Utrecht University

Thesis Project

Computing Science

Using Active Learning to Mitigate Sampling Bias: a Comparison of Two Algorithms

Supervisors:
Dr. ing. G.M. (Georg) Krempl

Dr. Albert Wong

Dr. Emiel Rorije

Dr. Ka Yin Leung

Co-reader:
Dr. M. (Thijs) van Ommen

Author:
Hannah Gathu

May 16, 2022

1 Abstract

In machine learning we often have to work with imperfect data. One way in which data can be imperfect is through sampling bias. When a data sample is biased, it doesn't accurately represent the population. This makes it challenging to train a well-generalizing and unbiased classifier, which can cause a machine learning algorithm to be unfair. A lot of techniques have been produced to deal with this challenge. These vary from pre-processing techniques, which focus on mitigating bias in the data before beginning training, to in-process techniques, which alter the training process to reduce bias, to post-processing techniques. In this thesis we compare two active learning algorithms. One of them is an in-process technique that is specifically designed to target sampling bias. We compare this technique to a more conventional simpler active learning algorithm. Based on experiments on benchmark data, we compare how both algorithms perform in terms of improving the performance of the classifier, making the sample more closely resemble the "population" and other factors. We find that the conventional active learning algorithm actually seems to outperform the sampling bias targeted algorithm on most judging criteria and for most settings. Besides that, we also do an experiment on biased toxicology data from the RijksInstituut voor Volksgezondheid en Mileu(RIVM).

Contents

1 Abstract **1**

2 Introduction **2**

3 Background and Related Work **3**
3.1 Sampling Bias 3
3.2 Measuring Sampling Bias 6
3.3 Mitigating Sampling Bias 7
3.3.1 Semi-supervised Learning 8
3.3.2 Active Learning 9
3.3.3 Semi-supervised Active Learning 10
3.4 Evaluation in Presence of Sampling Bias 11
3.5 Toxicology 12
3.5.1 Use of Machine Learning 13
3.5.2 Sampling Bias 15

4 Method **16**
4.1 Experimental Setup 16
4.2 Bias Simulation 17
4.3 Bias mitigation 17
4.4 Evaluation 19

4.5 Experimental parameters 20
4.6 Data Preparation 20
4.7 Toxicology Experiment 21

5 Data **21**
5.1 Toxicology Data 21
5.2 Benchmark Data 22

6 Results **24**
6.1 Performance metrics 24
6.2 Sample Distribution 24
6.3 Development of Class Imbalance 31
6.4 Toxicology Experiment 31

7 Discussion **46**
7.1 Benchmark Experiments 46
7.2 Toxicology Experiment 49
7.3 Limitations 50
7.4 Future Work 51

8 Conclusion **52**

A Toxicology Experiment **56**

2 Introduction

In this day and age, AI algorithms are all around us. They provide a lot of benefits and solutions to problems, but they also come with risks. AI algorithms have the potential to amplify or even create bias. One way in which bias can manifest itself in AI algorithms is in the form of sampling bias. Sampling bias refers to a bias in the collection of the (labelled) training data, which results in a sample that is not representative of the population.

When data that are used to train machine learning models suffer from sampling bias, this can result in unfairness. It can generate and amplify all kinds of biases, like bias based on race, gender, socio-economic status or age.

The chemicals risk assessment department of the RIVM would like to benefit from machine learning techniques in order to predict whether certain substances could be a potential risk to human health. The department would like to train prediction models to make predictions about whether a substance is harmful or find relevant features that are particularly predictive. Prediction models rely on random sampling to work best. Unfortunately, the chemical risk assessment department currently only has access to a dataset that does not have a random label distribution but a presumably biased one. Therefore, it is tough to trust how a prediction model based on the current data would do when it comes across entirely new substances. The most obvious solution would be to simply add more

labels in the unknown areas, but this is not easily feasible. The reason for this is that labeling an instance would require (further) testing which is costly and time-consuming. Other than that, there is also the ethical problem that we do not want to sacrifice animals for experiments that confirm what we already suspect (that a substance is safe). To improve the current model and its chances of successfully classifying entirely new substances, we need to look at smart ways to gain more information about the grey/looked over areas in the feature space, using the data that is available.

A lot of techniques have been produced to deal with this challenge. These vary from pre-processing techniques, which focus on mitigating bias in the data before beginning training,to in-process techniques, which alter the training process to reduce bias, to post-processing techniques. In this thesis we will take a look at two active learning algorithms that can be used as in-process sampling bias mitigation techniques. One of them is a 'conventional' active learning algorithm that just uses the uncertainty of instances to decide which instances to add to the sample. The other was specifically designed to combat sampling bias and takes adds another criterion to the uncertainty criterion. This criterion is specifically designed to make the sample more closely resemble the test set or population, thus mitigating sampling bias. We will test these techniques on benchmark data and see how well they perform against different degrees and types of bias.

3 Background and Related Work

A lot of existing literature about sampling bias focus on fairness and discrimination. Most of it deals with data from gathered from surveys or databases about humans. So it seems as though many of the literature about sampling bias comes from the human sciences field as opposed to the natural sciences field, which is where our application is situated.

However, there is some literature that are more situated in the natural sciences. In this thesis we will test some of the bias mitigation techniques mentioned in this literature to see how they hold up against different degrees of bias.

A lot of papers about sampling bias talk about cases where certain parts of the feature space are underrepresented in the sample (e.g. the percentage of people with a certain ethnicity is a lot smaller in the sample than in the population). However, it seems like not a lot of the literature deals with situations where some parts of the feature space are not represented at all in the sample, i.e. some parts of the feature space (that does contain instances in the population or test data) is not represented at all in the (labelled) sample.

3.1 Sampling Bias

An assumption that is common when training a supervised classifier, is that the training data and the data to be classified have the same underlying distribution. There are a lot of applications however, for which this assumption cannot be

fulfilled due to a lack of complete control over the way in which data is gathered or a lack of resources that are necessary for 'perfect' data collection. In the application of credit scoring for instance, the labelled data will consist purely of credit applications that have been accepted, because it is generally not possible to observe the repayment behaviour of a rejected applicant[10]. In such cases the training data suffers from sampling bias. In this section we will define sampling bias, discuss its effects and its impact on the fairness of a classifier.

Definition Most classifier learning algorithms assume that we have instances (x, y) that are drawn independently from a distribution D with domain $X \times Y$ where X is the feature space and Y are the possible labels[5]. To model a case in which sampling bias is present, Zadrozny instead assumes that instances (x, y, s) are drawn independently from a distribution D with domain $X \times Y \times S$ where S is $\{0, 1\}$. The variable s is 1 if instance x is part of the sample and 0 if it is not. In [5], Zadrozny mentions the following four ways s can depend on the instance (x, y):

1. If s is independent of both x and y, the sample is not biased.
2. If s is independent of y given x, i.e. $P(s|x, y) = P(s|x)$, the sample is biased but the bias only depends on the feature vector x. This is called co-variate shift[6].
3. If s is independent of x given y, i.e. $P(s|x, y) = P(s|y)$, the sample is biased but the bias only depends on the label y. There are methods for correcting this type of bias, which are discussed in
4. The fourth type of sampling bias is bias for which there is no independence assumption between x, y and s. It is more difficult to correct this type of bias, unless there is a known feature vector x_s for x that determines whether or not an instance is selected. I.e. $P(s|x_s, x, y) = P(s|x_s)$ for all instances.

Effects of Sampling Bias There seems to be consensus among papers on the fact that bias or sampling bias has negative effects in machine learning. However, few of them elaborate on what these effects are or prove that it has a negative effect. Zadrozny[5] does formally and mathematically show these effects. They do this specifically for the case where the sampling bias is only caused by the features, as in the second definition from the previous paragraph. They separate learners into two categories:

- local: the output asymptotically (i.e. as the size of the training set increases) only depends on $P(y|x)$
- global: the output asymptotically depends on $P(x)$ as well as $P(y|x)$.

They chose the terms "local" and "global", because $P(x)$ is a global distribution over the entire feature space, while $P(y|x)$ consists of many local distributions

for different feature vectors x. Local learners are not affected by the covariate shift type of sampling bias, because it doesn't affect $P(y|x)$. Global learners are affected however, because with this type of sampling bias, $P(x)$ is biased. In the paper Zadrozny show both mathematically and experimentally to which category, local or global, some popular learning methods belong. However, since the paper by Zadrozny only looks at the effects of sampling bias that is purely caused by the features, we can't rely on it to predict the effects of sampling bias in our setting, since the bias in the toxicology data is presumably caused by more than just the features. There are also papers in which they empirically show that the presence of bias has a negative impact on the classifier. In [18] for instance, Persello and Bruzzone add four different degrees of sampling bias to a Remote Sensing data set. Their experiments show that as the bias increases, the accuracy of the classifier decreases. In the experiment where the instances for the training set are randomly chosen, the resulting model has an accuracy of 95 percent on the testing set. As the training set becomes more biased, the accuracy goes down from 95 to around 70 percent and eventually the trained model is accurate in less than 60 percent of the cases.

Is sampling bias always bad? As mentioned in section 3.1, the holy grail of ways to collect training data is usually random sampling. However, there are cases in which random sampling is not feasible or not even the best way to get a well performing classifier.

As they highlight in [1], there are nuances to the sampling bias problem. In case of a class imbalance for instance, it could be that you get a better model by over-representing a certain group in comparison to the population.

Fairness When we are dealing with data on humans that contain features that are prone to discrimination, sampling bias can have an impact on the fairness of an AI algorithm. Often times a biased AI algorithm simply shows the bias that is already present in society, but it can also amplify or create bias, because the algorithms prefer easily quantifiable aspects of human behaviour over those which are hard or impossible to measure[11]. On top of that, some data is easier to access and use than others (this often manifests itself in the form of sampling bias). In particular, the ever-growing presence of social media in our society has provided a very easy way to collect and aggregate large amounts of data on humans. According to Tufekci [20], this has caused a qualitative shift in the scale, scope and depth of data analysis. A few platforms, like twitter, are often used to generate datasets without adequately taking its structural biases into account. Data sets that are gathered from sources like twitter often suffer from sampling bias, because they only include data on twitter users, which can be directly biased through sensitive features like socio-economic status, race and gender.

The discriminative impact that AI-systems can have has already been made apparent in several cases. For instance, the COMPAS system, which predicts the chance of re-offending, was found to be racially biased. Angwin et al. found that

COMPAS overestimated the risk values for black defendants, while at the same time underestimating the risk values for white defendants[19]. An example of gender bias in AI-systems is found in Google's Ads tool for targeted advertising, which was found by Datta et al. to show significantly fewer advertisements for high paid jobs to women than to men[21].

It seems like there had been a lot of debate about how to exactly define fairness. In [11], they state that a lot of formal mathematical definitions had been given in the previous years, but the problem of formally defining fairness was still open as discussion about the pros and cons of these measures was still missing. In [12], Corbett-Davies and Goel show the limitations and negative effects of using mathematical definitions of fairness and advice to focus on the consequences of such interventions. It seems like Sandra et al. Sandra et al.[13] share the opinion that formal or automatic measures of fairness do not suffice.

3.2 Measuring Sampling Bias

The most common way to measure sampling bias across the literature is to compare the feature and label distributions of the sample with that of the data to be predicted. This approach is sufficient to make sure your model performs well on your training data. However, if you want your model to be as unbiased as possible and to be able to extrapolate to different datasets in the field, you would have to compare aforementioned distributions of the sample to those of the population. However, it is rarely possible to know the exact distribution of the whole population, so instead we opt to compare it to a dataset that is as representative of/similar to the population as possible. In summary, the distribution that you compare it to needs to be as unbiased and representative of the population as possible.

There are several ways to compare the feature distributions. One way to do this is to compare the percentages of instances with certain values for a feature, like they do in table 1 and figure 2 of [1] and figure 4 of [2]. In the toxicology dataset, all features are continuous and numerical, so we could do something similar by looking at the percentages of instances that fall within certain intervals of feature values. For continuous values, you could also compare means to each other, like Fry et al. do in table 3 of [1]. When you do this, it is important to keep in mind the standard deviation. Another way to compare feature distributions is to split a certain feature into value intervals and plot it against some other feature, like they do in figure 3 of [1]. In figure 1 of an article by Richards et al.[3], feature distributions are compared by drawing scatter plots for two features of the sample and the testing data in one diagram and seeing how similar they are.

All aforementioned techniques to compare feature distributions are in the end a matter of eyeballing. There is no formal measure of when two plots differ enough from each other to be able to definitively say that there is a sampling bias present. The area might benefit from defining formal measures for sampling bias in addition to empirical ones. In [18] for example, Persello et al. use the

Jensen-Shannon divergence to measure the degree of bias in the training sets they compile for their experiments. The Jensen Shannon divergence is a method of measuring the similarity between two probability distributions. Persello and Bruzzone use it to measure the similarity between each of the training sets and the test set. A downside to this method could be that is doesn't take class imbalance or other desired bias into account. As mentioned in section 3.1, in case of a class imbalance in the test set or population a sampling bias towards the minority class could be desirable. If we purposefully oversample the minority class, the Jensen-Shannon divergence between the training and testing set will be large but we won't know whether this is completely caused by the desired sampling bias or there is undesirable bias involved.

3.3 Mitigating Sampling Bias

Most of the literature about bias or sampling bias focuses on mitigating bias. Many compare different methods for mitigating bias. In this section we will discuss some of these techniques.

A technique that is often used to mitigate sampling bias is reweighting. A reweighting technique adds different weights to different instances to make a model more balanced and less biased. For probabilistic classifiers for instance, a commonly used technique to deal with sampling bias is sample-reweighted loss minimization. In 2014 , Liu and Ziebart[16] adapted this technique to be able to handle more severe cases of sampling bias. In 2018 [14], Jeong et al. proposed a technique called GDF weighting for Neural Network interatomatic Potentials (NNP's) that they say effectively corrects sampling bias. It uses the Gaussian Density Function to calculate the density around a training instance and to reweight it accordingly. In 2019, Iosfidis and Ntousi[15] noted that the previously presented reweighting techniques for improving fairness were largely focused on maintaining accuracy. They noticed that many of the fairness-related datasets suffered from a class imbalance, in which case accuracy is not a good performance measure because it is biased towards the majority class. Therefore, they created AdaFair, a fairness-aware classifier that extends AdaBoost by updating the weights in a way that takes into account fairness as well as the classification error in each iteration. The classification error is not solely based on the accuracy but is calculated in a way that takes class imbalance into account.

However, Richards et al. conclude that in case of severely biased data, where there are voids or extremely undersampled areas in the training data, reweighting techniques would not cut it. They state that any classifier that is only trained on the training data will perform poorly in those areas and that no weighting method on the available training data can sufficiently improve the obtained results. Another popular way to mitigate bias in classifiers is by using techniques that are targeted at specific prejudices or features that cause bias. An example of a technique that focuses on removing prejudice is introduced in [17]. Kamishima et al. propose a technique that integrates a regularizer into a classifier to reduce

the effect of indirect prejudice. The purpose of the prejudice remover regulizer is to make sure the classification is independent of sensitive features (such as race, gender, religion etcetera). However, the method Kamishima et al. propose only works for discriminative probabilistic models such as logistic regression and naive Bayes, which makes it not universally applicable in settings where prejudice has to be removed from a classifier.

3.3.1 Semi-supervised Learning

Semi-supervised learning is a machine learning method that is between supervised and unsupervised learning. Just like supervised learning, it learns from labelled data, but it also tries to make the most of unlabelled data[7]. The main purpose of semi-supervised learning is to use the structure of the unlabelled data in the feature space to get a better idea of the distributions of the classes and to improve the classifier if it were only trained on the labelled data by also incorporating unlabelled data in the learning process. A lot of semi-supervised learning techniques can be found in the literature. The main semi-supervised learning methods can be grouped into the categories 1) self-training, 2) co-training, 3) generative probabilistic models, 4) semi-supervised SVM and 5) Graph-based semi-supervised learning. The book [7] goes into detail about these methods. We will briefly discuss some of these main methods in more detail:

(a) *Self-training* is one of the earliest introduced ways to use unlabelled data in classification[7]. First, it trains a classifier on the labelled data. At each iteration a part of the unlabelled data is labelled according to the current classifier and added to the training data. Usually, the unlabelled instances that are chosen by the self-training algorithm are those that have the most confident prediction. After adding the new instances and their labels to the training set, the classifier is retrained and the process repeats. With self-learning, you can use any supervised learning method to train your classifier.

(b) *Co-training* is a technique that involves training two classifiers, each using only part of the features, and letting them learn from each other. Co-training is based on two assumptions: 1) the features can be split into two sets that are conditionally independent given the class and 2) each of those sets is sufficient to train a decent classifier[8]. It starts by training two different classifiers on the training data, each using their own unique subset of the features. Then, for both classifiers, the unlabelled data is classified and the algorithm chooses the instances with the most confident labels. Those instances and their predicted labels are then added to the training set of the other classifier. After that, both classifiers are re-trained. This process repeats itself until either both training sets now contain all previously unlabelled instances or some other stopping criteria are met.
Co-training can also be performed with more than two subsets and corresponding classifiers. In that case, it is called multi-view learning[9]. With multi-view learning, the feature set is split into more than two subsets of

features (which are called views) that are used for training different classifiers. The algorithm than tries to jointly optimize all classifiers using the information obtained by the other classifiers. This is often done by combining the predictions of all other classifiers to label the instances to be added to the training set of a classifier.

3.3.2 Active Learning

The idea behind active learning is that a machine learning algorithm can learn more from the data it is trained on if instances are added to the training data based on how much information they add. Active Learning is often used to try to learn more from less data[4]. An active learner asks an oracle to label instances that it deems most valuable to learn from. The active learning algorithm iteratively chooses the instances that would provide most information, and asks you to label them. After they are added to the sample, the AL algorithm again chooses the most valuable instances and this process repeats until certain stopping criteria are fulfilled (e.g. you can't label more instances, the performance isn't improving anymore, the performance is sufficient etc.)

There are several different query strategies that can be used to determine which instance labels most improve the learner. The most used are pool-based scenario's, in which the learner selects the most valuable instances from a pool of unlabelled instances. One of the most common pool-based scenario's is uncertainty sampling. When an active learner uses uncertainty sampling, it queries the instances for which the label provided by the supervised classifier learner carries the least confidence.

It might feel counter-intuitive to let a learning algorithm pick which instances to add to your sample, since a common notion in Machine Learning is that random sampling is best. Therefore, active learning is mostly used in situations where random sampling is not feasible, or where only few instances can be labelled. Reasons for this could be that labeling instances is expensive, time-consuming or hard.

You might not think that the solution to sampling bias, i.e. not having a random sample, is to carefully select instances to add instead of randomly adding instances. You might even worry that carefully selecting instances to add to a biased sample would result in even more bias. However, in [3], Richards et al. state that this is not the case.

Richards et al. use AL to overcome sampling bias in variable star classification. When it comes to star data, some stars are easier to observe and label than others. They state that the stars that are well-understood and studied a lot are usually those that are brighter and more nearby, whereas there is a lack of knowledge and data on stars that are fainter or further away. This results in a bias in the data and a significant discrepancy between the labelled (training) data and the testing data. They state that this systematic difference between the training and testing data causes supervised classification methods to perform poorly, especially on the instances that are undersampled among the training data. In the variable star

data they use, some regions of the feature space are not at all represented in the sample, which causes "catastrophically bad extrapolation of the model in these regions".
One of the methods they use to overcome this problem, is AL. As mentioned above, a common method to find the most valuable instances to label is by uncertainty labeling. Richards et al. agree with this, but in addition to instances with low uncertainty, they believe the most valuable instances to add are those that lie in dense regions of the testing data feature space and scarce areas of the training data feature space. The reason for this is that they believe that in order for AL to be successful in mitigating sampling bias, it should be used to make the training set most closely resemble the set that we want to classify. Richards et al. introduce two new criteria for selecting the most valuable instances to label. The first criterion is aimed at selecting the instance of the testing set whose feature density is most undersampled by the training set. The criterion compares the proximity of a point to the instances in the test set to its proximity to the instances in the training set. If this ratio is high, this means that the point is undersampled in the training data and therefore would be valuable to add to the sample. The second criterion adds the uncertainty factor to this, which means that testing points that are harder to classify get higher weight.

The results obtained by using AL to mitigate sampling bias are promising. In [3], they conclude that "AL produces hugely significant improvements in performance within only a handful of iterations". The RF trained on the default training data has an accuracy of 65.5% whereas after nine AL iterations, RF gives an accuracy of 79.5%. For one of the star categories, the accuracy even rose from 1% to 70%. The percentage of testing data in which the classifier is confident (they define this as having a probability of over 0.5 percent to belong to a certain class), increases from 14.6% to 42.9%.

Richard et al. also test how their AL mitigation algorithm compares to an AL algorithm that only looks at the density of the sample versus the test set to query instances. They find that their algorithm performs a lot better. However, what they don't do is test their algorithm against the 'original' AL algorithm with only the uncertainty score, to see if their addition of the density score even improves upon AL with just uncertainty sample. This would be good to test, because it would be good to know whether this addition, which makes the algorithm more complex and time-consuming, is even worth it. This is why we make this comparison in this thesis.

3.3.3 Semi-supervised Active Learning

As illustrated by the results discussed in the previous paragraph, active learning can work well to mitigate sampling bias. However, there are settings where it is not feasible to let an oracle label instances that are queried by the algorithm. In order to still use active learning in such cases, we can combine it with semi-supervised learning.

Semi-supervised learning and active learning alike try to exploit unlabelled

data to improve the classifier. There is a difference in the way in which they attempt this however. One of the differences between the two is that in active learning, the queried instances are labelled by an expert. Therefore, we are able to select the instances with the most uncertain prediction, because we can completely rely on the expert to label them correctly. In semi supervised learning settings however, the classifier cannot be considered perfectly reliable in labeling the queried instances. Therefore, the query function used in AL and SSL are often based on very different, or sometimes even opposite criteria. In self-training for example, the algorithm usually selects the instances with the most confident labels, whereas in AL the instances with the least confident labels are often queried.

In [20], a self-training technique for the classification of hyperspectral images is proposed.Persello et al. propose a technique that integrates AL concepts into Semi-Supervised Learning. The technique that they end up with uses AL approaches in the query function, but still remains an SSL technique because it doesn't make use of an external party to classify new instances. The method operates in two steps: in the first step, confident candidate unlabelled samples are selected on the basis of spatial and spectral information; in the second step, an AL method is adopted for selecting the most informative samples among the candidate ones to be included in the training set. The technique they approach uses a Progressive semi-Supervised Support Vector Machine, which is an adaptation for self-learning specifically for SVM's.
Usually self-learning adds the most confident instances to the sample. Persello et al. integrate a diversity criterion (i.e. choosing instances that lie far away from each other in the feature space by looking at the kernel cosine-angular distance between points) into this step, which is a concept that is common in AL. Because in AL adding new instances to the sample has costs, they often incorporate a diversity criterion to make sure that you don't label redundant instances. In SSL this is usually less of a concern, because it is basically free to add new labels since they are given by the classifier. In case of sampling bias however, this is a clever criterion to add, because it lowers the chance of amplifying the bias through self-learning.

3.4 Evaluation in Presence of Sampling Bias

The COMPAS example that was mentioned in section 3.1, illustrates that accuracy isn't always a good performance measure in presence of sampling bias.

In supervised learning, a model is often evaluated by estimating the error on the testing set. A common assumption that is made when making such an estimation, is that the training and testing sets follow the same probability distribution[6]. When our training set suffers from sampling bias, this is not the case. Therefore, standard generalization error estimation methods such as cross-validation don't do well at estimating the error and result in poor model selection. In [5], Zadrozny even states that accounting for sampling bias in the evaluation of the model is

more important than accounting for sampling bias during the learning process.

In [6], Sugiyama et al. propose a method to mitigate the bias in the estimation of the error of a classifier when the training data suffers from covariate shift. Covariate shift is a specific form of bias in which the feature distributions of the training and test data differ, but the chance of an instance having a certain label given its feature vector are the same for the training and test set. The method they propose for achieving an unbiased estimate of the error on the testing set is Importance Weighting(IW)[6]. The idea of importance weighting is that we make sure that instances from dense regions of the testing feature space that are underrepresented in the training set count more towards the error than instances that are overrepresented by the training set. They do this by weighting instances of the training set by the ratio of feature density of the testing and training set. Specifically, they weight the training data by

$$w_i = \frac{P_{Test}(x_i, y_i)}{P_{Train}(x_i, y_i)} = \frac{P_{Test}(x_i)P_{Test}(y_i|x_i)}{P_{Train}(x_i)P_{Train}(y_i|x_i)} = \frac{P_{Test}(x_i)}{P_{Train}(x_i)}[6]. \quad (1)$$

Here, x_i is the feature vector of instance i and y_i is the label for instance i. An important thing to note is that this equation only holds when we assume that the probability of a specific instance to have a certain label is the same in the training set and the test set, i.e., $P_{Test}(y_i|x_i) = P_{Train}(y_i|x_i)$. This equation holds in a covariate shift setting but not necessarily in every sampling bias setting.

They use this weighting method to adjust the formula for k-fold cross validation, which is a commonly used technique for estimating the error of a classifier. K-fold cross validation works by repeatedly (k times) taking out a chunk of the training set, training the classifier on the remaining training set, calculating the error on the chunk that was left out and after doing this k times (with k disjoint, non-empty subsets), taking the average over the k errors. To adjust this calculation to account for bias, they simply weight all k calculated errors by their weight, which is calculated using equation (1). The rest of the formula is left unchanged.

Sugiyama et al. prove that this method does well at estimating the true error on the testing set if the probability density around an instance is known, but in practice, we don't know the exact value of this probability density p. We can, however, estimate it via several techniques. One of the methods Sugiyama et al. use is a Gaussian kernel density estimator.

3.5 Toxicology

Toxicology is the study of chemicals that have adverse effects on biological systems and the environment[22]. It focuses on the effects of such substances as well as how to diagnose and treat them. It is a scientific field that integrates principles and methods of fields such as biology, chemistry, pharmacology and medicine. Nowadays, chemicals are present in many things that we use in our day to day live. The foods we consume are often produced with additives, pesticides and fertilizers that can end up in the environment and the organisms that live in it,

including humans. Apart from foods, chemicals are also present in things like toys, furniture, prescriptive drugs, paints and much more. Because some of these chemicals can be harmful to living organisms or the environment, the study of these chemicals and regulation of usage is of high importance. The relationship between the dose and duration of exposure to a certain chemical and its effects on an organism is one of the main focuses in toxicology.

The toxicology field has traditionally relied upon in vivo tests, i.e. tests with live organisms, to determine the toxicity of chemicals. Things they test for include death, reproductive issues, reduced growth and more. For environmental toxicology they typically use organisms like fish and for testing the adverse effects on humans they typically use animals such as rats or rabbits The advantage of these traditional testing approaches is that they are very reliable, but the downside is that they are very time-consuming, costly and not very desirable from an ethical point of view. As a result of that, just a very small portion of the chemicals that humans and the environment are exposed to are fully tested in this traditional way.

This "too many chemicals, too little data" data gap has been recognized for nearly four decades. Computational toxicology was seen as a potential solution to help solve this problem by maximizing the data that is available. Upon initiative by the EPA (U.S. Environmental Protection Agency), the NRC (U.S. National Research Council) wrote the 2007 report "Toxicity Testing in the 21st Century"[23]. They found that a fundamental shift from animal testing being the main support for chemical safety decisions toward wide application of in vitro testing and predictive toxicology was needed. According to the report, this shift would result in rapid data generation, lower costs to obtain toxicity data and more targeted and hypothesis-driven toxicity studies. This report moved the EPA to form a partnership called Tox21, which aimed at developing efficient in vitro screening technology for toxicity testing.

The EU has also taken steps in regulating the use of chemicals. This is illustrated by initiatives such as REACH and CLP, started by the EU and European Chemicals Agency (ECHA). When a company, manufacturer or exporter wants to introduce a new chemical to the European market, they need to provide certain toxicity information based on the tonnage of the chemical that is used per year. Based on the category a certain chemical belongs to, it can be banned or limited in usage amount. A category of substances that was banned under the previously mentioned initiatives is known as Substances of Very High Concern (SVHC). This category consists of multiple subcategories of substances that have different toxic properties.

3.5.1 Use of Machine Learning

Due to the shift from in vivo methods towards in silico methods in the 21st century, machine learning has become more and more significant in predictive toxicology.

In predictive toxicology, machine learning is used to predict different endpoints, like how much of a substance is needed for 50 percent of a certain fish species to

die. Usually the data used is either the structural information of the substance or its physiochemical properties. The structure of a substance is often described in the from of a fingerprint. A fingerprint is a string of zeroes and ones which tell you whether a certain substructure is included in the chemical structure or not. Physiochemical properties can include things like how long it takes for a substance to reduce by half in the air, or the molecular weight of a chemical.

Machine learning is used for both regression and classification[24]. It is mostly used for regression, e.g. to predict the numerical amount of a substance that is needed to kill a certain percentage of the test species. Supervised learning techniques are more common than unsupervised or semi-supervised learning techniques. An example of supervised learning for regression being used in predictive toxicology is for QSAR models. QSAR models use a structure-activity relationship to predict certain toxicological properties of a chemical.

There are a lot of toxicology databases available. Examples include CheMBL, ToxCast and PubChem. These databases each focus on different types of compounds. For instance, CheMBL focuses on drug-like compounds whereas ToxCast focuses more on industrial chemicals. Imbalanced data is common in machine learning. There are a lot more non-toxic chemicals than toxic ones, so randomly gathered data is imbalanced by default. Therefore, when using machine learning with toxicology data there are often steps taken to counteract this balance, in order to train good quality models.

The most models used by toxicologists are Support Vector Machines and Random Forests. Wang et al. think this is because these models are easy to use as well as being well-known They also conclude that the performance of regression models is less than that of their classification counterparts. However, it is hard to compare the performance of those two in such a way, because regression is often harder to get right and it also depends on how close you want to be to the real answer. Wang et al. also conclude that hepatxicity, carcinogenicity/mutafenicity/genotoxicity and cardiotoxicity are the most common types of toxicity that have been predicted by machine learning models. The highest performing models for those carcinogen, genotox and mutagen are Random Forests and SVMs using maccs keys and PubChem fingerprints as inputs and relatively small datasets. An important note here is that this survey looked at machine learning models form different papers and the results/performances reported in those papers. Therefore, they all have different test and training sets and aren't really well comparable. Of course the performance of a model highly depends on the test and training sets.

To determine the quality of a model and the reliability of its results, validation methods such ads hold-out validation, k-fold cross-validation or LOO cross-validation are used. It is important to note that accuracy often isn't enough to determine the quality of a toxicology model. The reason for this is that in toxicology, it is preferred to overestimate the toxicity of a substances instead of underestimating it. Therefore some errors are worse than others.

Wang et al. conclude that they expect that more models will be developed in

the future to predict toxicity. They emphasize however, that much needs to be done to address what is in their eyes the main bottleneck at the moment: which is the quality and quantity of the data that is available to create datasets. They say that there are some toxicological endpoints that cannot be reliably predicted due to the lack of data. They specifically say: "If a computational model encompassing all of human toxicology is to be built, these gaps in data need to be addressed". Despite all that, there have been many machine learning models that have high performance for predicting drug toxicity, which demonstrates the fact that machine learning works well for predictive toxicology. As more data becomes available, they expect that ML models will become more attractive than expert-based systems due to their scalability. They even expect that the current models will outperform experts if more data would become available. They think that this might mean that in order to develop in silico methods in the future, the focus should be on generating new data.

3.5.2 Sampling Bias

Not much research has been published on Sampling bias in toxicology. We haven't been able to find articles that talk about sampling bias in machine learning for predictive toxicology. However, there has been mention of sampling bias in other fields and applications in toxicology. In [**?**] for instance, They mention sampling bias in the risk characterization of drugs during pregnancy. Another example is [**?**], where they also talk about sampling bias in self-reported substance use among emergency room patients. However, these articles don't talk about sampling bias in the toxicology predictions of substances, but about sampling bias in data that consists of human individuals.

A research that is more interesting to us, is that of David R. Fox[**?**]. He conducted research on sampling bias in data that is used in SSD modelling. He acknowledges that the assumption of random sampling of species for SSD modeling is violated. Species Sensitivity Distributions are a tool used for determining safe limits for chemical concentrations in surface waters. It works by taking results from toxicity tests on different aquatic animal species and fitting a curve through it. The fitted curve is then used to infer the maximum concentration that preserves enough proportion of the species in an aquatic environment. It is used a lot by jurisdictions all over the world. A lot of research had been done about SSD models and what is needed to for them to be successful. such as sample size, distributional assumptions and statistical estimation techniques. However, according to Fox, not a lot of research has been done into the effects of sampling bias in SSD-fitting.

The results of Fox's study are certainly interesting, but they don't necessarily resemble the effects sampling bias in toxicology would have in a machine learning setting. In SSD modelling the datasets used are very small compared to the amounts of data used in machine learning, because it only looks at one substance at a time. However, Fox's article does support the hypothesis that there is a problem with biased data in predictive toxicology.

Another mention of selection bias in toxicology research is in the form of feature

selection bias in QSAR modeling. This concept refers to the notion that using the same training dataset in both feature selection and learning can can result in overfitting, poor classification performance and/or biased predictions. This problem often occurs when there is not enough data available to use two different datasets for selecting the features and training the classifier. In [25], Idakwo et al. Feature selection bias is a problem in QSAR modeling, because there are rarely a large number of substances across the end points to be predicted.

In conclusion, it seems like there hasn't yet been research done on sampling bias in machine learning for predictive toxicology. It has however been acknowledged that the available data in the toxicology field could suffer from sparsity and bias.

4 Method

In this section, we will first try to convey on a more abstract level what our research goals and hypotheses are and how we are planning to test those. We will then get into the details of the setup of our experiments.

4.1 Experimental Setup

We want to test how the 'conventional' AL algortihm that only uses uncertainty querying, compares to Richard et al.'s algorithm that was specifically designed to combat sampling bias. We want to see how their ability to mitigate sampling bias compares for different degrees of bias and class imbalance. We mainly want to see how these two approaches compare in terms of efficacy, ease of implementation and running time. Our focus will lay on comparing the efficacy of both algorithms when it comes to mitigating bias. We judge this on two criteria:

- how much and how quickly does the performance of our model on our ('unbiased') test set improve?
- How representative of the 'population' is the distribution of our sample after x iterations?

To judge the first criterion we will look at performance metrics graphs and to judge the second criterion we will look at density plots of the sample. The general approach will be as follows. We will start by experimenting on the benchmark dataset(s). First, we split it into two parts, one of which we will use for evaluation. We then add bias to the other part and test performance of a Random Forest classifier before and after bias mitigation. We repeat these experiments for different combinations of experimental parameters and and 5 different splits of the original data in order to use 5-fold cross validation.

In the following sections we explain these steps in further detail.

4.2 Bias Simulation

The sampling bias in the RIVM toxicology data is thought to be partly certainty based. The substances that are labelled are mostly the ones that the experts are the most certain about or have the most knowledge on.

To simulate this in our benchmark data we will train a classifier on a random sample of the data and let that classifier classify the whole dataset. We will then divide the data into seven equally sized bins according to the posterior probability of said classifier. The middle bin(bin 3) contains the instances that are the most difficult to classify, whereas the outside bins (bin 0 and 6) contains the most clearcut instances. After binning the data we will randomly pick instances from each bin to add to our labelled set. To mimic the uncertainty bias, we will pick more instances from the bins with high certainty than from the low certainty bins. Because we want to do this for increasing degrees of bias, we will use a distribution function f with parameters a and k that determine the extremity of the curve. This function is defined as follows:

$$f(x) = \frac{g(x)}{g(0) + ... + g(6)}, \tag{2}$$

where

$$g(x) = a(x-4)^2 + k. \tag{3}$$

In figure 1, we have plotted two bin distributions for different values of a and k, resulting in different degrees of bias. As you can see, in figure 1a the sample will be more evenly distributed over the different bins, making it less biased than the sample picked with figure 1b as its underlying distribution.

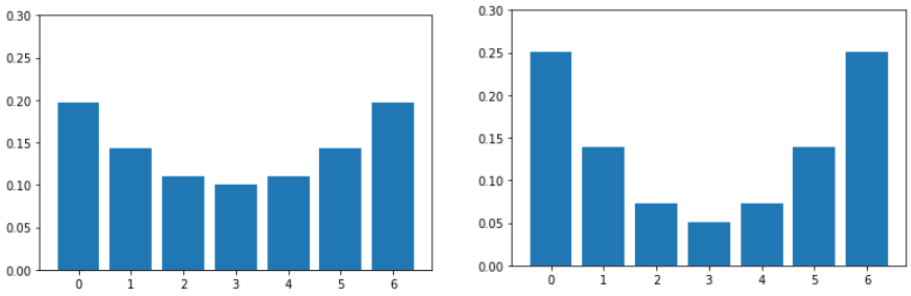
For each degree of bias, i.e. different (a, k) value pairs, we randomly pick $f(i) \cdot \text{sample\_size}$ instances from each bin i and together they form our labelled instances. All other labels are omitted to simulate data that suffers from sampling bias.

Another bias mechanism we use is auxiliary bias. For this we create variables $s1$ and $s2$ that are calculated from important features $pz1$ and $pz2$ by adding some noise. We then omit instances based on whether the $s1$ and $s2$ values lie within certain intervals. To vary the bias level, we simply vary the cut-off points used.

4.3 Bias mitigation

For bias mitigation. we will test the Active Learning technique from Richards et al. [3] In this section, we will refer to the population or the data that we want predict with our classifier as the test data.

We use a Random Forest model with B bootstrap samples. For each bootstrap sample a decision tree θ_b is built, that predicts the class of each instance from its feature vector $\mathbf{x}$. The RF estimate of probability that the instance with feature



(a) $a = 1/93$ and $k = 0.1$ (b) $a = 1/45$ and $k = 0.05$

Figure 1: Bin distributions for different values of a and k

vector $\mathbf{x}$ belongs to class y is just the empirical proportion of decision trees that predict class y, i.e.

$$\hat{P}_{RF}(y|\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B}\theta_b(y|\mathbf{x}), \tag{4}$$

The RF also provides a measure of the proximity of two feature vectors based on the decision trees. This distance is defined as

$$\rho(\mathbf{x}', \mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} I(\mathbf{x} \in T_b(\mathbf{x}')).$$

This is the proportion of trees for which the instances $\mathbf{x}$ and $\mathbf{x}'$ fall in the same terminal node, where I is a Boolean function that is 1 when this is the case and 0 otherwise. $T_b(\mathbf{x}')$ denotes the terminal node of instance $\mathbf{x}'$ in tree b.

The first AL selection procedure of the technique is focused on choosing instances that lie in areas that are undersampled in the training data compared to the test data. This is measured by

$$S_1(\mathbf{x}') = \frac{\sum_{\mathbf{x}\in U}\rho(\mathbf{x}', \mathbf{x})/N_{\text{Test}}}{\sum_{\mathbf{z}\in L}\rho(\mathbf{x}', \mathbf{z})/N_{\text{Train}}} \tag{5}$$

which is the ratio between the average proximity for $\mathbf{x}'$ in the testing (U) and training (L) set.

$S_1(\mathbf{x}')$ is large for instances from the test set that lie in regions of the feature space that relatively have high density in the test data compared to the training set.

The second AL selection criterion used in this technique is meant to select the instance $\mathbf{x}' \in U$ that maximizes the total amount of change in the predicted probabilities for the test data. This criterion is used to make sure that we only spend time on manually annotating instances whose labels most affect the predicted classifications. Richards et al.[3] attempt to approximate this by choosing

instances that have the highest uncertainty. Combined with the first criterion S_1, this results in the following selection criterion:

$$S_2(\mathbf{x}') = \frac{\sum_{\mathbf{x}\in U} \rho(\mathbf{x}', \mathbf{x})(1 - \max_y \hat{P}_{RF}(y|\mathbf{x}))}{\sum_{\mathbf{z}\in L} \rho(\mathbf{x}', \mathbf{z}) + 1}. \tag{6}$$

Here, $\hat{P}_{RF}(y|\mathbf{x})$ is as defined in equation 4 and $\max_y \hat{P}_{RF}(y|\mathbf{x})$ is the maximum of the class probabilities for instance $\mathbf{x}$. The details of deriving this criterion can be found in the appendix of [3].

S_2 is a weighted version of S_1 with the uncertainty, represented by $1-\max_y \hat{P}_{RF}(y|\mathbf{x})$ in the numerator. This means that S_2 gives higher weight to instances that are hard to classify, which causes the algorithm to focus more attention along class boundaries or other uncertain regions of the feature space, which should result in better performance.

Usually, the instances selected by the query function of the AL algorithm are manually labeled and added to the training set. Because in the setting that they perform their experiments in this is a lot more costly to do one by one, Richards et al.[3] use batch-mode querying instead of sequential querying. However, because we are using benchmark data we can simply obtain the label of an instance from the original dataset. Therefore we can just use sequential querying to make our algorithm less complicated.

4.4 Evaluation

As discussed in the section about evaluation, if we evaluate our experiments on biased data we risk the bias perpetuating in our evaluation. Therefore, we use benchmark data that is presumably unbiased or low in bias so that we can evaluate our experiments in an unbiased environment.

We are testing bias mitigation techniques. Therefore, we want to measure how biased our sample is after some number of iterations. Because we don't have a quantitative measure for the level of bias, we will try to approximate how biased our sample is by training a model on it and measuring it's performance on a presumably unbiased test set. We will do this as follows. Before we start our experiments on the benchmark dataset, we preserve around 20 percent of the original dataset for evaluation. We obtain this test set via stratified sampling, to make sure that our test set has roughly the same distribution as the data we use for our experiments. We don't touch this sample at all during our experiments, we solely use it for evaluation. After setting aside our test set, we add bias to the remaining data. We then split that data and use it for training and parameter tuning as normal. After we have fully trained our classifier, we it on the test set. We then perform our mitigation techniques on the data and again train a classifier. Subsequently, we evaluate that classifier on the test set and compare it to the classifier we obtained prior to bias mitigation.

To evaluate our experiments we use 10-fold cross validation with two repetitions, meaning that we will have 2 different splits of our benchmark dataset into

10 sets, each of which we will use as a hold-out test set once. We combine the results of these 20 runs in order to make our results and analyses more robust.

To measure our performance we will use PR AUC and balanced accuracy as our main metrics. Both of these metrics are robust to class imbalance. We have chosen PR AUC as one of our main metrics because it focuses on the positive class, which is the class we are most interested in. Because we don't want to solely focus on the positive class, we have chosen balanced accuracy as our second main metric because it takes both the negative and the positive class into account. Besides our main metrics, we will also look at recall, precision, AUC ROC, f1-score, f2-score, and the log loss.

4.5 Experimental parameters

one way we vary the bias level is by varying the noise and intervals for auxiliary variable bias. The lower the noise for the auxiliary variable bias, the more harsh the cut offs will be, so the more bias it will cause.Also, the smaller the auxiliary intervals are the more higher the bias and it matters where the cutoff points are. If the intervals don't include the most dense areas, than the bias will probably be more impactful on the performance of the model. Another way to vary the bias is by choosing using more extreme bin distributions for our uncertainty bias. The toxicology dataset (presumably) has a significant class imbalance in the unlabelled dataset, because there simply are a lot more non-toxic than toxic substances in the world. Therefore, we have chosen to add a 10/90 class imbalance to our benchmark data for most of our experiments. We will also do one experiment on data with bias level 3 without class imbalance, just to see how the two algorithms compare in that case. Keeping this in mind, we have chosen the following configurations for our experiments:

- bias level 1: uncertainty bias only, with a=1/45 and 0.05
- bias level 2: uncertainty bias only, with a=1/28 and k=0
- bias level 3: uncertainty bias with a=1/28 and k=0 and auxiliary variable bias, with auxiliary noise = 30, s1 interval (10,150) and s2 interval (10,200)
- bias level 4: uncertainty bias with bin distribution (0.4,0.1,0,0,0,0.1,0.4) and auxiliary variable bias with noise = 10, s1 interval (20,120) and s2 interval (20,150)
- bias level 5: no uncertainty bias, just auxiliary variable bias with auxiliary noise=10, s1 interval (10,80) and s2 interval(20,120)

4.6 Data Preparation

We start the preparation process by reducing the size of the benchmark dataset. We do this in order to make the running time of the experiments more manageable. We reduce the dataset size from 100.000 instances to 12000 instances

through random sampling. We make the assumption that the unbiasedness of the dataset is preserved in this process. After reducing the size of the dataset, we add noise to the endpoint, which is the molecular weight of the electron. We do this so that the predictive signal of the dataset more closely resembles that of the toxicology dataset. After adding the noise, we turn the regression problem into a classification problem by choosing a threshold for the molecular weight and splitting the dataset into instance that are bigger and smaller than that value. We do this for a 50/50 split and a 10/90 split of positive vs negative instances. After adding the classification labels, we use stratified 10-fold cross validation. After creating our folds, we add bias to each of the training sets to get a starting sample to apply the algorithms to. We calculate the bins for the uncertainty first, then we add the auxiliary variable bias and then we add the uncertainty bias.

4.7 Toxicology Experiment

Besides testing the two mitigation algorithms on our benchmark data, we will also apply them to our toxicology data. In order to be able to do this, we have to make some adjustments. Because we don't have the ability to label new instances, we will not be able to use sequential querying like we will do four our benchmark experiments. Therefore, we will run the algorithms once and select the top 20 substances with the highest query scores. Subsequently, a toxicology expert from the RIVM will look at the substances to see if they can say something about the substances that were selected by each algorithm. For instances, did the algorithms select substances that they would expect to be selected? Are the selected substances substances that are notoriously hard to come to a consensus about in the toxicology world? Perhaps the expert will even be able to give a judgement on which of the algorithms selected the most helpful or potentially informative substances from a toxicological point of view.

5 Data

We will conduct our experiments on benchmark data as well as toxicology data provided by the RIVM.

5.1 Toxicology Data

The RIVM toxicology data consists of around 700 labelled substances and over 60.000 unlabelled substances. The features consist of physiochemical properties of the substances, like their molecular weight or solubility coefficients. The presumption is that the set of unlabelled substances is unbiased and the set of labelled substances suffers from sampling bias. This becomes evident when we look at figure 2. There are areas of the unlabelled set that are not at all covered by the labelled data. In figure 3, we have shown a more detailed look at distributions for log kow and molecular weight. From this we can see that not only does the

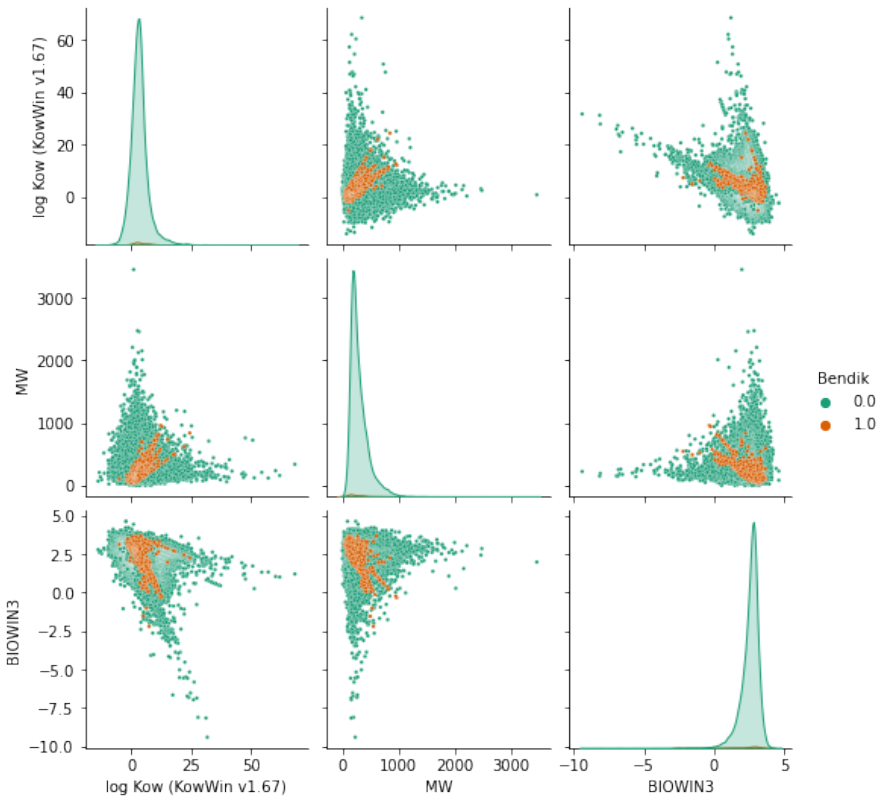


Figure 2: Scatterplots of the labelled toxicology set(orange) against the unlabelled data(green).

sample not cover the whole of the population, but also the centers of gravity of both sets are very different. The expectation from the toxicology expert's point of view, is that this bias is a mixture of auxiliary variable bias, uncertainty bias and bias caused by how easy substances are to label.

Because we presume that we will not be able to convincingly or in a trusty manner conduct and evaluate our experiments on this data only, we will conduct them on benchmark data first before applying our knowledge to the toxicology data.

5.2 Benchmark Data

Because our toxicology dataset is biased, we will not be able to trustfully analyse our results on this dataset. Therefore, we will be conducting our experiments on a benchmark dataset. Our aim is to have our benchmark dataset be as unbiased as possible, so that our results can also be unbiased. This means that ideally we

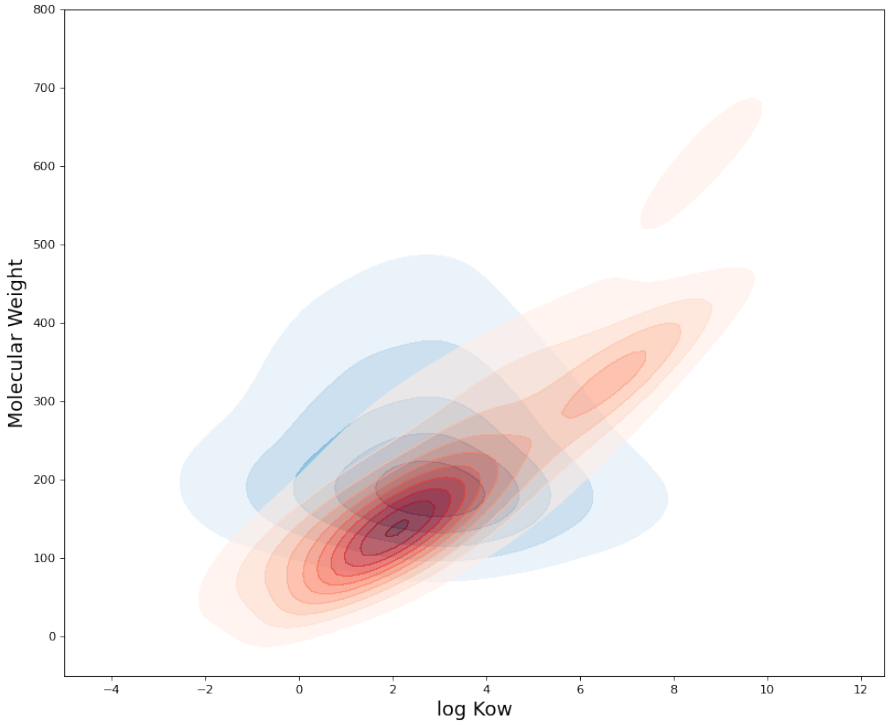


Figure 3: Contour plot of the labelled toxicology set(red) against the unlabelled data(blue).

want the data to have been obtained through random sampling. Furthermore, we want our benchmark data to be fully labeled, and similar to our toxicology dataset in terms of size, number of features, feature type (continuous), predictive power and covariances.

The benchmark dataset we have selected is the CERN Electron Collision Dataset. This dataset consists of 100.000 dielectron events in the invariant mass range 2-110 GeV. A dielectron event is a collision of two electrons. The dataset is publicly available from the CERN open data website. The dataset contains information like the total energy of both electrons, the transverse momentum of both electrons and the charge of both electrons. This dataset was selected with the intended task of creating a regression model which predicts the combined invariant mass of both electrons. Since the goal for the toxicology dataset is to find a classification model, we will convert the intended problem for the CERN dataset to a classification problem.

We selected the CERN Electron Collision Dataset as our benchmark dataset, because it is similar to the toxicology dataset in terms of the aforementioned criteria. It contains 100.000 instances with 19 features. Just like the toxicology dataset, all features are continuous. We tested the predictive power by training a model on the dataset with a binary endpoint. We created the endpoint by splitting the data at M= 21.3 GeV (the 50

6 Results

6.1 Performance metrics

Figures 4 through 15 show how the Area under the Precision Recall Curve (PR AUC) and balanced accuracy of the model change throughout the (first 4000) iterations of both algorithms for the five different bias levels. The blue line shows the development for the "conventional" active learning algorithm that just uses uncertainty sampling, whereas the orange line shows the progression for Richards et al.[3] algorithm that was specifically designed to mitigate sampling bias.

6.2 Sample Distribution

Figures16 through 21 show how the (pz1,pz2) distribution changes throughout the running of both algorithms. We have chosen to look at the distribution of these particular features because they are among the features with highest importance and because they were used to add the auxiliary variable bias. The latter means that for bias levels 3 through 5, the initial sample will miss big areas of the feature space, which makes it extra interesting to see which instances the algorithms select in those cases. In these figures, the middle column shows the distribution of the first 300 instances that are queried for the density algorithm and beneath it what the sample (the initial sample plus the 300 queried instances) looks like after 300 iterations. The third column shows the same but for the uncertainty only

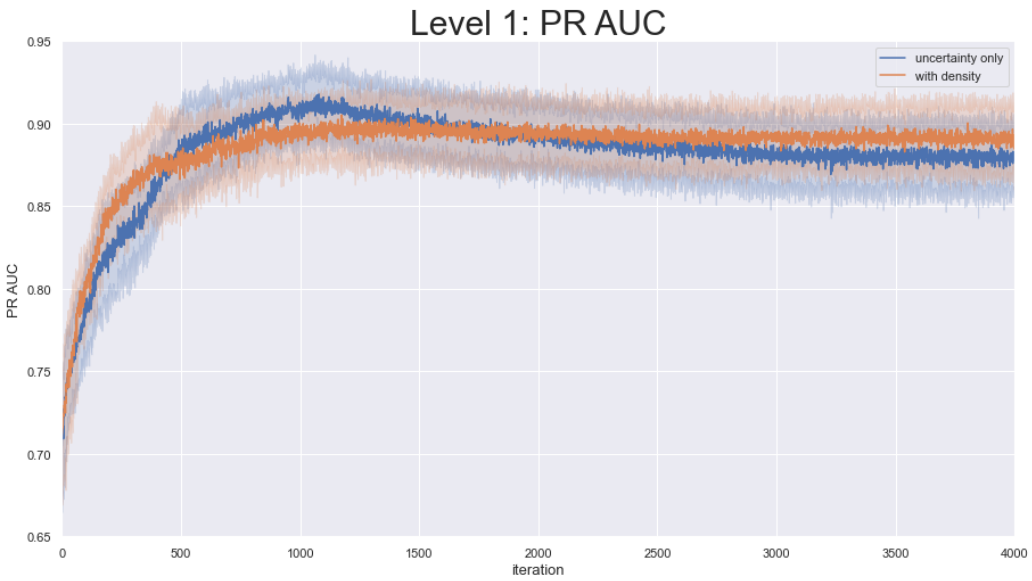


Figure 4: PR AUC of the RF model throughout both runs for bias level 1

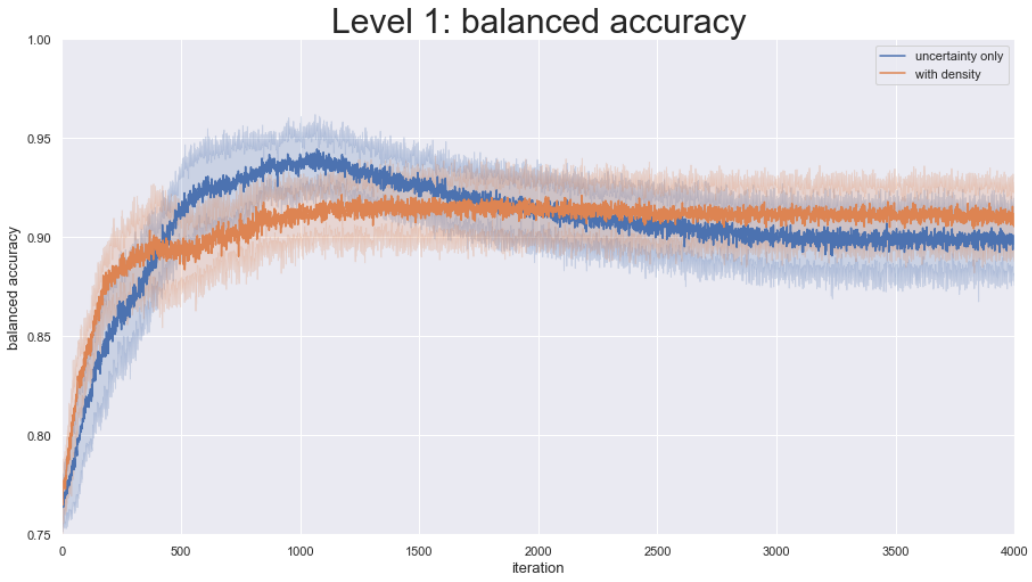


Figure 5: balanced accuracy of the RF model throughout both runs for bias level 1

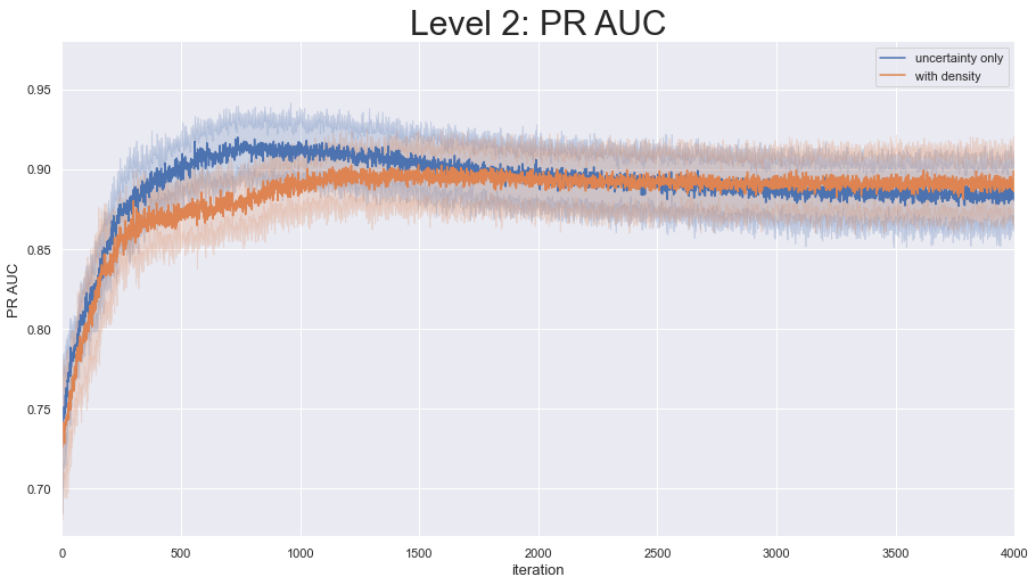


Figure 6: PR AUC of the RF model throughout both runs for bias level 2

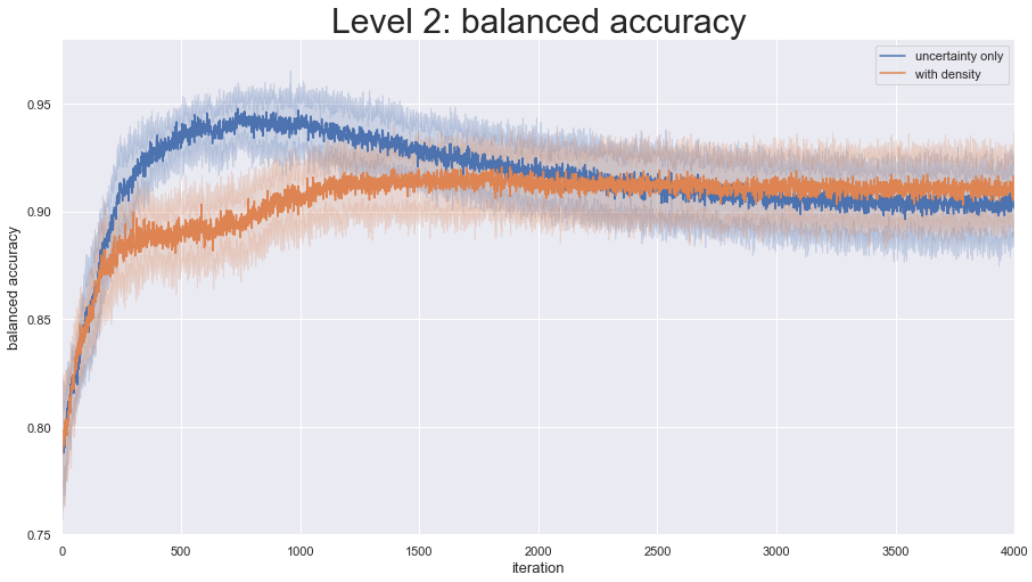


Figure 7: balanced accuracy of the RF model throughout both runs for bias level 2

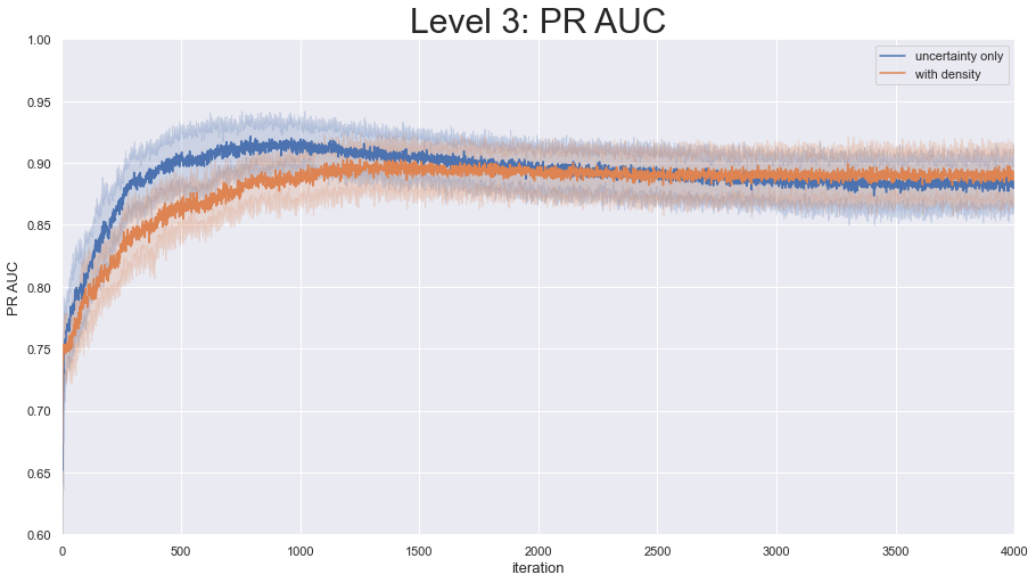


Figure 8: PR AUC of the RF model throughout both runs for bias level 3

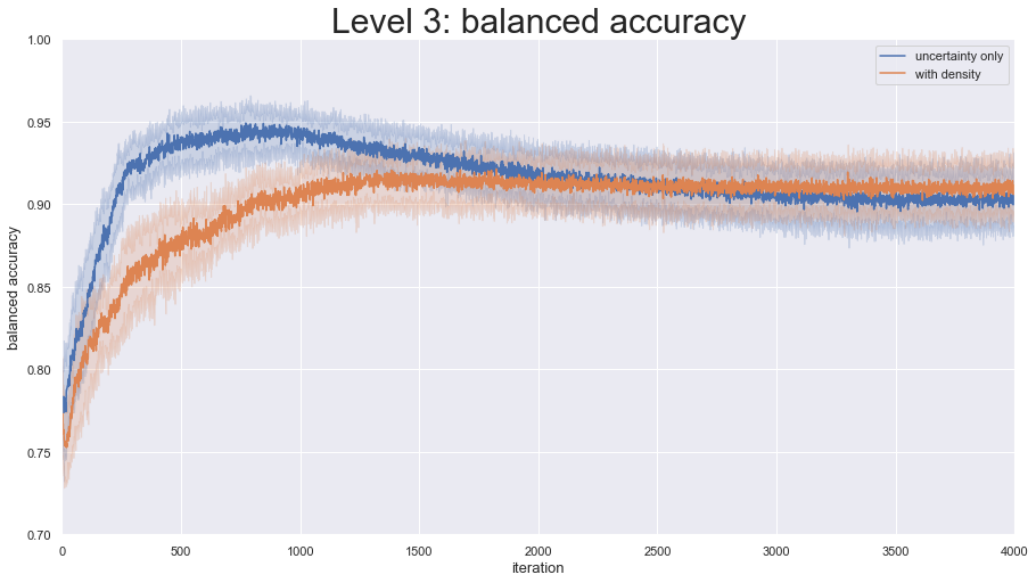


Figure 9: balanced accuracy of the RF model throughout both runs for bias level 3

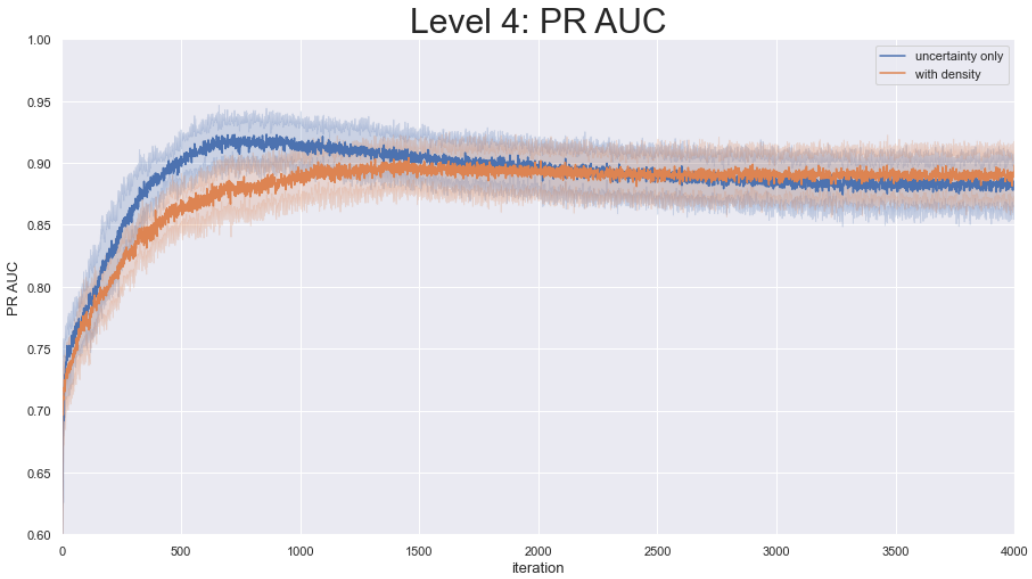


Figure 10: PR AUC of the RF model throughout both runs for bias level 4

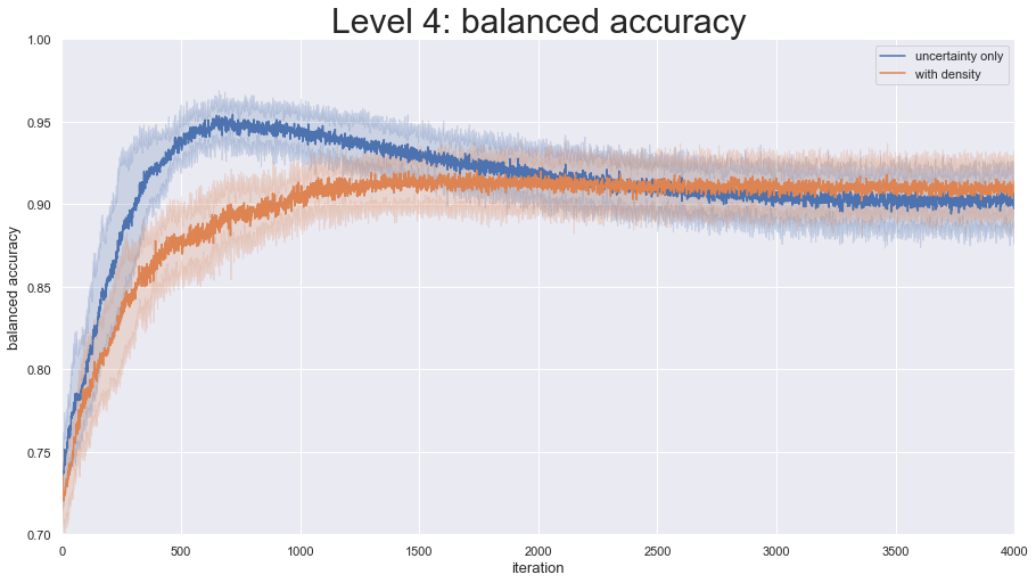


Figure 11: balanced accuracy of the RF model throughout both runs for bias level 4

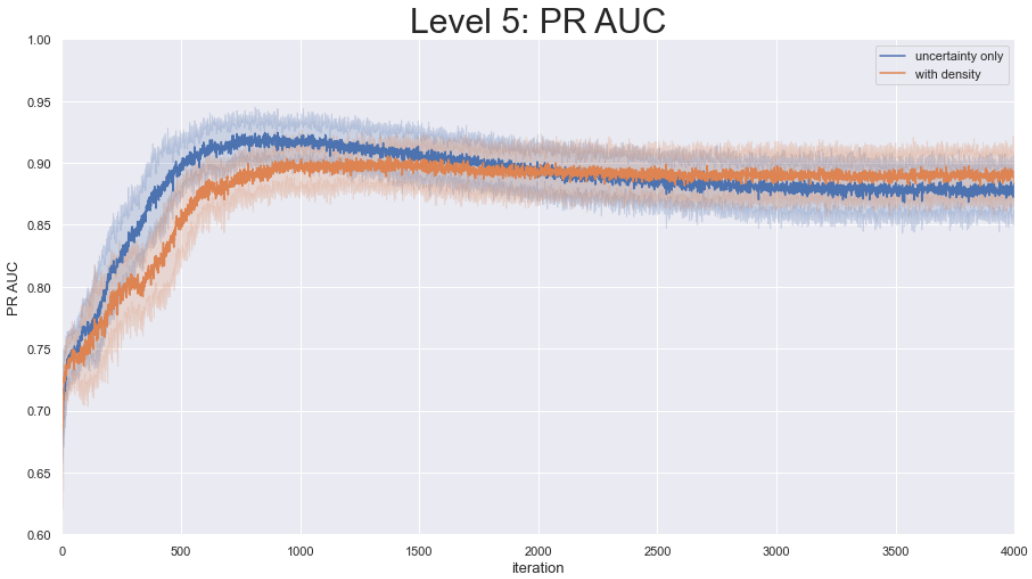


Figure 12: PR AUC of the RF model throughout both runs for bias level 5

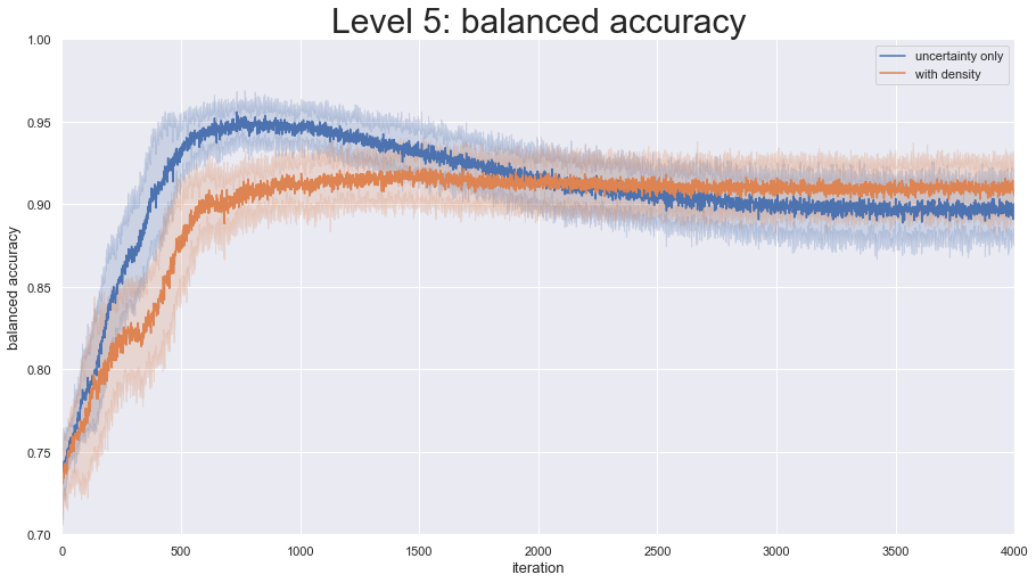


Figure 13: balanced accuracy of the RF model throughout both runs for bias level 5

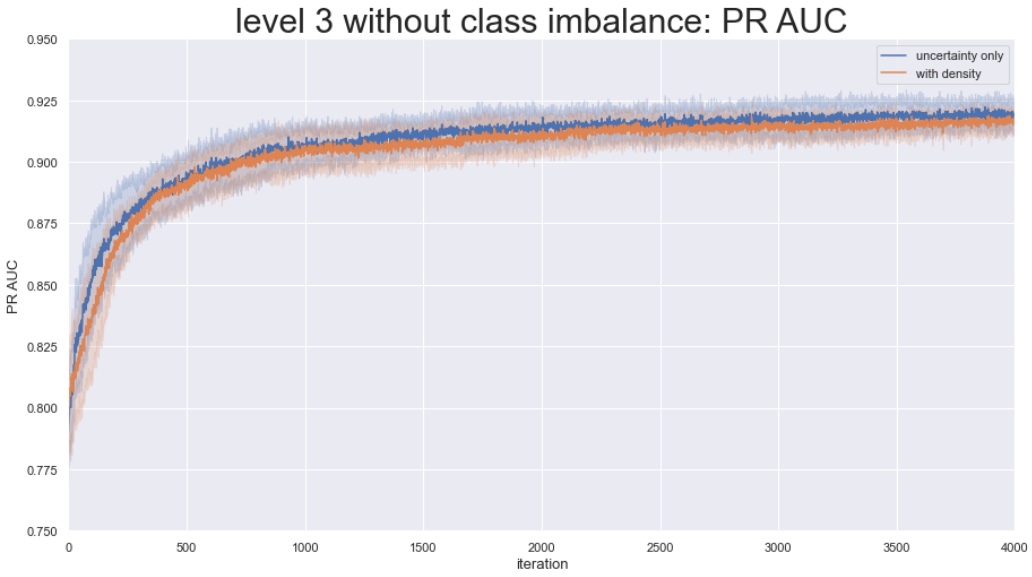


Figure 14: PR AUC of the RF model throughout both runs for bias level 3 without class imbalance

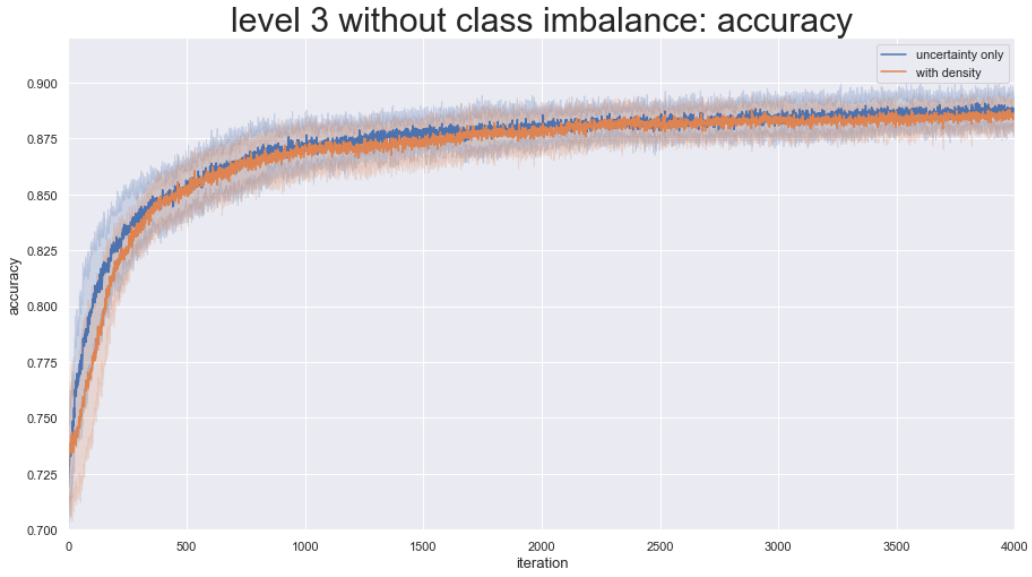


Figure 15: accuracy of the RF model throughout both runs for bias level 3 without class imbalance

algorithm. Because the (pz1,pz2) distribution is heavily concentrated around the center, we have also taken a look at the sample distribution development for the (eta1,eta2) distribution. This is shown in figures 22 through 27. We have chosen to show these distributions because eta1 and eta2 are also amongst the most important features.

6.3 Development of Class Imbalance

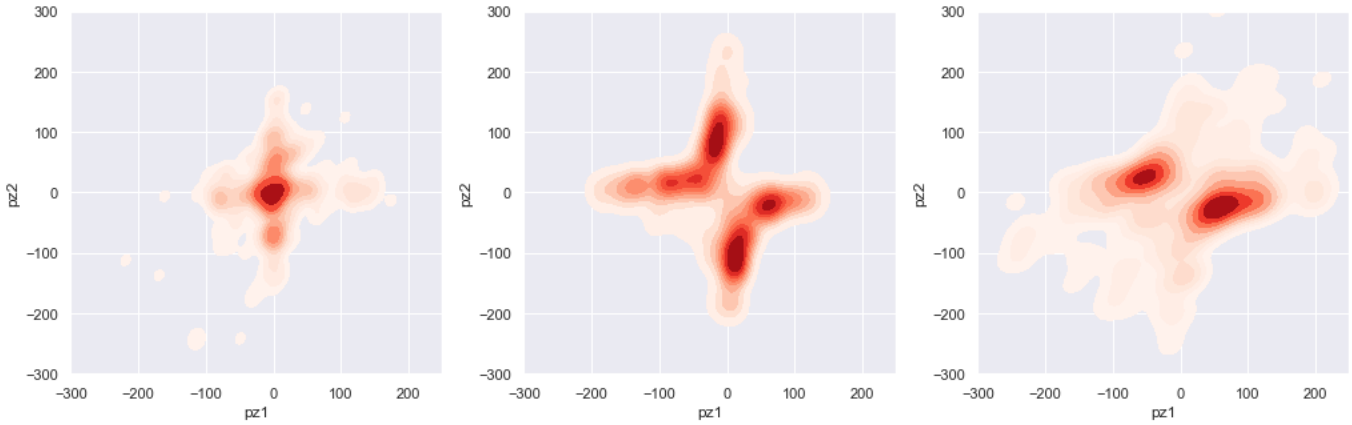
Tables 1 through 6 show the development of the class imbalance throughout the running of both algorithms. The second two columns show what percentage of the queried instances belong to the positive class, while the last two columns show what percentage of the sample as a whole belongs to the positive class. For the experiments where class imbalance is present, the positive class is the minority class.

Table 1: Class imbalance for bias degree 1

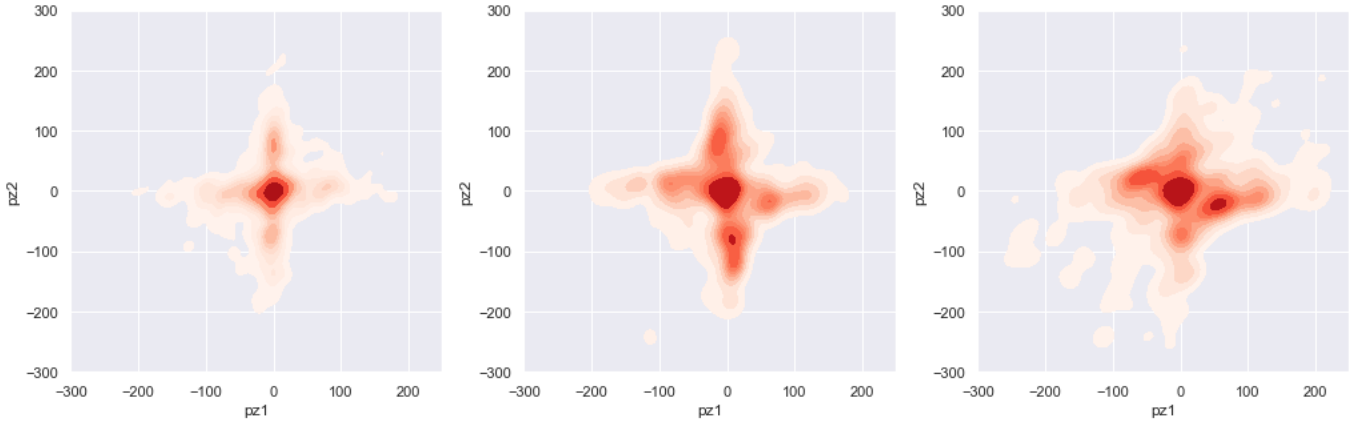
iteration	percentage queried		percentage in sample	
	w/ density	w/o density	w/ density	w/o density
0	-	-	18.1%	18.1%
25	52.8%	44.4%	20.7%	20.1%
50	55.6%	48.8%	23.4%	22.4%
100	55.5%	54.4%	27.4%	27.1%
250	48.8%	53.6%	32.0%	34.1%
500	40.5%	52.8%	32.1%	39.7%
750	36.5%	51.2%	31.2%	41.7%
1000	33.9%	49.4%	30.3%	42.1%
2500	24.7%	31.1%	24.0%	29.7%
4000	19.7%	23.0%	19.6%	22.6%

6.4 Toxicology Experiment

We applied both algorithms to the toxicology data provided by the RIVM. We let each algorithm select twenty substances. The selected substances and some of their descriptors and physiochemical properties can be found in tables 7 and 8 of the appendix.

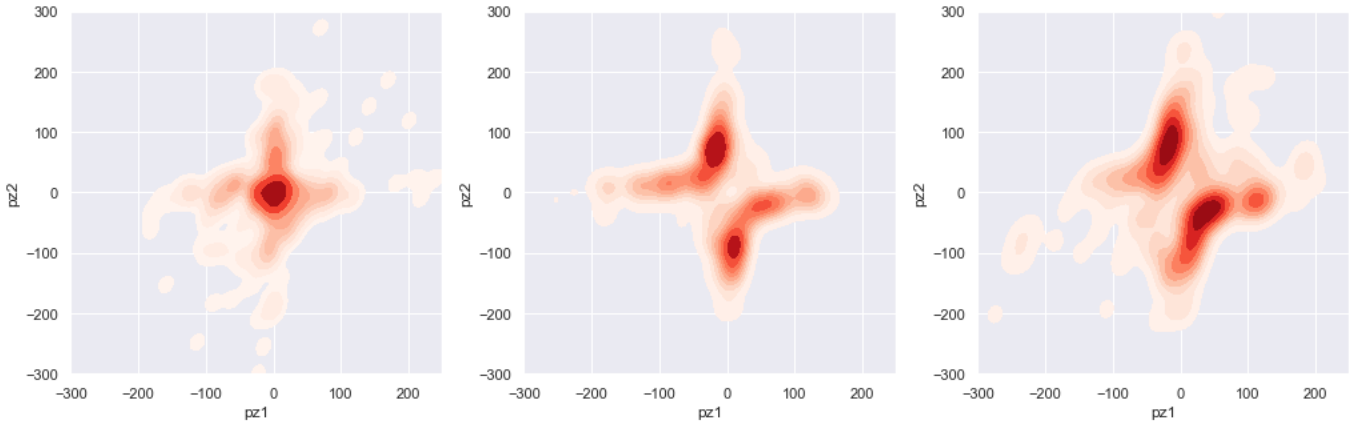


(a) initial sample (b) density first 300 queried (c) uncertainty first 300 queried

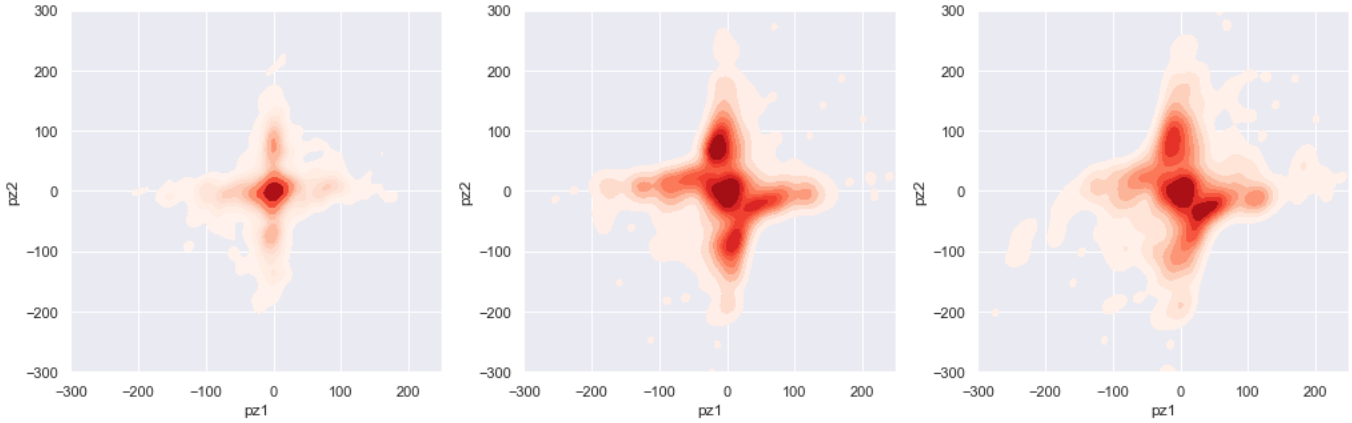


(d) Test set (e) density sample aft. 300 it. (f) uncertainty sample aft. 300 it.

Figure 16: (pz1,pz2) sample distribution for bias level 1

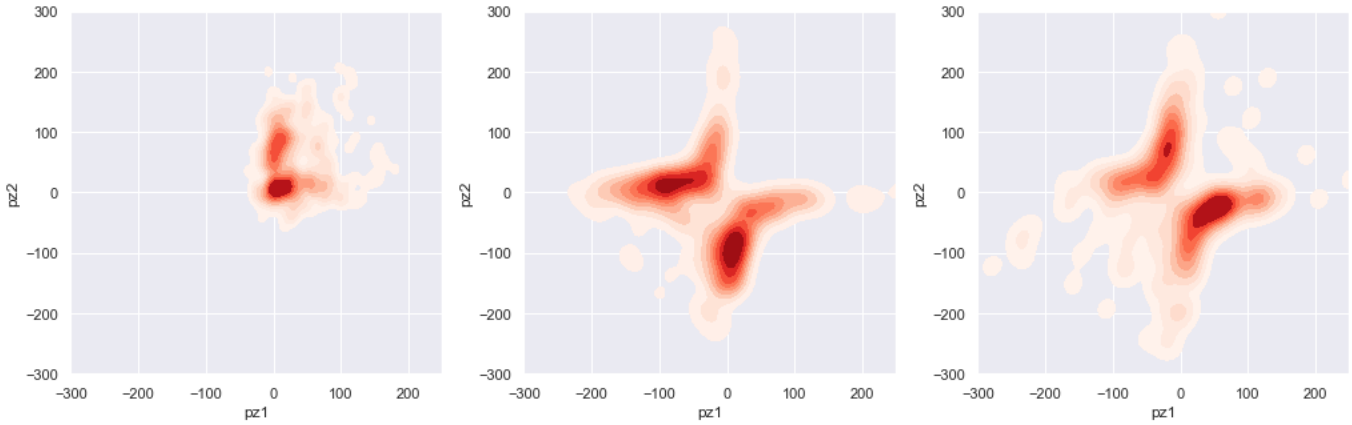


(a) initial sample (b) density first 300 queried (c) uncertainty first 300 queried



(d) Test set (e) density sample aft. 300 it. (f) uncertainty sample aft. 300 it.

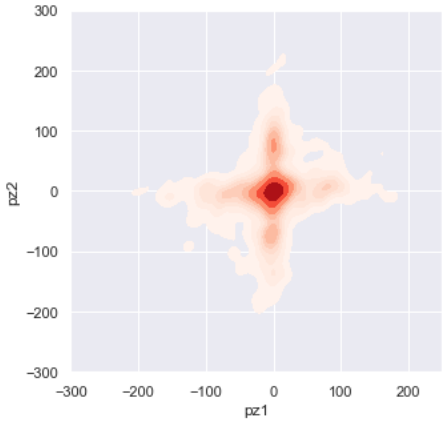
Figure 17: (pz1,pz2) sample distribution for bias level 2



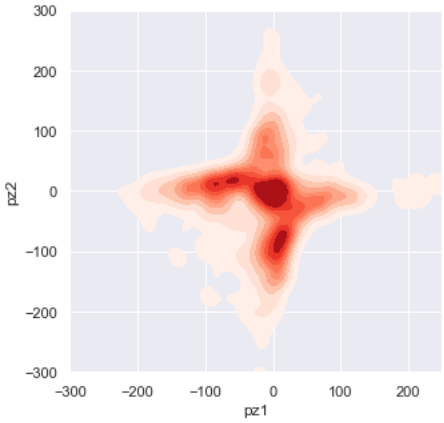
(a) initial sample

(b) density first 300 queried

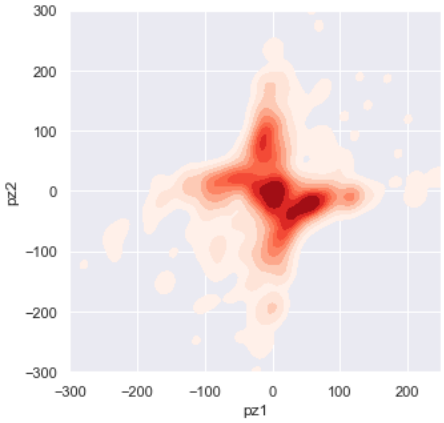
(c) uncertainty first 300 queried



(d) Test set

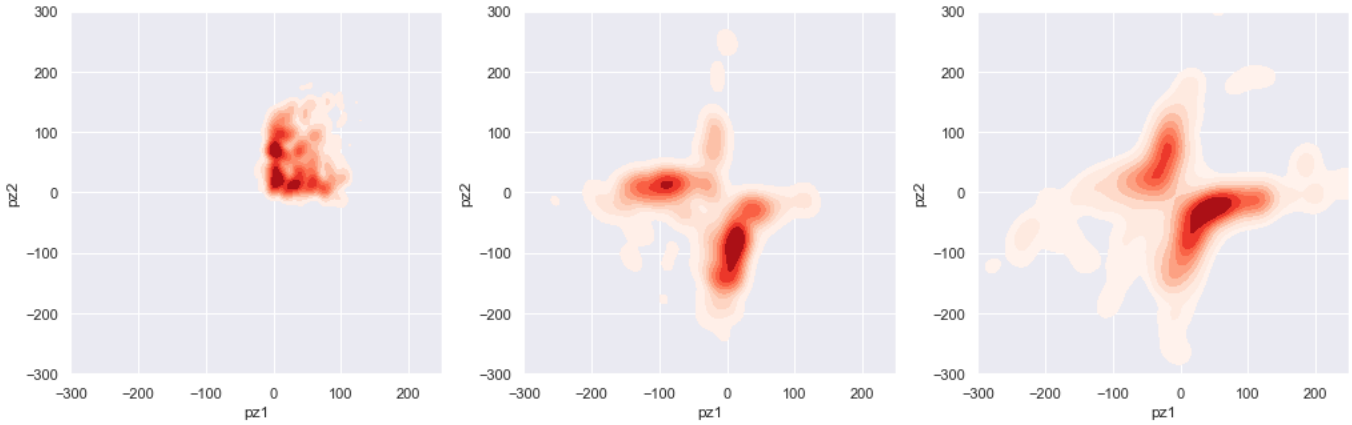


(e) density sample aft. 300 it.

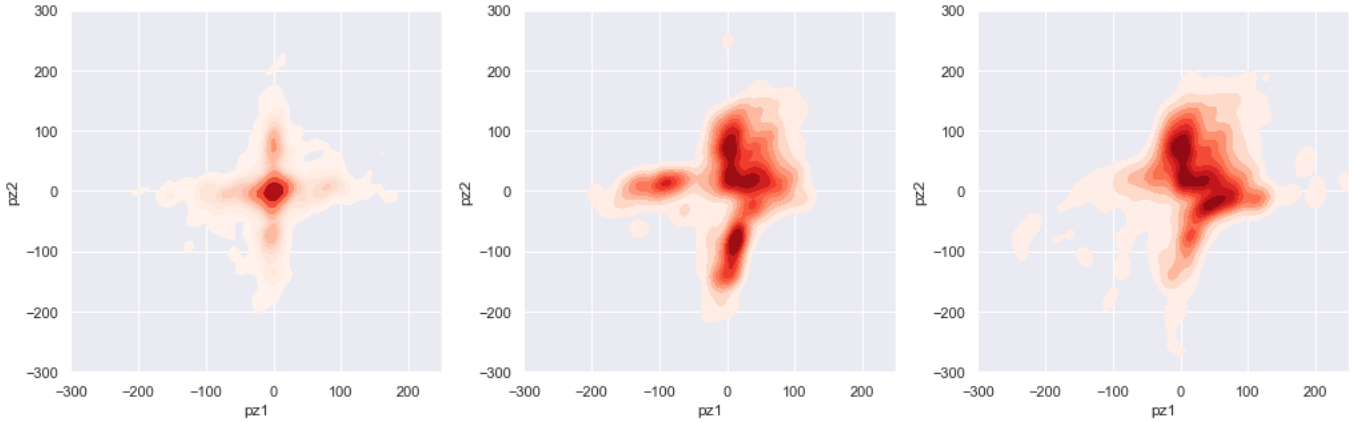


(f) uncertainty sample aft. 300 it.

Figure 18: (pz1,pz2) sample distribution for bias level 3

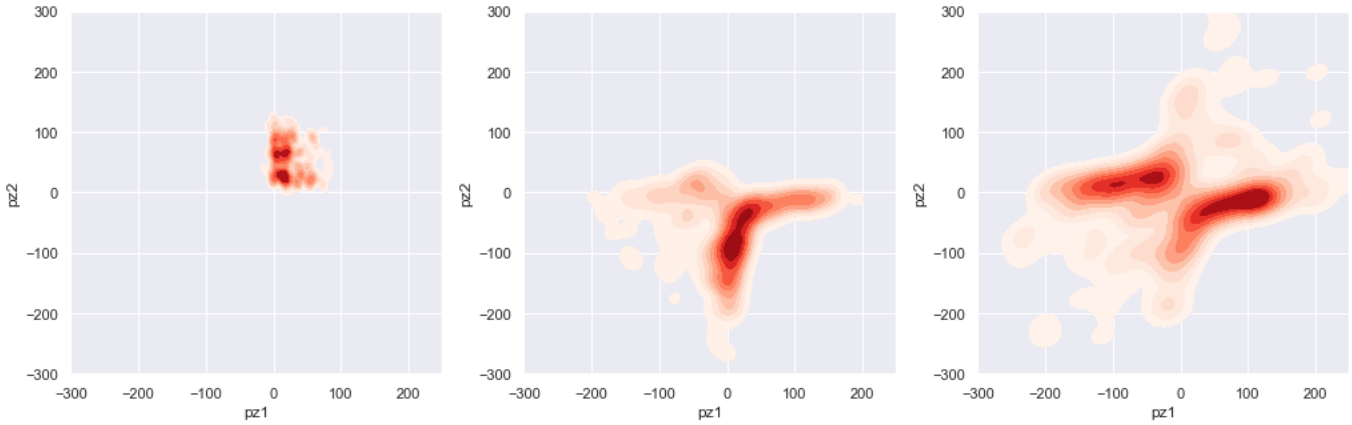


(a) initial sample (b) density first 300 queried (c) uncertainty first 300 queried



(d) Test set (e) density sample aft. 300 it. (f) uncertainty sample aft. 300 it.

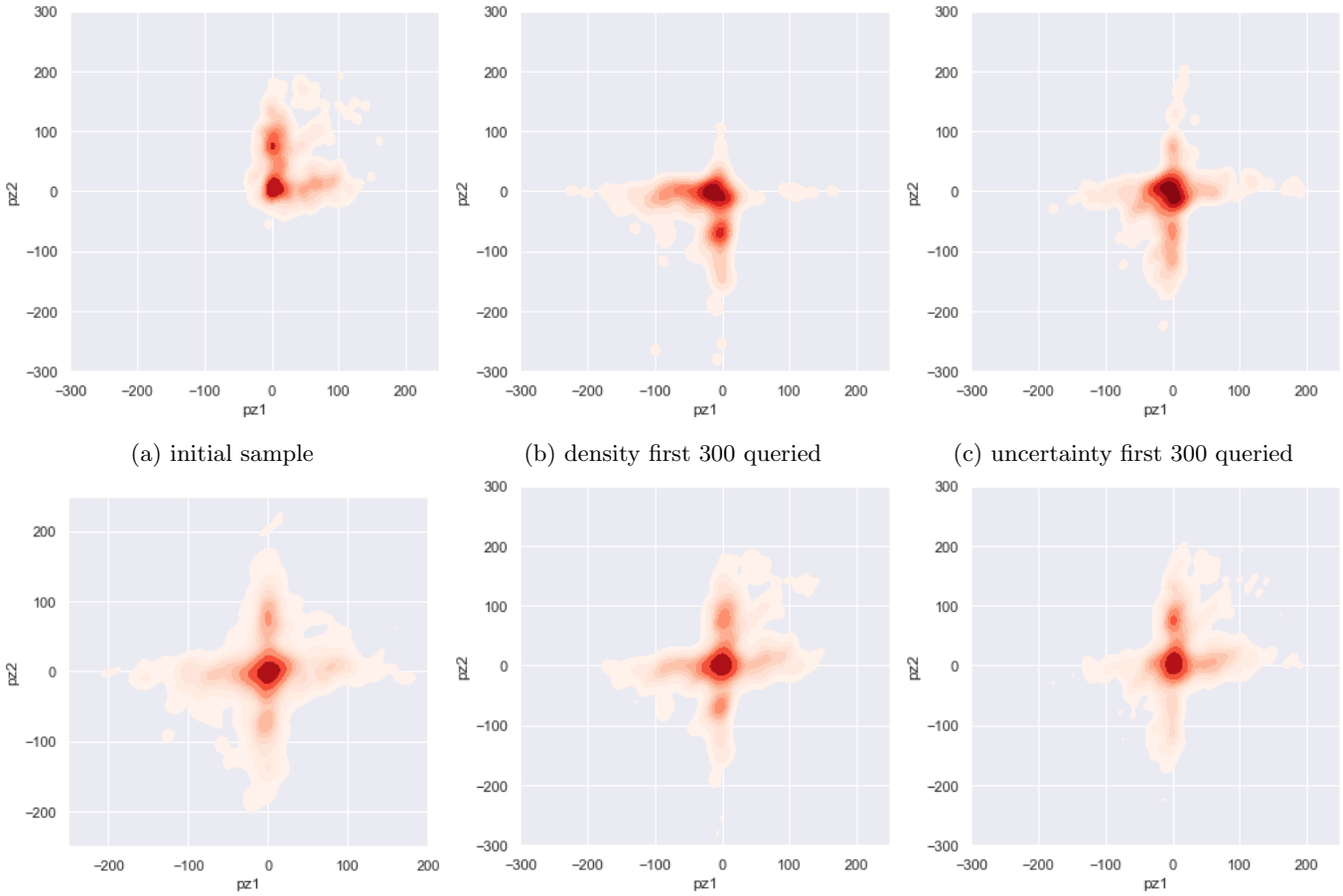
Figure 19: (pz1,pz2) sample distribution for bias level 4



(a) initial sample (b) density first 300 queried (c) uncertainty first 300 queried

(d) Test set (e) density sample aft. 300 it. (f) uncertainty sample aft. 300 it.

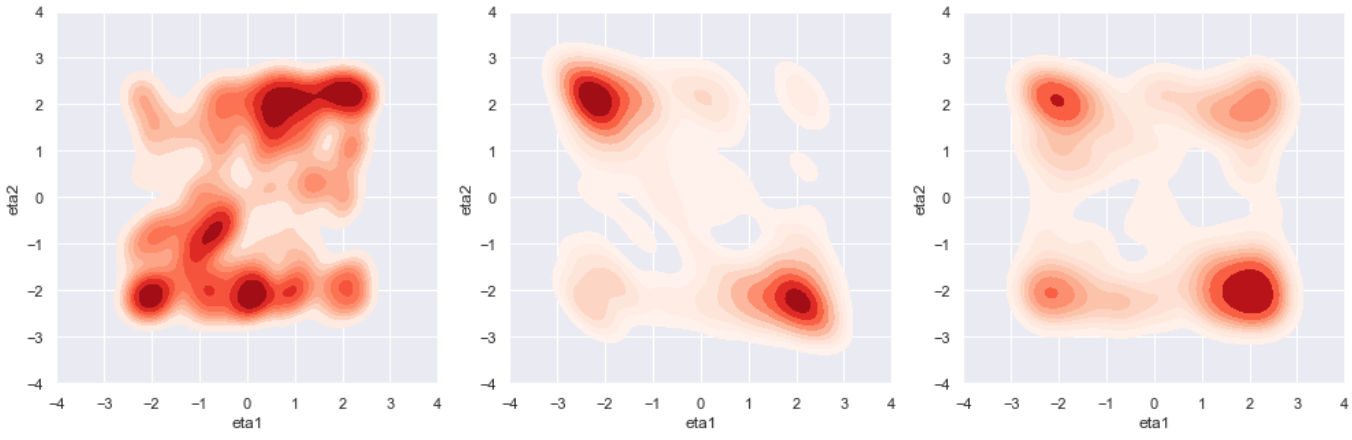
Figure 20: (pz1,pz2) sample distribution for bias level 5



(a) initial sample (b) density first 300 queried (c) uncertainty first 300 queried

(d) Test set (e) density sample aft. 300 it. (f) uncertainty sample aft. 300 it.

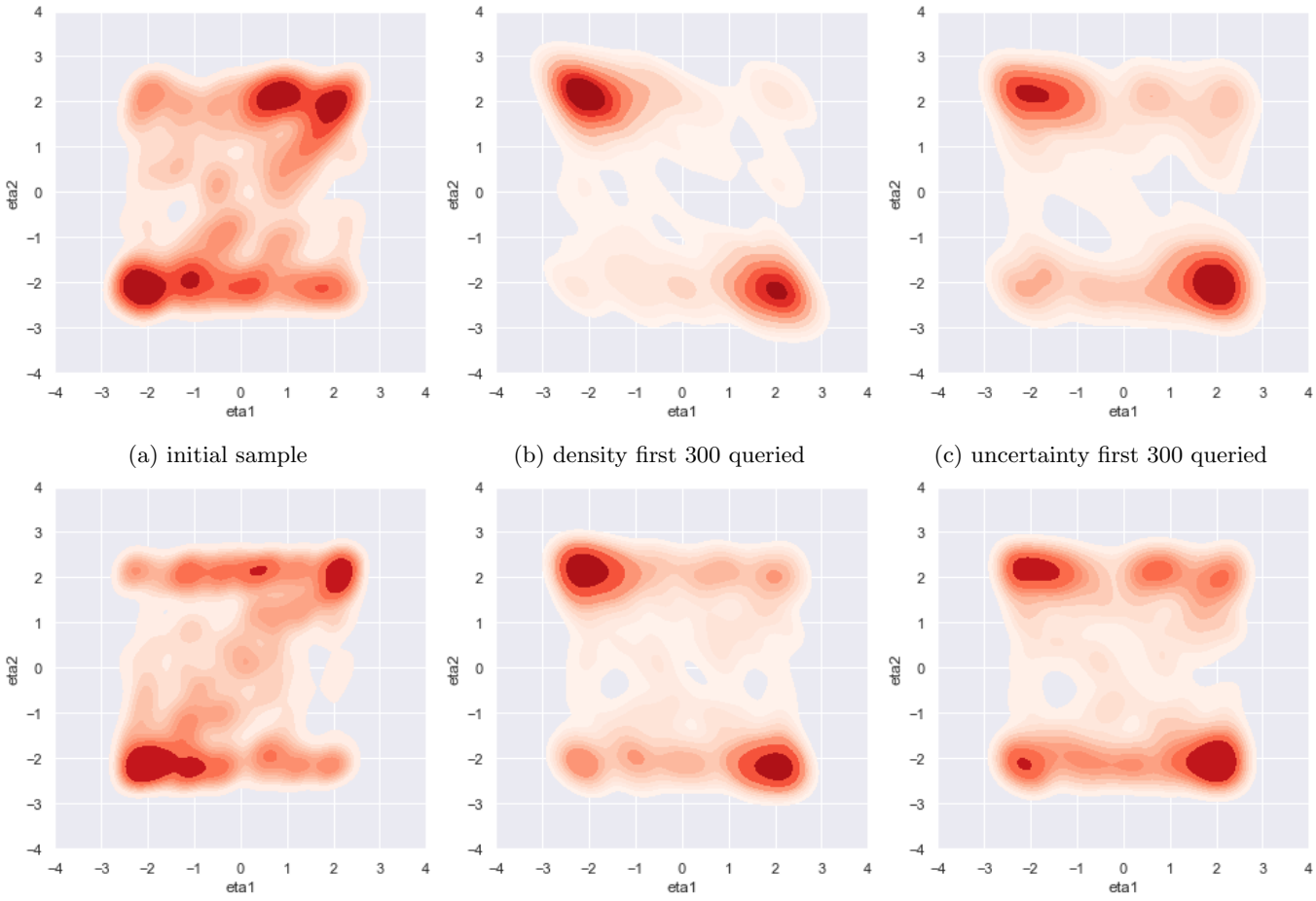
Figure 21: (pz1,pz2) sample distribution for bias level 3 without class imbalance



(a) initial sample (b) density first 300 queried (c) uncertainty first 300 queried

(d) Test set (e) density sample aft. 300 it. (f) uncertainty sample aft. 300 it.

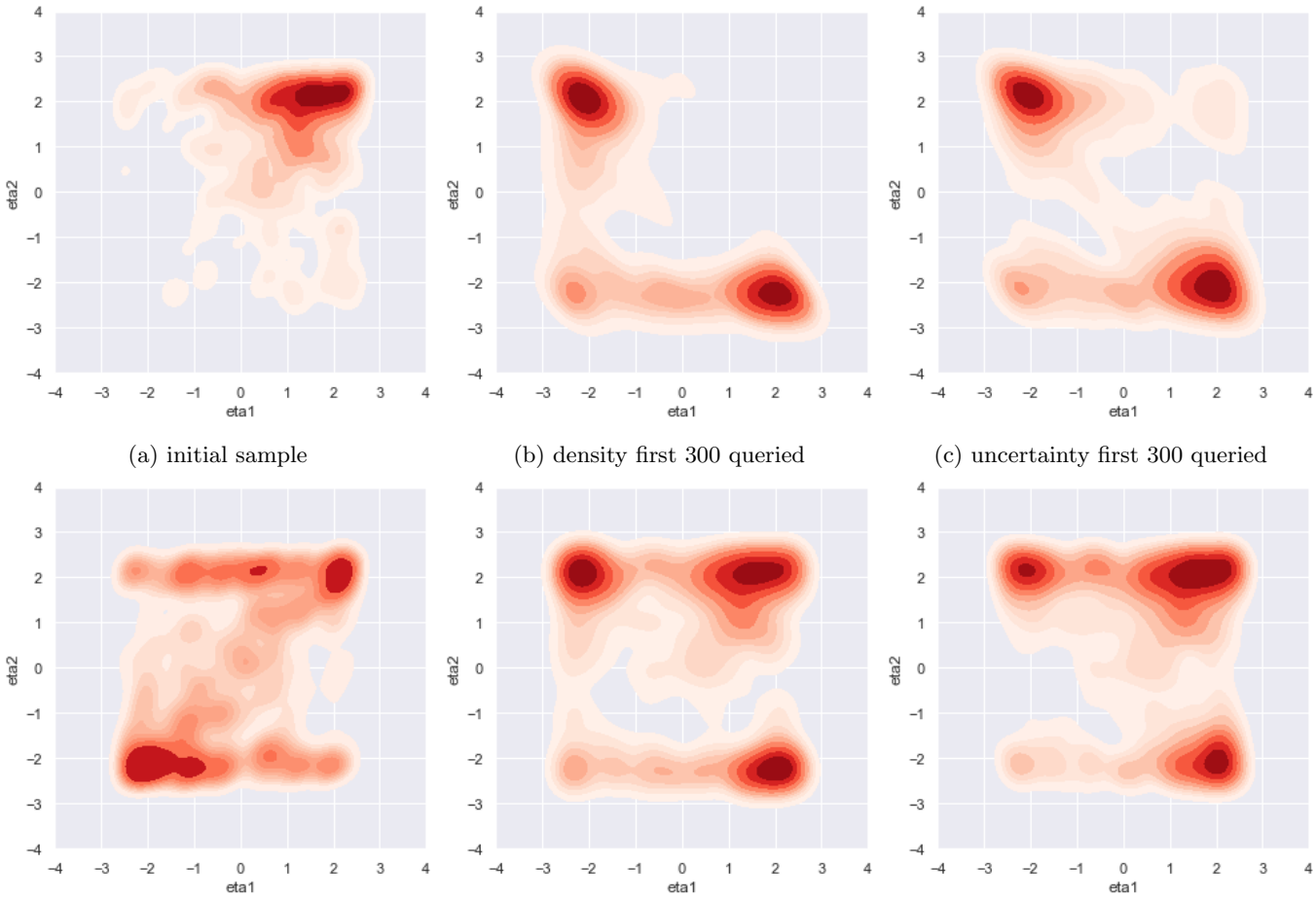
Figure 22: (eta1, eta2) sample distribution for bias level 1



(a) initial sample (b) density first 300 queried (c) uncertainty first 300 queried

(d) Test set (e) density sample aft. 300 it. (f) uncertainty sample aft. 300 it.

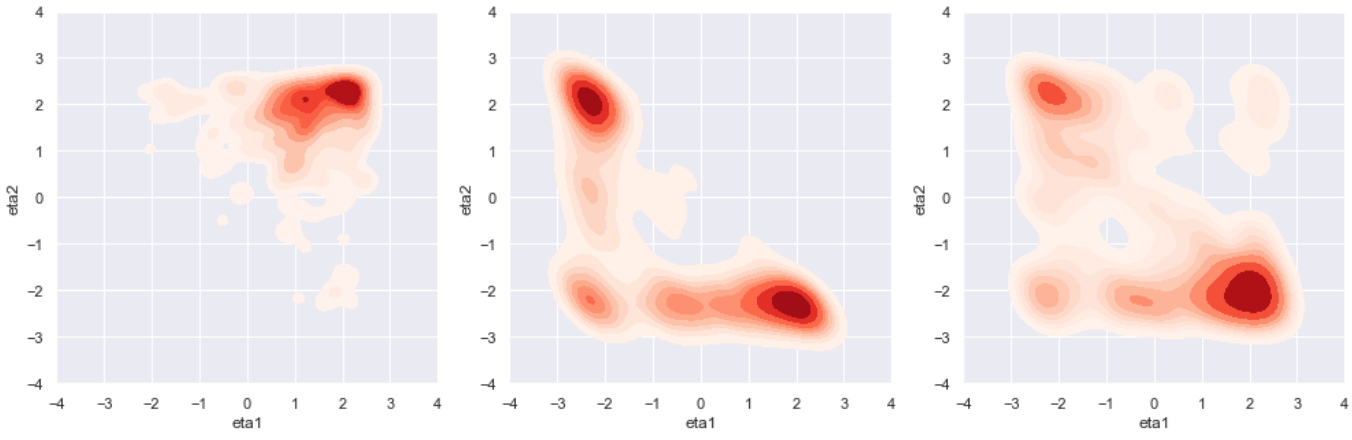
Figure 23: (eta1, eta2) sample distribution for bias level 2



(a) initial sample (b) density first 300 queried (c) uncertainty first 300 queried

(d) Test set (e) density sample aft. 300 it. (f) uncertainty sample aft. 300 it.

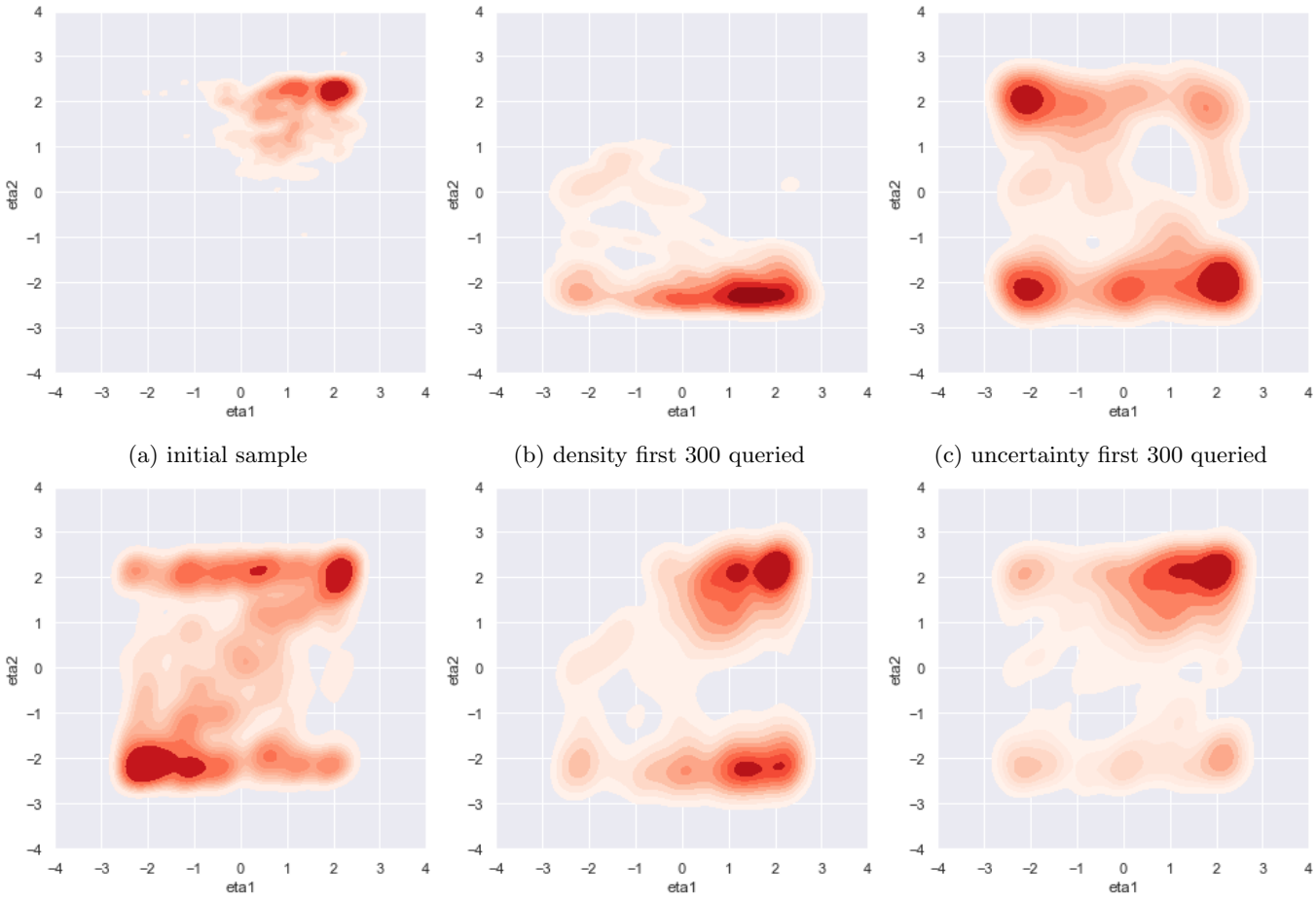
Figure 24: (eta1, eta2) sample distribution for bias level 3



(a) initial sample (b) density first 300 queried (c) uncertainty first 300 queried

(d) Test set (e) density sample aft. 300 it. (f) uncertainty sample aft. 300 it.

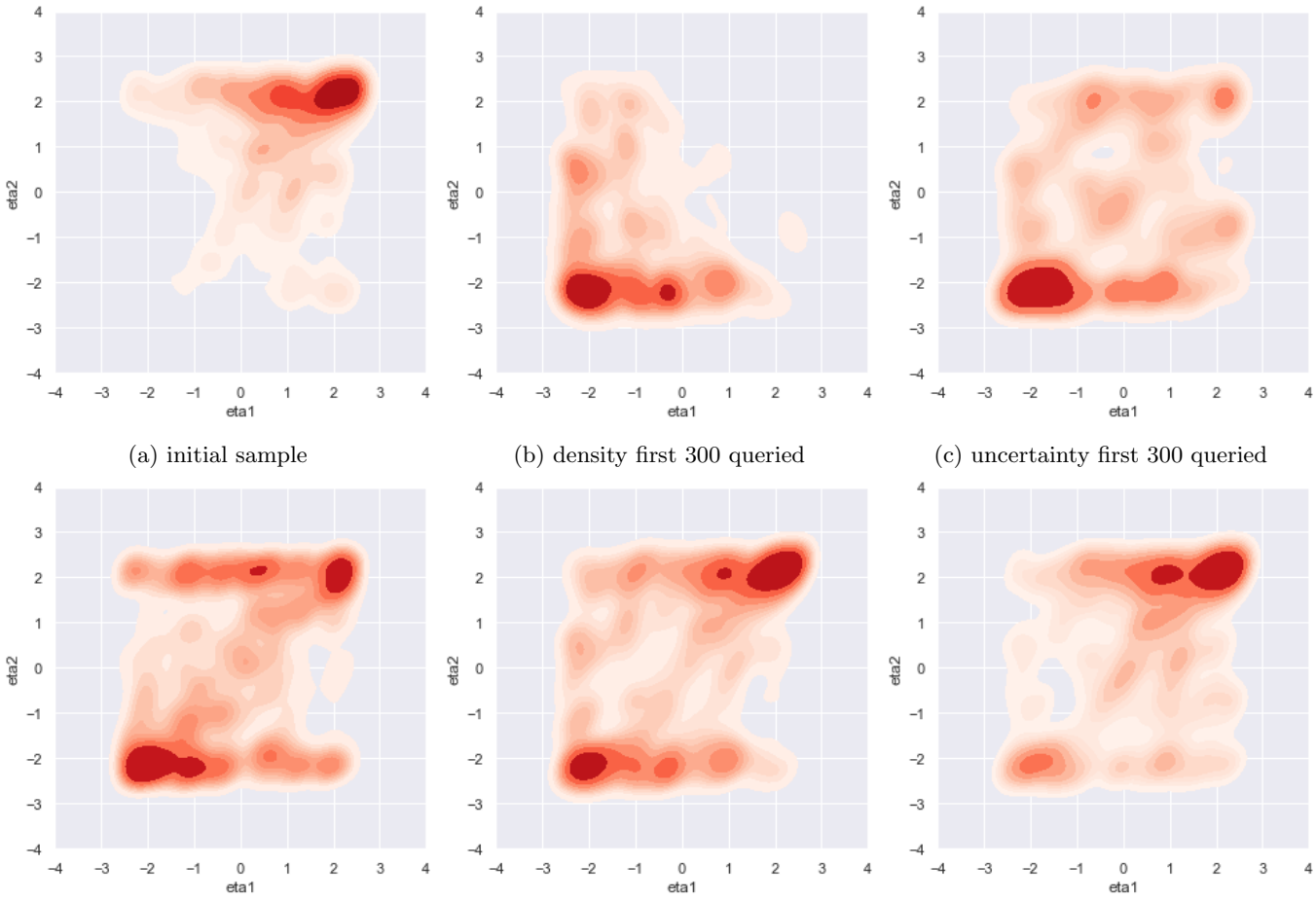
Figure 25: (eta1, eta2) sample distribution for bias level 4



(a) initial sample (b) density first 300 queried (c) uncertainty first 300 queried

(d) Test set (e) density sample aft. 300 it. (f) uncertainty sample aft. 300 it.

Figure 26: Sample distribution for bias level 5



(a) initial sample (b) density first 300 queried (c) uncertainty first 300 queried

(d) Test set (e) density sample aft. 300 it. (f) uncertainty sample aft. 300 it.

Figure 27: (eta1,eta2) sample distribution for bias level 3 without class imbalance

Table 2: Class imbalance for bias degree 2

iteration	percentage queried		percentage in sample	
	w/ density	w/o density	w/ density	w/o density
0	-	-	18.1%	18.1%
25	50.8%	48.8%	20.6%	20.4%
50	53.2%	49.8%	23.1%	22.6%
100	49.8%	52.1%	26.0%	26.5%
250	43.2%	53.5%	29.5%	34.1%
500	37.2%	51.7%	30.0%	39.0%
750	34.0%	49.9%	29.4%	40.8%
1000	32.1%	47.9%	28.9%	41.0%
2500	24.1%	30.5%	23.5%	29.2%
4000	19.4%	22.5%	19.3%	22.2%

Table 3: Class imbalance for bias degree 3

iteration	percentage queried		percentage in sample	
	w/ density	w/o density	w/ density	w/o density
0	-	-	18.1%	18.1%
25	27.2%	48.8%	18.8%	20.4%
50	43.0%	54.4%	21.6%	23.2%
100	44.5%	56.5%	24.7%	27.6%
250	42.7%	55.4%	29.2%	35.0%
500	38.7%	52.1%	30.9%	39.3%
750	35.3%	50.2%	30.4%	41.0%
1000	33.1%	48.2%	29.6%	41.2%
2500	24.4%	30.9%	23.7%	29.5%
4000	19.5%	22.6%	19.4%	22.3%

Table 4: Class imbalance for bias degree 4

iteration	percentage queried		percentage in sample	
	w/ density	w/o density	w/ density	w/o density
0	-	-	21.4%	21.4%
25	37.2%	55.2%	22.6%	24.0%
50	44.8%	58.0%	24.8%	26.7%
100	45.3%	56.5%	27.4%	30.2%
250	42.8%	55.1%	31.1%	36.7%
500	39.6%	52.9%	32.8%	41.1%
750	36.6%	50.7%	32.3%	42.4%
1000	33.7%	49.1%	30.9%	42.7%
2500	24.5%	31.0%	24.2%	30.0%
4000	19.6%	22.7%	19.8%	22.6%

Table 5: Class imbalance for bias degree 5

iteration	percentage queried		percentage in sample	
	w/ density	w/o density	w/ density	w/o density
-	-	0.0%	18.8%	18.8%
25	41.6%	53.6%	20.5%	21.4%
50	46.4%	55.8%	22.7%	24.1%
100	45.9%	55.3%	25.6%	27.9%
250	41.1%	53.1%	28.9%	34.4%
500	39.7%	51.2%	31.9%	39.1%
750	36.9%	49.9%	31.7%	41.0%
1000	34.1%	48.6%	30.5%	41.7%
2500	24.8%	31.6%	24.2%	30.2%
4000	19.9%	23.1%	19.8%	22.8%

Table 6: Class imbalance for bias degree 3 with no class imbalance

iteration	percentage queried		percentage in sample	
	w/ density	w/o density	w/ density	w/o density
0	-	-	50.0%	50.0%
25	24.4%	45.2%	48.0%	49.6%
50	34.2%	49.6%	47.8%	49.9%
100	45.7%	52.4%	48.9%	50.6%
250	52.6%	52.4%	51.2%	51.1%
500	52.0%	50.9%	51.2%	50.6%
750	51.0%	50.6%	50.7%	50.5%
1000	50.6%	50.6%	50.5%	50.5%
2500	50.3%	49.5%	50.3%	49.6%
4000	49.9%	49.2%	49.9%	49.2%

7 Discussion

7.1 Benchmark Experiments

Our main goal in this thesis was to determine how the conventional active learning algorithm and Richard et al.'s active learning algorithm compare to each other. In this section, we will first discuss how they compare in terms of ease of implementation and running time. After that we will get into the more important comparison, namely how they compare in their ability to mitigate sampling bias. Then we will discuss the results of our experiment on the toxicology data, talk about the limitations of our experiments and give suggestions for future work.

Implementation difficulty and running time Even though it is not the main goal of this thesis, it is also important to look at how both algorithms compare in terms of complexity of implementation and running time. This could be a deciding factor when choosing which algorithm to use.

The uncertainty algorithm is a lot easier to implement than the density algorithm. It requires a lot less functions and the functions that are needed are all very simple in nature. The uncertainty algorithm could technically be used with any machine learning technique that provides a measure of uncertainty or class probability scores. The density algorithm as given by Richards et al.[3] however, specifically uses the random forest model to calculate the proximity of instances and can therefore only be used with Random Forest models. The fact that the density algorithm is more complicated to implement, also makes it more prone to mistakes and more difficult to test.

In terms of running time, the density algorithm takes significantly longer to run. For our benchmark experiments we did 4000 iterations on a dataset consisting of 12000 instances. For the density algorithm this took approximately 160 hours to run, compared to 3 hours for the uncertainty algorithm.

One could of course decide that this difference in implementation and running time would be worth it, if the density algorithm does a better job at mitigating sampling bias. In the next few paragraphs we will discuss whether this is the case.

Performance metrics Next, we will compare the two algorithms in terms of the progression of the performance metrics of the model over the iterations. To determine this, we looked at plots of the performance metrics as shown in section 6.1.

For the case without class imbalance, both algorithms perform more or less the same in terms of model performance metrics. They both improve rapidly at first, until the improvement slows down.

Now we will discuss the experiments with class imbalance. Both algorithms exhibit similar behaviour in terms of the development of the rate of improvement of the performance metrics. For both algorithms the PR AUC and balanced accuracy increase very rapidly at first until at some point the rate of increase

decreases. Another thing that we can see in common is that for both algorithms the rate of improvement decreases the higher the bias level is and it takes longer for them to reach their optimum.

For the uncertainty algorithm we can see that, for all bias levels, it reaches an optimum and then starts decreasing. We can see the same for the density algorithm, but a lot less pronounced. Once it reaches its optimum, it roughly stays there. A very interesting thing to see is that at some point during the decreasing trend of the uncertainty algorithm, it even crosses the density algorithm and starts performing worse. This is especially noticeable for bias level 1.

Now for the cases with class imbalance. For bias level 2 and up, the uncertainty algorithm performs better than the density algorithm for all metrics except precision. For bias level 1, the density algorithm performs better at first, but gets overtaken by the uncertainty algorithm around the 400 iteration mark.

We did not expect the density algorithm to perform worse than the uncertainty algorithm in most cases. So why is this the case? One of the possible explanations we've found for this fact, is the difference between the two algorithms in what class the selected instances belong to. This is illustrated by the tables in section 6.3. For bias level 1, the density algorithm starts of selecting more instances from the positive (minority) class than the uncertainty algorithm. This switches around between the 100th and 250th iteration. Because for all other bias levels the density algorithm always selects less positive instances than the uncertainty algorithm, this could be a possible explanation for why the first iterations for bias level 1 are the only iterations where the density algorithm beats the uncertainty algorithm. This being an explanation for the difference in performance makes sense. If the model has more positive instances to learn from, we would expect it to become better at recognising those instances, thus improving the recall. Improvement of the recall means improvement of the PR AUC and f2-score. Another result that supports the class imbalance being one of the major explanations for the difference in performance, is the fact that both algorithms perform roughly the same for the case without class imbalance, for which both algorithms have roughly the same level of class imbalance throughout, as can be seen in table 6.

However, this difference in how many positive instances are selected by both algorithms probably isn't the full explanation for the difference in performance. For bias level 1, the switch between which algorithm selects more positive instances happens between the 100th and 250th iteration, whereas the switch in which algorithm performs better happens around the 400th iteration.

Sample distribution development Another way to get an idea of how well both algorithms do at mitigating sampling bias is by looking at how the sample distribution develops throughout the iterations. We hoped that looking at some distribution density plots for important features would give us an idea of how well the algorithms fare at making the sample more closely resemble the test set. We looked at the plots of section 6.2 for this.

We will first take a look at the (pz1,pz2) density plots. As expected, the density

algorithm seems to do a better job at selecting instances from regions that have high density in the test set. This is especially clear when we look at the sample distributions for the experiments with auxiliary bias(bias levels 3, 4 and 5), for which all initial instances are located in the upper right quadrant of the (pz1, pz2) feature space. As expected, the density algorithm has a heavier focus on the instances outside of the top right quadrant, whereas the uncertainty algorithm has a less strong emphasis on that area. This means that after 300 iterations, from just looking at both distributions, the density algorithm distribution is more similar to that of the test set. Therefore, we would expect the density algorithm to also have a better performance after 300 iterations. It doesn't however.

A possible explanation for this could be that despite there being a clear difference between the distribution of the instances selected by both algorithms, both sets of instances still are very highly focused on the center, which is also by far the most dense region of the test set.

Another explanation could be that the fact that all instances of the initial sample lie in a certain quadrant does not necessarily have to mean that that quadrant is well represented in the sample. Despite an area being dense with instances, it could be that instances with certain characteristics are not present due to the uncertainty bias. The fact that the uncertainty algorithm does query instances from that area does indicate this, as well as the fact that the uncertainty algorithm results in better performance metrics, even though it is less focused on the instances outside of the quadrant where all initial instances were located.

Because the (pz1,pz2) plots are so heavily clustered around the center, we also took a look at the (eta1, eta2) plots to gain more insight. The biggest standout when we look at the (eta1, eta2) density plots is the result for bias level 5. The initial sample is mostly situated in the top right quadrant, so it is to be expected that the density algorithm does not select any instances from that region in the first 300 iterations. However, as we can see from figure 26, the density algorithm also does not select any instances from the top left corner, even though that part is not represented in the initial sample. This phenomenon does not occur for all lower bias levels. In fact, for all other bias levels the algorithm actually selects a lot of instances from the top left corner. Apart from that, the density algorithm also does not select many instances from the bottom left corner, even though that is a very dense region in the test set. Hence, it seems like the density algorithm does not do a very good job at making the sample more representative in terms of its eta distribution for such a high bias level.

A possible explanation for this could be the fact that the proximity measure used by the density algorithm is dependent on the model. This could mean that for such a high bias level the proximity measure becomes biased as well and therefore fails to recognize that the two left corners are highly underrepresented in the initial sample.

These results raise the question: is the added density criterion redundant? An interesting observation that can be made from the sample distribution plots is that, though not as extremely as the density algorithm, the uncertainty algorithm

also has a clear focus on the areas that are underrepresented in the sample. This is especially clear if we look at bias levels 3 to 5. This gives the impression that the uncertainty algorithm inherently has the effect that the sample becomes more representative of the population and therefore inherently combats sampling bias. One could even argue that this effect/phenomenon makes the addition of the density criterion redundant.

If we look at it from a more theoretical standpoint, it makes sense that the uncertainty algorithm inherently exhibits this behaviour. Essentially, classification model uses the training set to find some kind of correlation between the features of an instance and its class. The decision trees in a random forest quite literally do just that. A logical consequence of this is that a new instance is often more difficult to classify if there are not many similar instances in the training set, especially if it lies near a decision boundary. Therefore it makes sense that when selecting the instances which have the highest uncertainty, we are more likely to select instances from underrepresented areas.

Despite the uncertainty algorithm inherently selecting instances from areas that have low density in the sample, one could still expect the density criterion to improve on this. The uncertainty algorithm does not take the density in the test set into account. This means that outliers, i.e. instances that lie in low density areas of the sample that also have low density in the population or test set, are equally likely to be selected as instances that lie in low density areas of the sample and high density areas of the population/test set. Our distribution plots clearly show that the uncertainty algorithm indeed queries more outliers. Preventing this is one of the motivations Richards et al.[3] give for adding the density criterion, which they have succeeded in.

7.2 Toxicology Experiment

We let both algorithms select 20 substances. We asked a toxicology expert from the RIVM to take a look at the selected substances and shine their light on them. The first thing the expert noticed, was that the substances selected by the density algorithm were all known borderline cases, whereas for the substances selected by the uncertainty algorithm that was less obvious. All instances selected by the density algorithm have a BioWin 3 score between 2.33 and 2.72, i.e. a half-life in water from 26 to 56 days. In the toxicology field, this is exactly known as the borderline region of the persistence estimation where an expert would wonder whether the substance only just meets the PBT criterium (a half-life of over 40 days), or only just breaks down quickly enough (a half-life of under 40 days) to be categorized as non-persistent and therefore non-PBT. This is less obvious for the substances selected by the uncertainty algorithm, where the BioWin3 values vary from 0.33(half-life of 2800 days) to 2.57(half-life of 35 days).

Another thing that the expert noticed is that there is not a single overlapping substance between the sets of substances selected by both algorithms.

While looking at the structure of the selected substances, the expert noticed

that the ones queried by the density algorithm all have a roughly similar structure. For the most part, they are biphenyl and bisphenol-like substances, which means that they consist of two (or more) rings that are not directly connected to each other, but are connected through a bond (biphenyl) or one or more carbon bonds.

An example is the substance "CAS 5450-24-8". This is an aromatic ring with a hydroxide group and a second ring attached to that. The tert-butyl group as substituent - which causes a substance to have a more difficult/slower break down process - is frequent among the twenty substances selected by the density algorithm.

The substances selected by the uncertainty algorithm are a bit more diverse in terms of their structure. Interestingly enough, among them is a PFAS (perfluor) bond, which is a substance that is currently being discussed a lot among toxicologists who are deciding whether they should be completely disallowed it in production. This substance is a clear precursor of a well known PBT substance. These kind of 'precursors' of PBT substances are known to be difficult to predict for all models, which means it makes sense for them to be selected by the active learning model.

In summary, the density algorithm selected more substances that the expert would expect to be selected from a toxicological point of view, namely well-known borderline/disputed substances. the substances selected by the uncertainty algorithm were more diverse in terms of their structure, and one of the substances selected by the uncertainty algorithm is currently being disputed by toxicologists.

7.3 Limitations

With this discussion, we want the reader to keep in mind that while these results can give an indication of the ability of both algorithms to mitigate bias, we cannot be a hundred percent sure that these results actually fully tell us how much sampling bias is left in the model and sample. For instance, while the performance metrics of the model on a presumably unbiased test set expectedly is a good indication of how much sampling bias is left in our model, it probably does not have a hundred percent correlation with the amount of sampling bias there is. The same goes for looking at the sample distribution plots to judge how representative the sample is of the population. While 2d density plots of the most important features could be a good indicator of this, they of course show a simplified picture of the instances in the sample. There are many features that all interplay with each other, which is difficult to visualize in a picture. Also, we need to keep in mind that we only tested the algorithms in the setting without class imbalance for one bias level. it could be that for lower bias levels without class imbalance, the density algorithm beats the uncertainty algorithm in terms of performance metrics, similarly to the setting with class imbalance and bias level one. Therefore, we don't want the reader to completely write Richards et al.'s [3] algorithm off based on our results and findings.

The experiment we performed on the toxicology data also has its limitations.

For one, we only looked at the first twenty substances that were selected, whereas if one were to use one of the algorithms to add instances to their data, they might add more than that. Secondly, we had to select the substances in batch mode instead of sequentially, whereas the algorithms as we used them were not necessarily meant to be used in batch mode. Another obvious limitation is the fact that we weren't actually able to add the substances to the data to see how a model trained on it would improve, which could be something to look into in the future.

7.4 Future Work

In this section we will discuss suggestions for future work that could be useful in furthering the research on sampling bias.

The first thing that could be helpful is the invention of a formal measure for sampling bias. This would not only be helpful because it could prevent sampling bias in models going undiscovered, and help confirm sampling bias if there is already a suspicion, but also because it could make the testing of sampling bias mitigation algorithms easier and more trustworthy. Instead of relying on performance metrics or density plots to tell us how much sampling bias is still present, we could just see how the bias level as calculated through the formal measure would decrease as the algorithm runs.

A second idea for future work is to improve the density algorithm. There still seems to be potential and a lot of sense in the idea of preventing the AL algorithm from selecting too many outliers, thereby speeding up the mitigation process. However, as discussed in the previous sections the density algorithm as it is now does not seem to improve on conventional active learning. We can think of two ways that could be interesting to look at for improving the algorithm.

The first would be the addition of a class criterion, which favors the minority class if the distribution of the sample is low. Of course low is vague description, but our advice would be to test what works best for this. One could even go up to favoring the minority class if its contribution to the sample is any less than 50 percent. We suggest this because from our experiments on class imbalanced data, the downfall of the algorithm seems to be recognising the instances from the minority class, which is clear from the balanced accuracy as well as the recall.

Another improvement suggestion would be to exchange the proximity measure that is used by the algorithm with a measure that is independent of the model, in order to eliminate the risk of bias seeping into the proximity values and therefore into the selection of instances by the algorithm. Another benefit of this would be that the algorithm wouldn't be tied to random forest anymore and would become more versatile because one would be able to use it with more different classification models.

In the toxicology field, possible future work could be to label new substances.

An impediment to this is that precisely those substances whose absence is suspected to be causing bias are most difficult to label by toxicologists. There is ongoing discussion among toxicologists on a lot of these substances and for some substances there is not yet a consensus among toxicologists on whether they should be labeled PBT or not. However, since the gap between the 'definitely toxic' and 'definitely non-toxic' substances seems to be so vast - based on our discussions with the toxicology expert -, an idea could be to at least try to add more substances to the labelled set that are suspected to be closer to the decision boundary, in order to mitigate some of the suspected bias. One way to do this could be by using one of the active learning algorithms discussed in this thesis to select substances and actually label these substances. This would be a very long and difficult process, but maybe things like in vitro testing, transfer learning or using existing models could aid in making this process a bit more manageable.

At the RIVM, a lot of promising steps are being made towards integrating more usage of machine learning into the toxicology field and seeing how this can aid toxicology prediction and reduce the need for animal testing. We expect a lot of progress to be made in this direction in the coming years.

Lastly, an important suggestion for future work would be to test these bias mitigation technique on data that is more prone to the kind of bias that affects the fairness of a machine learning algorithm, e.g. race, age or gender bias. One benefit of testing mitigation techniques on such data would be that it could be easier to determine how much bias is left in the model, because the bias has a more clear focus. There are also already formal measures for such discriminatory bias in machine learning algorithms. Apart from these benefits it would of course also aid the movement towards more fair machine learning, which is an very important in the current climate, where more and more fairness issues in artificial intelligence are coming to light. It is important to be careful with our use of machine learning, especially when people are involved, and not be too fixated on just the performance of the algorithms we use to make decisions that affect people's lives.

8 Conclusion

In summary, these results indicate that the uncertainty algorithm does equally as well or better than the density algorithm in terms of performance metrics of the model, in all cases except for high class imbalance combined with very low bias. Looking at the sample distribution development for both plots, it is hard to say that one algorithm has a better ability to make the sample more closely resemble the 'population'. One thing that is clear is that the density algorithm tends to have a heavier focus on the areas that are dense in the 'population' and not dense in the sample. This seems to hold until the bias level is too high, which puts the algorithm at risk of missing certain undersampled areas. Another thing that is noticeable is that the density algorithm queries a lot less outliers than the density algorithm, which is the main reason Richard et al. gave for creating the

algorithm. Taking all this and the fact that the uncertainty algorithm is much quicker to implement and run into account, as it stands, we would choose the uncertainty only algorithm over Richard et al.'s algorithm with the added density score.

However, we think changes could be made to the density algorithm that could improve its performance. One of these possible changes is the addition of a component that favors the current minority class in the sample, which could help with the accuracy on the minority class as well as the recall(if the minority class is the positive class like in our setting). Another possible alteration would be to exchange the current proximity measure by one that is independent of the model, which would eliminate the risk of perpetuating bias and make the algorithm more versatile.

During the toxicology experiment the density algorithm selected more substances that the expert would expect to be selected from a toxicological point of view, namely well-known borderline/disputed substances. the substances selected by the uncertainty algorithm were more diverse in terms of their structure. From this, we could conclude that the density algorithm does a better job at selecting those borderline cases which are precisely the substances whose absence is presumed to cause bias. But the fact that we only looked at the first 20 selected substances and the limitation of not being able to do sequential querying means that our experiment on the toxicology data was too limited to draw ultimate conclusions. Therefore, possible future work would be to actually label new substances in undersampled regions of the the feature space, possibly with aid of one of the algorithms discussed in this thesis, to get a better idea of how well these techniques work for the specific toxicology data. Not only would this further the research we did in this thesis, but it could also contribute towards safer and more trustworthy toxicology prediction models, that could advance the field and reduce the need for animal testing.

Other possible work that we see for the future is creating a formal bias measure for sampling bias as well as testing the algorithms on data about humans that contain discrimination-prone features like race, gender, income or age, to see how well they fare in terms of improving fairness.

References

[1] Fry, A., Littlejohns, T. J., Sudlow, C., Doherty, N., Adamska, L., Sprosen, T., Collins, R., Allen, N. E. (2017). Comparison of Sociodemographic and Health-Related Characteristics of UK Biobank Participants with Those of the General Population. American Journal of Epidemiology, 186(9), 1026–1034.

[2] Reddy, S., Dávalos, L. M. (2003). Geographical sampling bias and its implications for conservation priorities in Africa. Journal of Biogeography, 30(11), 1719–1727.

[3] Richards, J. W., Starr, D. L., Brink, H., Miller, A. A., Bloom, J. S., Butler, N. R., Berian James, J., Long, J. P., Rice, J. (2012). Active learning to overcome sample selection bias: Application to photometric variable star classification. Astrophysical Journal, 744(2).

[4] Settles, B. (2009). Computer Sciences Active Learning Literature Survey. January.

[5] Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004, 903–910.

[6] Sugiyama, M., Krauledat, M., Müller, K.-R. (2014). Covariate Shift Adaptation by Importance Weighted Cross Validation. Journal of Machine Learning Research, 8, 985–1005.

[7] Chapelle, O., Scholkopf, B. and Zien, A., 2006. Semi-supervised learning. 2006. Cambridge, Massachusettes: The MIT Press View Article. http://www.acad.bg/ebook/ml/MITPress-

[8] I. Dopido, J. Li, P. R. Marpu, A. Plaza, J. B. Dias, and J. A.Benediktsson, "Semisupervised self-learning for hyperspectral imageclassification," IEEE Transactions on Geoscience and Remote Sensing,vol. 51, no. 7, pp. 4032–4044, 2013.

[9] Zhao, J., Xie, X., Xu, X. and Sun, S., 2017. Multi-view learning overview: Recent progress and new challenges. Information Fusion, 38, pp.43-54.

[10] Kozodoi, N., Katsas, P., Lessmann, S., Moreira-Matias, L., Papakonstantinou, K. (2020). Shallow Self-learning for Reject Inference in Credit Scoring. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11908 LNAI, 516–532.

[11] Ntoutsi, E., Fafalios, P., Gadiraju, U., Iosifidis, V., Nejdl, W., Vidal, M. E., Ruggieri, S., Turini, F., Papadopoulos, S., Krasanakis, E., Kompatsiaris, I., Kinder-Kurlanda, K., Wagner, C., Karimi, F., Fernandez, M., Alani, H., Berendt, B., Kruegel, T., Heinze, C., ... Staab, S. (2020). Bias in data-driven artificial intelligence systems—An introductory survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(3), 1–14.

[12] Corbett-Davies, S. and Goel, S., 2018. The measure and mismeasure of fairness: A critical review of fair machine learning. arXiv preprint arXiv:1808.00023.

[13] Wachter, S., Mittelstadt, B. and Russell, C., 2021. Why fairness cannot be automated: Bridging the gap between EU non-discrimination law and AI. Computer Law Security Review, 41, p.105567.

[14] Jeong, W., Lee, K., Yoo, D., Lee, D. and Han, S., 2018. Toward reliable and transferable machine learning potentials: uniform training by overcoming sampling bias. The Journal of Physical Chemistry C, 122(39), pp.22790-22795.

[15] Iosifidis, V. and Ntoutsi, E., 2019, November. Adafair: Cumulative fairness adaptive boosting. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (pp. 781-790).

[16] Liu, A. and Ziebart, B., 2014. Robust classification under sample selection bias. Advances in neural information processing systems, 27, pp.37-45.

[17] Kamishima, T., Akaho, S., Asoh, H., Sakuma, J. (2012). Fairness-aware classifier with prejudice remover regularizer. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7524 LNAI(PART 2), 35–50.

[18] Persello, C., Bruzzone, L. (2014). Active and semisupervised learning for the classification of remote sensing images. IEEE Transactions on Geoscience and Remote Sensing, 52(11).

[19] Angwin, Larson, Mattu, Kirchner (2016). Machine Bias - Risk Assessment in Criminal Sentencing. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[20] Tufekci, Z., 2014, May. Big questions for social media big data: Representativeness, validity and other methodological pitfalls. In Eighth international AAAI conference on weblogs and social media.

[21] Datta, A., Tschantz, M.C. and Datta, A., 2014. Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination. arXiv preprint arXiv:1408.6491.

[22] Thomas F. Schrager. What is toxicology? https://web.archive.org/web/20070310233247/http://www.toxicologysource.com/whatistoxicology.html

[23] Krewski, D., Acosta Jr, D., Andersen, M., Anderson, H., Bailar III, J.C., Boekelheide, K., Brent, R., Charnley, G., Cheung, V.G., Green Jr, S. and Kelsey, K.T., 2010. Toxicity testing in the 21st century: a vision and a strategy. Journal of Toxicology and Environmental Health, Part B, 13(2-4), pp.51-138.

[24] Wang, M. W. H., Goodman, J. M., Allen, T. E. H. (2021). Machine Learning in Predictive Toxicology: Recent Applications and Future Directions for Classification Models. Chemical Research in Toxicology, 34(2), 217–239.

[25] Idakwo, Gabriel, Joseph Luttrell IV, Minjun Chen, Huixiao Hong, Ping Gong and Chaoyang Zhang. "A Review of Feature Reduction Methods for QSAR-Based Toxicity Prediction." Challenges and Advances in Computational Chemistry and Physics (2019)

Appendix A Toxicology Experiment

CAS 11 digits	SMILES	k OH (AOPv1.92) atmospheric	t1/2 atmosphere in hours	BIOWIN3	t1/2 (water)hrs
071501-12-7	C1(=O)C(C(C)CC)CCCC1C(C)CC	3.64E-11	1.06E+01	2.7118	6.50E+02
067990-31-2	O(c(c(O)ccc1C(C(CC2C3(C)C)C3C)C2)c1)C	4.48E-11	8.60E+00	2.3351	1.36E+03
084473-68-7	C(=O)(SCC(C1CCC(C)=CC1)C)C	1.03E-10	3.73E+00	2.7299	6.28E+02
093843-00-6	N(CCC1)(C1)CC=C(c(cccc2)c2)C(CCCC3)C3	1.89E-10	2.04E+00	2.371	1.26E+03
024344-21-6	C1(=O)C(C2=CCCCC2)CCCC1C3=CCCCC3	1.98E-10	1.94E+00	2.6056	8.00E+02
020201-60-9	c1(ccc(C2CC(CC23)C(C)C3(C)C)cc1OC)O	4.48E-11	8.60E+00	2.3351	1.36E+03
067859-99-8	c1(C(OC\C=C(\C)/CC\C=C(/C)\C)=O)c(N)cccc1	2.13E-10	1.81E+00	2.6003	8.08E+02
005875-45-6	c1(cc(C(C)(C)C)ccc1C(C)(C)C)O	8.92E-11	4.32E+00	2.3753	1.25E+03
093858-52-7	N(c(c1)cccc1)CCc2cc(C)c(C)cc2	7.03E-11	5.48E+00	2.3637	1.28E+03
097552-88-0	C1(C=CC=CC1(CCc(c2)cccc2)C3CCCCC3)O	1.69E-10	2.28E+00	2.47	1.04E+03
006264-59-1	N(=Cc(ccc1)o1)c2ccc(Cc(ccc3N=Cc(ccc4)o4)cc3)cc2	2.05E-10	1.88E+00	2.3411	1.34E+03
093951-24-7	n1ccn(C(c2c(C)cc(C)cc2C)=O)c1c(c3)cccc3	9.80E-11	3.93E+00	2.355	1.30E+03
074298-63-8	c12N(Cc3ccc(Cl)cc3)C(C)=Nc1cccc2	9.67E-11	3.98E+00	2.3504	1.32E+03
017222-56-9	C(c(c1)cccc1)(c(c2)cccc2)(O)CCCCCCC	2.01E-11	1.92E+01	2.7053	6.58E+02
035106-15-1	C(OC1CC(C)CCC1C(C)C)(OCC)=O	2.00E-11	1.92E+01	2.6946	6.72E+02
002398-96-1	O(c(ccc(c1ccc2)c2)c1)C(N(c(cccc3C)c3)C)=S	4.01E-10	9.59E-01	2.3983	1.20E+03
093982-70-8	C(C(C)C)(OC(C)\C=C\C1C(C)(C)CCC=C1C)=O	1.53E-10	2.52E+00	2.5429	9.04E+02
005450-24-8	c1(ccc(C(C)(C)C)cc1C2CCCCC2)O	5.88E-11	6.55E+00	2.4551	1.07E+03
003689-76-7	n1(Cc2ccc(cc2)Cl)c3c(cccc3)nc1C	9.67E-11	3.98E+00	2.3504	1.32E+03
085650-85-7	c12c(cc(C)c(C)c1C(C)CC)cccc2	1.21E-10	3.18E+00	2.5054	9.73E+02

Table 7: Substances queried by the density algorithm.

CAS 11 digits	SMILES	k OH (AOPv1.92) atmospheric	t1/2 atmosphere in hours	BIOWIN3	t1/2 (water)hrs
071501-12-7	C1(=O)C(C(C)CC)CCCC1C(C)CC	3.64E-11	1.06E+01	2.7118	6.50E+02
067990-31-2	O(c(c(O)ccc1C(C(CC2C3(C)C)C3C)C2)c1)C	4.48E-11	8.60E+00	2.3351	1.36E+03
084473-68-7	C(=O)(SCC(C1CCC(C)=CC1)C)C	1.03E-10	3.73E+00	2.7299	6.28E+02
093843-00-6	N(CCC1)(C1)CC=C(c(cccc2)c2)C(CCCC3)C3	1.89E-10	2.04E+00	2.371	1.26E+03
024344-21-6	C1(=O)C(C2=CCCCC2)CCCC1C3=CCCCC3	1.98E-10	1.94E+00	2.6056	8.00E+02
020201-60-9	c1(ccc(C2CC(CC23)C(C)C3(C)C)cc1OC)O	4.48E-11	8.60E+00	2.3351	1.36E+03
067859-99-8	c1(C(OC\C=C(\C)/CC\C=C(/C)\C)=O)c(N)cccc1	2.13E-10	1.81E+00	2.6003	8.08E+02
005875-45-6	c1(cc(C(C)(C)C)ccc1C(C)(C)C)O	8.92E-11	4.32E+00	2.3753	1.25E+03
093858-52-7	N(c(c1)cccc1)CCc2cc(C)c(C)cc2	7.03E-11	5.48E+00	2.3637	1.28E+03
097552-88-0	C1(C=CC=CC1(CCc(c2)cccc2)C3CCCCC3)O	1.69E-10	2.28E+00	2.47	1.04E+03
006264-59-1	N(=Cc(ccc1)o1)c2ccc(Cc(ccc3N=Cc(ccc4)o4)cc3)cc2	2.05E-10	1.88E+00	2.3411	1.34E+03
093951-24-7	n1ccn(C(c2c(C)cc(C)cc2C)=O)c1c(c3)cccc3	9.80E-11	3.93E+00	2.355	1.30E+03
074298-63-8	c12N(Cc3ccc(Cl)cc3)C(C)=Nc1cccc2	9.67E-11	3.98E+00	2.3504	1.32E+03
017222-56-9	C(c(c1)cccc1)(c(c2)cccc2)(O)CCCCCCC	2.01E-11	1.92E+01	2.7053	6.58E+02
035106-15-1	C(OC1CC(C)CCC1C(C)C)(OCC)=O	2.00E-11	1.92E+01	2.6946	6.72E+02
002398-96-1	O(c(ccc(c1ccc2)c2)c1)C(N(c(cccc3C)c3)C)=S	4.01E-10	9.59E-01	2.3983	1.20E+03
093982-70-8	C(C(C)C)(OC(C)\C=C\C1C(C)(C)CCC=C1C)=O	1.53E-10	2.52E+00	2.5429	9.04E+02
005450-24-8	c1(ccc(C(C)(C)C)cc1C2CCCCC2)O	5.88E-11	6.55E+00	2.4551	1.07E+03
003689-76-7	n1(Cc2ccc(cc2)Cl)c3c(cccc3)nc1C	9.67E-11	3.98E+00	2.3504	1.32E+03
085650-85-7	c12c(cc(C)c(C)c1C(C)CC)cccc2	1.21E-10	3.18E+00	2.5054	9.73E+02

Table 8: Substances queried by the uncertainty algorithm.