, H, I, and J for all $L$. The highest scores are highlighted.

LS-KNN generally outperformed both LS-RBF and RF in terms of classification accuracy and AUC as visualized in Figure 7. RF and LS-RBF achieved similar accuracy, but RF performed better than LS-RBF when considering the AUC metric. This is true for $L_{50}, L_{100}$ and $L_{200}$, whereas $L$ grows large LS-RBF takes the lead in terms of accuracy. For large $L$, all three methods performed equally well in terms of AUC. It can be seen that LS-RBF and RF benefited...
the most in terms of a higher increase in performance when using more labeled
data. LS-KNN is most favorable when labeled data is scarce.

The performance of each method applied to each specific data set in a transductive setting can be seen in Appendix B.1.1. For most data sets, the curves of each diagram are moderately similar to as they are perceived in Figure 7. However, Tables 5 and 6 highlight that there is a significant variance in how they performed on each specific data set, meaning that each method did not achieve consistent performance on all data sets. For example, examining data set E (see Figure 24), the difference in accuracy for \(||L|| < 200\) is more significant between LS-KNN and the other methods than when applied to other data sets. Another interesting observation is the achieved performance on data set J (see Figure 29), where the RF model outperforms LS-RBF and LS-KNN in both accuracy and AUC by up to 20% and 6% respectively. LS-RBF is also seen deviating from the trend by outperforming LS-KNN in terms of accuracy on that particular data set.

6.1.3 Performance: Inductive Setting

![Figure 8](image)

(a) Mean accuracy. (b) Mean AUC.

Figure 8: Mean accuracy and AUC obtained by applying RF, RF-RBF, and RF-KNN to data sets D, E, F, G, H, I, and J using 10-fold cross-validation.

<table>
<thead>
<tr>
<th>Size of (L)</th>
<th>RF ((\pm 1.44))</th>
<th>RF-RBF ((\pm 3.67))</th>
<th>RF-KNN ((\pm 3.13))</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>79.45</td>
<td>79.91</td>
<td>86.12</td>
</tr>
<tr>
<td>100</td>
<td>85.72 ((\pm 3.45))</td>
<td>84.93 ((\pm 3.5))</td>
<td>88.93 ((\pm 3.08))</td>
</tr>
<tr>
<td>200</td>
<td>88.85 ((\pm 2.72))</td>
<td>88.66 ((\pm 2.72))</td>
<td>90.92 ((\pm 2.31))</td>
</tr>
<tr>
<td>500</td>
<td>91.44 ((\pm 2.28))</td>
<td>92.19 ((\pm 2.11))</td>
<td>92.69 ((\pm 1.78))</td>
</tr>
<tr>
<td>1000</td>
<td>92.95 ((\pm 2.04))</td>
<td><strong>93.98</strong> ((\pm 1.79))</td>
<td>93.34 ((\pm 1.57))</td>
</tr>
</tbody>
</table>

Table 7: Mean accuracy \((\pm 1\text{std})\) over data sets D, E, F, G, H, I, and J for all \(L\). The highest scores are highlighted.
<table>
<thead>
<tr>
<th>Size of $L$</th>
<th>RF</th>
<th>RF-RBF</th>
<th>RF-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>91.06 (±3.31)</td>
<td>81.56 (±3.91)</td>
<td>88.27 (±3.37)</td>
</tr>
<tr>
<td>100</td>
<td>93.55 (±2.48)</td>
<td>87.09 (±3.51)</td>
<td>91.36 (±3.05)</td>
</tr>
<tr>
<td>200</td>
<td>95.21 (±1.75)</td>
<td>91.25 (±2.36)</td>
<td>93.47 (±2.04)</td>
</tr>
<tr>
<td>500</td>
<td>96.72 (±1.21)</td>
<td>94.91 (±1.59)</td>
<td>95.42 (±1.47)</td>
</tr>
<tr>
<td>1000</td>
<td>97.54 (±0.96)</td>
<td>96.73 (±1.24)</td>
<td>96.3 (±1.25)</td>
</tr>
</tbody>
</table>

Table 8: Mean AUC (±1std) over data sets D, E, F, G, H, I, and J for all $L$. The highest scores are highlighted.

In Figure 8, it can be seen that using the originally labeled sets combined with the inferred labels provided an improvement in predictive accuracy for the RF model. In terms of AUC, however, it is noted that the RF model using only the initial labeled set $L$ without the pseudo-labels from LS consistently outperformed RF-KNN and RF-RBF. This is the case for most data sets (see Appendix B.1.2), especially when $|L| > 50$. A reason for this will be further discussed in Section 7.

RF is deemed a more appropriate choice for this scenario and setting since AUC is a better metric overall, as previously stated. For example, the difference in mean AUC for $L_{50}, L_{100}$ and $L_{200}$ between RF and LS-KNN is 2.79%, 2.19% and 1.74% respectively in favour to RF. The difference between RF and LS-RBF is significantly larger of 9.5%, 6.46% and 3.96% respectively.

It is again shown that the performance is highly dependent on the underlying data set. This dependency can be observed when examining the high variance in performance across the corresponding data sets as depicted in Tables 7 and 8.
6.2 Label Spreading: Scenario 2

6.2.1 Hyperparameters

![Heatmaps depicting the average accuracies achieved when applying the LS model on data sets A, B and C using different combinations of hyperparameters. The performance on each data set can be seen in Figures 18, 19, 20, 21 and 22. The labeled data sets L_{50}, L_{100}, L_{200}, L_{500} and L_{1000} were sampled from A, B and C using k-Means clustering.](image)

(a) Kernel: KNN  
(b) Kernel: RBF

Figure 9: Heatmaps depicting the average accuracies achieved when applying the LS model on data sets A, B and C using different combinations of hyperparameters. The performance on each data set can be seen in Figures 18, 19, 20, 21 and 22. The labeled data sets L_{50}, L_{100}, L_{200}, L_{500} and L_{1000} were sampled from A, B and C using k-Means clustering.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>neighbors</th>
<th>( \gamma )</th>
<th>( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBF</td>
<td>-</td>
<td>1</td>
<td>0.95</td>
</tr>
<tr>
<td>KNN</td>
<td>21</td>
<td>-</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Table 9: Hyperparameters selected based on results in Figure 9

Tuning the hyperparameters of LS-KNN and LS-RBF when \( L \) was sampled using the k-Means clustering algorithm increased their predictive accuracy. The behavior of the methods used with different sizes of \( L \) can be inspected more closely in Appendix A.2. Allowing the neighbor parameter of LS-KNN to be a value in the range of 21-81 provided better accuracy for all sizes of \( L \). LS-RBF performed better with \( \gamma \in \{1, 5, 15\} \). For both methods, the \( \alpha \) parameters were chosen equally to scenario 1.
6.2.2 Performance: Transductive Setting

(a) Mean accuracy.

(b) Mean AUC.

Figure 10: Mean accuracy and AUC obtained by applying RF, LS-RBF, and LS-KNN to data sets D, E, F, G, H, I, and J using a k-Means clustering algorithm to sample the labeled data sets.

<table>
<thead>
<tr>
<th>Size of $L$</th>
<th>RF</th>
<th>LS-RBF</th>
<th>LS-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>75.39  (±3.79)</td>
<td>87.5  (±2.72)</td>
<td><strong>88.71</strong> (±2.74)</td>
</tr>
<tr>
<td>100</td>
<td>75.11  (±3.88)</td>
<td>90.51 (±2.54)</td>
<td><strong>91.67</strong> (±2.42)</td>
</tr>
<tr>
<td>200</td>
<td>79.55  (±4.55)</td>
<td>92.99 (±2.16)</td>
<td><strong>93.44</strong> (±2.15)</td>
</tr>
<tr>
<td>500</td>
<td>90.33  (±3.89)</td>
<td><strong>94.68</strong> (±1.80)</td>
<td>94.64 (±1.90)</td>
</tr>
<tr>
<td>1000</td>
<td>94.69  (±1.89)</td>
<td><strong>95.66</strong> (±1.42)</td>
<td>95.42 (±1.63)</td>
</tr>
</tbody>
</table>

Table 10: Mean accuracy (±1std) over data sets D, E, F, G, H, I, and J for all $L$. The highest scores are highlighted.

<table>
<thead>
<tr>
<th>Size of $L$</th>
<th>RF</th>
<th>LS-RBF</th>
<th>LS-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>93.10  (±2.72)</td>
<td>92.31 (±2.72)</td>
<td><strong>93.56</strong> (±1.53)</td>
</tr>
<tr>
<td>100</td>
<td>94.33  (±2.69)</td>
<td>94.74 (±2.09)</td>
<td><strong>95.84</strong> (±1.91)</td>
</tr>
<tr>
<td>200</td>
<td>95.91  (±1.91)</td>
<td>96.27 (±2.04)</td>
<td><strong>96.99</strong> (±1.78)</td>
</tr>
<tr>
<td>500</td>
<td>97.47  (±1.19)</td>
<td>97.28 (±1.49)</td>
<td><strong>97.82</strong> (±1.37)</td>
</tr>
<tr>
<td>1000</td>
<td>98.34  (±0.78)</td>
<td>97.95 (±1.02)</td>
<td><strong>98.39</strong> (±0.91)</td>
</tr>
</tbody>
</table>

Table 11: Mean AUC (±1std) over data sets D, E, F, G, H, I, and J for all $L$. The highest scores are highlighted.

For any $|L| < 1000$, both LS-RBF and LS-KNN outperformed RF in terms of mean accuracy over all data sets, as visualized in Figure 10. From this, it can be concluded that when $L$ is small, the use of LS is favorable over RF. When $L$ grows, the performance of LS-RBF and LS-KNN increases, however, less so than that of RF. For $L_{100}$ and $L_{500}$, the mean accuracy of RF increases by approximately 16%. For $L_{1000}$, RF and LS-KNN perform equally well, and
LS-RBF takes a slight lead by 0.24%. All three methods performed relatively well in terms of AUC, with LS-KNN performing slightly better for $|L| \leq 500$. A closer look at how the methods performed on each data set can be seen in Appendix B.2.1.

An interesting observation is that the mean accuracy of RF had a large decrease when sampling from clusters instead of the random sampling done in scenario 1. However, the AUC score is still quite high, showcasing that the RF model is still a good classifier. The reasoning for this may be the effect of having the decision threshold set to 0.5. This is further discussed in Section 7.

An abnormal observation can also be seen in the mean accuracy diagram in Figure 10, namely that the predictive accuracy of RF decreases when $L$ goes from a size 50 to 100. By examining the behavior of RF on each individual data set, it is observed that its performance on data set G is what influences that decrease the most, as seen in Figure 40. It was stated in Section 5.1.2 that when sampling from clusters, it cannot be assumed that $L_{50}$ is a subset of $L_{100}$. If the data points sampled in $L_{50}$ are better suited for the training process of the RF model than the data points in $L_{100}$, it could explain the performance decrease.

Another example is that this decrease is not evident in the mean AUC diagram, indicating that this problem is also about the chosen decision threshold for the accuracy metric.

The previous observations regarding RF being considerably better on data set J in terms of both accuracy and AUC can be seen in Figure 25 for this scenario and setting as well. Tables 10 and 11 still depict a high variance in the performance between all data sets. This inconsistent behavior between the data sets highlights that the performance of the methods is highly dependent on the data they are applied to. Sampling from clusters depicts a small decrease of the standard deviation values for small $L$ in terms of mean AUC for all data sets compared to scenario 1.
6.2.3 Performance: Inductive Setting

(a) Mean accuracy.

(b) Mean AUC.

Figure 11: Mean accuracy and AUC obtained by applying RF, RF-RBF, and RF-KNN over data sets D, E, F, G, H, I, and J using 10-fold cross-validation and a k-Means clustering algorithm to sample the labeled data sets.

<table>
<thead>
<tr>
<th>Size of $L$</th>
<th>RF</th>
<th>LS-RBF</th>
<th>LS-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>73.98 (±5.20)</td>
<td>88.26 (±2.72)</td>
<td>89.64 (±2.74)</td>
</tr>
<tr>
<td>100</td>
<td>75.51 (±4.18)</td>
<td>90.97 (±2.84)</td>
<td>92.09 (±2.53)</td>
</tr>
<tr>
<td>200</td>
<td>79.49 (±5.24)</td>
<td>93.03 (±2.43)</td>
<td>93.62 (±2.26)</td>
</tr>
<tr>
<td>500</td>
<td>89.99 (±3.88)</td>
<td>94.64 (±1.85)</td>
<td>94.70 (±1.90)</td>
</tr>
<tr>
<td>1000</td>
<td>93.94 (±2.04)</td>
<td>95.24 (±1.66)</td>
<td>95.11 (±1.72)</td>
</tr>
</tbody>
</table>

Table 12: Mean accuracy (±1std) over data sets D, E, F, G, H, I, and J for all $L$. The highest scores are highlighted.

<table>
<thead>
<tr>
<th>Size of $L$</th>
<th>RF</th>
<th>LS-RBF</th>
<th>LS-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>92.89 (±2.75)</td>
<td>90.64 (±2.16)</td>
<td>91.55 (±2.22)</td>
</tr>
<tr>
<td>100</td>
<td>94.23 (±2.58)</td>
<td>93.68 (±2.28)</td>
<td>94.51 (±2.11)</td>
</tr>
<tr>
<td>200</td>
<td>95.58 (±2.11)</td>
<td>95.68 (±2.04)</td>
<td>96.15 (±1.91)</td>
</tr>
<tr>
<td>500</td>
<td>97.26 (±1.24)</td>
<td>97.21 (±1.55)</td>
<td>97.31 (±1.54)</td>
</tr>
<tr>
<td>1000</td>
<td>98.02 (±0.89)</td>
<td>97.82 (±1.32)</td>
<td>97.73 (±1.31)</td>
</tr>
</tbody>
</table>

Table 13: Mean AUC (±1std) over data sets D, E, F, G, H, I, and J for all $L$. The highest scores are highlighted.

Regarding the performance in the inductive setting, in terms of mean accuracy, both RF-RBF and RF-KNN consistently outperformed RF, with RF-KNN taking the lead over RF-RBF when $|L| \leq 200$. The LS methods are both more favorable when $L$ is small. The reason for the overall decrease in accuracy for small $L$, compared to scenario 1, will again be discussed in Section 7.
The decrease in performance of the RF model between $L_{50}$ and $L_{100}$ is not prevalent as in the transductive setting when considering the mean accuracy on all data sets. The performance on each data set can be seen in Appendix B.2.2. By observing the behavior of the RF model on data sets G and H (Figures 47 and 48 respectively), it can be noted that RF decreases in accuracy as $L$ goes from a size 50 to 100 but not enough to influence the mean.

Regarding mean AUC, all three methods performed somewhat equally for $L_{200}$, $L_{500}$ and $L_{1000}$. RF performed better (a difference of 1.34%) when only using the initially sampled $L$ rather than combining them with the pseudo-labels of LS when $||L|| = 50$. The mean AUC of RF is still high compared to the mean accuracy, again highlighting that the chosen decision threshold might be at fault.

The same trend that the models are not performing consistently across all data sets can be seen in Tables 12 and Tables 13 by observing the standard deviations. It was again observed in this scenario that RF achieved better performance using only $L$, as opposed to combining it with the pseudo-labels of LS, when applied to data set J (see Figure 50).

7 Discussion

The research group at PharmBio was interested in examining SSL methods that could generalize well on to any similar type of data used in ligand-based activity modeling where the binding target is not necessarily the same each time. For that reason, multiple data sets were used in the hyperparameter tuning process for the LS methods. By doing so, the choice of hyperparameters is not biased towards one specific data set but averaged over multiple different ones to minimize the bias and variance of the estimated performance.

There are issues with how the tuning process was carried out that need to be acknowledged. The obvious one is the choice of values for each hyperparameter. It would be better to try a more detailed parameter search using equal-sized steps between each value as it could allow for better-chosen hyperparameters, which could increase the overall performance of the applied methods. In hindsight, it would have been better also to tune the hyperparameters to maximize AUC since it is a better metric overall. Another one is that including data sets A, B, and C in the tuning process was completely random. It could be that the molecules in these data sets are too similar to one another and too dissimilar to the molecules in the other data sets used for evaluation. The misrepresentation of the diversity in possible data sets could make the methods too biased on data sets similar to A, B, and C. A possible solution would be to include more data sets to see if they produce the same hyperparameters. Another solution could be to pick data sets representing the diversity of data that these methods may potentially be applied to. However, for that to be a possibility, more profound knowledge in the field of pharmaceutical bioinformatics and the data sets themselves is required.

The goal of maximizing the predictive performance of the LS method was
successful in that the applied LS methods (especially LS with a KNN kernel) ended up performing better than the RF model on most data sets in all scenarios and settings in terms of accuracy. This applied to the AUC metric when considering the transductive setting. These results were satisfactory since the effectiveness of SSL methods in a transductive setting was of most importance to the research group.

An interesting observation when comparing the behavior of the methods in each scenario is that using a clustering approach to sample the labeled data set \( L \) will increase the accuracy of the LS methods but decrease it for the RF model. By comparing the Figures 7 and 10 that depicts the performances in the transductive setting, it can be seen that the mean accuracy of LS-KNN and LS-RBF goes from being at approximately 84-90% and 80-88% in scenario 1 to 88-94% and 87-93% in scenario 2 whilst \(|L| \leq 200\). All models had their performances increased in terms of AUC when sampling \( L \) from clusters, with LS-RBF benefiting the most. The positive effect in performance by sampling from clusters is evident in Figure 12, which depicts the performance increase in scenario 2 compared to scenario 1 in a transductive setting.

The reasoning for the increased performance when sampling from clusters could be directly attributable to graph-based SSL methods operating on the smoothness and clustering/manifold assumptions 11. By sampling data points randomly, \( L \) has a good chance of containing outliers, meaning that \( L \) may end up with a large amount of data that do not represent the class distribution in the surrounding region of the feature space. Since pairwise similarity is the basis of label propagation algorithms, it can be assumed that the more common
neighbors two nodes share, the larger the probability that they belong to the same class [26]. Since LS methods operate on this belief, they might predict that the label of a node close to the outlier belongs to the same class. This could, over time, diminish the method’s potential with correctly classifying the unlabeled data. When \( L \) is sampled by picking data points closest to the centroids of the generated clusters and assuming that the assumptions mentioned earlier are satisfied in the data sets, \( L \) has a greater chance of containing data points indicative of the surrounding class representation. In other words, having an initial labeled data set containing data that better represents the groupings (relations) in the corresponding data sets could improve the predictive capabilities of the LS methods, which could explain the increase in performance. A study in seed sampling (the sampling of \( L \)) using clustering methods shows that it can improve the performance of SSL methods [17].

The RF model went from a mean accuracy of 81-88% to 76-80%, a decrease by almost 8% when sampling from clusters. However, the mean AUC score is still high, and similar behavior can be observed in the inductive setting. The high AUC score is indicative of the model still being a good classifier when considering all possible decision thresholds \( \tau \in [0, 1] \). Table 14 portrays the mean TN, FP, FN, and TP obtained by the applied RF model on data set F at the decision threshold \( \tau = 0.5 \). The high count in FP is what ultimately affects the decrease of accuracy. The table also depicts that the training set \( L \) is overly represented by the positive class. In Table 2, it can be seen that data set F has a negative class representation of 51%. This means that the negative class is underrepresented in the training set even though they are the majority in the data set used for evaluation. This imbalanced training set will weaken the predictive accuracy of the model [22, 21]. It begs the question that a threshold at 0.5 might not be ideal for this model on this particular data set since the overly represented positive class in the training set will cause the model to predict many false positives. A possible solution would be to increase the decision threshold for the overrepresented class in the training set.

<table>
<thead>
<tr>
<th>Size of ( L )</th>
<th>Negative</th>
<th>Positive</th>
<th>TN</th>
<th>FP</th>
<th>FN</th>
<th>TP</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.28</td>
<td>0.72</td>
<td>2730</td>
<td>2324</td>
<td>32</td>
<td>4899</td>
</tr>
<tr>
<td>100</td>
<td>0.27</td>
<td>0.73</td>
<td>2892</td>
<td>2287</td>
<td>23</td>
<td>4871</td>
</tr>
</tbody>
</table>

Table 14: Columns two and three depict the mean ratio of the positive and negative class representations in the labeled data set \( L \) of size 50 and 100. The last four columns represent the mean true negatives, false positives, false negatives and true positives obtained from applying the RF model on data set F using a decision threshold \( \tau = 0.5 \). The mean values are calculated by acquiring the individual results of the RF model in the transductive setting in scenario 2.

The RF model consistently performed better using only the initially sampled \( L \) in terms of AUC when observing the results obtained from the inductive setting in scenario 1. In scenario 2, the gap is minimal and even indistinguishable in some cases. This entails that the RF model was much better at separat-
ing the classes in scenario 1 when using $L$ exclusively in the training process than using $L$ with the inferred labels of the unlabeled data obtained by the LS methods. The use of inferred labels (or pseudo-labels) in the fitting process is similar to how Self-Training (see Section 3.3.5) algorithms behave. Predictive errors early on in the iterative process of a Self-Training algorithm tend to reinforce themselves by generating incorrect labels. This will logically hurt the performance of any underlying classifier and their ability to estimate decision boundaries since they are trained on incorrectly labeled data. This could be why the RF model performs better when only using the initially sampled $L$. Since it was shown that the clustering approach improves the predictive performance in the inference process of the LS methods, it is rational that the inferred labels can be used to increase the AUC score of the RF model in scenario 2.

While the RF model could perform well on data set J, this was not the case for the LS methods. A reason could be that the SSL assumptions might not be satisfied on that particular data set, which leads to the unlabeled data set being rendered useless. Another reason could be that the hyperparameter tuning of the LS methods was not satisfactory when applied to data set J. It was also shown in a similar study that SSL is not always superior to SL in QSAR modeling and that there is no preferred SSL method that is known to always work in any similar types of environments.

From the results, it is evident that SSL can be favorable over SL in scenarios where the amount of labeled data is minimal. The LS methods performed better than the RF model in predictive accuracy and AUC on almost all data sets, especially in the transductive setting. In the inductive setting of scenario 1, the RF model achieved better AUC using only $L$ for all sizes of $L$. Comparing the LS methods and the two kernels, the LS method with a KNN kernel would generally outperform the other. It was clear that all methods benefited greatly by increasing the labeled data set. Using a sample-from-clusters approach also increased their performance. In the transductive setting, LS-RBF and RF could, in most cases, only reach the potential of LS-KNN when $L$ contained more than 250 labeled data points.

8 Related Work

Using semi-supervised learning in QSAR modeling to incorporate vast amounts of available unlabeled data with the typically small amount of labeled data is not new. One study examined the performance of four different SSL methods, including transductive support vector machines (TSVM) and Label Spreading (referred to as LLGC), on three different data sets, comparing them against commonly used SL algorithms. It was concluded that, although SSL can be used to achieve better predictive accuracy over SL when a small amount of labeled data is used, the improvements depended heavily on the data set and the method used. In the end, no claim that SSL would consistently outperform or even perform just as well as SL could be made.

Another study proposed a semi-supervised learning framework that in-
volves deep learning (DSSL) when performing drug discovery using virtual screening (VS). VS is a process where libraries containing ligands are searched to identify structures that will most likely bind to a drug target. The study utilizes a SSL method known as pre-training using stacked autoencoders (SAE). Pre-training with SAE takes advantage of unlabeled data to guide the decision boundary towards interesting regions before being trained in a supervised setting using deep learning on the available labeled data. They found that DSSL outperformed other algorithms in terms of overall classification accuracy.

Active Learning (AL) is similar to SSL in that it aims to lessen the dependency on labeled data. This is done by incorporating an oracle (expert in the field) that conducts the labeling process and deploying a querying strategy to determine what subset of the available unlabeled data to label that would maximize the performance gained by any underlying machine learning model. One report highlights the use of AL in a similar setting to this thesis, namely screening compounds for binding activity towards a target molecule. Various querying strategies were studied that would select batches to be labeled while minimizing the number of biochemical laboratory experiments needed. It was shown that a querying strategy based on the maximum margin hyperplane generated by support vector machines outperformed a random selection strategy and a strategy that would select molecules to label that were closest in distance to molecules that possess the binding activity.

Another study explored the effectiveness of AL in QSAR modeling using pool-based sampling on the same data sets used in this thesis. Two querying strategies were primarily examined, namely uncertainty sampling and Query-By-Committee (QBC), with an initial labeled data set of 50, 100, 200, and 500 compounds. Different batch sizes for each query were also investigated. It was shown that uncertainty sampling using a RF base classifier was, on average, slightly better than QBC, which used an ensemble of models consisting of a RF model and a KNN model. However, QBC was shown to outperform uncertainty sampling for the initial queries, suggesting that the ensemble method has better predictive performance than the single RF model before applying AL to the problem. A comparison of these querying strategies was made to a random sampling control. When querying in batches of size ten, the different strategies showed potential by performing better than the random control. Increasing the batch size decreases the model efficiency by performing worse than random sampling. It was concluded that AL has the potential to reduce the number of experiments needed. In future work, it was suggested that an increase in the performance of QBC could be achieved by performing hyperparameter tuning on the KNN classifier. However, more research on how to improve the querying strategies is necessary.

9 Conclusion

In this thesis, the effectiveness of graph-based semi-supervised learning using a Label Spreading method was compared to a Random Forest model in building
models that predict the presence of some biochemical activity in molecules under different experimental conditions. These conditions included a transductive and inductive setting, and the initial labeled data set being either randomly sampled or sampled from generated clusters using the k-Means algorithm. Each experiment was conducted using different amounts of labeled data relative to the unlabeled data. It was concluded that the LS method using a k-Nearest Neighbors kernel was favorable in most settings when the amounts of labeled data were low. However, it was also clear that the LS methods did not always surpass the performance of the RF model.

Sampling the labeled data after applying a clustering method improved the predictive capabilities of all methods. Clustering is thus a recommended approach to sampling the initial labeled data set when conducting similar modeling experiments that incorporate SSL.

In conclusion, this thesis is evidence that using SSL can be beneficial in scenarios where the labeled data is scarce and the unlabeled data is abundant. However, the improvements are heavily dependent on the underlying data since all applied methods seemed to experience a high variance in performance when comparing the mean performance across all data sets. Using Label Spreading as a generalized method for any similar type of data could not be decided as there is no one-size-fits-all method.

Future Work

In future work, a thorough investigation of different molecular descriptors that could potentially improve the predictive capabilities of the applied methods would be interesting to conduct since feature selection was not performed for this thesis. The exclusive use of Morgan fingerprints as input features was decided by the thesis supervisors. However, feature selection is known to improve performance in similar modeling scenarios.

Section mentions that molecular fingerprints often use the Tanimoto similarity equation when measuring the similarity between two fingerprints. The kernels used by the LS methods incorporated the Euclidean distance when it could be more reasonable to use the Tanimoto similarity instead. The creation and use of kernels based on the Tanimoto similarity equation could thus be a future endeavor to explore.

Since only one semi-supervised and supervised method was investigated, it would be valuable to research how well other SSL and SL methods fare against the Label Spreading method and the Random Forest model in QSAR modeling. It would also be beneficial to conduct experiments on data sets that have not been artificially balanced.

It was discussed with the research group that in a real-world scenario, only a tiny fraction of the acquired data would have the sought-after biochemical activity. In these scenarios, the positive predictions are deemed more important than the negative predictions. This scenario could be simulated in future work by applying the methods to data sets with a significant portion of the positive labeled data removed. However, this invalidates the use of accuracy since any
classifier that predicts the negative class for all molecules could still achieve a high accuracy score. There are more appropriate metrics used when estimating the performance of methods used on imbalanced data sets such as the $F_1$-score, $F_2$-score and the area under the precision-recall curve [13].

The combination of SSL and Active Learning (AL) has been studied [14, 9] before, where graph-based methods were paired up with different AL querying strategies, such as entropy reduction uncertainty sampling and estimated classification error reduction, respectively. Both studies concluded that the combination of the two showed promising results, making it interesting to investigate.

References


A  Selecting Hyperparameters: Label Spreading

A.1 Scenario 1

Figure 13: Heatmaps depict the average accuracy of applying the Label Spreading model to data sets A, B, and C using each combination of hyperparameters shown in Section 5.3.1 and a labeled data set of size 50.

Figure 14: Heatmaps depict the average accuracy of applying the Label Spreading model to data sets A, B, and C using each combination of hyperparameters shown in Section 5.3.1 and a labeled data set of size 100.
Figure 15: Heatmaps depict the average accuracy of applying the Label Spreading model to data sets A, B, and C using each combination of hyperparameters shown in Section 5.3.1 and a labeled data set of size 200.

Figure 16: Heatmaps depict the average accuracy of applying the Label Spreading model to data sets A, B, and C using each combination of hyperparameters shown in Section 5.3.1 and a labeled data set of size 500.

Figure 17: Heatmaps depict the average accuracy of applying the Label Spreading model to data sets A, B, and C using each combination of hyperparameters shown in Section 5.3.1 and a labeled data set of size 1000.
A.2 Scenario 2

Figure 18: Heatmaps depict the average accuracy of applying the Label Spreading model to data sets A, B, and C using each combination of hyperparameters shown in Section 5.3.1 and a labeled data set of size 50.

Figure 19: Heatmaps depict the average accuracy of applying the Label Spreading model to data sets A, B, and C using each combination of hyperparameters shown in Section 5.3.1 and a labeled data set of size 100.
Figure 20: Heatmaps depict the average accuracy of applying the Label Spreading model to data sets A, B, and C using each combination of hyperparameters shown in Section 5.3.1 and a labeled data set of size 200.

Figure 21: Heatmaps depict the average accuracy of applying the Label Spreading model to data sets A, B, and C using each combination of hyperparameters shown in Section 5.3.1 and a labeled data set of size 500.

Figure 22: Heatmaps depict the average accuracy of applying the Label Spreading model to data sets A, B, and C using each combination of hyperparameters shown in Section 5.3.1 and a labeled data set of size 1000.
B Plotting Performance Metrics: Label Spreading

B.1 Scenario 1

B.1.1 Transductive Setting

Figure 23: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set D using 10-fold random sub-sampling.

(a) Accuracy.  
(b) AUC.

Figure 24: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set E using 10-fold random sub-sampling.

(a) Accuracy.  
(b) AUC.
Figure 25: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set F using 10-fold random sub-sampling.

Figure 26: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set G using 10-fold random sub-sampling.

Figure 27: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set H using 10-fold random sub-sampling.
Figure 28: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set I using 10-fold random sub-sampling.

Figure 29: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set J using 10-fold random sub-sampling.
B.1.2 Inductive Setting

Figure 30: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set D using 10-fold cross-validation and randomly sampled labeled data sets.

Figure 31: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set E using 10-fold cross-validation and randomly sampled labeled data sets.
Figure 32: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set F using 10-fold cross-validation and randomly sampled labeled data sets.

Figure 33: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set G using 10-fold cross-validation and randomly sampled labeled data sets.
Figure 34: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set H using 10-fold cross-validation and randomly sampled labeled data sets.

Figure 35: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set I using 10-fold cross-validation and randomly sampled labeled data sets.
B.2 Scenario 2

B.2.1 Transductive Setting

Figure 36: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set J using 10-fold cross-validation and randomly sampled labeled data sets.

Figure 37: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set D using a k-Means clustering algorithm to sample the labeled data sets.
Figure 38: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set E using a k-Means clustering algorithm to sample the labeled data sets.

Figure 39: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set F using a k-Means clustering algorithm to sample the labeled data sets.
Figure 40: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set G using a k-Means clustering algorithm to sample the labeled data sets.

Figure 41: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set H using a k-Means clustering algorithm to sample the labeled data sets.
Figure 42: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set I using a k-Means clustering algorithm to sample the labeled data sets.

Figure 43: Mean classification accuracy and AUC of RF, LS-KNN, and LS-RBF when applied to data set J using a k-Means clustering algorithm to sample the labeled data sets.
B.2.2 Inductive Setting

Figure 44: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set D using 10-fold cross-validation and a k-Means clustering algorithm to sample the labeled data sets.

Figure 45: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set E using 10-fold cross-validation and a k-Means clustering algorithm to sample the labeled data sets.
Figure 46: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set F using 10-fold cross-validation and a k-Means clustering algorithm to sample the labeled data sets.

Figure 47: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set G using 10-fold cross-validation and a k-Means clustering algorithm to sample the labeled data sets.
Figure 48: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set H using 10-fold cross-validation and a k-Means clustering algorithm to sample the labeled data sets.

Figure 49: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set I using 10-fold cross-validation and a k-Means clustering algorithm to sample the labeled data sets.
Figure 50: Mean classification accuracy and AUC of RF, RF-KNN, and RF-RBF when applied to data set J using 10-fold cross-validation and a k-Means clustering algorithm to sample the labeled data sets.