CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Predicting multiple chemical contexts using multi-label classification and predictors

Master's thesis in Computer science and engineering

Gustav Lahti

Agnes Mårdh

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

Predicting multiple chemical contexts using multi-label classification and predictors

Gustav Lahti and Agnes Mårdh

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Predicting multiple chemical contexts using multi-label classification and predictors
Gustav Lahti and Agnes Mårdh

© Gustav Lahti and Agnes Mårdh, 2021.

Supervisor: Simon Olsson, Department of Computer Science and Engineering
Advisor: Samuel Genheden, AstraZeneca
Advisor: Thierry Kogej, AstraZeneca
Examiner: Devdatt Dubhashi, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2021

Predicting multiple chemical contexts using multi-label classification and predictors
Gustav Lahti and Agnes Mårdh
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Drug discovery is a time and resource intensive process. Machine learning is one way of speeding up the process. One important task is to choose suitable conditions – solvents, catalysts etc. – for a reaction to optimize the amount of product from the reaction. The purpose of this thesis was to investigate ways to improve condition prediction. In this thesis the condition prediction is limited to *chemical contexts*, or sets of conditions, and the reaction class Buchwald-Hartwig that is common in drug discovery.

First, we evaluate two models using two approaches for multi-label classification to predict several possible chemical contexts for a reaction. We evaluate both a neural network and a binary relevance model. Second, we present a model for condition prediction of a chemical library used for parallel synthesis. Last, Venn-ABERS predictors were added on top of these models to evaluate the impact of model calibration on these tasks. However, calibrating the scores with Venn-ABERS predictors did not improve our results.

All models show potential in improving condition prediction. We consider both models for the multi-label classification task to be well-performing. Also, both models performed better than the naive models. The novel model for condition prediction for chemical libraries also showed good results which out-performed naive classifiers.

Keywords: reaction prediction, condition prediction, cheminformatics, machine learning, drug development, multi-label classification, predictor, model calibration

Acknowledgements

We want to express our gratitude to our supervisor Simon Olsson for all his support and guidance in this project. We would also like to thank our advisors on AstraZeneca, Thierry Kogej and Samuel Genheden, for their help and for making their dataset and research available to us. Last, thanks also to all who have helped us with our thesis and given feedback.

Gustav Lahti and Agnes Mårdh, Gothenburg, June 2021

Contents

List of Figures

List of Tables

1

Introduction

Drug discovery is a time and resource intensive process [1]. Every step currently requires substantial human intervention. Chemists use chemical reactions to synthesize the target drug molecule. In a reaction, reactants react with each other and create one or several product molecules. The yield is a measure of a reaction's success or effectiveness based on the quantity of moles that could be obtained in the product in relation to the reactants. To increase a reaction's yield, conditions such as catalysts, solvents, and other reagents can be added to the reaction. Let us call a *chemical context* the set of conditions that are added to the reaction. This step in the drug discovery process is called synthesis, where chemists combine experimental conditions to build a chemical context which maximizes the yield of a target compound. This task is very complex requiring experienced chemists [2]. Furthermore, small differences in the context can have a large impact on the outcome [3]. There are several challenges in the traditional approach, a chemist's knowledge may be limited or biased towards certain conditions. In addition, two similar reactions might not work with the same chemical context [3].

A chemical library is a collection of similar compounds and their known properties. To obtain a chemical reaction, multiple reactions are run in parallel. A number of M reactants are combined with N other reactants. The reactions result in $M \times N$ sets of product molecules that can be added to the library at the same time. $M \times N$ is typically small compared to the space of possible molecules and reactions, resource use must thus be effective. One can optimize the chemical context to get as high yield of product as possible. To speed up the parallel synthesis one can apply the same context to all reactions.

There have been several attempts at reducing the required manual labor with the use of computers. Many of the earlier approaches are rule-based and have been helpful, but the rules impose limitations to what can be predicted and suggested by the system. Machine learning has been used since the 90s in drug discovery [4], one area of interest is synthesis prediction tasks [4, 5]. Nair *et al.* [6] believes that experienced chemists will still be needed but that machine learning tools will become lab assistants working with chemists.

One critical aspect of synthesis prediction receiving little attention is predicting the optimal chemical context to get the target compound. In this thesis, we will describe our approaches to improve condition prediction. First, we present two different multi-label models that give multiple plausible chemical contexts for a given reaction. We also evaluate the multi-label models and compare them to a

current single-label model for this synthesis prediction task. Second, we describe a model that can take a set of reactions and return the optimal chemical context for this set. This optimal context could then be used for creating a chemical library and running parallel synthesis. Last, we evaluate calibration of these models. We used Venn-ABERS predictors, which calibrates the score from a classifier to better represent probabilities. These calibrated probabilities are more useful than just class membership as it can provide information about likelihood when making decisions. Calibrating the scores and quantifying the uncertainty of predictions is crucial both in drug discovery and medical applications [7–9].

1.1 Related work

Reaction prediction is a complex yet important task. Coley *et al.* [2] implemented a model which predicts the major product of a reaction by ranking a list of candidates. Their model could predict the major product with a top-1 accuracy of 72% and top-5 of 91%. Top-k accuracy means that the result can be found among the k highest predicted. Schwaller *et al.* [10] developed a model called molecular transformer for predicting the product of a reaction. It outperformed all similar models in the literature with a top-1 accuracy above 90%.

Segler and Waller [11] improved the traditional rule-based expert systems by using machine learning together with data-extracted reaction templates for reaction prediction. The reaction templates describes the reactions and is part of the rule-based system. Their model shared some properties with a traditional rule-based system but all work with the rules were made with machines rather than humans. One of the improvements with deep learning was that resolving conflicts was learned rather than having to be encoded by hand. They got a top-10 accuracy of 97% for reaction prediction. However, like any rule-based system, the model cannot predict anything outside the rule space.

Schwaller *et al.* [12] have developed a model with a regression layer that predicts the yield of the reaction. On one kind of dataset they achieved good results (R^2 score of 0.96) but using another dataset with patent data the model could not learn as the yield data was noisy. They state that knowing the yield for a reaction is important as a low yield reaction in the beginning of a reaction chain can affect all later reactions negatively. Schwaller *et al.* explains that a large factor of the yield value is the conditions used.

Finding a suitable chemical context for a given chemical reaction poses a separate but related problem. Gao *et al.* [3] developed a neural network that returns the top-k suggestions for catalysts, solvents, and temperatures for a given organic reaction. Their model is able to predict conditions with an exact match to the recorded context (one catalyst, one solvent, and one reagent) with a top-10 accuracy of 66% and a top-3 accuracy of 57%. By using different similarity thresholds and methods to find close matches instead of exact matches, they managed to improve the top-10 accuracy to 70% and the top-3 accuracy to 60%. Fu *et al.* [13] used a deep learning model to determine the best catalysts, catalyst loader and temperature for a given

reaction. They achieved high yields (73% to 97%) experimentally for new sets of reactants by using the context suggested by the model.

AstraZeneca has previously made a condition prediction model (unpublished work). It is a feedforward network that takes one reaction as the input and predicts one suitable chemical context for this reaction. AstraZeneca's model predicted the ground truth context with a top-1 accuracy of about 72% and a top-3 accuracy of 87% for a limited class of chemical reactions called the Buchwald-Hartwig amination or coupling.

Maser *et al.* [14] have developed two different kinds of models (relational graph convoluted network and gradient-boosting machine) for multi-label classification of conditions. The models used binary relevance method and classifier chaining, which are both common in multi-label classification tasks. The approach presented by Maser *et al.* could effectively solve the multi-label classification task and showed promising results. However, the learning task is still highly unexplored and would benefit from further research.

Model calibration is important especially in drug development, where synthesizing a drug molecule takes place over many consecutive reactions. The previously mentioned molecular transformer for reaction prediction also predicts the uncertainty of its reactions [10]. The riskier steps could thus – when possible – be placed in the beginning of the synthesis so that a failure shows up fast. A traditional scoring classifier gives a score as the prediction, which is turned into a class membership based on a threshold for the score. A classifier can show good discrimination, that is, the predicted class membership is often correct. However, the scores can still be unreliable when it comes to likelihood of class membership. Even though calibration is an important issue, calibration is not evaluated to the same extent as discrimination in the literature [8]. Predictors, which is one method of model calibration, has not been applied to condition prediction, but in other areas of drug design where they have shown great results [7, 10, 15, 16].

1.2 Purpose

Condition prediction is an area that has received little attention [4, 5] and the research that has been made has focused on predicting one chemical context [3, 13]. There has been only a few applications of predictors in drug development and none in condition prediction. Moreover, suggesting one chemical context suitable for many reactions has, to the best of our knowledge, not been done.

The purpose of this project is to fill this gap by exploring different approaches to improve condition prediction. We hypothesize that condition prediction can be improved to be more useful to chemists with the use of multi-label classification. Multi-label classification could add more useful alternatives for chemists to investigate. We also hypothesize that a model suggesting one chemical context that works well for many reactions would improve efficiency in a lab setting as several reactions could be performed simultaneously.

The two different models developed can thus be seen as two separate tools in a

lab setting. The first model can be used to find a suitable context to optimize one reaction. The second model instead tries to find the best context for several reactions. The second model either builds on the first model or solves the task in an alternative approach.

Our last hypothesis is that adding predictors on top of these models can provide useful information about the uncertainty which could guide chemists in their choices of which chemical context to use.

1.3 Limitations

Our focus and aim for the project is to improve condition prediction by developing an approach for prediction of many chemical contexts for one reaction and an approach for finding one context that works for many reactions. To reach our goal we have to impose some limitations on the project's scope and our process. The representation form for the molecules and conditions will be the same as AstraZeneca's model uses and alternatives will not be evaluated. We will focus on getting the model to work with the reaction class Buchwald-Hartwig amination or coupling. We have chosen this reaction class since it is commonly used in drug development, and AstraZeneca currently has a model used in production which we can benchmark against. We therefore do not anticipate our model to be applicable for all types of chemical reactions but instead a specialized class of reactions. With this limited scope, we hope to outperform other existing methods within this important reaction-class. Regarding the condition prediction, we will not predict temperatures, concentration, pressure, etc. but only a set of molecules. We ignore the temperature since it is not included in the current model, and because the temperature data is either sparse or nonexistent in our dataset. The other possible conditions are not included because they are not included in the dataset. We will not test and evaluate several predictors but rather only use the probabilistic Venn-ABERS predictor. Because, calibrated probabilities as output seem to be more useful and intuitive in a lab setting than other approaches. Also, Venn-ABERS predictors have shown good results in similar applications. We will only compare two general approaches for the multi-label classification task. First, we will adapt the current AstraZeneca model to multi-label classification, so we can compare the current model with ours. Second, we will use binary relevance which is one of the most common approaches in multi-label classification tasks [17, 18].

1.4 Contributions

In this thesis we make the following contributions to condition prediction.

- Develop and evaluate a multi-label classification model
- Develop and evaluate a novel model predicting one context for a chemical library
- Evaluate Venn-ABERS predictors added on top of these models

1.5 Outline

The rest of the report is structured as follows. Chapter 2 gives an explanation of concepts, such as multi-label classification and predictors, that are important to know to understand the rest of the thesis. Chapter 3 describes the development and evaluation of the models and predictors. Chapter 4 contains an analysis of the dataset as well as a description of the pre-processing of the data. In Chapter 5 we present the results for the multi-label classification condition prediction task. In chapter 6 we present the results for the condition prediction for chemical libraries. In Chapter 7 we discuss our results and suggest future work. Last, Chapter 8 concludes our thesis.

2
Background

In this chapter we will introduce some methods for multi-label classification, predictors and different representations for molecules and reactions. We will also show what research has been done in these areas that is relevant to our project.

2.1 Multi-label classification

A multi-label classification task is similar to a traditional classification task, with the difference that each data point can have more than one label. It could for example be multiple genres for a book or movie. In our case the model will return multiple chemical contexts for a single chemical reaction, as it is possible for a reaction to happen with several different contexts. Multi-label classification can be achieved either with problem transformation methods or algorithm adaptation methods [17].

Multi-label classification models need other metrics than single-label classification models. This is a direct consequence of the output potentially containing multiple positive results, as we can get a partially correct prediction.

2.1.1 Problem transformation methods

Problem transformation methods reformulates the multi-label classification task into one or more simpler tasks. These methods are also called algorithm independent methods as they can be used with any traditional classifier.

The most popular method is binary relevance, also knows as one vs rest, where the problem is transformed to multiple binary classification tasks [17]. For every label, a binary classifier is trained to determine if a sample belongs to that label or not. Thus, the number of binary classifier increases linearly with the number of labels. One advantage this method has is that each classifier will be specialized in learning the pattern of only one label. However, because the classifiers are trained independently, no dependency between labels can be learned. Classifier chaining is a method which is suitable if the labels are dependent [14]. Classifier chaining builds on binary relevance by adding an order to the classifiers where the output of the first label is added to the second classifier's input and so on. This method is thus more complex compared to binary relevance as the optimal order is a learned parameter.

Another approach, called label-powerset [18], is to join the labels in the dataset such

that each data point has a single label corresponding to one or more labels in the old dataset [17]. Thus, the problem has been transformed to a traditional single-label multi-class classification problem. A disadvantage with this approach is that there could be many labels that only have few examples each. There is an extension to the Label-Powerset method called RAndom K-labELsets or RAKEL, which takes class correlation into account and also avoids the drawback of power sets with very few examples [18].

2.1.2 Algorithm adaptation methods

Algorithm adaptation methods, in comparison to problem transformation methods, use models that have been extended to be able to output multiple labels more directly. Several algorithms have been adapted where some of them use similar techniques as the problem transformation methods but for a specific algorithm [17].

Clare and King [19] used a decision tree algorithm and modified the entropy calculation by having it depend on the relative frequencies of the classes to make their model work for multi-label classification.

Zhang *et al.* [20] introduced a modification of the k-Nearest Neighbor (kNN) algorithm for multi-label classification called ML-kNN. In the traditional kNN, the class of an unknown sample is the most common class amongst the k nearest neighbors. ML-kNN uses the kNN algorithm for each label independently. For every label the algorithm looks at all of of the k nearest neighbors and a neighbor is considered positive if it contains that label and negative if it does not contain that label. It can seem similar to the binary relevance method, but the difference is that ML-kNN is dependent on the kNN algorithm and that ML-kNN uses Bayesian statistical inference.

A feedforward network can be adapted by setting the last layer's activation function to `sigmoid` instead of `softmax` for example and having an appropriate loss function. The `sigmoid` function produces a score between zero and one independently for all classes. `Softmax`, on the other hand, calculates the scores so that they all add up to one. `Sigmoid` is thus the suitable option to be able to predict multiple labels independently.

2.2 Model calibration

The score from classifiers does not give the exact probability but can rather be seen as an indication. However, a model's scores can be calibrated to better represent probabilities. Predictors are one method of model calibration. They are added on top of models to calibrate this score so that it can more accurately show the probability of some label. The calibration can be done by feeding the predictor a subset of the data, the calibration set, which is separate from the training and validation sets. Conformal and Venn-ABERS predictors are two alternatives that have been used in drug development [7, 15, 16]. The main difference is that conformal predictors have a fixed error rate of including the true label in the output set and Venn-ABERS predictors give calibrated probabilities.

2.2.1 Conformal predictors

A conformal predictor takes a confidence level from the user and returns the confidence interval for a specific model in the form of a set of outputs, called prediction regions [21].

A classifier, h, trains on the training set Z_t. By using h, a non-conformity function, $f(z_i)$, is applied to the calibration set, Z_c, to get the non-conformity scores, α_i (see Equation 2.1). The non-conformity function measures how close the prediction was to the true label. Statistical inference from the non-conformity scores gives the prediction regions for new examples. The statistical inference procedure is based on the ideas of p-values and hypothesis testing.

$$\alpha_i = f(z_i) = \Delta[y_i, h(x_i)] \tag{2.1}$$

Here Δ is an anonymous function measuring similarity between predicted and true label.

One example could be an image classification of fruits where we want a confidence level of 95%. For one of the fruits the prediction region could then be {*strawberry*, *raspberry*, *apple*}, where the true label is *strawberry*. Three potential labels are given, which suggests that the model is not too certain of the true label. Overall, the true label will be included in the prediction region with the defined confidence level, in this case 95%. The uncertainty is given by the size of the prediction region, where a smaller region means that the model is more certain.

2.2.2 Venn-ABERS predictors and isotonic regression

Vovk and Petej [22] introduced Venn-ABERS predictors which are based on Venn predictors. They also prove that Venn-ABERS share important properties from Venn predictors. Venn predictors are perfectly calibrated, which is an important property stating that the probabilities are matched by observed frequencies. For example, if the calibrated probability is 0.1 then 10% of the predictions with this score should be true positives. Venn predictors are multiprobabilistic which means that the output is a set of probabilities (p_0, p_1) that can be seen as the lower and upper bound. This set of probabilities is most useful when $p_0 \approx p_1$. However, Vovk and Petej emphasize the importance of having a set of probabilities as the difference between p_0 and p_1 provides the level of uncertainty.

Venn-ABERS predictors take inspiration from a calibration method by Zadrozny and Elkan [23] which uses isotonic regression. One difference is that Zadrozny and Elkan's approach has shown to give poor calibration in some cases where Venn-ABERS predictors performs better. Applying isotonic regression means that a non-decreasing function is fit to the scores. Since it can be any non-decreasing function there is a large flexibility and complex patterns can be learned. Venn-ABERS predictors uses isotonic regression and fits a non-decreasing function to the scoring function of the classifier to calibrate the scores.

2.2.3 Multi-label predictors

There has been some research on adding predictors to multi-label classification task, but to our knowledge only on conformal predictors.

Conformal predictors have been used in a multi-label setting with good results [24–27]. Papadopoulus [24] states that knowing the uncertainty of prediction is even more important in multi-label classification as the uncertainty is much greater. For the multi-label classification setting, a probability is added to each output set from the conformal predictor indicating the probability of those labels being the true labels. The study found that adding conformal predictors to multi-label classification tasks gives better performance and the information about uncertainty proved useful [24]. In a comparative study of three multi-label approaches (instance reproduction, binary relevance and power set) it was found that they all can be calibrated well but that binary relevance has a higher prediction efficiency and lower computational cost than the other two [26]. Using problem transformation, random forest and conformal predictors to solve a multi-label classification task, Wang *et al.* [27] were able to create a medical diagnosis tool that performed well with confidence levels ranging between 80% and 100%.

2.3 Chemistry

A Buchwald-Hartwig amination or coupling is a common reaction type in drug design. It is often used to connect a molecule containing aromatic halogen with a molecule containing nitrogen, often with the help of a palladium catalyst [28]. This reaction, however, is under research to try to find more efficient contexts.

A reaction could have have many possible variations, which is the same reactants and product but a different chemical context. All of these cannot be experimentally tested. Additionally, the dataset contains only a fraction of those tested. Knowing this, we cannot simply say that if we have a data point for a variation with a high enough yield we have a working variation, and otherwise not. A false positive will occur when the predicted context differs from the ground-truth (contexts that have been used and can be found in the dataset). These false positives might not necessarily be bad, it is possible that the model has simply predicted a working variation that has not been tested experimentally.

2.4 Representation of molecules and reactions

In *cheminformatics* information and computer science is used to solve chemistry problems, with applications in drug discovery and development. There are different ways of representing a molecule or reaction in cheminformatics. Some representations could be more suitable as input to computer models while other representations could be easier to understand for humans. No representation is perfect for cheminformatics and some information loss is to be expected. An example of different representations for the same molecule is illustrated in Figure 2.1.

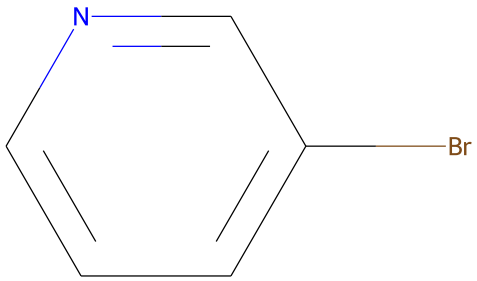
Molecules are often represented as 2D graphs or 2D drawings. This works well for humans, but it is not as easy to understand for computers, and other representations have been developed, many of which are based on graphs. Recently, with the graph neural networks, graphs have started to be used as input for different machine learning tasks [29].

A common molecular representation for machine learning applications are fingerprints. Fingerprints capture the chemical structure well and translate to a representation that can be understood by a computer [29, 30]. One kind of fingerprint is Extended Connectivity Fingerprints (ECFP), which are circular fingerprints calculated with a variant of the Morgan algorithm [31]. ECFP are created by first initializing an identifier for each atom in a molecule, for example the atom number. Then, for each atom the algorithm looks at its neighborhood and stores the information in an array. The array becomes the new identifying number for the atom with the use of a hashing function. Then, duplicate identifiers are removed. This process is repeated with an increasingly large radius, until a certain defined radius has been reached. Another type of fingerprint is the `RDKit` fingerprint [32]. In contrast to ECFP, the `RDKit` fingerprint is topological. The `RDKit` fingerprint is calculated by first finding all sub-graphs of the molecule that are within a defined range of sizes. These sub-graphs are hashed and used to set one (or more) bits in the fingerprint bit vector.

Line notations, based on strings, are also common and has the advantage of being human readable. The most popular line notation is the Simplified Molecular Input Line Entry System (SMILES) [29]. The SMILES representation is unambiguous and is created by traversing the molecule. SMILES strings are however not unique for a molecule, which means that it can relatively easily be used for data augmentation, but also that they cannot as easily be used as identifiers.

Another common line notation is the International Chemical Identifier (InChI). An InChI is a representation of a molecule that consists of several different layers that corresponds to some property of the molecule. These layers include (but are not limited to) skeletal connections, or the main chain of atoms that the rest of the molecules branch off of, and hydrogen layers, which describes where the hydrogen atoms are [34]. The structure of a molecule can be extracted from an InChI, but in return the length might become too long to easily search for among a list of keys. InChIKeys on the other hand are 27 characters long hashes of InChIs, which results in InChIKeys being more convenient to have as an identifier, but the molecular structures cannot be extracted from them algorithmically.

There are also different representations for reactions, see Figure 2.2 for an example. The context can be included or excluded from the representation. An ECFP for a reaction can be constructed by subtracting the reactant fingerprints from the fingerprint for the corresponding product [29]. Reaction SMILES represent a reaction by concatenating the SMILES for reactants, reagents, and product, as can be seen in Figure 2.2b.



(a) **2D drawing**

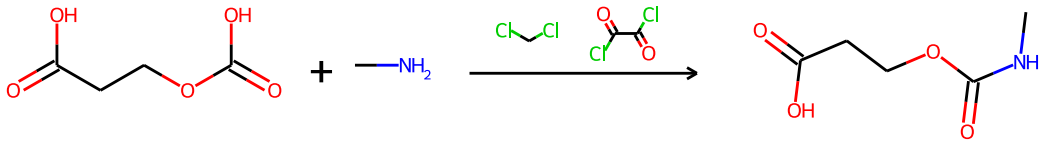
`Brc1cccnc1`

(b) **SMILES string**

`10011100001001000111011010100001`

(c) **Bit vector of Morgan fingerprint.** Note that fingerprints usually are much longer, for example 1024 bits, but to illustrate it only 32 bits are used.

Figure 2.1: Three different representations of the same molecule. Created with the `RDKit` tool [33].



(a) **2D drawing**. The reactants are to the left of the arrow seperated with a plus sign. The chemical context is on top of the arrow. The product is to the right of the arrow.

`C(COC(=O)O)C(=O)O.NC>C(Cl)Cl.C(=O)(C(=O)Cl)Cl>CNC(=O)OCCC(=O)O`

(b) **SMILES string** The molecules are separated with a dot. Between the two '>' is the chemical context or reagents. To the left of this is the reactant and to the right is the product.

`1 0 0 0 -1 -1 0 -1 1 0 0 0 0 0 0 1 0 0 -1 -2 0 0 1 0 0 1 0 0 0 0 1 0`

(c) **Vector of a Morgan fingerprint**. Note that fingerprints usually are much longer, for example 1024 bits, but to illustrate it only 32 bits are used.

Figure 2.2: Three different representations of the same reaction. Created with the `RDKit` tool [33].

3
Methods

In this chapter we describe our approach to solving the two condition prediction tasks: multi-label condition prediction and condition prediction for chemical libraries. First, we present our method for analyzing the dataset. We also motivate our choices of methods and explain our different models for multi-label classification of chemical contexts. Finally, we describe our approach for condition prediction for chemical libraries.

3.1 Dataset analysis

An analysis of the dataset was performed to provide insight to the data and task at hand. We looked at the distribution of contexts and how many variations there were for each reaction. Additionally, we evaluated if there was any correlation between different contexts to help our choice of approach to solve the multi-label classification task.

The analysis was made with a multi-label perspective as the data was going to be used in a multi-label classification task. First, the label cardinality and density was calculated to measure how multi-label the dataset was [17]. Second, the imbalance of labels was measured, as imbalance is a common problem of multi-label datasets [35].

Label cardinality (LC) quantifies how many labels are used per example of the dataset on average. Label density quantifies how often the labels are used on average. Even though the two equations are related, $\text{LabelCardinality}(D) = |L| \times \text{LabelDensity}(D)$, two datasets with the same cardinality score can have different densities. These measurements can be useful when choosing the right method for solving a multi-label problem as well as evaluating how well a model replicates the distribution of labels.

$$\text{LabelCardinality}(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}|Y_i| \tag{3.1}$$

$$\text{LabelDensity}(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i|}{|L|} \tag{3.2}$$

For these equations, $|Y_i|$ is the number of true labels for data point i, $|D|$ is the size of the dataset, and $|L|$ is the number of unique labels.

Chartre *et al.* [35] proposed methods for measuring imbalance in multi-label datasets as it cannot be measured using the same methods as for single-label datasets, since there can be several labels with high and low presence. The first measurement is the imbalance ratio per label, see Equation 3.3. The ratio between the majority label and all the other labels is calculated, where a higher value represents a higher imbalance. The second measurement is the mean of the imbalance ratios for all labels, see Equation 3.4. It measures how many more samples there are with the majority label compared to samples with minority labels on average. The last measurement is the coefficient of variance of the imbalance ratios, see Equation 3.5. Both variance and mean are calculated since two different multi-label datasets can have the same mean but different variance or vice versa.

$$\text{IRLbl(y)} = \frac{\text{argmax}_{y'=Y_1}^{Y_{|Y|}}(\sum_{i=1}^{|D|} h(y', Y_i))}{\sum_{y'=Y_1}^{|D|} h(y', Y_i)}, h(y, Y_i) = \begin{cases} 1, y \in Y_i \\ 0, y \notin Y_i \end{cases} \tag{3.3}$$

$$\text{meanIR} = \frac{1}{|Y|} \sum_{y=Y_1}^{Y_{|Y|}} (\text{IRLbl(y)}) \tag{3.4}$$

$$\text{CVIR} = \frac{\text{IRLbl}\sigma}{\text{meanIR}}, \text{IRLbl}\sigma = \sqrt{\sum_{y=Y_1}^{Y_{|Y|}} \frac{(\text{IRLbl(y)} - \text{meanIR})^2}{|Y| - 1}} \tag{3.5}$$

Here D is the multi-label dataset and Y the set of labels. Y_i is the i-th label and $|Y|$ is the number of labels.

3.2 Features

Before the data was used to train a model the reactions needed to be represented in a way that a machine learning model can understand. Our chosen method was to use reaction fingerprints similar to those described by Patel *et al.* [36], David *et al.* [29], and Schneider *et al.* [37]. A reaction fingerprint is a way to represent the changes that take place during a reaction. The reaction fingerprints we used consist of a concatenation of an Extended Connectivity FingerPrint (ECFP) and an `RDKit` reaction fingerprints, see more about these fingerprints in Section 2.4. To obtain a reaction fingerprint, all of the reactant fingerprints were subtracted from the product fingerprint. This was done for both types of fingerprints. There was only one product for the reactions present in the dataset, which was the reason for the assumption of one product. The resulting reaction fingerprint was an integer vector that could be mapped onto many different reactions that undergo similar changes. Or in other words, the reaction fingerprints and reactions had a one-to-many relationship [36].

To decide what fingerprint, or combination of fingerprints, to use as well as the size, three combinations of fingerprint sizes were tested for both the algorithm adaptation and binary relevance models. In Table 3.1 we can see the results for the algorithm

Table 3.1: This table shows the LRAP, Jaccard accuracies, and label cardinalities for the algorithm adaptation model for various fingerprint sizes and combinations. The chosen fingerprint size and its scores are in bold.

ECFP size	RDKit-FP size	LRAP	Jaccard	true/predicted LC
512	**512**	**0.81**	**0.69**	**1.02/1.02**
1024	0	0.78	0.64	1.04/0.99
512	0	0.77	0.61	1.04/0.89

Table 3.2: This table shows the LRAP, Jaccard accuracies, and label cardinalities for the binary relevance model for various fingerprint sizes and combinations. The chosen fingerprint size and its scores are in bold.

ECFP size	RDKit-FP size	LRAP	Jaccard	true/predicted LC
512	**512**	**0.65**	**0.64**	**1.02/1.16**
1024	0	0.60	0.59	1.04/1.13
512	0	0.57	0.56	1.04/1.36

adaptation model. As can be seen, there was not a big difference, although a combination of 512 bits ECFP and 512 bits of RDKit fingerprint was slightly better than only using ECFP. In Table 3.2 we can see that this held for the binary relevance model as well, although here the performance increase was slightly larger. It is important to stress, however, that we got a slightly higher true label cardinality when using only ECFP. This suggests that there were more reactions that were represented by the same fingerprint. For this fingerprint we will possibly get more contexts, and therefore a higher label cardinality.

In Table 3.3 we can see that both of the fingerprints consisting only of ECFP had a lower number of reactions with only one context and more that have multiple contexts, which further suggests that the fingerprints using only ECFP are more general. The increased generality does not fully explain the difference in distribution of the number of contexts between the fingerprints. It explains why we saw an increase in reactions with two, three, or more contexts, but not the big decrease in reactions with one context. This decrease probably comes from similar reactions

Table 3.3: This table shows the distribution of the number of reactions that has a specific number of contexts for three combinations of fingerprint sizes.

Fingerprint sizes		Number of reactions with X contexts				
ECFP	**RDKit-fp**	**1**	**2**	**3**	**>3**	**>0**
512	512	4537	603	88	57	5285
1024	0	3411	603	137	162	4313
512	0	3411	603	137	162	4313

with the same context having the same fingerprint when only using ECFP but not when also using `RDKit` fingerprint.

3.3 Multi-label classification of chemical contexts

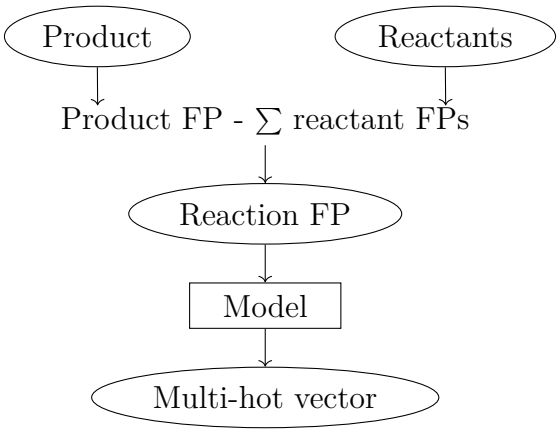


Figure 3.1: This diagram shows how we go from having a product and a set of reactants, to having a score vector describing what contexts the model estimates would work well for the given reaction.

We have developed and evaluated multi-label classification using two approaches. One of the models was also evaluated against the current AstraZeneca model.

Both of our models take a product and a set of reactants that have been turned into a reaction fingerprint as an input. The output from this model is a multi-hot vector, where each index corresponds to a specific context, or label, and a value close to one means that the model predicts that the corresponding context will work for the input reaction. This is meant to help chemists find multiple contexts that could work for a reaction. This workflow is described in Figure 3.1.

3.3.1 Single-label model

We have based one of our multi-label models on an already existing single-label model developed at AstraZeneca, and we also used this single-label model as a reference point to evaluate that multi-label model. Figure 3.2 visualizes the architecture of the neural network that makes up the single-label model. The single-label model is a feedforward network where the input is a concatenation of an ECFP and an `RDKit` fingerprint as integer vectors. The output is a one-hot vector where each bit corresponds to a chemical context. Only the context with the highest yield was kept for each reaction. This pre-processing step was done since the model is a single-label classification model, and thus any worse performing contexts could hinder the model from predicting the best context rather than than help it.

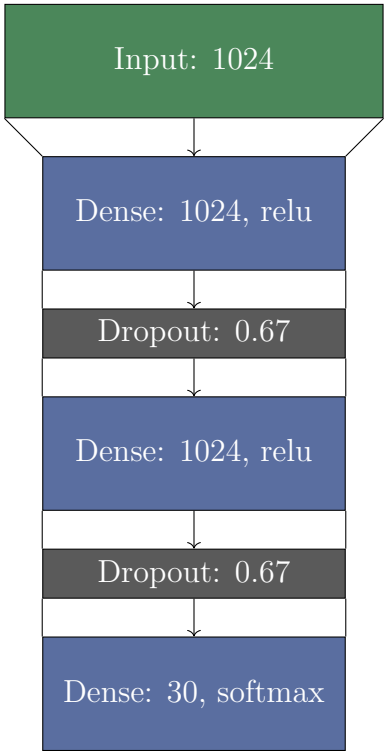


Figure 3.2: This diagram shows the structure of the single-label model, which is a feed-forward neural network with `softmax` as the final activation layer to get one predicted context.

3.3.2 Pre-processing

The data was pre-processed differently for the multi-label models than for the single-label model. For the multi-label models, all combinations of chemical contexts and reactions that appear were kept. Also, the output vector was a multi-hot vector representing several promising contexts, in contrast to the single-label model's one-hot vector. In the single-label model, yield was used to calculate a weight factor for the model to indicate a context's success rate. We did not investigate ways to apply weights to the multi-label models. We considered trying to predict the yield for each context to take the yield into consideration. This approach was however discarded before evaluating it thoroughly as the yield data turned out to be too noisy. Yield can be noisy because the same reaction and chemical context can be observed multiple times with differing yields, which means that unless a variation appears many times it would be difficult to find a good value for the yield. In addition, the problem would have transformed into a ranking or regression problem, which is more complex to handle and evaluate than a classification problem.

3.3.3 Algorithm adaptation

Our first approach at developing a multi-label classifier was using an algorithm adaptation model, see Figure 3.3 for the general structure of the neural network making up the algorithm adaptation model. The model is a feedforward network

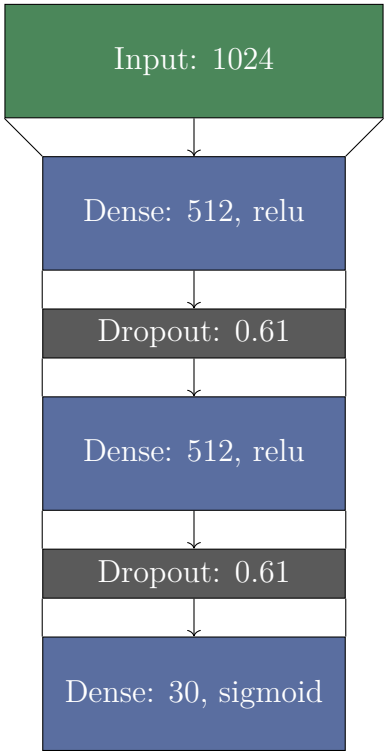


Figure 3.3: This diagram shows the structure of our algorithm adaptation model, which is a feed-forward neural network with `sigmoid` as the final activation layer to get more independent predictions.

using `TensorFlow`'s `Sequential` [1] [38] model. Since we are using fingerprints with a combined length of 1024 bits as input, our first layer has 1024 nodes. The values for the coming hyper-parameters were found using a kind of grid search, see Section 3.3.5. The model has one hidden layer of size 512. Rectified Linear Unit (`relu`) [2] [39] is used as activation between the layers, and `sigmoid` [3] is used as the final activation layer. A `dropout` layer [4] [40] is used between layers, with a `dropout` rate of 0.61. The `dropout` layers are used to decrease the risk of overfitting. The model was trained using the `Adam` optimizer [5] [41] with a learning rate of 8.9×10^{-4}. The model uses binary crossentropy [6] [42] as its loss function since each context only has two classes, true (1) or false (0).

3.3.4 Binary relevance

Our second approach was a problem transformation method using `scikit learn`'s `MultiOutputClassifier` [7] [43] as binary classifier. A simplified illustration of the

[1] `https://www.tensorflow.org/api_docs/python/tf/keras/Sequential`
[2] `https://www.tensorflow.org/api_docs/python/tf/keras/activations/relu`
[3] `https://www.tensorflow.org/api_docs/python/tf/keras/activations/sigmoid`
[4] `https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout`
[5] `https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam`
[6] `https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy`
[7] `https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputClassifier.html`

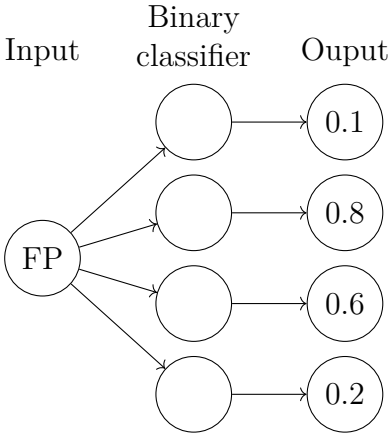


Figure 3.4: This diagram shows the structure of a simplified version of our binary relevance model.

binary relevance model's architecture can be seen in Figure 3.4. While a classifier chaining approach was considered, it was ultimately decided against as there was no correlation between the labels in our dataset, as can be seen in Section 4.2.

Our model uses support vector classifiers (`SVC`) [8] as the base estimator. The regularization parameter, `C`, is set to 10.67, this value was found using grid search, see Section 3.3.5. The parameter `C` is used to determine how to delimit the different sections of feature space, where a low value makes the shape smoother, and a high value makes the shape more complex by trying to include correct samples and exclude incorrect samples more aggressively. `Gamma`, the kernel coefficient, is set to "auto", which is the inverse of the feature size. This parameter determines how much influence each data point has in the training. The `class_weight` is set to "balanced", which means that the two classes, 0 and 1, are weighted inversely proportional to their frequency for each estimator. The `class_weight` is how the support vector machine handles class imbalance and that is also one motivation behind choosing this classifier as the base estimator. The rest of the parameters are left at their default values.

3.3.5 Hyper parameter tuning and training

The models were trained in several steps. First, the optimal values for hyper-parameters of each model were found with the use of grid search with `Optuna` [44] and `KFold` cross validation [9] with five folds. The grid search was performed with 200 trials for the algorithm adaptation model and 50 trials for the binary relevance model. Each trial consisted of one set of values for the hyper-parameters for cross validation training. The binary relevance model had fewer trials due to the smaller search space.

The hyper parameters and search spaces for the algorithm adaptation model can be

[8] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

[9] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html`

Table 3.4: This table shows the hyper parameters and search space that the grid search used for the algorithm adaptation model. We note sets with curly brackets, i.e. $\{a, b\}$ is a set containing a and b, while square brackets is a range of values, i.e. $[0, 1]$ means values ranging from 0 to 1.

Hyper parameter	**Search space**	**Best value**
Batch size	$\{32, 64\}$	32
Number of epochs	$\{10, 15, 20\}$	15
Number of hidden layers	$\{1, 2, 3\}$	1
Learning rate	$[10^{-5}, 5 \times 10^{-3}]$	8.9×10^{-4}
`Dropout` percentage	$[0\%, 90\%]$	61%

found in Table 3.4. The binary relevance model only has one hyper parameter, `C`, with the search space ranging from 10^{-2} to 10^{3}, and the best value found in the grid search was 10.67.

Label ranking average precision (LRAP) [10] [43] was used as the score to maximize in the grid search. It was used as we wanted to get a good ranking, or ordering, of the predictions. The ordering of predictions can be seen as the order a chemist tries the reaction, and then it is important that the predictions first in the order are the most promising options. It was also important that ground-false labels were ranked lower than any false negatives. We tried a few other scores, but they did not give as good results as LRAP regarding the ranking.

LRAP computes for each ground-truth label how many other labels that were ranked higher and were also ground-truth.

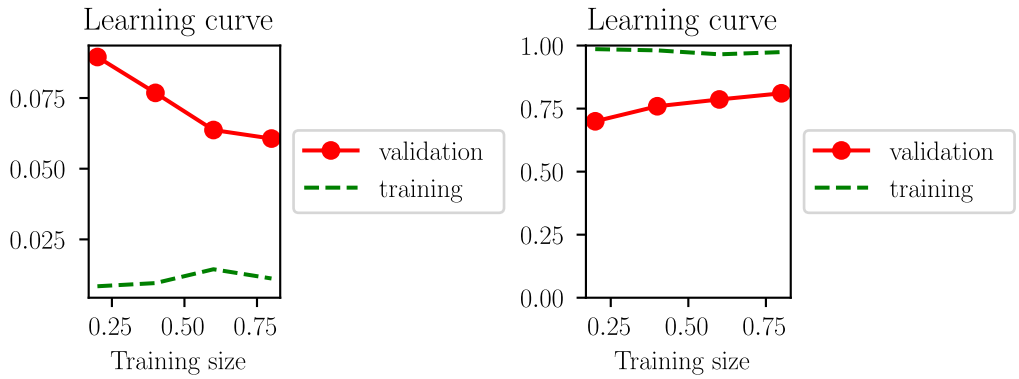
$$\text{LRAP}(H, D) = \frac{1}{|D|} \sum_{i=0}^{|D|-1} \frac{1}{||Y_i||_0} \sum_{j=0}^{|Y_i|} \frac{|\mathcal{L}_{ij}|}{|\mathcal{R}_{ij}|} \tag{3.6}$$

Where D is a multi-label dataset consisting of pairs (x_i, Y_i) where x_i is the feature and Y_i is the vector of ground-truth labels, H is a multi-label classifier that takes a feature x_i and returns a vector $H(x_i)$ containing the prediction, and $||A||_0$ is the number of non-zero elements in A. Additionally, $\mathcal{R}_{ij}$ is a set of all elements Y_{ik} such that $k \neq j$, and $H(x_i)_k \geq H(x_i)_j$, and $\mathcal{L}_{ij}$ is a set of all elements Y_{ik} such that $Y_{ik} \in \mathcal{R}_{ij}$, and $Y_{ik} = 1$. Or in other words, $\mathcal{R}_{ij}$ is a set of all elements in the true vector for feature x_i that got a higher prediction score than the jth label, and $\mathcal{L}_{ij}$ are all the ground-true labels ranked higher than the jth label for feature x_i.

The models were trained and evaluated on test data, where 20% of the data was used as test data. The data was split into train and test data using `sklearn`'s `train_test_split` [11] [43]. The test ratio of 20% was chosen as it is a standard value, giving a large portion of the data to training but still leaving enough data to reasonably evaluate the model. In Figure 3.5 and Figure 3.6 we can see the

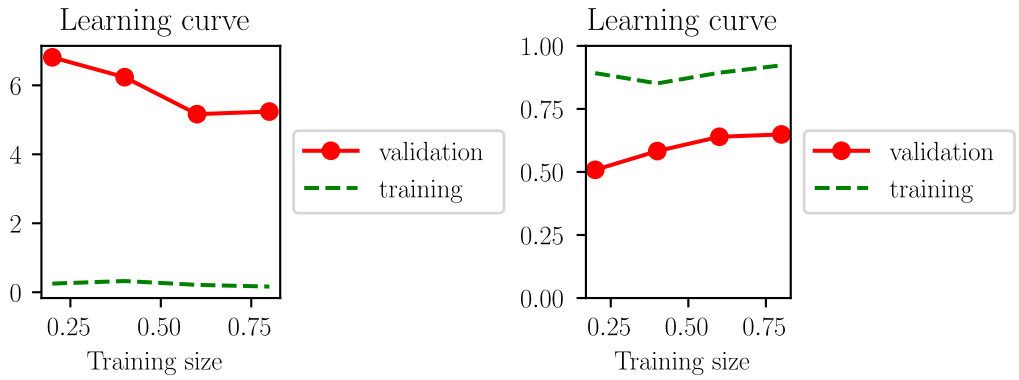
[10] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.label_ranking_average_precision_score.html`

[11] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`



(a) Binary crossentropy **(b)** Label ranking average precision

Figure 3.5: Learning curves for different training sizes, given as ratios of the whole dataset, for the algorithm adaptation model.



(a) Binary crossentropy **(b)** Label ranking average precision

Figure 3.6: Learning curves for different training sizes, given as ratios of the whole dataset, for the binary relevance model.

binary crossentropy and LRAP scores for different training sizes. The large distance between the lines in these figures suggest a high variance. We can see that the variance decreases with a training set ratio of at least 0.6.

Lastly, a model was created with the hyper-parameters and trained on the entire dataset. This process was repeated for each model.

3.3.6 Metrics

To evaluate our models several metrics were used. The metrics we used to evaluate our model were Hamming loss [12], Jaccard score [13], ranking loss [14], LRAP [15] [43], and binary crossentropy [16]. These metrics can be used in a multi-label classification task and handle partial correctness of the prediction.

Hamming loss [17] is the Hamming distance between the predicted and true set of labels, and can be seen as performing the XOR operation on the two sets of labels.

$$\text{HammingLoss}(H, D) = \frac{1}{|D|} \sum_{i=0}^{|D|-1} \frac{||Y_i \oplus H(x_i)||_0}{|L|} \tag{3.7}$$

Where $\oplus$ is the XOR-operator and L is the set of labels. A Hamming loss is a value between zero and one where zero corresponds to no loss, or edit distance, while 1 corresponds to the predicted set of labels being the inverse of the true set of labels. Hamming loss has the disadvantage that it punishes false positives as much as false negatives, while in our case false positives are not as bad as false negatives.

Jaccard score is a similarity score and is defined by taking the intersection of the true and predicted labels and divide it by the union.

$$\text{Jaccard}(H, D) = \frac{1}{|D|} \sum_{i=0}^{|D|-1} \frac{||Y_i \cap H(x_i)||_0}{||Y_i \cup H(x_i)||_0} \tag{3.8}$$

Where D is a multi-label dataset consisting of pairs (x_i, Y_i) where x_i is the feature and Y_i is the vector of ground-truth labels, H is a multi-label classifier that takes a feature x_i and returns a vector $H(x_i)$ containing the prediction, and $||A||_0$ is the number of non-zero elements in A. The Jaccard score can be seen as a measure of partial accuracy. The Jaccard score, just like the Hamming loss, punishes false positives and false negatives equally.

The label ranking loss looks at a ranking in the predicted labels and a perfect score of 0 is given when the true labels are the predicted labels with the highest ranking.

$$\text{LabelRankingLoss}(H, D) = \frac{1}{|D|} \sum_{i=0}^{|D|-1} \frac{|\mathcal{W}_i|}{||Y_i||_0(|D| - ||Y_i||_0)} \tag{3.9}$$

Where $\mathcal{W}_i$ is the set of all pairs of integers (k, l) such that $H(x_i)_k \leq H(x_i)_l$, $Y_{ik} = 1$, and $Y_{il} = 0$, or in other words k and l are indices for label pairs where a ground truth, the positive labels in the data, is ranked lower than or equal to a ground falsehood, the negative labels in the data.

[12] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.hamming_loss.html

[13] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard_score.html

[14] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.label_ranking_loss.html

[15] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.label_ranking_average_precision_score.html

[16] https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy

Lastly, the binary crossentropy compares the predicted values to the vector of ground-truth labels, if the actual and predicted vectors are the same the loss will be 0.

$$\text{BinaryCrossentropy}(H, D) = \frac{1}{|D|} \sum_{i=0}^{|D|-1} Y_i \times \log(H(x_i)) + (1 - Y_i) \times \log(1 - H(x_i)) \tag{3.10}$$

We also evaluated how many false positives that would need to be included before all ground truths have been predicted following the ordering of the predictions. We called this the False Positive Ranking Inclusion (FPRI) and its definition can be found in Definition 3.1. The metric was used to evaluate the relevance in the ranking as ground truths should be higher predicted than ground falsehoods.

Definition 3.1. *False Positive Ranking Inclusion* is the fraction of false positives that are predicted with a score higher than or equal to all ground-truths' predicted scores.

For example, given true vector [0, 1, 1, 0] and predicted vector [0.5, 0.3, 0.8, 0.2] we get a FPRI score of 0.33. It means that to predict all ground-truths we would have to include one false positive out of three total predictions.

3.3.7 Comparison to other models

To further evaluate the performance of our models we compared them to other models. We compared the performance to baseline models to make sure that the model learns a good mapping of the data. We also compared our algorithm adaptation model to the single-label model.

3.3.7.1 Comparison to baselines

We used two baselines for our model. The first was a `DummyClassifier` [17] from `sklearn` [43]. The `DummyClassifier` always predicted the most frequent set of chemical contexts from the training set. This baseline represents a valid approach in a lab setting.

The second baseline was a model that had trained on the data, but the order for the true outputs had been shuffled. Say, for example, that we have a very small dataset consisting of the inputs $apple, banana, shoe$ and outputs $fruit, fruit, clothes$, then the shuffled model might get the outputs $clothes, fruit, fruit$. As can be seen in this example, some instances of the more common labels might still be paired with the correct input. This can make it even more difficult for the model to learn as it is trying to group input that fits the label with input that does not. The goal of doing this shuffling was that this model should not be able to isolate the different parts of feature-space that correspond to certain labels. Any model that learns the non-shuffled data should outperform this shuffled model.

[17] `https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html`

3.3.7.2 Comparison to single-label model

We compared our algorithm adaptation model to the single-label model developed at AstraZeneca to see if they would predict similar results under the assumption that it would be beneficial if the single-label and multi-label models agreed. The single-label model was initially trained on another dataset than the one we used, so the first step was to retrain the single-label model with our dataset. The comparisons between the single-label model and multi-label model was made with the algorithm adaptation model only. This was done because the models are similar in architecture, thus we wanted to show that our multi-label model performs at least as well as the single-label model. Even though the same dataset was used there were some pre-processing differences to creating the single-label data to the multi-label data. All contexts used for a reaction were used for the multi-label model. In contrast, only the context with the highest yield was used for each reaction for the single-label model. This causes minor differences in the distribution of the contexts for the data depending on whether it was pre-processed to be single-label or multi-label.

We looked at the performance for both models in terms of the top-k distribution and the distribution of predicted and actual contexts. Top-k in this case means that the true context was predicted as the rank k context. To properly compare the two models, we only used data with one true context, because having multiple ground-truths would make the top-k definition less clear. By looking at the top-k distribution one can compare the predictive power of the models. The distribution of contexts was analyzed to see if they compare roughly the same contexts as frequent.

To compare the predictions between the models we looked at how highly the single-label predicted context would be ranked by the multi-label model for the same reaction. This calculation showed how well the models agree with each other for the same data. For this to be a reasonable comparison the test data consisted of the intersection between the two models' test data to make sure no model had trained on the data. Any reactions with multiple true contexts were discarded since it could otherwise appear that the two models disagree when they might not.

3.4 Condition prediction for chemical libraries

As the final part of the project we have developed a model that can predict a chemical context that works well for a chemical library. Our approach was to have one binary classifier for each of the top five most common contexts, where each classifier predicts how well the reactions would work with its context. The approach was chosen as it is simple and the contexts do not have to be learned in relation to each other. In addition, there is no sequence or ordering of the input.

3.4.1 Overall structure

Figure 3.7 shows the overall workflow for our model. The input is a set of reactions, which are turned into reaction fingerprints as described in Section 3.2. These fingerprints are given as input to all of the classifiers. There is one classifier per most

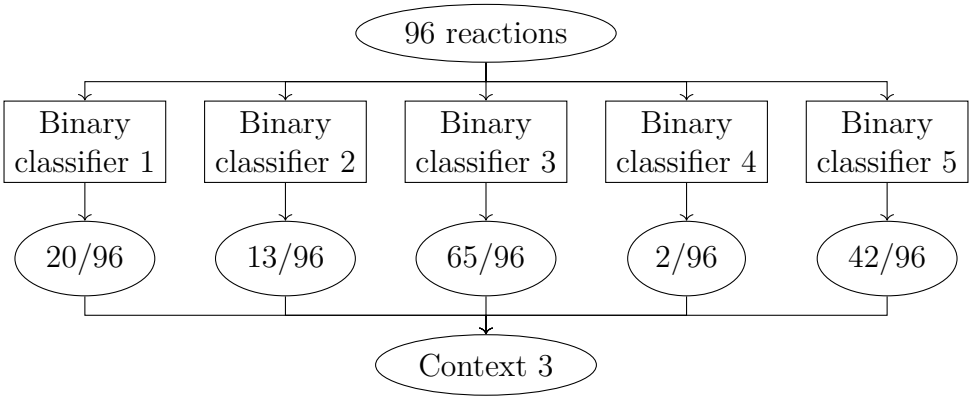


Figure 3.7: This diagram shows the structure of our model that will predict a context that works well for a chemical library. The top level shows the input to the classifiers. The classifiers predict how many of the reactions would work with their respective contexts. Finally, the context associated with the highest prediction is chosen as the output.

common context in the dataset. We used the five most common contexts. This limit was chosen because we observed a decrease in performance as we used less and less common contexts. Each classifier determines how likely each reaction is to work for a specific context and returns a score vector. These vectors are used to calculate the fraction of reactions that have a high enough score. Finally, the context that is most likely to work for the largest number of reactions is chosen as the output.

3.4.2 Classifier

In Figure 3.8 you can see the basic structure of the classifiers and how the information flows through them. First, a reaction fingerprint was fed to the binary classifier. The binary classifier then predicts whether that reaction was likely to work with the context associated with that classifier or not. The output score from the binary classifier was fed to a Venn-ABERS predictor which calibrates the score to turn it into a probability. This process was repeated for each reaction fingerprint, and the calibrated probabilities are sent forward as a vector.

We used `SVC` [18] as the binary classifier. We chose this as it is commonly used and we observed that it performed better than random forest classifier [19] during our development. The regularization parameter `C` is set to values between 13.02 and 192.50 depending on the context. These values were found using a grid search. Just like with our Binary Relevance-model, `Gamma` was set to "auto", and the `class_weight` was set to "balanced".

[18] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

[19] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

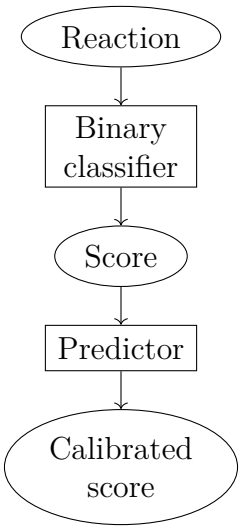


Figure 3.8: This diagram shows the data-flow through the classifier in our model for predicting a context for a chemical library.

3.4.3 Predictor

We added a Venn-ABERS predictor to each binary classifier to calibrate the scores to better represent probabilities. Venn-ABERS was chosen because it has shown good results in the literature.

A Venn-ABERS implementation [20] based on Vovk's paper [22] was used for the calibration. This implementation is an inductive Venn-ABERS predictor which means that some of the training set has to be used for the predictor to calibrate the scores. Because we could not use all of the training data with the predictor we had to re-train the classifiers on the smaller proper training data. First, the dataset was divided into training, test, and calibration sets using `sklearn`'s `train_test_split` [21] [43]. First the test set was created consisting of 20% of the dataset, the dataset was then split into the training and calibration sets consisting of 80% and 20% of the remainder.

Each binary classifier was trained on its proper training data in the same way as described in Section 3.4.4. Having a calibration set means that we have less training data to train the model before applying the predictor. For the regular classifiers the training set consisted of 80% of the data, and for this model the training set was 64% of the dataset. The effect on having a smaller training set for the binary classifiers' performance was investigated by comparing it to the model where we trained on 80% of the data.

The calibration was performed for each of the classifiers with the calibration and test sets. The prediction score and label of each of the calibration examples were used to calibrate the test prediction scores. The output is multi-probabilistic as it returns two probabilities for each test example. The two probabilities, p_0 and p_1,

[20] `https://github.com/ptocca/VennABERS`

[21] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

were turned to one probability with Equation 3.11 [22]. Based on this probability, or calibrated score, a prediction was derived in the same way as for the uncalibrated score.

$$p = \frac{p_1}{1 - p_0 + p_1} \tag{3.11}$$

The calibration was evaluated for each classifier. The predictor and calibration was evaluated with the same metrics used for the evaluation of the binary classifier to evaluate the difference between the uncalibrated and calibrated scores. The same was done to the other evaluation approaches for the binary classifiers. These evaluations were made to see how the calibrated scores affected the performance.

For each of the binary classifiers, a calibration plot was created and the difference in the multi-probabilistic output was investigated as well. A calibration of both the calibrated and uncalibrated scores was used to see the difference between the two in regards to a perfect calibration. A calibration plot shows the association between the predicted and actual probabilities. A perfect calibration is where the predicted and actual probabilities are the same. Furthermore, investigating the difference of the multi-probabilistic output, $|p_0 - p_1|$, gave us a measure on the uncertainty.

3.4.4 Hyper parameter tuning and training

During the training process a separate dataset was created for each classifier. The same pre-processing as for the multi-label classification task was applied to the dataset. However, instead of keeping the 30 most common contexts, all contexts were kept. In addition, we created separate Y-vectors for each classifier. Each Y-vector corresponds to a specific context. Each classifier was fed all reaction-fingerprints and a Y-vector. For each classifier, the data was divided into training-, test-, and calibration-sets using `train_test_split` [22] [43] where we stratified over the Y-vector, meaning that the fraction of ground-truths stayed the same between the different sets. The stratification was done to handle the imbalance in the dataset, so that the test set has the same label distribution as the training set.

Each binary classifier's hyper-parameter was found with grid-search using `Optuna` [44] and `KFold` cross validation [23] with five folds to optimize the hyper-parameter before training on their respective datasets. The number of trials, that is the number of values explored in the hyper-parameter grid search, was 50 since there was only one hyper-parameter and the models did not improve significantly after this. For the grid search we only used the hyper parameter `C` with search space ranging from 10^{-2} to 10^{3}, just as with the `SVC` in the binary relevance model. We used `sklearn`'s area under receiver operating characteristic curve (ROC AUC) score [24] as the score to optimize in the grid search, because it was important to get overall good performance. Furthermore, we wanted the model to predict all ground-truths

[22] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

[23] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html`

[24] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html`

as positives, see Section 2.3 for the explanation why. We tried different scores, such as f1-score and precision. Using these scores gave a good overall performance but did not retrieve all the positives at the same rate as when using recall or ROC AUC. We did not use recall as the optimization score even though it had slightly better recall (retrieval of ground-truths), because the overall performance was much worse. Thus, ROC AUC was chosen since it had a good overall performance and predicted many true positives.

3.4.5 Evaluation

Our approach was primarily evaluated by analyzing each binary classifier's performance. We used several metrics as well as confusion matrices for evaluation. We also looked at the ROC curve to investigate the true positive rate compared to the false positive rate. The added predictor's effect on performance was evaluated as well, see Section 3.4.3. For each classifier, a `DummyClassifier` was added as a baseline. The predictions generated by the `DummyClassifier` followed the training class distribution instead of predicting the most frequent class. This strategy was chosen as it is a more realistic baseline to compare to. A shuffled classifier was also used as baseline.

The metrics we used were precision [25], recall [26], f1-score [27] and ROC AUC [28]. We used these as they are all common in binary classification tasks. They are defined in the following equations where T stands for true, F for false and P for positive and N for negative. For example, TN is the number of true negatives. The precision tells how many of the predicted positives that are ground-truths (see Equation 3.12).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.12}$$

The recall tells how many of the ground-truths were predicted (see Equation 3.13).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.13}$$

The F1 score can be seen as a weighted average of the precision and recall (see Equation 3.14). The F1 score ranges between 0 and 1, where 1 is the best and 0 the worst.

$$\text{F1} = \frac{2 * precision * recall}{precision + recall} \tag{3.14}$$

The ROC AUC score is the area under the ROC curve, with an optimal score of 1. The ROC curve gives an indication of how well a binary classifier can separate

[25] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html`

[26] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html`

[27] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html`

[28] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html`

between the two classes. The curve is the fraction of positives that are true positives with respect to the fraction of negatives that are true negatives at various thresholds.

We also evaluated the model as a whole by extracting 96 data points for validation. With this validation data we looked at how often a context was in the dataset compared to how often it was predicted. It was done to see if we predicted one context more than others and if we predicted the contexts more or less often than they appeared in the dataset.

Lastly we predicted one context for a chemical library from AstraZeneca containing 89 reactions that are run in parallel. This chemical library had been used in practice and we compared our predicted context to the context used in practice. By performing this task we could see how it would work in practice and give an indication of our model's performance. Since it was only one library file and if our predicted context would differ, we cannot say if the predicted context would work better or not, we cannot draw too strong conclusions based on this result.

4

Dataset analysis

In this chapter we analyze the dataset we have used for condition prediction. We also describe the pre-processing we have done and analyze the data from a multi-label classification perspective.

During the project we have been using a dataset extracted from AstraZeneca electronic notebooks (ELN). Electronic notebooks are written by chemists and contain data from synthesized reactions. The data are from two phases in the drug discovery process: Medicinal Chemistry (MedChem) and Pharmaceutical Development (PharmDev) which is later in the process. MedChem is part of the discovery and pre-clinical phase where chemists synthesize and investigate good drug candidate molecules. PharmDev is part of the later pre-clinical and clinical phases where the chemists optimize the reactions to synthesize the drug target molecules.

Only data from the Buchwald-Hartwig reaction class was used in this project and that is the only part of the dataset we will analyze here.

4.1 AstraZeneca ELN and pre-processing

The reactions in the dataset are represented by a sequence of InChIKeys [34] representing the molecules, paired up with a string telling what role the molecule played in the reaction. The roles are reactant, product, precatalyst, ligand, base, solvent, catalyst, and reagent. The reactants and product will later be used to form the feature, while the rest of the molecules make up the chemical context and will be our models' labels. To be able to build upon earlier code and easily compare to an older model another representation, SMILES strings are added to the dataset. These SMILES strings are used to create reaction SMILES by concatenating a reaction's reactants, chemical context, and product with special characters. There is also other important information in the dataset, namely the yield, the year the variation was tested, what class the reaction belongs to, and where the data originates from (MedChem or PharmDev).

In Table 4.1 we see the number of variations, reactions, and chemical contexts in the dataset and their source. A variation is a reaction with a specific chemical context. For there to be many reactions with multiple contexts in the dataset we would expect the number of variations to be higher than the number of reactions, since a variation is a reaction paired with a chemical context. However, note that no pre-processing has been applied yet. What can be said, however, is that while

Table 4.1: The number of variations, reactions, contexts, and total number of data points respectively in the dataset split into the sources of the data.

	PharmDev	**MedChem**	**Both**
# Variations	754	33358	34112
# Reactions	167	21615	21769
# Contexts	702	4619	5257
Total size	1296	42981	44277

Table 4.2: This table shows the size of the dataset after the different steps of the filtering. For this specific filtering we set minimum yield to 20%, and were only interested in Buchwald-Hartwig. The last row shows the size when the top 30 most frequent contexts are kept. This last step is not performed for the condition prediction task for libraries, as all contexts are kept.

Filter	**Dataset size**
Before filtering	44277
Remove too high yield	43972
Remove too low yield	11762
Remove other reaction classes	11210
Remove too infrequent contexts	6291

PharmDev might have a higher number of variations per reaction, it is completely overshadowed by MedChem which supplies most of the data, leading to a dataset with less than two variations per reaction.

During the pre-processing we filter out data that is unimportant or faulty in some way. In the first step of the filtering process we remove any entries with a yield higher than 100% as this is faulty data. After that we remove data with a yield lower than 20%, since we want to train the model on well-performing variations. This also removes all unsuccessful variations from the dataset. The main reason is that the dataset does not provide an explanation for why the variation failed, it could be because the variation does not work in general, but it could also mean that some external factor harmed the experiment. Even if this was not the case, however, it would increase the complexity of our model as we would need to differentiate between not having tested a variation, and having a failed test. Then, we remove all reactions that do not belong to Buchwald-Hartwig. While the dataset should already consist of Buchwald-Hartwig reactions, it appears that some reactions have either been mislabeled, or have accidentally been included in the dataset. Finally we remove all entries that use a context that is not in the 30 most common contexts. This value was chosen as our binary relevance model had difficulties handling the dataset if too many contexts were used as it would for some splits receive no ground truths for some contexts. Removing the entries with less common contexts makes the dataset less unbalanced as it removes the contexts that do not have enough data points to properly train the model to find how reactions that use them look. The

size of the datasets before, during, and after the filtering can be seen in Table 4.2.

Once the data has been filtered we encode the contexts to one-hot vectors. We then use these one-hot vectors to make a multi-hot vector for each reaction and remove all but one instance of each reaction from the dataset. This changes the number of rows in the dataset to 5285.

4.2 Processed dataset

Table 4.3 shows the distribution of the number of reactions that had a specific number of variations. These numbers are fetched before calculating the multi-hot vectors. As is clearly depicted by this table the vast majority of reactions were only used once, meaning that a majority of the reactions will only have one context. The mean number of variations were 1.2 and the maximum number of variations for one reaction was 10.

Table 4.3: This table shows how many reactions has a specific number of variations in the dataset.

Number of variations	Number of reactions
1	4537
2	603
3	88
>3	57

Figure 4.1 depicts how many times each context was used in the dataset. As can be seen, the dataset is somewhat unbalanced despite the removal of the less frequent contexts. While this is not by any means optimal, we have not explored any further ways to mitigate this.

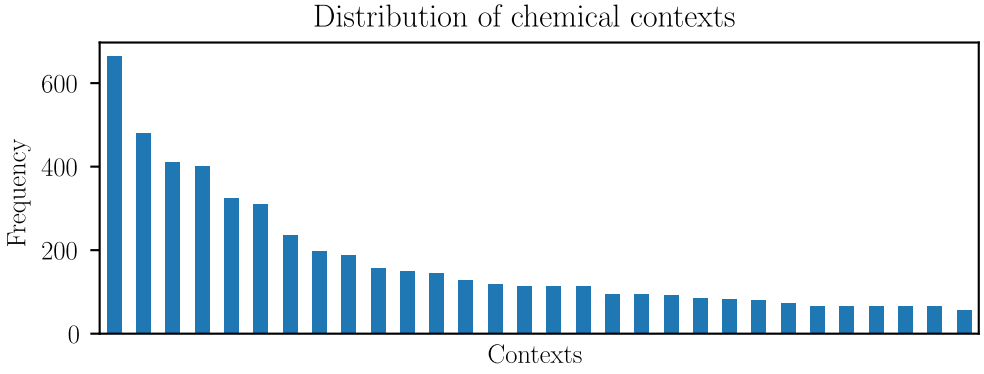


Figure 4.1: This plot shows how many times each context appears in the dataset.

4.2.1 Multi-label perspective

We have analyzed the data from a multi-label perspective by looking at two kinds of measures. First, we looked at label cardinality and label density which measures how

multi-label the data is (see Equation 3.1 and Equation 3.2). Multi-label dataset often suffer from imbalance [35], so we also measured the imbalance in terms of the mean and variance of the imbalance ratios per label (see Equation 3.4 and Equation 3.5).

In the processed dataset there are only two data points left from from the PharmDev data, the rest are removed in pre-processing and filtering. The PharmDev data had a much higher label cardinality than the MedChem data. After the filtering of faulty data but before the multi-hot vectors are created the PharmDev data (132 data points) has a label cardinality of 6.0 while the MedChem data (11078 data points) with a label cardinality of about 1.2. This difference arises from the different goals of MedChem and PharmDev. Since MedChem is only investigating drug candidates it is enough if a context that works is found. In contrast, during PharmDev the chemists tried to optimize the reactions, and thus tested more contexts for each reaction. Since MedChem make up almost all of the data points, the total label cardinality when the multi-hot vectors have been created is slightly above 1.0, with a label density of about 0.03. We can conclude that the data, on average, is single-label. The low label cardinality causes some issues. First, it means that the model might learn to reproduce the low label cardinality in its predictions. Second, the low label density means that each column in our multi-hot vector is sparse, making it harder for the model to learn when to predict a certain label.

The calculation of the imbalance was done after the dataset had been fully pre-processed, that is, after all the filtering and creation of multi-hot labels. The mean of the imbalance ratios per label is approximately 6 and the variance approximately 0.5. A mean above 1.5 and variance above 0.2 is considered imbalanced [35]. Thus, our multi-label dataset can be considered to be imbalanced.

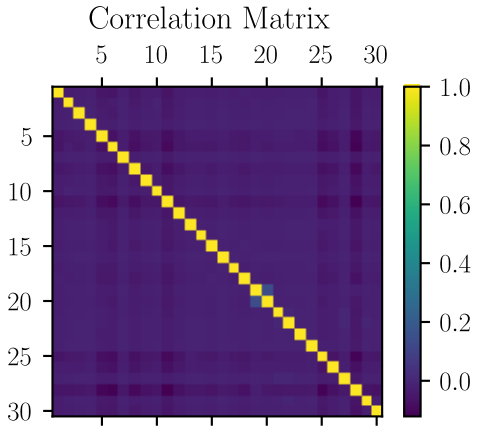


Figure 4.2: This plot is a heat map for how often a pair of labels appear for the 30 most frequent labels. A bright and yellow spot means that they appear together often, while a dark and blue spot means that they do not appear often together. The yellow diagonal is expected as this corresponds to a label appearing with itself.

In Figure 4.2 we see the correlation between the chemical contexts in the dataset. The bright yellow diagonal line with score one shows that a context appears with itself. Since the rest of the heat map is a darker blue (scores close to zero) we can conclude that there is no meaningful correlation between the contexts in our dataset.

No correlation between the contexts also mean that a model can predict a context for a reaction independently of any other context. It is not a surprising result as the data is mostly single-label and therefore most contexts appear as the only context to a reaction.

5

Multi-label classification of chemical contexts

In this chapter we show our results on multi-label classification to solve a condition prediction task. We show results from both an algorithm adaptation model and a binary relevance model. We will also compare our results to AstraZeneca's single-label model and baseline models. Here we describe the performance on the test data. To see the performance on the training data see Appendix B.

5.1 Learning curve evaluation

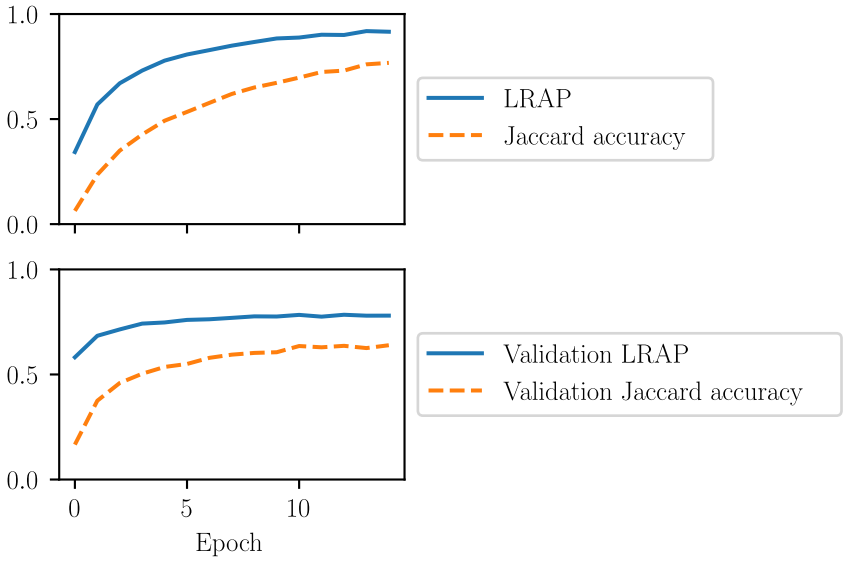
The metric scores for the different epochs in both training and validation data for the algorithm adaptation model can be seen in Figure 5.1. It shows that the model performs very well on label ranking average precision, Jaccard accuracy, Hamming loss, label ranking loss and the loss function (binary cross-entropy loss). Our binary relevance model does not train over several epochs and is only evaluated once.

5.2 Analysis of predicted labels

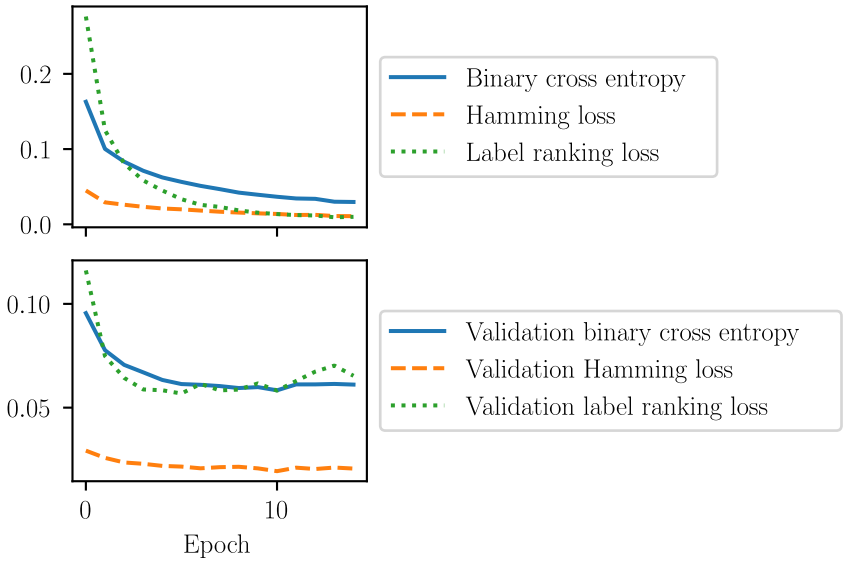
When analyzing the predicted labels we look at how many labels we predict on average per reaction and how it compares to the true labels. Figure 5.2 shows that most of the time the contexts are predicted at the same rate as they appear in the dataset. However, the algorithm adaptation model starts to slightly under-predict the contexts that appear more than around 50 times in the dataset. Nevertheless, all contexts are predicted by both models. That the dots follow the dashed line, indicates that our models handles the unbalanced dataset quite well.

The label cardinality is about the same for the true and predicted outputs (see Table 5.1). The algorithm adaptation model has slightly higher label cardinality for the training but lower for the test data compared to the label cardinality of the actual data. The binary relevance model has a higher label cardinality in both cases, although it still decreases in the test compared to the training. It is clear that the model most often predicts only one context. If the data itself was more multi-label our models would likely get higher label cardinalities for the predicted output as well.

We can see the fractions of true positives, true negatives, false positives and false



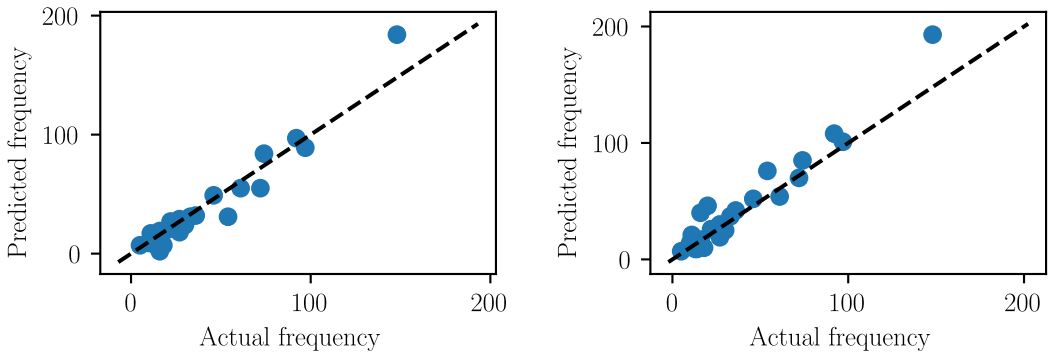
(a) Scores



(b) Losses

Figure 5.1: Plots over accuracy scores and losses in training and validation data for the algorithm adaptation model.

negatives for each of the eight most common contexts in the confusion matrices in Figure 5.3. The confusion matrices of the two models are similar. For some contexts the binary relevance model is slightly better and in other cases the algorithm adaptation model is slightly better. Overall, both models are predicting almost all ground-falsehoods as negative. The models cannot distinguish the positives as accurately. For some contexts, many false negatives are given, which is likely because there are few true positives in the dataset making it more difficult for the model to learn. False positives are more preferable than false negatives in our case because it means that the model could have proposed something useful that the chemists have

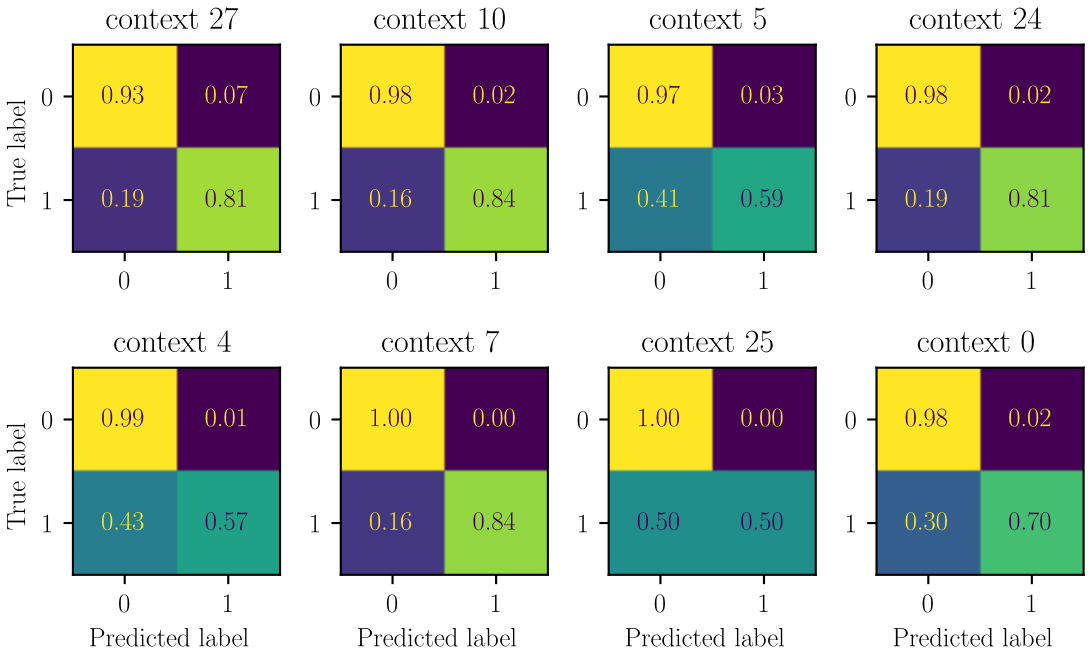


(a) Algorithm adaptation model **(b)** Binary relevance model

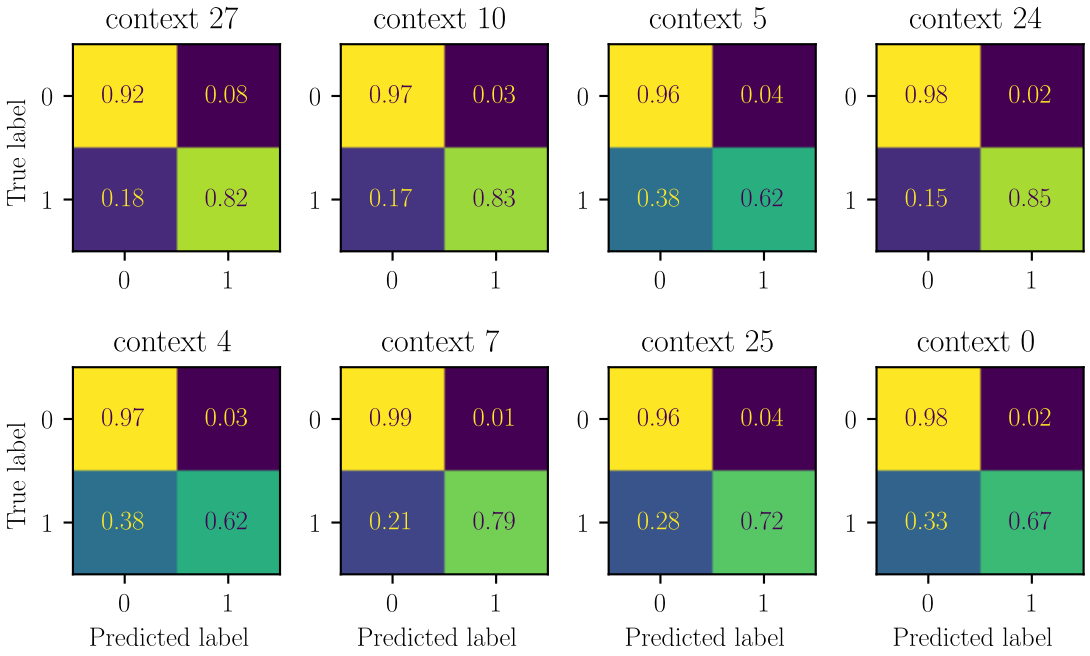
Figure 5.2: Plots how the predicted distribution of contexts behaves in comparison to the actual distribution on the test data. One point shows how many times that context appears in the actual and predicted data. The dashed line represents a frequency that is the same in the actual and predicted data.

Table 5.1: Label cardinality, average number of labels, for the true and predicted vectors in both test and training data.

	Ground-truth	Algorithm adaptation model	Binary relevance model
Train	1.01	1.07	1.20
Test	1.01	0.96	1.03

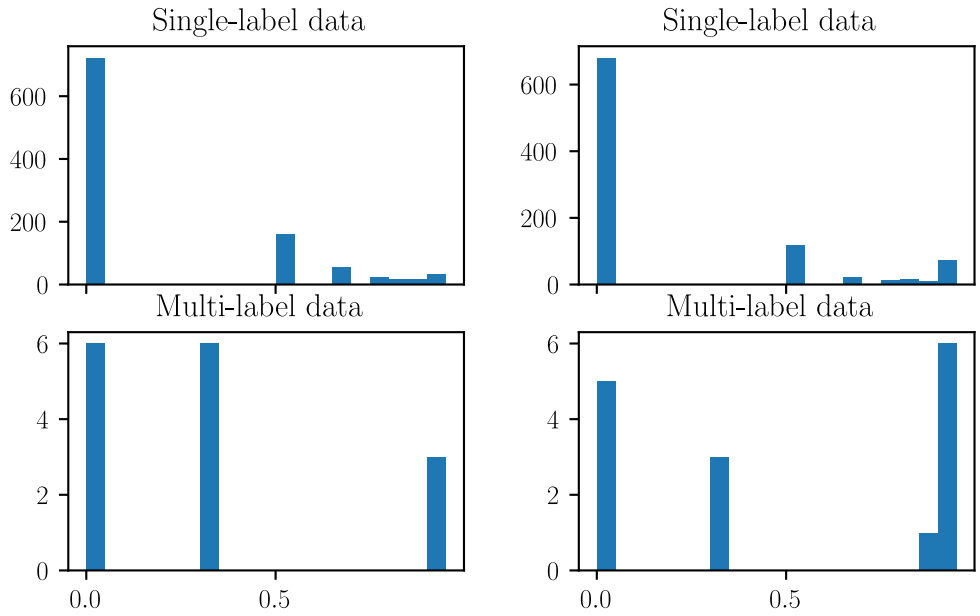


(a) Algorithm adaptation model



(b) Binary relevance model

Figure 5.3: Confusion matrices, one per context, for the ten most frequent contexts in the dataset. They are all normalized with regards to the true labels. Optimal matrices would have ones in the top left and bottom right cells, as this would mean all predictions are true positives and true negatives. Yellow cells show a value close to one and purple values close to zero.



(a) Algorithm adaptation model **(b)** Binary relevance model

Figure 5.4: Plot of the distribution of scores for the predictions where the score is calculated using Definition 3.1. This score is on the x-axis and is shared for the top and bottom plots. It shows how large the fraction of false positives we have per prediction if all true positives are included based on the `sigmoid` score for each context. An optimal value is zero as that is when our top predictions are all true positives and no false positive. The worst score is close to one where one of the true contexts is the lowest predicted. The top plot is for the scores where there is only one true context, and the bottom plot is for the cases where there is more than one true context.

not tried yet. It is clear from the confusion matrices that some contexts are more difficult to learn than others. For example both models performs quite poorly for context 5 and both models perform quite well for context 27.

In comparison, most of the confusion matrices on the training data for both models show good performance (see Appendix B). It shows that the models can predict the contexts in the training data very well. The reason why these confusion matrices differ from the confusion matrices for the test data is likely due to the data not being fully representative of the ground truth. Thus we get a tendency of overfitting. For example, this representation does not distinguish between failed and not tried contexts for reactions. Further explanation could be that the data is imbalanced and contain mostly zeroes.

Figure 5.4 shows the False Positive Ranking Inclusion (FPRI) (see Definition 3.1). For most data points, the scores are close to zero which mean that the true positives are predicted higher than false positives. We can see that for the single-label data, the true context is in most cases the highest predicted context. If there were more predicted labels one could be reasonably sure that the ranking in prediction scores could guide the chemist in choosing a good context for the reaction. As there are only 15 instances of multi-label test data it is difficult to draw a conclusion for the multi-label cases. It could be that the model gets a good results on this partly because the hyper-parameters were chosen when optimizing rank precision, which is related to the score presented in the figure. The algorithm adaptation model performs slightly better than the binary relevance model in this metric.

5.3 Comparison to naive models

We trained two naive models as baselines to compare our models to. We used a `DummyClassifier` that predicted the most frequent vector each time. In addition to that, we also trained our models with the true labels shuffled. The comparisons can be found in Table 5.2 and Table 5.3.

Table 5.2: A comparison between our algorithm adaptation model, a `DummyClassifier` which just predicts the most frequent vector, and our algorithm adaptation model trained on shuffled labels. A bold number shows that a model has outperformed other models in that score substantially.

	Algorithm adaptation model	Shuffled algorithm adaptation model	`DummyClassifier`
LRAP	**0.80**	0.26	0.03
Jaccard accuracy	**0.66**	0.00	0.00
Binary crossentropy loss	**0.06**	0.14	0.52
Hamming loss	0.02	0.03	0.03
Label ranking loss	**0.04**	0.31	1.00

In one of the scores, Hamming loss, the performance is roughly the same for all models. The naive models probably perform well on the Hamming loss because the vectors are mostly zeroes and therefore there will be few incorrect predictions.

Table 5.3: A comparison between our binary relevance model, a `DummyClassifier` which just predicts the most frequent vector, and our binary relevance model trained on shuffled labels. A bold number shows that a model has outperformed other models in that score substantially.

	Binary relevance model	Shuffled binary relevance model	`DummyClassifier`
LRAP	**0.64**	0.09	0.03
Jaccard accuracy	**0.64**	0.07	0.00
Binary crossentropy loss	**0.34**	2.12	0.52
Hamming loss	0.02	0.14	0.03
Label ranking loss	**0.26**	0.82	1.00

For all other scores, our models outperform the naive models. Thus, we can conclude that the models learned how to relate the input data to the output from the training. Note also that the algorithm adaptation model performs better than the binary relevance model.

5.4 Comparison to single-label model

The comparison between the single-label and multi-label model is inherently difficult as they are trying to solve very different tasks. Also, the multi-label model can have partially correct answers, which cannot be said about the single label model. Additionally, the models have slightly different datasets due to differences in pre-processing, further increasing the difficulty. Moreover, since the data is mostly single-label we cannot tell if we would get better results with more multi-label data.

Figure 5.5 shows the top-k distribution of both the single-label model and the multi-label model. It shows that in the majority of the data points the ground truth is the highest ranked context for both models. The plot indicates that the multi-label model is almost as predictive as the single-label model. For a better comparison, only single-label data was considered for the multi-label model. It meant that 1042 data points could be used out of a total of 1057 data points.

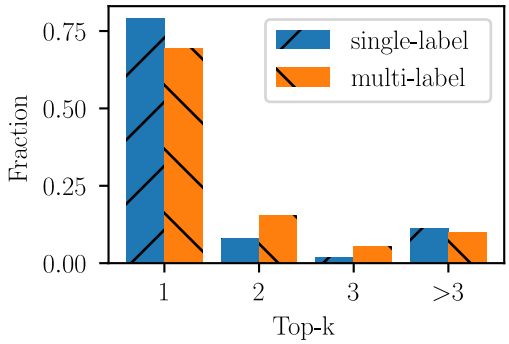


Figure 5.5: The top-k distribution for the single label and multi-label algorithm adaptation model respectively. A prediction being top-k means that the true context is the kth highest predicted.

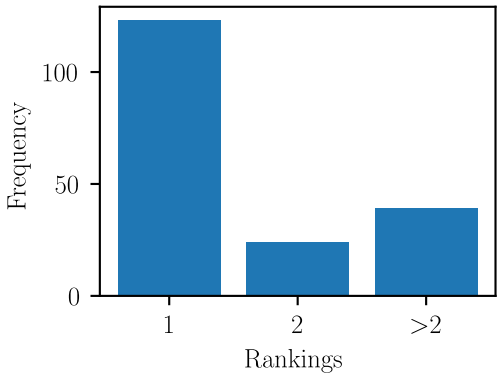


Figure 5.6: The plot shows how the one prediction from the single-label model ranks given the multi-label ranking from the algorithm adaptation model. For example, if the ranking is 1 it means that the single-label prediction is the highest predicted context in the multi-label output. In general, the lower the ranking the higher level of agreement.

Table 5.4: The number of times each model was correct and incorrect. And also, when one model was correct when the other model was incorrect.

	Single-label model	**Multi-label model**
# correct	112	85
# incorrect	74	101
# correct when other is incorrect	28	1

The single-label prediction was compared to the multi-label prediction to see how highly the multi-label model ranked the predicted context by the single-label model. The data used was a common test set which consisted of all data points that were in both the single-label and multi-label models' test data. This new test set was created to make sure there was no overlap between the test set and either model's training set. It is rather small with only 186 data points. However, most of the contexts appear at least a few times and no context appear much more than any other. The small test set is therefore considered as representative for the data as a whole. As can be seen in Figure 5.6, the multi-label model most often ranks the context predicted by the single-label model the highest in its prediction. It shows that the models agree with each other to a high extent.

For the common test set, we can see the number of correct and incorrect predictions for both models in Table 5.4. The single-label model gets a majority correct, and is correct many times when the multi-label model is not, while the multi-label model is incorrect most of the time and is only right once when the single-label model predicts incorrectly. The single-label model is better, but it is also a small dataset so it is difficult to conclude something with certainty.

From these comparisons of the two models we find that our multi-label model performs slightly worse than the single-label model for single-label data. While this

could just be the result of having a small dataset, it could also be a real difference. This could arise from the difference in scores to optimize with the grid search, difference in loss functions, or the difference in pre-processing of the data. Either way, the slight drop in performance might not be a problem, as the models serve different purposes, and receiving multiple probable contexts instead of only one might be worth the trade-off of a slightly lower performance. Further comparison can be found in Appendix C.

6

Condition prediction for chemical libraries

In this chapter we show the results from our model that solves the task of condition prediction for chemical libraries. This model predicts one context which is suitable for many reactions. The model consists of several binary classifiers, each predicting if a reaction would work with its context. There is one classifier for each of the five most common contexts.

The performance of the approach is mostly evaluated by analyzing each of the binary classifiers' performance. We compare the performance of our classifiers to naive classifiers. We also show how adding a predictor for each classifier affect the results. The evaluation presented here will be on the classifier and predictor for the third most common context as that classifier's results was considered representative of all models. The representative classifier will be called Model 3 from now on. Results for all the classifiers can be found in Appendix D. The evaluation of all the predictors can be found in Appendix E. We also show the evaluation of the overall aggregate model. Lastly, we present the results on condition prediction for a chemical library using our model.

6.1 Dataset

The dataset consists of AstraZeneca electronic notebooks of synthesized reactions. It is the same dataset that we used for the previous modelling. An analysis of the dataset can be found in Chapter 4. There is, however, one deviation from that dataset: we keep all contexts in the filtering rather than just the 30 most common, which results in 9230 data points. We keep all contexts to increase the size of the dataset, and unlike the previous models, this aggregate model only predicts on a subset of the contexts in the dataset.

The most common context appears 679 times in the dataset, the second 470 times, the third 331 times and the fourth and fifth 305 and 304 times respectively. The most common context is more than twice as common as the third to fifth most common. It could lead to a bias towards the most common context as that has been part of more reactions.

6.2 Evaluation

In parallel synthesis it is essential to get as much product as possible. The positive examples need to be found by the model, since they have proved to be successful. In addition, the negative samples actually could be positive since we have no separation between *tested and does not work* and *not tested* for any reaction. It could be that some of the *not tested* examples could work when tested. Thus, high recall is more important than high precision in our case.

Table 6.1: The precision, recall, f1 and ROC AUC (area under receiver operating characteristic curve) scores for our model, a `DummyClassifier` and a shuffled model. Where the `DummyClassifier` predictions were the same as the distribution of the training data and the shuffled model trained on a shuffled Y vector. Where one classifier outperforms the others substantially, that score is in bold.

	Our model	**Shuffled model**	**`DummyClassifier`**
Precision	**0.18**	0.04	0.05
Recall	**0.87**	0.41	0.05
F1 score	**0.30**	0.08	0.05
ROC AUC	**0.91**	0.49	0.50

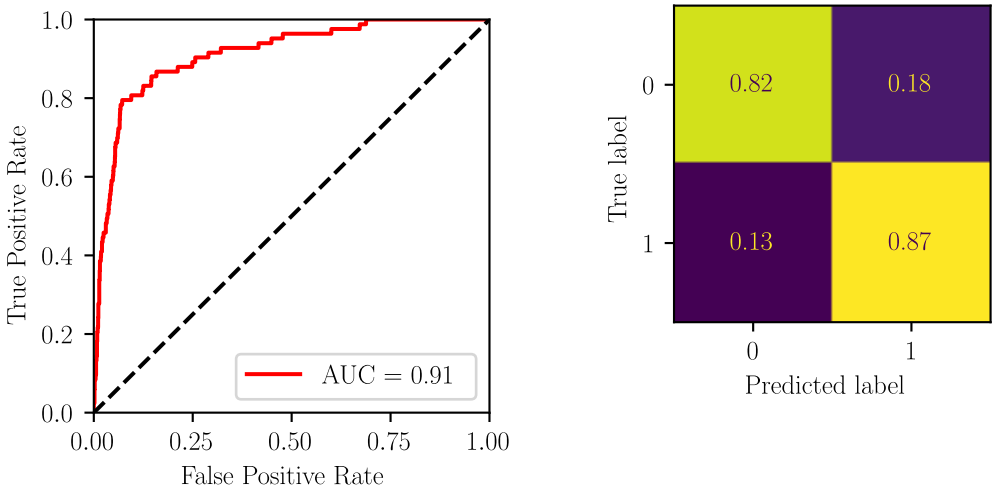
Model 3 has a high ROC AUC score and recall score but lower precision and F1 score (see Table 6.1). A high recall means that our model is good at finding all the positive examples. The lower precision, and thus also f1 score, indicates that quite many of the classifier's positive predictions are ground-falsehoods.

The confusion matrix for Model 3 (see Figure 6.1b) is normalized with respect to the true labels, and shows that a large fraction of the ground-true labels are predicted as positive. A large fraction of the ground-false labels are also predicted as negative. The fraction for the true and predicted negative is somewhat lower than the fraction for the positive examples. However, since there are many more negative examples in the dataset they will have a larger effect. This explains why the f1 score is quite low even though the recall is quite high.

The ROC curve can be seen in Figure 6.1a and is a measure of how well a binary classifier performs at different discriminating thresholds. The AUC is 0.91, which is close to the optimal value 1. An AUC of 1 would mean that the model predicts the correct class for all input. The AUC score of 0.91 shows that the classifier has learned to discriminate between the positive and negative examples well. The dashed line represents an AUC of 0.5, where the model would predict correctly 50% of the time, like a coin flip.

6.2.1 Comparison to naive models

We compared Model 3 to two naive models to get a baseline. The first naive model was a `DummyClassifier` that predicted 0's and 1's with respect to the distribution of the training data. The second naive model simply trained on a shuffled Y-vector



(a) Receiver Operating Characteristic Curve for Model 3 on test data.

(b) Confusion matrix for Model 3 on test data.

Figure 6.1: The Receiver Operating Characteristics curve and confusion matrix for Model 3.

and this model we call the shuffled model. A comparison between these models can be seen in Table 6.1. It is clear that Model 3 outperforms both naive models. These results shows that our model has learned a good representation of the data and has predictive power.

The shuffled model seems to perform slightly better than the `DummyClassifier`, especially when it comes to recall. Since the precision is low for the shuffled model it means that the shuffled model predicts many of the ground-falsehoods as positive. Therefore, it seems like the shuffled model predicts ones to a higher extent than the `DummyClassifier` and thus gets a better recall but poorer precision. Our model also has a high recall at the expense of a poorer precision score, but with overall better performance. One reason behind this could be that the shuffled model and our model have similar architecture, where both are optimized based on the ROC AUC score, unlike the `DummyClassifier`.

6.3 Evaluation with predictors

When evaluating the predictor and calibration we used a calibrated score which combines the two probabilities, p_0 and p_1, we get as output from the calibration. We used this score instead of p_0 or p_1 as it captured the performance from the calibration and also because we did not notice a big difference. The median difference from all the samples is about 0.005, and suggests that the predictor is confident about its calibration.

Table 6.2: The precision, recall, f1 and ROC AUC (area under receiver operating characteristic curve) scores for our model's predictions as well as the calibrated scores. Where one classifier significantly outperforms the others, that score is in bold.

	Our model	**Predictor**
Precision	0.15	**0.44**
Recall	**0.84**	0.24
F1 score	0.26	0.31
ROC AUC	0.90	0.90

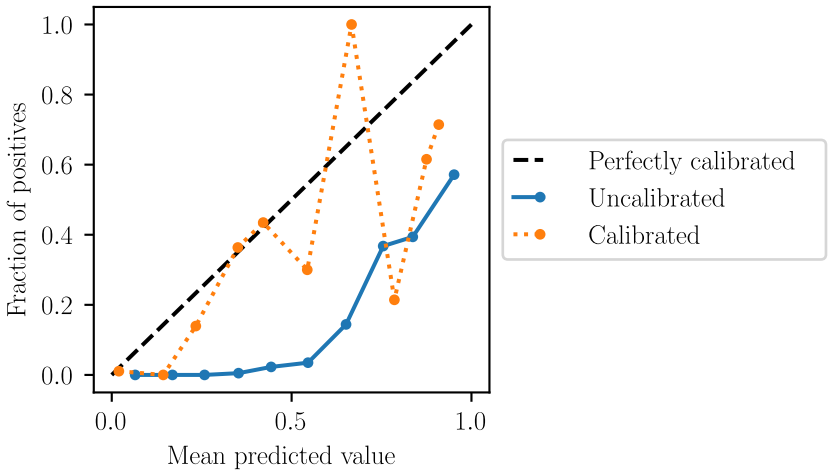
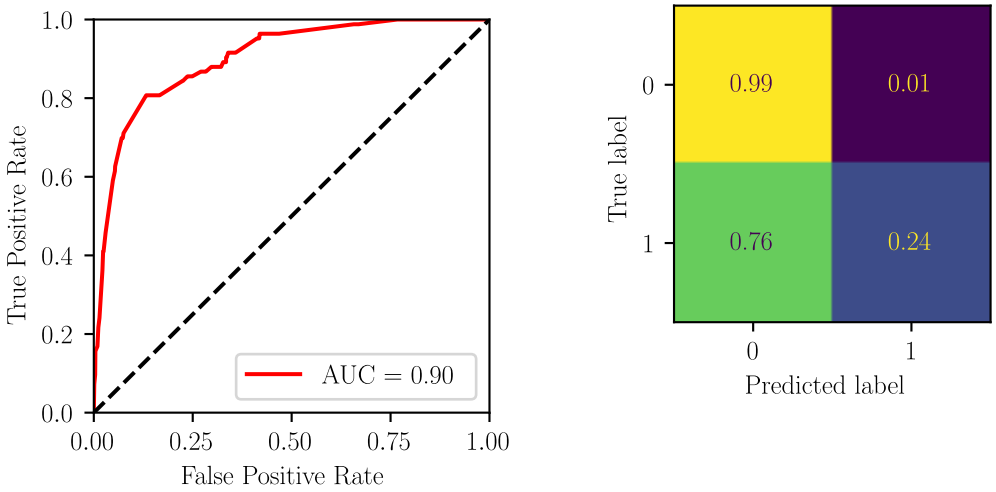


Figure 6.2: Calibration curve for the predictor applied on top of Model 3 on test data. The dashed line shows a perfect calibration. The blue solid line shows how well our model is calibrated. The yellow dotted line shows how well the predictor has calibrated the scores.

When the predictors are applied to the classifiers less training data is available for use since a calibration set is needed for the calibration. When training on a smaller training set, the performance is slightly worse. The new scores compared to the calibrated scores from the predictor can be seen in Table 6.2. The calibrated scores gives a much higher precision and slightly better f1 score, but an equal ROC AUC score and much worse recall.

The calibration curve can be found in Figure 6.2. Neither the model nor the predictor yields perfectly calibrated scores. Although, the calibrated scores are closer to perfect for scores below 0.6. It could be that the calibration gets worse for higher scores since there are few actual and predicted positives.

In the confusion matrix for the predictor (see Figure 6.3b) we can see that it retrieves almost all ground-falsehoods but hardly any of the ground-truths. The confusion matrix shows that the predictor calibrates the score to a lower score so that many of the examples are predicted as negative. Since retrieving the ground-truths is



(a) Receiver Operating Characteristic Curve for the predictor applied on Model 3 on test data.

(b) Confusion matrix for the predictor applied on top of Model 3 on test data.

Figure 6.3: The Receiver Operating Characteristics curve and confusion matrix for the predictor on Model 3.

important to our task, the calibration works worse for this reason.

The ROC curve for the predictor can be seen in Figure 6.3a. The area under the curve is 0.9, which is close to both our model (Figure 6.1a) and the optimal score. One reason behind the high ROC AUC score is that the model has a higher precision, but a lower recall than the classifier.

6.4 Evaluation of aggregate model

We also evaluated the aggregate model, consisting of the five classifiers. In Figure 6.4 you can see how often each context appear in the predictions and dataset respectively. All five contexts are close to each other in this plot which shows that no context is predicted or presented in the dataset that much more than the others. It is clear that all contexts are predicted more than they appear in the dataset, which is not a surprise when we are encouraging false positives, to some extent, as they could actually be correct.

6.4.1 Prediction of chemical library

We let the model predict a chemical context for a chemical library file consisting of 89 Buchwald-Hartwig reactions. We then compared this to the true context that had been used in practice for these reactions. The true context can not be found in our

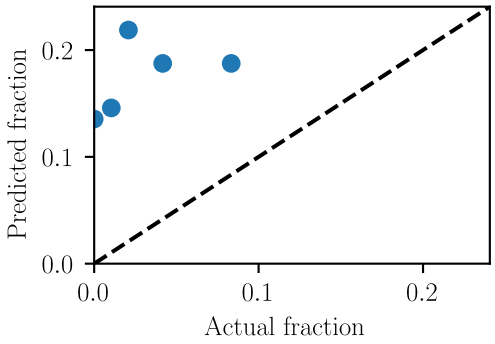


Figure 6.4: A scatter plot over how often each context appears in the dataset compared to how often it is predicted. Both numbers are fractions of how many times it is predicted or in the dataset in relation to the number of predictions. The dashed line shows when the predicted and actual fractions are equal.

top five most common contexts. This is to be expected since the chemical context that is used depends on multiple factors, such as when the variation was tested, who is running the experiment, and the availability of chemicals. It was therefore not likely that we would find the context used in our five contexts. The only molecule they had in common was the base, which some of the top five contexts had.

Our model predicts the second most common context and predicts that it will work for about 88% of these reactions. This context has no molecule in common with the true context. The predictor predicts the most common context, which has the same base as the true context.

It is difficult to draw any conclusions from these results as the true context for the library was not present in our data. We also do not know how well our predicted context would work for these reactions when tested, it could be equivalent or different in either direction. Because synthesis is time consuming, it was not possible to test our prediction in practice for this project. However, we have shown that the classifiers have a good performance which indicates that such a model could work in the lab.

7

Discussion

The models for both tasks showed predictive power and performed well on the evaluations. The algorithm adaptation model gets a label ranking average precision score of 0.8 and a top-1 accuracy of about 0.7 for single-label data. The binary relevance model got slightly worse results. We presented a novel model for condition prediction for chemical libraries. The classifiers got a ROC AUC score between 0.89 and 0.95. We also added Venn-ABERS predictors to the models to calibrate the scores. Unfortunately, the predictors did not calibrate the models successfully.

There are some issues left and these will be discussed and related to other research. We will also identify areas for future research. Last, we will discuss the ethical considerations for this project.

7.1 Dataset and modelling

The dataset was found to be both imbalanced and mostly single-label. Especially the predictors seems to have suffered from the imbalance in the dataset. More attempts could be done to handle this to see how that would affect the results.

One problem with the current approach, and data, is that all negative samples are seen as negative and we have no distinction between *tested and does not work* and *not tested*. It could be that some of these negative samples should be positive but we have no way of knowing that. One way we handled this was to not punish the models too much on false positives. We allowed a lower precision to achieve a higher recall for the condition prediction of chemical libraries model.

We limited our scope to only handle one reaction class, Buchwald-Hartwig coupling or amination. Buchwald-Hartwig is the most prominent reaction class in drug development, which is why we chose it. We also believe that it is important to show good results for this class especially. Our models are specialized on this reaction class and show good results on this data. We have not tested our model on any other reaction class, so we cannot draw any conclusions on how generalized our model is. It is possible that a more generalized model would be more useful than a specialized, like ours. This topic could be further explored and our approach could easily be extended to use more and other reaction classes.

7.2 Multi-label classification

One of the largest challenges to the multi-label condition prediction has been that the data has been more single-label than multi-label. It seems to have affected our results as the predictions are rather single-label as well. It would be interesting to see the results with a more multi-label dataset for this task.

Cerri *et al.* [18] evaluated different multi-label classification approaches on two datasets, where one was similar to ours when it comes to the low label cardinality. They also got low Hamming losses for the less multi-label data. They found that support vector machines (SVM) worked well for many approaches. The classifier we used for the binary relevance model was `SVC`, which is a type of SVM. Cerri *et al.* state that the algorithm adaptation models can lead to better results but also more complex models. But they also inform that similar results can be achieved with problem transformation approaches. We got slightly better results for the algorithm adaptation model, but the results were very similar. We suspect that it has something to do with one of two factors. The first is that neural networks are able to learn more complex feature spaces than support vector machines. The second is that the algorithm adaptation model is able to handle the low label density better since it always has at least one ground-true label.

The dataset, like many other multi-label datasets [35], was imbalanced. Handling imbalance in a multi-label dataset is more difficult than in a single-label dataset. Chartre *et al.* [35] state that oversampling works better than undersampling since deleting an instance can affect multiple labels appearing in that sample. They suggest oversampling approaches that takes their imbalance measures for multi-label datasets into consideration. For a binary relevance approach it can be handled better since the dataset can be split into several binary datasets. Stratification, thresholding the decision boundary, or oversampling/undersampling could have been applied to each individual dataset [45]. We did not try to handle the imbalance for either approach but it may have given better results.

We only compared two models for the multi-label classification task. Although, the results are promising, it is possible that other approaches would give even better results.

7.3 Model calibration

The model calibration with Venn-ABERS predictors does not work well for any of our models and it seems to be due to the imbalance. We might have gotten different results if we had used another method for calibration. For multi-label tasks, conformal predictors has been more used [24–27] and it is possible that a conformal predictor could work better for the multi-label classification task.

Venn-ABERS predictors uses isotonic regression which fits a non-decreasing function to the scoring function to calibrate the scores. An alternative is logistic regression presented by Platt *et al.* [46] as a way to get probabilities from SVMs, where a sigmoid function is applied to the scoring function instead. An advantage with

isotonic regression is that it can be applied on any distortion, while logistic regression is limited to the sigmoid function [47]. An disadvantage is that isotonic regression is prone to overfit since it needs more data to learn the more complex function [47]. For the models using SVMs it could be that logistic regression would be a better choice as they follow a sigmoid shape. Additionally, potential overfitting and ways to mitigate this could be investigated.

Adding Venn-ABERS predictors to our multi-label models did not give good results. One difficulty is that Venn-ABERS is intended for single-label tasks. While we managed to calculate scores for our binary relevance model, all of our metrics decreased as compared to not using a predictor. Additionally we noticed our calibration could only find calibrated scores for a small portion of the probability interval and that was scores closer to zero. In contrast, the model's scores ranged from 0 to 1. The poor performance could be due to low label density as well as the imbalance of the dataset, which would explain why the calibrated scores only were on the lower end.

The predictors work poorly for the condition prediction for chemical library task as well. It works poorly because it is essential to get as many of the true positives predicted positive as possible. One explanation for the poor performance could be that the data is imbalanced and there are very few true positives. Furthermore, the predictor is better than our model at retrieving the true negatives but worse in retrieving the true positives. It could be that the predictor has enough negative samples to get the negatives right but not enough positive samples to predict positives correctly. With more positive samples we might have gotten a better overall prediction.

7.4 Condition prediction

Most condition prediction research does not predict contexts, but rather predict each condition individually. Gao *et al.* [3] predicted each condition individually and got high accuracies for each individual task. They got significantly lower results on the task of finding the correct context with a top-3 accuracy of 57.3%. Our model got a top-1 accuracy of about 70% on single-label data. It indicates that it is easier to find exact matches when training with sets of conditions than using the individual predictions to create the set. On the other hand, predicting them individually could lead to more flexibility to predict what kind and how many conditions you need with a high accuracy. However, if the model will be used by a robot with no human intervention the complete context is needed. Another advantage with predicting the complete context is that you have more data of the molecules working together in a reaction. Otherwise, with every individual condition predicted you introduce an error which will propagate for the context. Another disadvantage would be that you cannot predict anything outside the contexts in the dataset such as a new context, but that would also mean that there is no data for that new context so it would not be useful in practice.

As far as we know, no condition prediction has been used for chemical libraries even though it would be useful in practice. Although we have not extensively evaluated our model as a tool for condition prediction of chemical libraries, it shows promising

results. The individual simple binary classifiers perform well, especially for retrieving the true positives.

7.5 Future work

We believe a comparison between predicting the conditions individually and as a set (chemical context) could improve condition prediction research further. The same dataset could be used, but pre-processed differently where one version (A) would have the conditions separately and the other version would have the conditions as sets or chemical contexts (B). One model would then be trained on A and another similar model trained on B. A comparison of the two models' performance would guide future research as to which option is most appropriate.

The multi-label classification task would benefit from more multi-label data. When there is more multi-label data, for example from PharmDev, it would be interesting to re-train the multi-label models and evaluate the results.

More research needs to be done on condition prediction for chemical libraries. Especially to evaluate it in practice. We also believe that incorporating a feedback loop as a form of active learning could be useful. In the lab, when a chemist chooses another context than the predicted one, the model can learn from this instance. As more and more experiments are done, the model will continue to learn. It would also be a way to keep the model up to date with what is actually used in the lab.

We also believe that predictors and model calibration in general can benefit from future research. It is an area with little research, especially regarding practical applications. One possible area of future work could be how predictors are affected by imbalanced data. Future work could also be focused on applying predictors to condition prediction and further experiment with it to see if it could work in practice.

7.6 Ethical aspects of optimizing reactions

There are some ethical considerations for this project. Tools, like the models developed in this project, for optimizing chemical reactions can potentially be used to create any compound. The target compound of the reaction could be one harmful to life on earth or the environment. In the wrong hands the tool can be used for developing chemical weapons instead of developing drugs to treat diseases. Also, since our models optimizes yield there could be side-effects for some reactions such as toxic byproducts. While we are aware of these ethical concerns, the scenarios do not seem very likely to us. Various research on optimizing reaction has been done without being used to harm people. In addition, our project has the potential of speeding up the drug development process which could help a lot of people. We believe the possible benefit to society outweigh the risk of doing harm.

8

Conclusion

The purpose of this thesis was to research how condition prediction can be improved. We hypothesized that condition prediction can be improved to be more useful to chemists with the use of multi-label classification. While we still believe that, our model is not predicting several chemical contexts, likely due to the mostly single-label data. In the future when there is more multi-label data available, the model could possibly learn to predict several contexts, and hence also be more useful.

We also hypothesized that a model suggesting one chemical context working well for many reactions would improve efficiency in a lab setting as several reactions could be performed simultaneously. Our model was only evaluated on one library file where the context they used for those reactions was not part of our data. Thus we cannot draw any conclusions about how our model would work in practice. This model is novel as we find no examples in the literature on solving this task with machine learning. We hope that these promising results could inspire to more research in this area.

We cannot conclude how these models would work as tools in the lab for chemists as they have not been evaluated in this setting. However, all models showed to be predictive and outperformed the naive classifiers. The algorithm adaptation model for the multi-label classification task performed slightly worse than the single-label model on single-label data. Therefore, it seems likely that these models could work in practice.

Our final hypothesis was that adding predictors on top of these models can provide useful information about the uncertainty which could guide chemists in their choices of which chemical context to use. We did not manage to prove this hypothesis for any of the two tasks, as they did not calibrate the predictions in a good way. Getting well calibrated probabilities would be useful and it might still be possible to achieve this. For example by using another type of predictor or processing the dataset in another way.

Lastly, we conclude that condition prediction would benefit from further research. Especially when it comes to predictors as they have yet not been used for condition prediction but would provide useful information.

Bibliography

[1] S. M. Paul, D. S. Mytelka, C. T. Dunwiddie, C. C. Persinger, B. H. Munos, S. R. Lindborg, and A. L. Schacht, "How to improve R&D productivity: the pharmaceutical industry's grand challenge," *Nature Reviews Drug Discovery*, vol. 9, no. 3, pp. 203–214, 2010, ISSN: 1474-1784. DOI: 10.1038/nrd3078. [Online]. Available: https://doi.org/10.1038/nrd3078.

[2] C. W. Coley, R. Barzilay, T. S. Jaakkola, W. H. Green, and K. F. Jensen, "Prediction of organic reaction outcomes using machine learning," *ACS Central Science*, vol. 3, no. 5, pp. 434–443, 2017. DOI: 10.1021/acscentsci.7b00064.

[3] H. Gao, T. J. Struble, C. W. Coley, Y. Wang, W. H. Green, and K. F. Jensen, "Using machine learning to predict suitable conditions for organic reactions," *ACS Central Science*, vol. 4, no. 11, pp. 1465–1476, 2018. DOI: 10.1021/acscentsci.8b00357.

[4] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke, "The rise of deep learning in drug discovery," *Drug discovery today*, vol. 23, no. 6, pp. 1241–1250, 2018.

[5] A. C. Mater and M. L. Coote, "Deep learning in chemistry," *Journal of chemical information and modeling*, vol. 59, no. 6, pp. 2545–2559, 2019.

[6] V. H. Nair, P. Schwaller, and T. Laino, "Data-driven chemical reaction prediction and retrosynthesis," *CHIMIA International Journal for Chemistry*, vol. 73, no. 12, pp. 997–1000, 2019.

[7] R. Buendia, T. Kogej, O. Engkvist, L. Carlsson, H. Linusson, U. Johansson, P. Toccaceli, and E. Ahlberg, "Accurate hit estimation for iterative screening using venn–abers predictors," *Journal of Chemical Information and Modeling*, vol. 59, no. 3, pp. 1230–1237, 2019, PMID: 30726080. DOI: 10.1021/acs.jcim.8b00724.

[8] B. Van Calster, D. J. McLernon, M. Van Smeden, L. Wynants, and E. W. Steyerberg, "Calibration: The achilles heel of predictive analytics," *BMC medicine*, vol. 17, no. 1, pp. 1–7, 2019.

[9] W. Chen, B. Sahiner, F. Samuelson, A. Pezeshk, and N. Petrick, "Calibration of medical diagnostic classifier scores to the probability of disease," *Statistical methods in medical research*, vol. 27, no. 5, pp. 1394–1409, 2018.

[10] P. Schwaller, T. Laino, T. Gaudin, P. Bolgar, C. A. Hunter, C. Bekas, and A. A. Lee, "Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction," *ACS central science*, vol. 5, no. 9, pp. 1572–1583, 2019.

[11] M. H. Segler and M. P. Waller, "Neural-symbolic machine learning for retrosynthesis and reaction prediction," *Chemistry–A European Journal*, vol. 23, no. 25, pp. 5966–5971, 2017.

[12] P. Schwaller, A. C. Vaucher, T. Laino, and J.-L. Reymond, "Prediction of chemical reaction yields using deep learning," *Machine Learning: Science and Technology*, vol. 2, no. 1, p. 015 016, 2021.

[13] Z. Fu, X. Li, Z. Wang, Z. Li, X. Liu, X. Wu, J. Zhao, X. Ding, X. Wan, F. Zhong, D. Wang, X. Luo, K. Chen, H. Liu, J. Wang, H. Jiang, and M. Zheng, "Optimizing chemical reaction conditions using deep learning: A case study for the suzuki–miyaura cross-coupling reaction," *Org. Chem. Front.*, vol. 7, no. 16, pp. 2269–2277, 2020. DOI: 10.1039/D0QO00544D.

[14] M. R. Maser, A. Y. Cui, S. Ryou, T. J. DeLano, Y. Yue, and S. E. Reisman, "Multilabel classification models for the prediction of cross-coupling reaction conditions," *Journal of Chemical Information and Modeling*, 2020. DOI: 10.1021/acs.jcim.0c01234.

[15] E. Ahlberg, S. Winiwarter, H. Boström, H. Linusson, T. Löfström, U. Norinder, U. Johansson, O. Engkvist, O. Hammar, C. Bendtsen, *et al.*, "Using conformal prediction to prioritize compound synthesis in drug discovery," in *The 6th Symposium on Conformal and Probabilistic Prediction with Applications,(COPA 2017), 13-16 June, 2017, Stockholm, Sweden*, Machine Learning Research, 2017, pp. 174–184.

[16] E. Ahlberg, L. Carlsson, A. Gammerman, V. Vovk, Z. Luo, E. Smirnov, and R. Peeters, "Using venn-abers predictors to assess cardio-vascular risk," 2018, pp. 1–15.

[17] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.

[18] R. Cerri, R. R. da Silva, and A. C. de Carvalho, "Comparing methods for multilabel classification of proteins using machine learning techniques," in *Brazilian Symposium on Bioinformatics*, Springer, 2009, pp. 109–120.

[19] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *European conference on principles of data mining and knowledge discovery*, Springer, 2001, pp. 42–53.

[20] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[21] G. Shafer and V. Vovk, "A tutorial on conformal prediction," *CoRR*, vol. abs/0706.3188, 2007. [Online]. Available: http://arxiv.org/abs/0706.3188.

[22] V. Vovk and I. Petej, "Venn-abers predictors," *arXiv preprint arXiv:1211.0025*, 2012.

[23] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 694–699.

[24] H. Papadopoulos, "A cross-conformal predictor for multi-label classification," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*, Springer, 2014, pp. 241–250.

[25] A. Lambrou and H. Papadopoulos, "Binary relevance multi-label conformal predictor," in *Symposium on Conformal and Probabilistic Prediction with Applications*, Springer, 2016, pp. 90–104.

[26] H. Wang, X. Liu, I. Nouretdinov, and Z. Luo, "A comparison of three implementations of multi-label conformal prediction," in *International Symposium on Statistical Learning and Data Sciences*, Springer, 2015, pp. 241–250.

[27] H. Wang, X. Liu, B. Lv, F. Yang, and Y. Hong, "Reliable multi-label learning via conformal predictor and random forest for syndrome differentiation of chronic fatigue in traditional chinese medicine," *PloS one*, vol. 9, no. 6, e99565, 2014.

[28] Y. Sunesson, E. Limé, S. O. N. Lill, R. E. Meadows, and P. O. Norrby, "Role of the base in buchwald-hartwig amination," *Journal of Organic Chemistry*, vol. 79, pp. 11 961–11 969, 24 Dec. 2014, ISSN: 15206904. DOI: 10.1021/jo501817m.

[29] L. David, A. Thakkar, R. Mercado, and O. Engkvist, *Molecular representations in AI-driven drug discovery: a review and practical guide*, Sep. 2020. DOI: 10.1186/s13321-020-00460-5.

[30] T. Stepišnik, B. Škrlj, J. Wicker, and D. Kocev, "A comprehensive comparison of molecular feature representations for use in predictive modeling," *Computers in Biology and Medicine*, vol. 130, Mar. 2021, ISSN: 18790534. DOI: 10.1016/j.compbiomed.2020.104197.

[31] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of Chemical Information and Modeling*, vol. 50, pp. 742–754, 5 May 2010, ISSN: 15499596. DOI: 10.1021/ci100050t.

[32] G. Landrum. (2021). "The rdkit book," [Online]. Available: https://www.rdkit.org/docs/RDKit_Book.html (visited on 02/05/2021).

[33] RDKit. (2021). "RDKit," [Online]. Available: http://www.rdkit.org/ (visited on 02/05/2021).

[34] S. R. Heller, A. McNaught, I. Pletnev, S. Stein, and D. Tchekhovskoi, "Inchi, the iupac international chemical identifier," *Journal of Cheminformatics*, vol. 7, 1 May 2015, ISSN: 17582946. DOI: 10.1186/s13321-015-0068-4.

[35] F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera, "Addressing imbalance in multilabel classification: Measures and random resampling algorithms," *Neurocomputing*, vol. 163, pp. 3–16, 2015.

[36] H. Patel, M. J. Bodkin, B. Chen, and V. J. Gillet, "Knowledge-based approach to de novo design using reaction vectors," *Journal of Chemical Information and Modeling*, vol. 49, pp. 1163–1184, 5 May 2009, ISSN: 15499596. DOI: 10.1021/ci800413m.

[37] N. Schneider, D. M. Lowe, R. A. Sayle, and G. A. Landrum, "Development of a novel fingerprint for chemical reactions and its application to large-scale reaction classification and similarity," *Journal of Chemical Information and Modeling*, vol. 55, pp. 39–53, 1 Jan. 2015, ISSN: 15205142. DOI: 10.1021/ci5006614.

[38] F. Chollet *et al.* (2015). "Keras," [Online]. Available: https://github.com/fchollet/keras (visited on 02/22/2021).

[39] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines."

[40] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," Jul. 2012. [Online]. Available: http://arxiv.org/abs/1207.0580.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014. [Online]. Available: http://arxiv.org/abs/1412.6980.

[42] A. Buja, W. Stuetzle, and Y. Shen, 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.184.5203&rep=rep1&type=pdf.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[44] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[45] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, and X. Geng, "Binary relevance for multi-label learning: An overview," *Frontiers of Computer Science*, vol. 12, no. 2, pp. 191–202, 2018.

[46] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[47] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 625–632.

A

Glossary

Reagent: A substance that is added to aid a chemical reaction.

Condition: A reagent or other property, such as temperature, that is controlled to aid a chemical reaction.

Reactant: Molecules that are present before a reaction has occurred.

Product: The resulting target molecule from a reaction.

Yield: Percentage of target molecule that could be obtained related to the quantity of reactants in the reaction. A measurement of how effective or successful a reaction was.

Chemical context: A set of conditions or reagents used in combination for a reaction. For example ligand, solvent(s), catalyst(s) and pre-catalyst(s). In this thesis, it is the chemical context that is predicted.

Chemical library: A chemical library is a collection of compounds and their known properties. To obtain the chemical library, compounds are synthesized via reactions, often in parallel. A number of M reactants which have something in common such as a common scaffold are combined with N other reactants. When these react we will get $M \times N$ product molecules to the library at the same time.

B

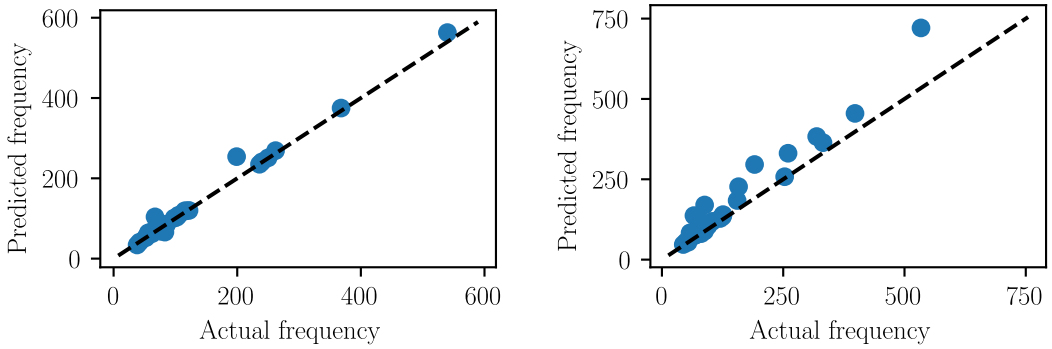
Training results for multi-label models

Here follows the training results for both multi-label classification models.

In Figure B.1 we can see that the binary relevance model over-predicts in training while the algorithm adaptation model follows the dashed line closely.

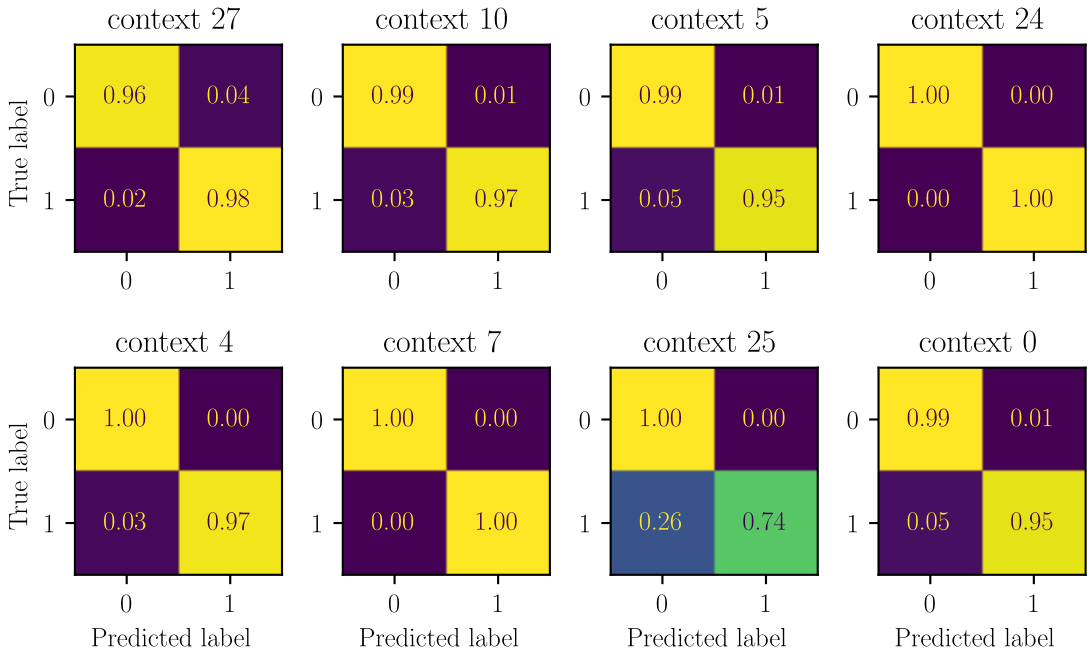
The confusion matrices of the models in Figure B.2 shows that the model has learned the training data, at least for the ten most frequent contexts. The binary relevance model has more or less perfect scores for many of the confusion matrices.

The False Positive Ranking Inclusion (FPRI) (see Definition 3.1) can be seen in Figure B.3. For both models, the predictions on the training data rarely includes many false positives. The algorithm adaptation model performs somewhat better than the binary relevance model in this metric.

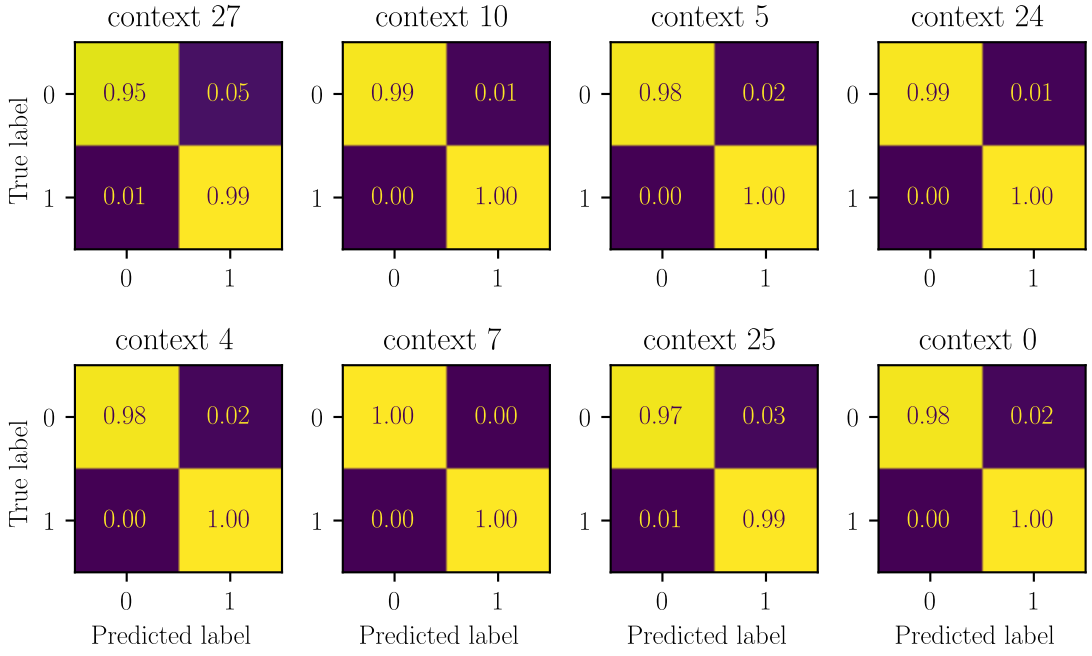


(a) Algorithm adaptation model **(b)** Binary relevance model

Figure B.1: Plots how the predicted distribution of contexts behaves in comparison to the actual distribution on the test data. One point shows how many times that context appears in the actual and predicted data. The dashed line represents a frequency that is the same in the actual and predicted data.

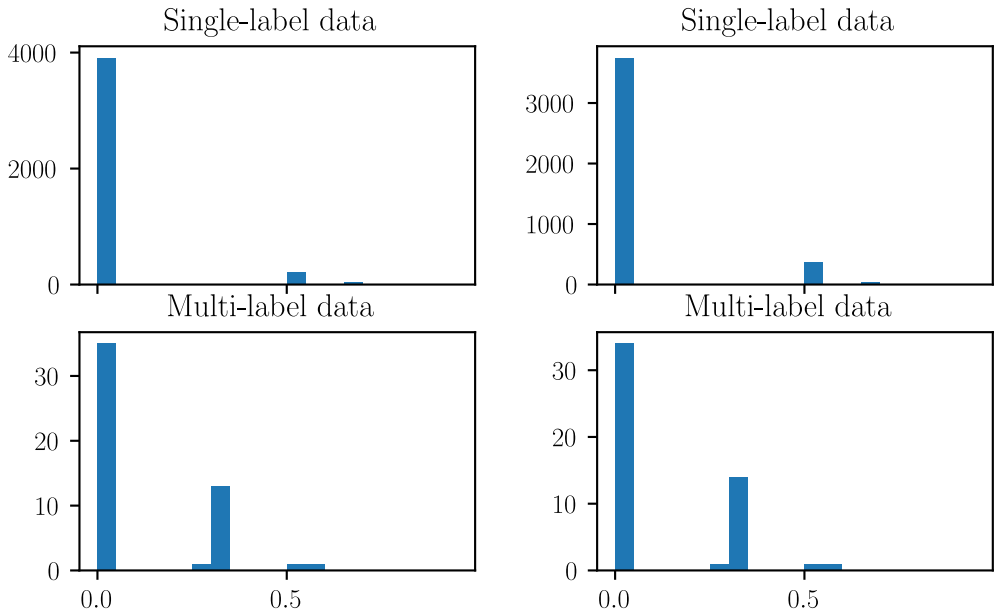


(a) Algorithm adaptation model



(b) Binary relevance model

Figure B.2: Confusion matrices, one per context, for the ten most frequent contexts in the dataset. They are all normalized with regards to the true labels. Optimal matrices would have ones in the top left and bottom right cells, as this would mean all predictions are true positives and true negatives. Yellow cells show a value close to one and purple values close to zero.



(a) Algorithm adaptation model **(b)** Binary relevance model

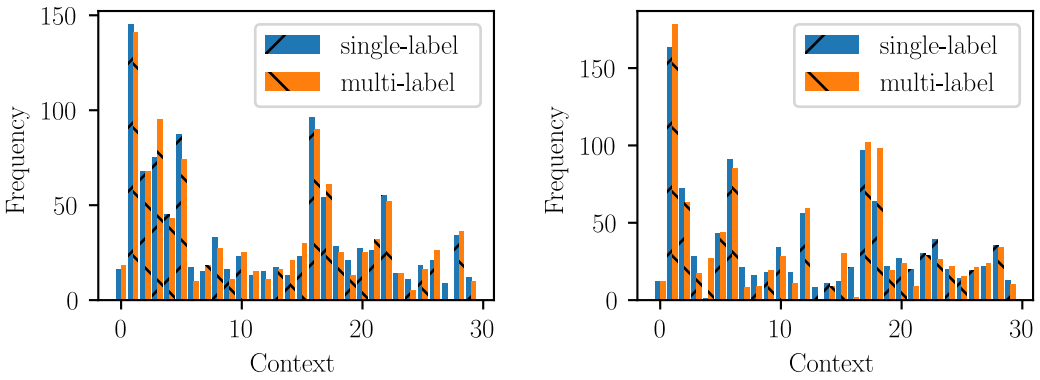
Figure B.3: Plot of the distribution of scores for the predictions where the score is calculated using Definition 3.1. This score is on the x-axis and is shared for the top and bottom plots. It shows how large the fraction of false positives we have per prediction if all true positives are included based on the `sigmoid` score for each context. An optimal value is zero as that is when our top predictions are all true positives and no false positive. The worst score is close to one where one of the true contexts is the lowest predicted. The top plot is for the scores where there is only one true context, and the bottom plot is for the cases where there is more than one true context.

C

Comparison to single-label model

For the comparison to the single-label model we looked at the distribution of predicted contexts by the two models as well as the distribution of actual contexts in the two datasets.

Figure C.1 shows the actual and predicted context distribution. It is clear from the figure that the contexts appear with roughly the same frequency in both datasets. The predicted distribution differs a bit more. For most contexts the frequency is similar, although there are a few contexts that appear many more times in one or the other set.



(a) Actual **(b)** Predicted

Figure C.1: The context distribution (predicted and actual) for the single label and multi-label algorithm adaptation model respectively. It shows how many times each context is present in the dataset and predictions.

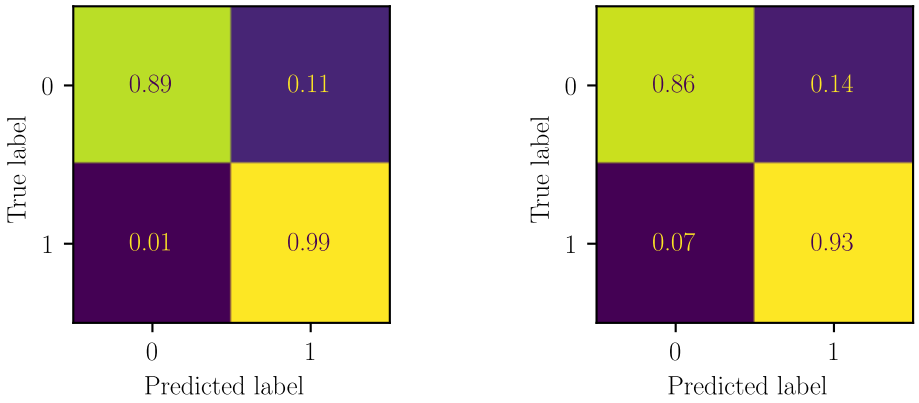
D

Results from all classifiers for condition prediction on chemical libraries

The performance of all the classifiers used to solve the condition prediction on chemical libraries task is shown in this appendix.

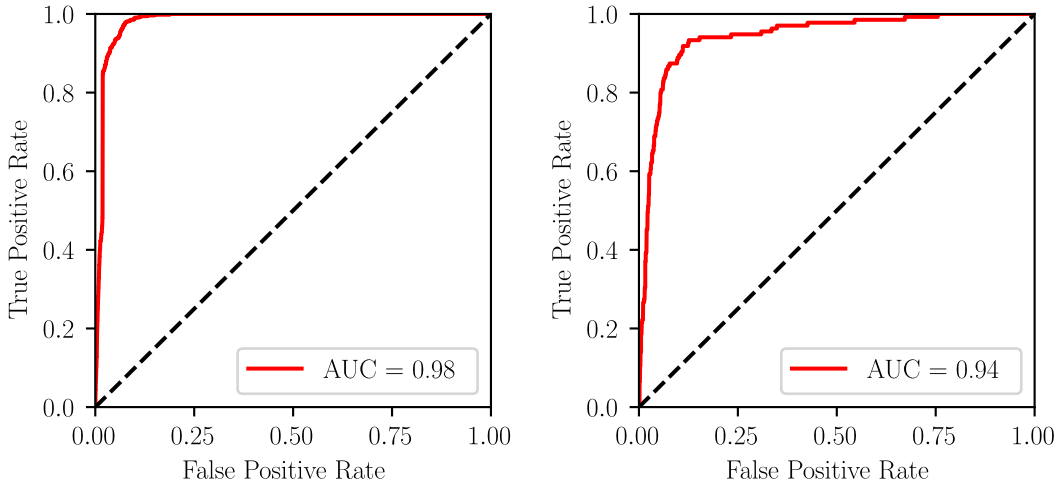
For model one see Figures D.1 and D.2 and for model two see Figures D.3 and D.4. Figures D.5 and D.6 shows the performance of model three that was chosen as representative to be analyzed in the main text. Lastly, model four's performance can be seen in Figures D.7 and D.8 and model five's performance in Figures D.9 and D.10.

The models all perform well and have quite similar performance. They all show a slightly better performance on training data than the test data, where some even predicts all ground-truths in the training data correctly. The overfitting of the classifiers could be due to the imbalance of the dataset.



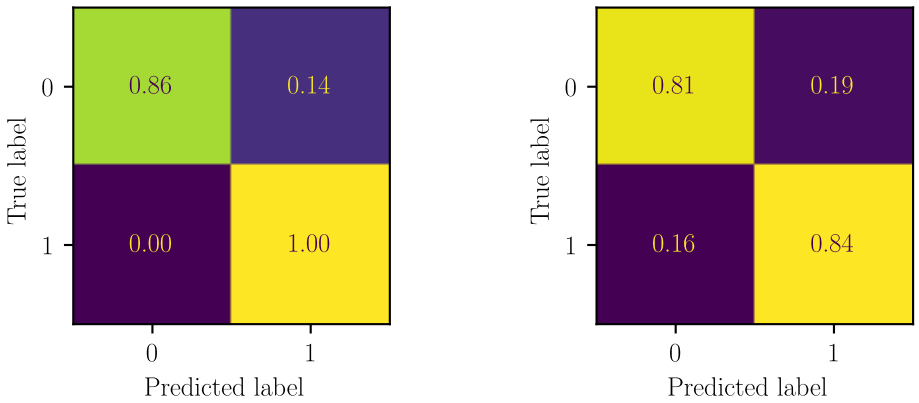
(a) Training **(b)** Test

Figure D.1: Confusion matrix for the first model. It is normalized with regards to the true labels. An optimal matrix would have ones in the top left and bottom right cells, as this would mean all predictions are true positives and true negatives. Yellow cells show a value close to one and purple values close to zero.



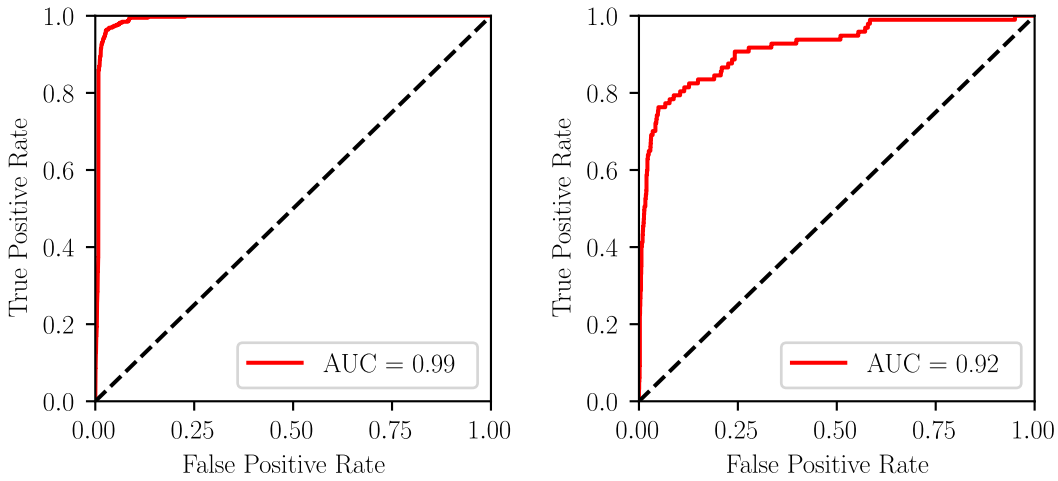
(a) Training **(b)** Test

Figure D.2: Receiver Operating Characteristic Curve for the first model. The area under the curve (AUC) is 1 if the model makes all predictions correctly. The dashed line forms a baseline of 0.5 AUC when the model is as accurate as a coin flip.



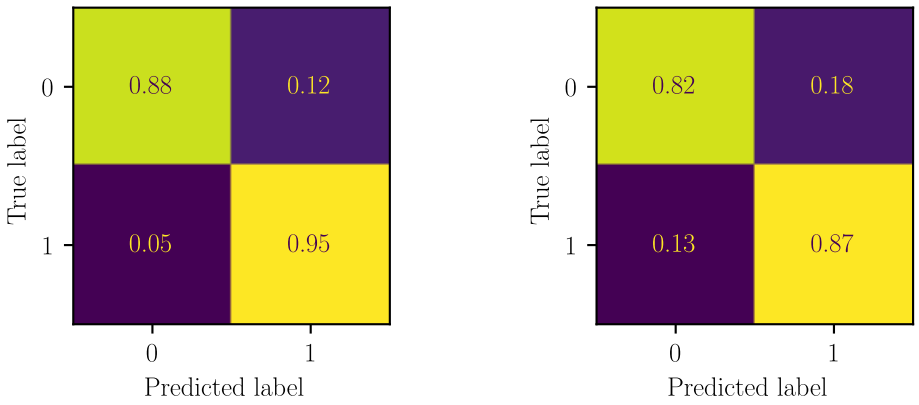
(a) Training **(b)** Test

Figure D.3: Confusion matrix for the second model. It is normalized with regards to the true labels. An optimal matrix would have ones in the top left and bottom right cells, as this would mean all predictions are true positives and true negatives. Yellow cells show a value close to one and purple values close to zero.



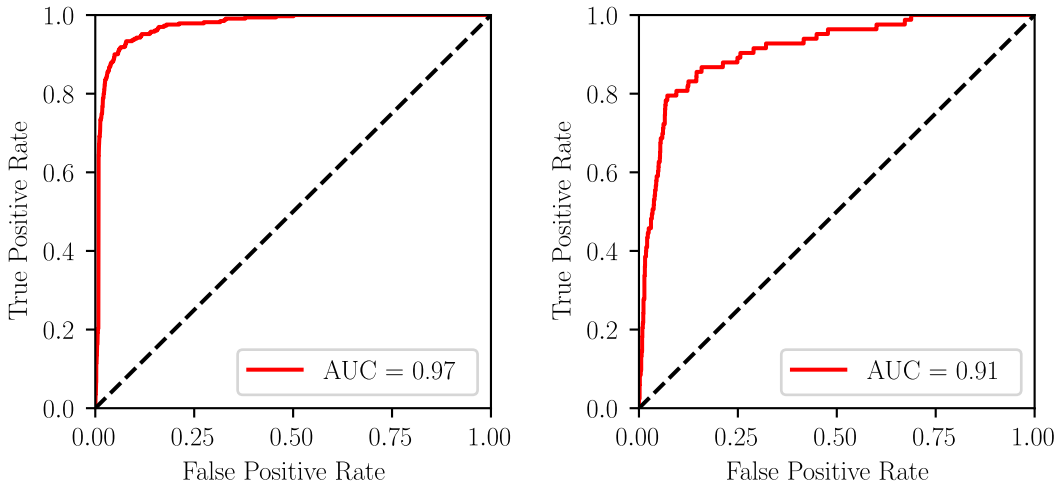
(a) Training **(b)** Test

Figure D.4: Receiver Operating Characteristic Curve for the second model. The area under the curve (AUC) is 1 if the model makes all predictions correctly. The dashed line forms a baseline of 0.5 AUC when the model is as accurate as a coin flip.



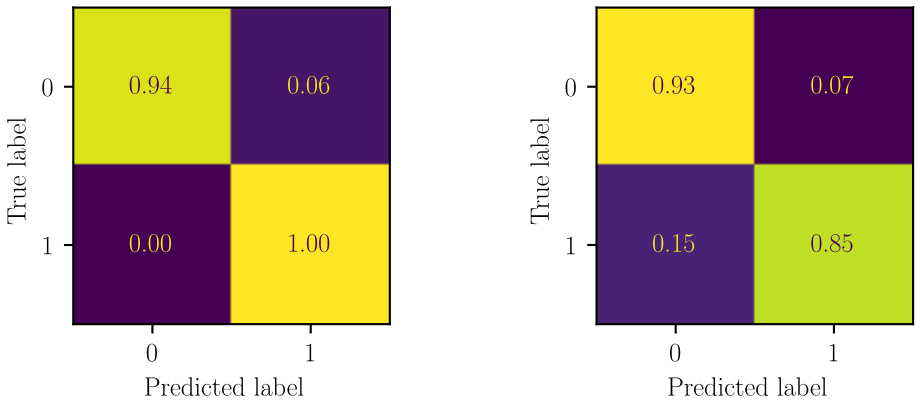
(a) Training **(b)** Test

Figure D.5: Confusion matrix for the third model. It is normalized with regards to the true labels. An optimal matrix would have ones in the top left and bottom right cells, as this would mean all predictions are true positives and true negatives. Yellow cells show a value close to one and purple values close to zero.



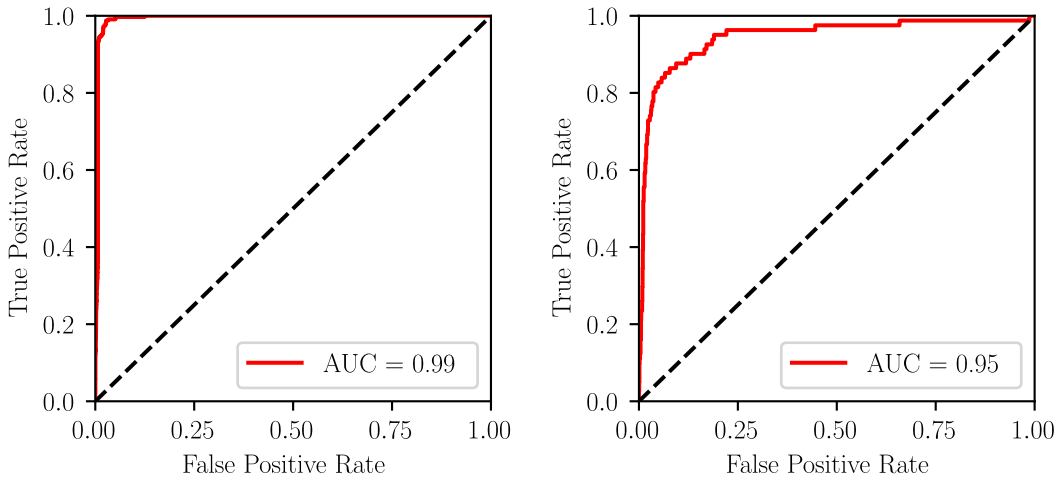
(a) Training **(b)** Test

Figure D.6: Receiver Operating Characteristic Curve for the third model. The area under the curve (AUC) is 1 if the model makes all predictions correctly. The dashed line forms a baseline of 0.5 AUC when the model is as accurate as a coin flip.



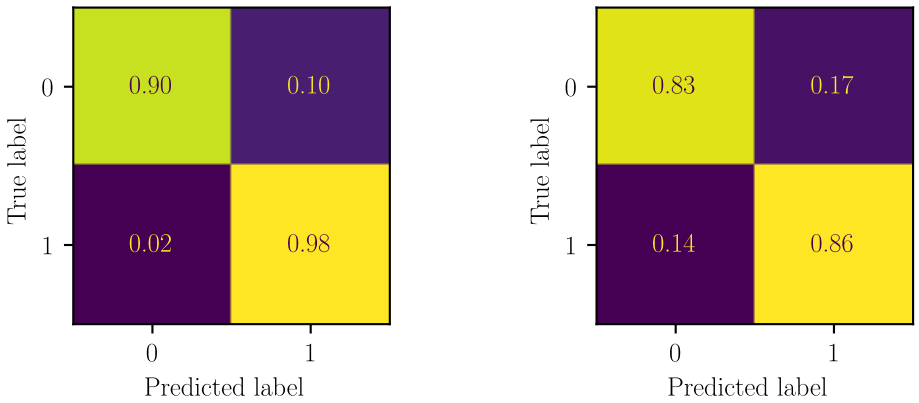
(a) Training **(b)** Test

Figure D.7: Confusion matrix for the fourth model. It is normalized with regards to the true labels. An optimal matrix would have ones in the top left and bottom right cells, as this would mean all predictions are true positives and true negatives. Yellow cells show a value close to one and purple values close to zero.



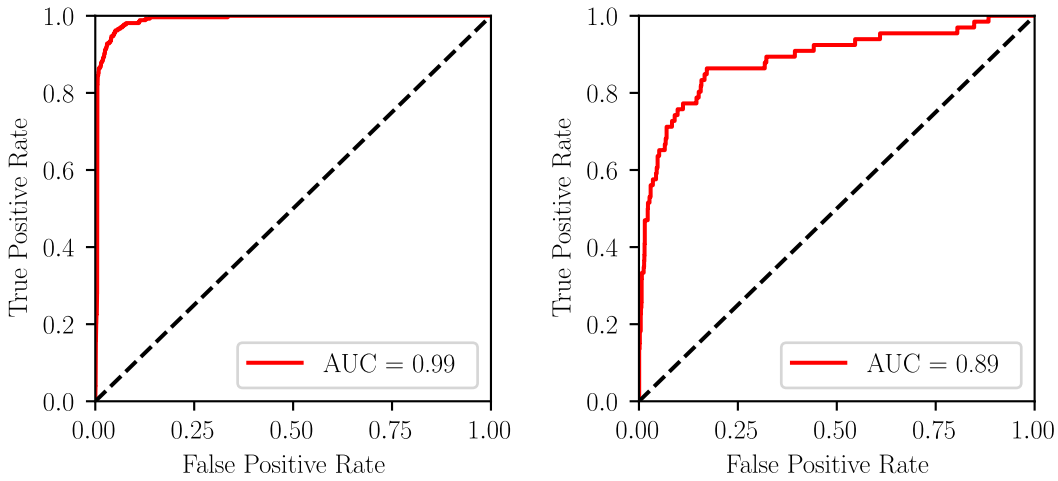
(a) Training **(b)** Test

Figure D.8: Receiver Operating Characteristic Curve for the fourth model. The area under the curve (AUC) is 1 if the model makes all predictions correctly. The dashed line forms a baseline of 0.5 AUC when the model is as accurate as a coin flip.



(a) Training **(b)** Test

Figure D.9: Confusion matrix for the fifth model. It is normalized with regards to the true labels. An optimal matrix would have ones in the top left and bottom right cells, as this would mean all predictions are true positives and true negatives. Yellow cells show a value close to one and purple values close to zero.



(a) Training **(b)** Test

Figure D.10: Receiver Operating Characteristic Curve for the fifth model. The area under the curve (AUC) is 1 if the model makes all predictions correctly. The dashed line forms a baseline of 0.5 AUC when the model is as accurate as a coin flip.

E

Results from all predictors for condition prediction on chemical libraries

Here follows the results from the calibration with Venn-ABERS predictors for the condition prediction on chemical libraries. Each predictor is evaluated in terms of the calibration curve, receiver operating curve (ROC) and confusion matrix.

A calibration curve shows how well the scores or probabilities are calibrated, where a perfect calibration (marked with a dashed line) is where the predicted score matches the fraction of positives for that score. The blue solid line shows how well our model is calibrated. The yellow dotted line shows how well the predictor has calibrated the scores.

The receiver operating characteristic curve shows how well a binary classifier can distinguish between positives and negatives. The area under the curve (AUC) is 1 if all predictions are correct. The dashed line forms a baseline of 0.5 AUC where the predictions are as accurate as a coin flip.

A confusion matrix shows the fraction of true positives, false negatives, true negatives and false positives. The confusion matrix is normalized with regards to the true labels. An optimal matrix would have ones in the top left and bottom right cells, as this would mean all predictions are true positives and true negatives. Yellow cells show a value close to one and purple values close to zero.

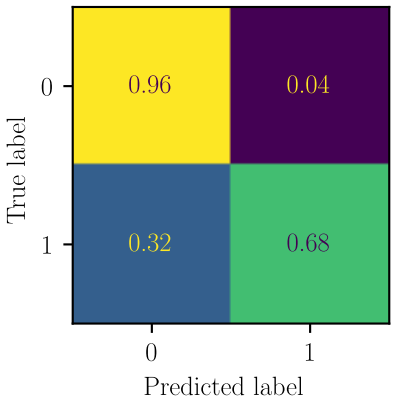
These plots can be found in Figure E.1 for predictor 1, Figure E.2 for predictor 2, Figure E.3 for predictor 3 (evaluated in the main text), Figure E.4 for predictor 4 and finally, Figure E.5 for predictor 4.

From the confusion matrix, we can see that the predictors performs worse when it comes to retrieving the ground-truths compared to the classifiers (see Appendix D). We can also see in the calibration curves that the predictors lack calibrated scores above 0.8 or 0.9 which also suggests that. Moreover, the true negatives are predicted to a higher extent than the underlying classifiers. It could be due to the data imbalance that the predictor calibrates well in regards to true negatives and badly in regards to true positives.

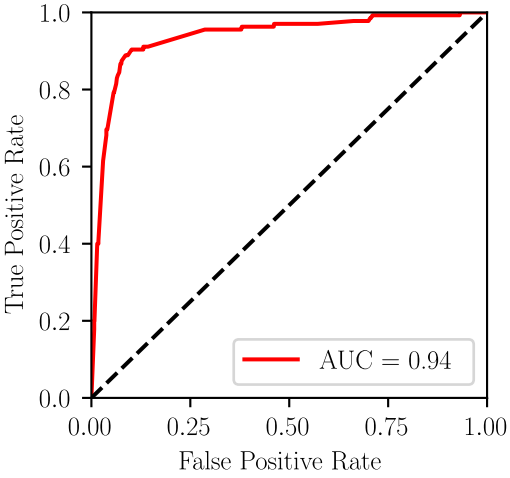
The performance for the predictors varies quite a bit, for example predictor 1 looks rather well calibrated in comparison to predictor 2 or 3. It is unclear why the

performance differs between the predictors as it doesn't seem to be fully connected to the label imbalance and the performance for the classifiers does not vary as much.

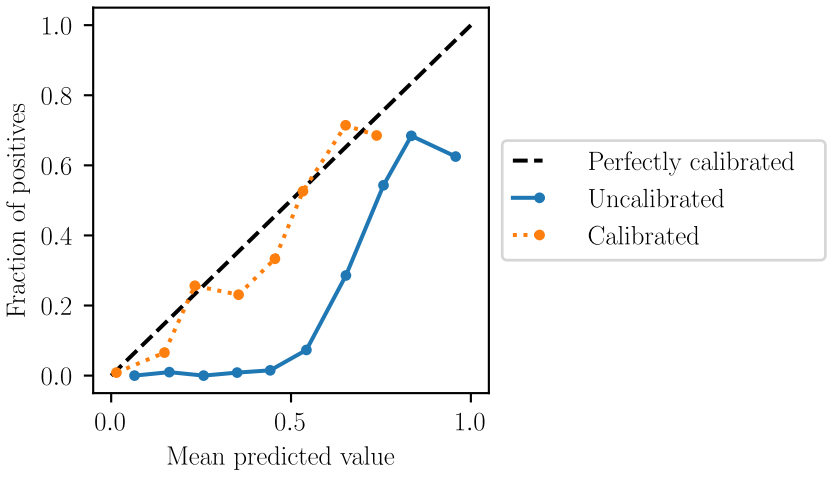
In general, the calibration seems to work to some degree but not very well in our case because the imbalance of the data as well as the importance of predicting the ground-truths as positives.



(a) Calibration curve

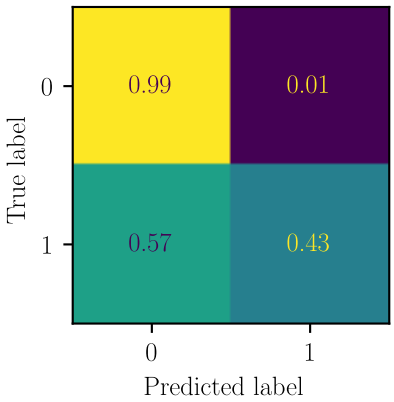


(b) ROC curve

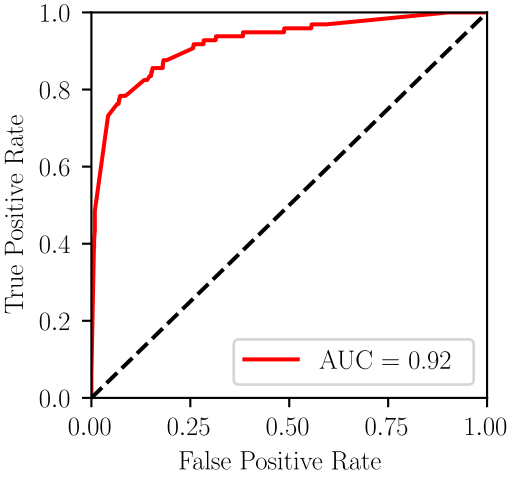


(c) Confusion matrix

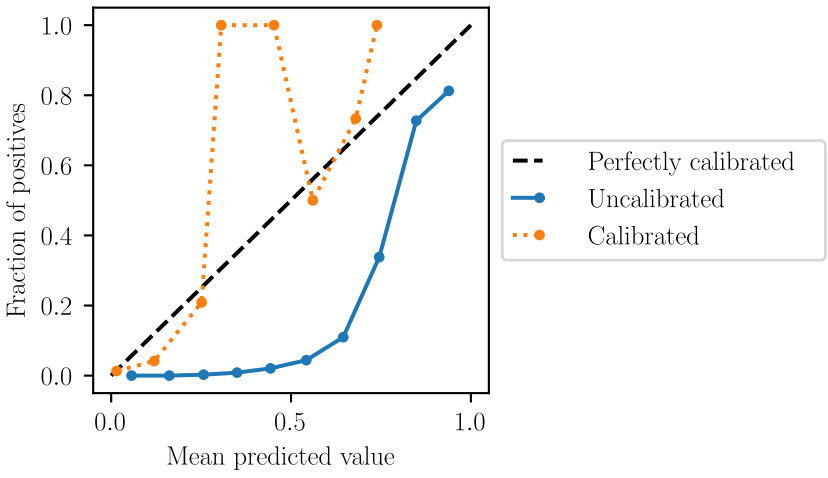
Figure E.1: Performance of predictor 1



(a) Calibration curve

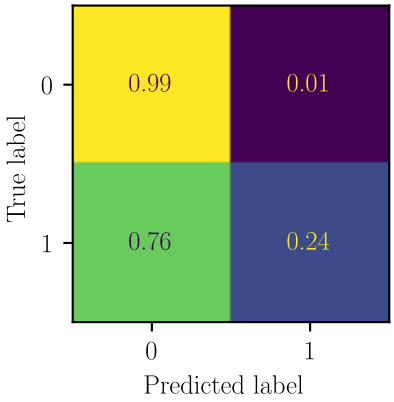


(b) ROC curve

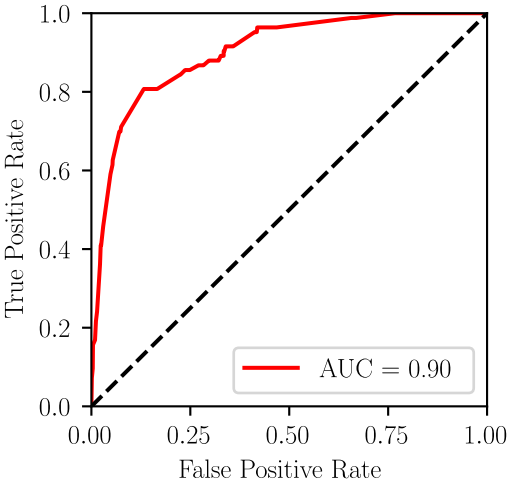


(c) Confusion matrix

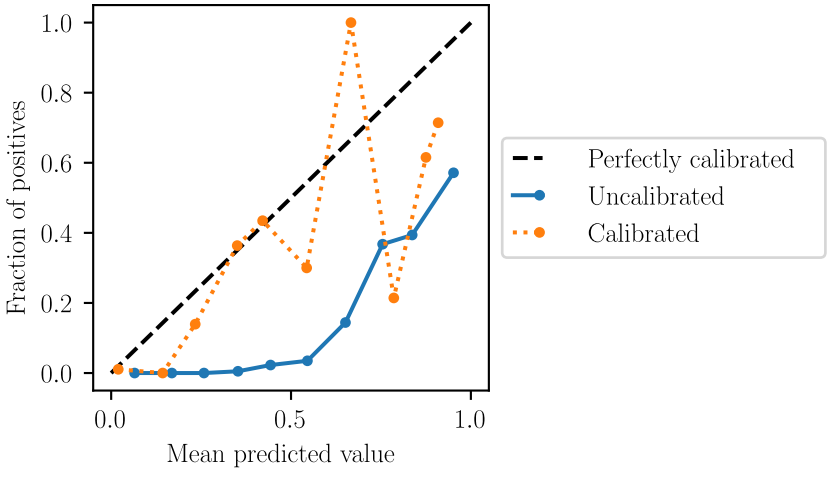
Figure E.2: Performance of predictor 2



(a) Calibration curve

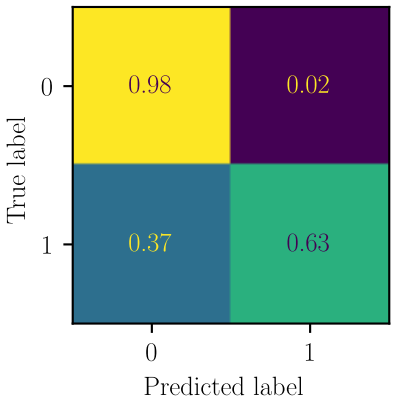


(b) ROC curve

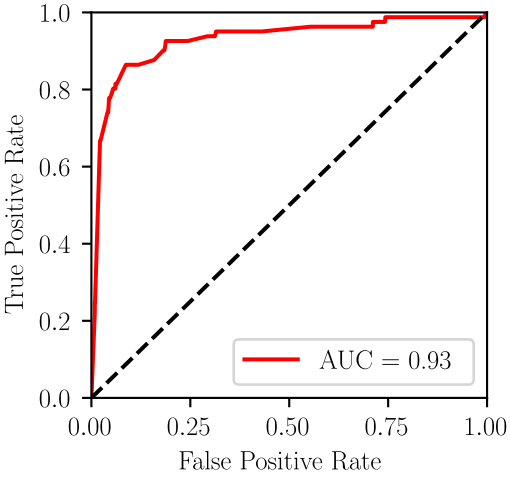


(c) Confusion matrix

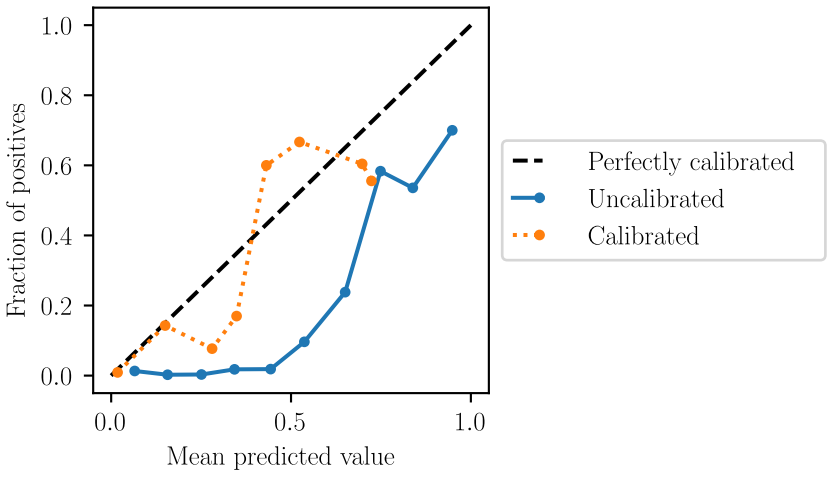
Figure E.3: Performance of predictor 3



(a) Calibration curve

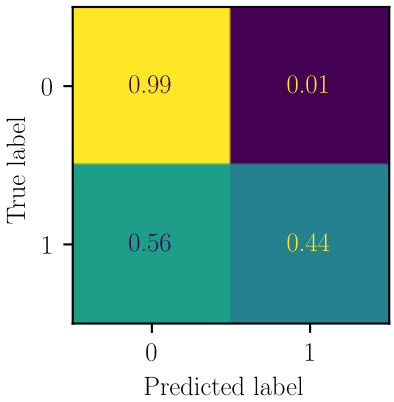


(b) ROC curve

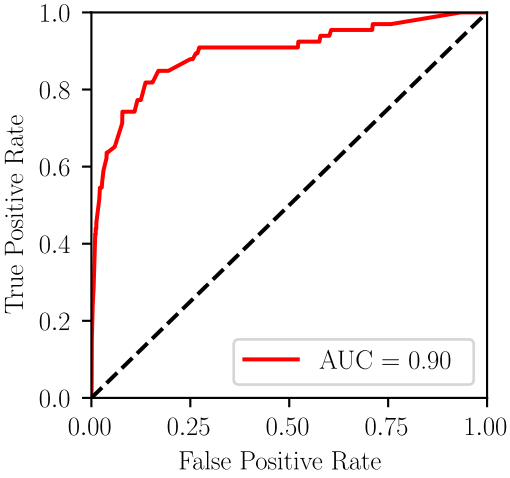


(c) Confusion matrix

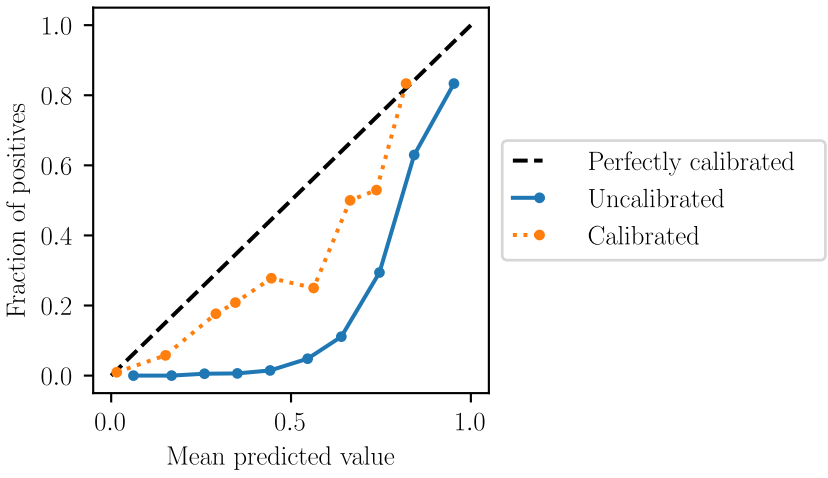
Figure E.4: Performance of predictor 4



(a) Calibration curve



(b) ROC curve



(c) Confusion matrix

Figure E.5: Performance of predictor 5