Excape WP1. Conformal Predictors

Paolo Toccaceli*, Ilia Nouretdinov*, Zhiyuan Luo*
Vladimir Vovk*, Lars Carlsson** and Alex Gammerman*

*Royal Holloway, University of London
**Astra Zeneca, Sweden

Abstract

The report summarises some preliminary findings of WP1.4: Confidence Estimation and feature significance. It presents an application of conformal predictors in transductive and inductive modes to the large, high-dimensional, sparse and imbalanced data sets found in Compound Activity Prediction from PubChem public repository. The report describes a version of conformal predictors called Mondrian Predictor that keeps validity guarantees for each class. The experiments were conducted using several non-conformity measures extracted from underlying algorithms such as SVM, Nearest Neighbours and Naïve Bayes. The results show (1) that Inductive Conformal Mondrian Prediction framework is quick and effective for large imbalanced data and (2) that its less strict i.i.d. requirements combine well with training set editing algorithms such as Cascade SVM. Among the algorithms tested with the Mondrian ICP framework, Cascade SVM with Tanimoto+RBF kernel appeared to be best performing one, if the quality criteria are precision, recall and number of uncertain predictions. The report also describes briefly the parallelization approach that allowed to distribute the computational load and reduce execution time.

1 Opening remarks

This memo documents the activity of the team working on WP1.4. The objectives of the work package are in Table 1.

This report deals specifically with item 1.4.1 and provides some account for the insights gained while applying transductive and inductive conformal prediction to large and highly imbalanced data sets common in the domain of Compound Activity Prediction.

2 Introduction

The overall objective is to predict compounds' bioactivities for the pharmaceutical industry using scalable machine learning algorithms. Within this context, the task of WP1.4 is to provide uncertainty quantification by applying conformal predictors and other techniques. This Report outlines the basic ideas of conformal predictors and describes a set of experiments using Transductive and Inductive Conformal Prediction (TCP and ICP). There are two major aims here: 1) to apply TCP and ICP to strongly imbalanced datasets of compounds in order to select active and non-active compounds;

WP1.4 Confidence estimation and feature significance (RHUL and AZ)

- 1.4.1.Optimization of the transductive and inductive conformal prediction framework (CP) for imbalanced and big data [RHUL].
- 1.4.2. Estimation of the feature significance using CP approaches and methods developed in 1.2. [RHUL]; Addressed challenges: confidence estimation.
- 1.4.3. Integration of the CP with methods developed in 1.2 [RHUL, AU, UL].
- 1.4.4. Platt scaling for probabilistic confidence estimation for supervised algorithms developed in Task 1.2. [UL]; Addressed challenges: confidence estimation.

Table 1: Work package WP1.4

2) to experiment with very large sets of high-dimensional data to find the most accurate and computationally efficient algorithms.

The questions we want to consider are:

- What is the most effective learning strategy for very large and strongly imbalanced datasets?
- What kernels could be used?
- What non-conformity measures can we use for efficient and accurate predictions?
- How to develop parallelization approach that would allow us to distribute the computational load and reduce execution time?

3 Transductive Conformal Prediction

In this section, we present the main concepts and their rationale. The reader is referred to the many publications for all the details - see, for example, [13] [6] [12].

We also assume that the reader is familiar with the general framework of classification, i.e.,

- a **training set** made of **examples** (x_i, y_i), where x_i is an **object** generally represented as a vector of **attributes** and y_i is a **label** taking a finite number of values, indicating for instance the class to which the example belongs
- a **test set** made of objects x_i, whose label we are asked to predict

The approach followed by Conformal Prediction revolves around the notion of Conformity or rather of Non-Conformity.

Intuitively, one way to view the problem of classification is that of assigning a label $\hat{y}$ to an object x so that the example $(x, \hat{y})$ does not look out of place among the

training examples $(x_1, y_1), (x_2, y_2), \ldots, (x_\ell, y_\ell)$. To find how "strange" the new example is in comparison with the training set, we use the Non-Conformity Measure (NCM) to measure $(x, \hat{y})$.

The advantage of approaching classification in this way is that this leads to a principled way to quantify the uncertainty of the prediction, under certain rather general hypotheses.

Although there is no universal method, a Non-Conformity Measure can in principle be extracted from any machine learning algorithm (as long as one has access to its internals). Note that we are not necessarily interested in the actual classification resulting from such underlying algorithm. What we are really interested in is an indication of how "unusual" an example appears, given a training set.

Armed with a NCM, it is possible to compute for any example (x, y) a *p-value* that reflects how good the new example from the test set fits (or conforms) with the training set. A more accurate and formal statement is that, chosen an $\epsilon \in [0, 1]$ it is possible to compute p-values for test objects so that they are (in the long run) smaller than ϵ with probability at most ϵ. Note that the key assumption here is that the examples in the training set and the test objects are *independent and identically distributed* (in fact, even a weaker requirement of *exchangeability* is sufficient).

The idea is then to compute for a test object a p-value for every possible choice of the label, that is for every possible *completion*.

Once the p-values are computed, they can be put to use in one of the following ways (among others):

- Given a significance level, ϵ, a *region predictor* outputs for each test object the set of labels (i.e., a region in the label space) such that the actual label is not in the set no more than a fraction ϵ of the times.
- A prediction is given, alongside its *confidence* and its *credibility*

Now that we have laid down the general direction, let's go into a bit more detail.

3.1 Non-Conformity Measure

The Non-Conformity Measure is a real-valued function

$$\mathcal{A}(\{z_1, z_2, \ldots, z_n\}, z)$$

where the $\{\ldots\}$ denotes a bag or multiset, i.e., a collection in which there can be multiple instances of the same entity (whereas a set can contain no "copies"). The Non-Conformity Measure expresses how non-conform with respect to the bag the object z is.

Transductive[1] and Inductive Confidence Machines differ subtly in the way the Non-Conformity Measures are computed and used.

[1]Transductive refers to the idea of Transduction, which was proposed by Vapnik as an alternative to Induction. In Transduction, no model is built from training data (contrary to what happens in Induction). Whereas in Induction the model is meant to provide a prediction for every possible test object, Transduction aims only at providing the specific prediction for the given test object, avoiding the generation of a model valid for all possible predictions. An example of a transductive algorithm is k Nearest Neighbours. Theoretically, Transduction enjoys tighter bounds on the error rate.

In the Transductive setting, the Non Conformity Measure for a training example or for a completion $z_i = (x_i, y_i)$ is

$$\alpha_i = \mathcal{A}(\wr z_1, z_2, \ldots, z_{\ell+1}\wr \backslash z_i, z_i)$$

where $\wr z_1, z_2, \ldots, z_n\wr \backslash z_i$ denotes the bag $\wr z_1, z_2, \ldots, z_n\wr$ with z_i removed.

In practical terms, this means retraining the underlying classification algorithm after removing the example z_i from the bag (which in fact plays the role of the training set).

3.2 p-values

The p-value is computed as

$$p_y = \frac{|i = 1, \ldots, (\ell + 1) : \alpha_i \geq \alpha_{\ell+1}|}{\ell + 1}$$

where ℓ is the number of training examples and index $\ell + 1$ denotes the test object. Note that the p-value is for a specific completion, as alluded to by the subscript y which indicates a dependency on the specific choice of label.

So, for every test object there are as many p-values as possible label values (i.e., as possible completions $z_{\ell+1} = (x_{\ell+1}, y_{\ell+1})$). Note that the all α_i too need to be recomputed for every candidate label value (note there is a subtle dependency: the completion $z_{\ell+1}$ is part of the bag used to compute α_i).

3.3 Region predictor

The motivation for the notion of a *region predictor* is to give a prediction that will not be wrong (in the long term) more often than a given fraction ϵ of the times. Intuitively, one can see that in order to satisfy this guarantee, it is no longer possible to emit one and one only label, but sometimes it will be necessary to "hedge one's bets" and come up with a range of possibilities.

Chosen a **significance level** $\epsilon \in [0, 1]$, for every test object the region predictor outputs the following:

$$\Gamma^\epsilon = \{y \in Y : p_y > \epsilon\}$$

where Y is the set of the label values.

The region prediction Γ_y is a set and it can be empty or contain one or more labels. This is a key difference with respect to the usual framework of classification, in which a prediction is always provided and is unique. This has to do with the fact that the region predictor offers the guarantee that in the long term the actual label of a test object (drawn from the same distribution as the training data and under exchangeability) will not be in the predicted region (hence the prediction is wrong) more often than a fraction ϵ of the times.

3.4 Confidence and Credibility

An alternative way to report the prediction is by providing a point *prediction* with associated *confidence* and *credibility*.

We define confidence, credibility, and prediction as follows:

confidence : $\sup\{1 - \epsilon : |\Gamma^\epsilon| \leq 1\}$, i.e. the greatest $1 - \epsilon$ for which Γ^ϵ is (at most) a single value

credibility : $\inf\{\epsilon : |\Gamma^\epsilon| = 0\}$, i.e. the smallest ϵ for which Γ^ϵ is empty.

prediction : Γ^ϵ for $1 - \epsilon$ equal to the confidence. Note that with this definition the prediction is never multiple and usually contains exactly one label.

It can be argued that the reporting of prediction and confidence is analogous to the reporting of the observed level of confidence in statistics. The credibility aids in avoiding overconfidence in a prediction (e.g. when the object $x_{\ell+1}$ is unusual).

Limiting ourselves to the case in which only two values (say, 0 and 1) are allowed for the label, the prediction is the label value y for which p_y is largest, the confidence is $1 - \min(p_0, p_1)$ and the credibility is $\max(p_0, p_1)$, where p_0 and p_1 are two p-values that correspond to two possible completions.

4 Conformal prediction: Mondrian framework

The error rate guarantee of the region predictor seen above does not distinguish among label values. This creates a problem especially in imbalanced data sets, such as those encountered in this exercise. The problem occurs because with imbalance the guarantee can be met while the error rate for the less prevalent class could be disproportionately higher. Consider for example a guarantee of 1% error rate in a classification problem with a data set in which one class accounts for 1% of the population. The classifier could simply classify every test object as belonging to the more prevalent class and this would satisfy the guarantee, despite resulting in 100% error rate on the less prevalent class.

It is possible to offer per-label-value guarantees with Mondrian Conformal Prediction. In general Mondrian[2] Conformal Predictors split all examples (x_n, y_n) into categories $k(n, x_n, y_n)$ and set a separate significance level ϵ_k for each category. For the purposes of this report, we'll consider only label-conditional Conformal Prediction $k(n, x_n, y_n) = y_n$.

The fundamental advantage of Mondrian Conformal Prediction is that it can guarantee that in the long run the p-values assigned to objects of each type k are smaller than ϵ_k with probability at most ϵ_k.

The output of Mondrian Conformal Prediction can be interpreted in the same ways as described above: region prediction or prediction with confidence and credibility (plus other ways not described here).

4.1 p-values for a Label-conditional Conformal Prediction

The key difference introduced with the label-conditional CP is in the way the p-values are calculated. The p-value for a hypothesis $y_{\ell+1} = y$ about the label of test object $x_{\ell+1}$ is defined as follows:

$$p(y) = \frac{|\{i = 1, \ldots, (\ell+1) : y_i = y, \alpha_i \geq \alpha_{\ell+1}\}|}{|\{i = 1, \ldots, (\ell+1) : y_i = y\}|}$$

The difference with respect to the earlier definition of p-value is that the comparisons are restricted to the α_i associated with training examples with the **same** label as the hypothetical completion.

As discussed in the previous section, this transforms the global guarantee into a per-label-value or more formally into a label-conditional guarantee.

[2]The name derives from the fact the graphical representation of the specific type of division in categories (the taxonomy) considered here reminds one of the distinctive style of the Dutch painter Piet Mondrian.

4.2 Inductive Conformal Prediction

In the Transductive mode presented so far, in principle the α_i have to be recomputed for every completion. This can be a prohibitive computational burden. For instance in the case of using an SVM as underlying, one has to retrain $\ell + 1$ SVMs (each with ℓ training examples) for every completion (this is because for each α_i, the bag, which is what we are training on, is different: it is in fact the whole training set plus the hypothetical completion, with the i-th element removed). Luckily, it is possible to conceive of a different mode of operation for a Conformal Prediction.

In the Inductive Conformal Prediction, the training set is divided into a *proper training set* and a *calibration set*. The proper training set is used to train the underlying Machine Learning algorithm and the calibration set is used to compute the α_i on the basis of the trained underlying machine learning algorithm.

This results in a much more manageable computational load. The training of the underlying classifier occurs once only. Roughly speaking, the α_i are obtained as a by-product of the machine learning algorithm prediction which is generally much less onerous that the training phase[3].

In our experiments, we applied the Inductive Conformal Prediction (catering for big data sets) to a label-conditional setting (catering for imbalance), so we'll focus on that next.

4.3 Label-conditional Inductive Conformal Prediction

To combine the notion of Mondrian Conformal Prediction with that of Inductive Conformal Prediction, we have to revise the definition of p-value seen for the Mondrian case so that it incorporates the change brought about by splitting the training set and evaluating the α_i only on the calibration set.

It is customary to split the training set at index h so that examples with index $i \leq h$ constitute the proper training set and examples with index $i > h$ (and $i \leq \ell$) constitute the calibration set.

The p-values for a hypothesis $y_{\ell+1} = y$ about the label of $x_{\ell+1}$ are defined as

$$p(y) = \frac{|\{i = h+1, \ldots, \ell+1 : y_i = y, \alpha_i \geq \alpha_{\ell+1}\}|}{|\{i = h+1, \ldots, \ell+1 : y_i = y\}|}$$

In other words, the formula above considers only α_i associated with those examples in the calibration set that have the same label as that of the completion we are currently considering (note that also $\alpha_{\ell+1}$ is included in the set of α_i used for the comparison). As in the previous forms of p-value, the fraction of such α_i that are greater than or equal to $\alpha_{\ell+1}$ is the p-value.

Finally, it is important to note that ICPs can be applied under less restrictive conditions. The requirement of i.i.d. can in fact be dropped for the proper training set, as the i.i.d. property is relevant only for the populations on which we calculate and compare the α_i, that is, the calibration and testing set.

[3]Obviously, this is not quite true for (transductive) algorithms such as kNN for which there is no real training phase - hence, the use of the adverb 'generally'...

5 Underlying Algorithms: SVM and Cascade SVM

As discussed in the previous section, Conformal Prediction hinges on the notion of nonconformity score. The general approach for defining in concrete terms a nonconformity score involves an underlying Machine Learning algorithm from which it is possible to extract a measure of how well a test example fits (or conforms) with the training examples.

SVMs have been widely been used in QSAR approach [14] and SVM has been used here as one of the underlying algorithms. It can be argued that the main limitation of this approach arises from the limited scalability of the SVM. Data sets of the size considered here (100k+) are big enough to pose some technical problems. The Quadratic Programming optimizers involved in the training of an SVM are known to have a time complexity of the order of $O(n^3)$ (although widely available implementations are thought to be somewhere between $O(n^2)$ and $O(n^3)$) and in terms of space complexity the (dense) Gram matrix has $O(n^2)$ entries.

The application of SVM to large-scale data sets is an active research area [16, 1, 15, 4]. Since this topic is not part of the RHUL team focus for this project, we decided to limit ourselves to an approximate approach inspired to the technique called CascadeSVM [8], which is described in more detail in the Appendix. This approach allowed us to carry on with application of Conformal Prediction, without stalling on the dependency on the underlying machine learning algorithm.

For the purposes of the present discussion, it is sufficient to note that the CascadeSVM decomposes the training over a large set in an appropriate sequence of SVM trainings over smaller sets that form a partition of the training set. The end result is, under certain conditions, the same as the one that would train over the entire set in one go. The CascadeSVM has also the desirable property that is possible to obtain a suboptimal result by stopping early in the sequence, thereby trading off accuracy against computational effort.

6 Kernels

A key factor for the performance of the SVM classifier is the choice of the kernel. We experimented with the Tanimoto similarity[4] and then composed it with the Polynomial Kernel and the Gaussian RBF Kernel. See Table 2, where $A = (a_1, a_2, \ldots, a_p)$, $B = (b_1, b_2, \ldots, b_p)$, $a_i, b_i \in \mathbb{N}^0$.

7 Implementation

7.1 The development environment

The choice of the tools to carry out this experiment was influenced primarily by the exploratory nature of this effort. For this reason, tools and programming languages and environments that support interactivity and rapid prototyping were preferred to those that enable maximal CPU and memory efficiency. The language adopted was Python 3.4 and the majority of programming was done using IPython Notebooks in the Jupyter environment. Chemoinformatics functions were provided by the `rdkit`[10] package. Parallelization and computation distribution were supported by the `ipyparallel`[2] pack-

[4] See [7] for a proof that Tanimoto Similarity is a kernel.

Tanimoto Coefficient	$T(A,B) = \frac{\sum \min(a_i, b_i)}{\sum a_i + \sum b_i - \sum \min(a_i, b_i)}$
Tanimoto with Polynomial kernel	$TP(A,B) = (T(A,B) + 1)^d$
Tanimoto with Gaussian RBF	$TG(A,B) = e^{-\frac{\lvert T(A,A)+T(B,B)-2T(A,B)\rvert}{\gamma}}$

Table 2: Tanimoto similarity

age. SVM and other machine learning facilities were provided by the `scikit-learn`[11] package (the SVM reuses the well-established `libsvm`[3] implementation).

The computations were run on a 8-core server with 32GB of RAM, running OpenSuSE.

8 Experimental Results

We set out to apply Mondrian Conformal Prediction, which as shown in the previous section offers per-class validity guarantees, to a sample data set. The following sections detail the challenges we were faced with, the choices we made and the results we obtained at the various stages.

8.1 The data

The first step was to obtain a sample data set that would be representative of the types that are encountered in Compound Activity Prediction. One of the authors set out to select a few relevant data sets from publicly-available sources. In the end, we concentrated mainly on one, namely `827.svm`. This data set originates from BioAssay AID 827 in the PubChem public repository[5].

The original data set provides the name and/or chemical structure of the tested compound alongside the outcome of the test, either as a number (Viability as a percentage) or as Active/Inactive classification. (The way in which the classification is derived from the Viability outcome is described in the PubChem page for the test and is not relevant for the present discussion).

In the data set for this experiment, each tested compound is described by a variable number of *signature descriptors* [5] derived for us by AZ from the chemical structure of the compounds itself. Each signature corresponds to the number of occurrences of a given labelled subgraph in the molecule graph. The resulting data set can be viewed as a relatively sparse matrix of attributes (the signatures on the columns) and examples (the compounds on the rows).

Notice that the number of attributes exceeds the number of compounds in this training set. This creates a danger of overfitting and advocates the use of some form of regularization.

[5]Accessible at https://pubchem.ncbi.nlm.nih.gov/bioassay/827 "High Throughput Screen to Identify Compounds that Suppress the Growth of Cells with a Deletion of the PTEN Tumor Suppressor".

Total number of examples	=	138,287
Number of features	=	165,786
Number of non-zero entries	=	7,711,571
Density of the data set	=	0.00034
Active compounds	=	1,658
Inactive compounds	=	136,629
Unique set of signatures	=	137,901

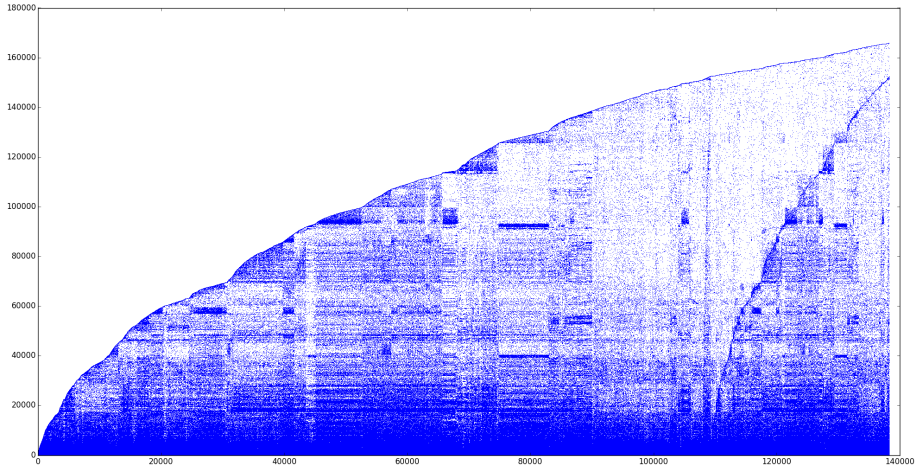


Figure 1: Matrix of the training data. The x-axis lists the compounds and the y-axis the attributes. A dot corresponds to a non-zero value for a feature

In addition to high dimensionality and sparsity, the other defining characteristic of the data set is its class imbalance, i.e., the vast difference in the representation of the Active and Inactive classes. The number of active compounds was in fact 1.2% of the total.

Finally, there are distinct compounds that have the same set of signature descriptors. In fact, as shown in the table, there are 386 more compounds than unique sets of signature descriptors. Interestingly, there are 4 pairs of distinct compounds that share the same descriptors but belong to different classes.

8.2 Criteria for model assessment

It seems that the assessment of the quality of a classifier for drug discovery is still very much an unresolved question. Jain & Nicholls [9] for instance are quite critical of the current lack of consensus. We decided to report the actual confusion matrix and let the readers draw their conclusions. In our view, precision (ratio of true positives to total number of predicted positives) and recall (ratio of true positives to total number of true positives) are the most relevant criteria. In the case of conformal predictors, also the number of uncertain predictions is of interest (of course, the lower the better).

8.3 SVM

As mentioned already in sec. 5, there are practical limits on the size of the training set for an SVM, therefore SVM needs to be parallelised to be applied to a larger set. On our platform, it was possible to push the training set size to 20,000, but this resulted in effectively commandeering the server on which we were running the training for a considerable duration. So we limited ourselves to a much more manageable training set size of 10,000.

With a training set of this size, the performance of the classifier was consistently unsatisfactory, as less than 1 in 10 of the Active test compounds was correctly classified.

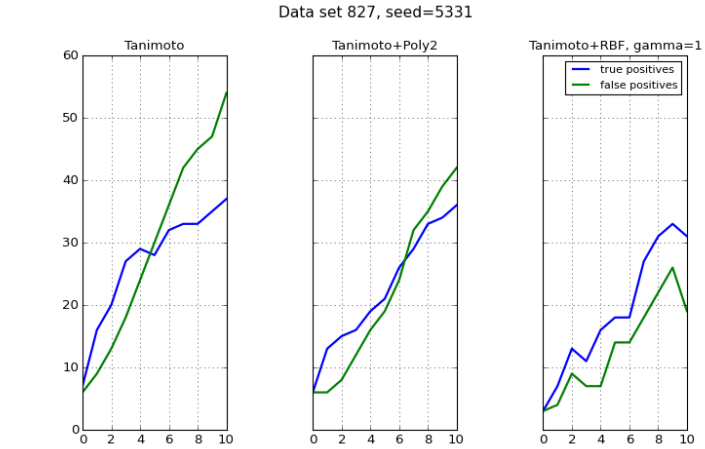
It was evident that improvement in classification required that a larger fraction of the available data set be used for training.

8.4 Cascade SVM

In fig. 2, one can see the progressive improvement in the SVM performance for successive stages of the CascadeSVM (i.e., as more blocks of training examples have been processed). The y-axis has counts of True Positives and False Positives. The tables at the bottom provide the confusion matrices. The 3 subplots refer to three different choices of kernel, namely Tanimoto, Tanimoto+Poly, Tanimoto+RBF as discussed in sec.6.

The CascadeSVM consisted of 11 stages, using in total 12 training sets of 10,000 examples each (the initial stage uses two sets). The performance was evaluated on a testing set of 10,000 examples (the testing set and the training sets were of course kept separate).

The class imbalance was addressed with the use of per-class weighting of the C hyperparameter, which results in a different penalization of the margin violations. The per-class weight was set inversely proportional to the class representation in the training set.



	True Act	True Inact
Pred Act	37	54
Pred Inact	70	9839

	True Act	True Inact
Pred Act	36	42
Pred Inact	71	9851

	True Act	True Inact
Pred Act	31	19
Pred Inact	76	9874

Figure 2: CascadeSVM results for seed=5331 The y-axis has counts of *true positives* (blue), i.e., Active compounds predicted as Active, and *false positives* (green), i.e., Inactive compounds predicted as Active.

Of the three kernels, the combination of Tanimoto and RBF appears to achieve the best *precision*, i.e., $\frac{\text{True positives}}{\text{True positives+False positive}}$, although it was the worst for *recall*.

In the most computationally intensive case (Tanimoto+RBF), the 11 stages of the CascadeSVM requires approx. 2 hours and 30 minutes over 7 cores.

8.5 Transductive Conformal Predictor (TCP): a non-conformity measure

For SVM, the nonconformity score α_i for an example (x_i, y_i) was calculated with the function that maintains SVM's order of non-SV examples and SV examples:

$$\alpha_i = \begin{cases} -y_i d(x_i) + 1 & \text{if } \alpha_{\text{Lagr},i} = 0 \\ \alpha_{\text{Lagr},i} & \text{if } 0 < \alpha_{\text{Lagr},i} < C \\ C - y_i d(x_i) + 1 & \text{if } \alpha_{\text{Lagr},i} = C \end{cases} \quad (1)$$

where d is the decision function of SVM trained on all the examples.

Note that the computation of each α_i requires the training of an SVM on a training set to which we add the object (x_i, y_i) for which we want compute the p-value.

The CascadeSVM was used as underlying algorithm for the Transductive Conformal Predictor. More precisely, the underlying algorithm was the SVM obtained at the last stage of the CascadeSVM.

Two key observations made it possible to compute the α_i efficiently.

1. At least with the training set sizes being used (around 10,000), the vast majority of the CPU time is actually spent computing the Gram matrix. The training of the SVM in itself takes seconds, whereas the Gram matrix calculation from scratch takes tens of minutes.

2. When adding one example to a training set and we have already computed the Gram matrix for that training set, the new Gram matrix can be obtained by computing just one additional column (row).

These two observations, coupled with the fact that it is relatively easy to parallelize the Gram matrix computation and the Gram matrix update, enabled us to apply the Transductive CP to a test set of 10,000 examples with execution times of 7 hrs to 1 day over 7 cores.

8.6 TCP using SVM with Tanimoto Kernel

The CascadeSVM whittled the 120,000 training set down to 8,477 examples (6,963 inactive and 1,514 active).

The resulting SVM was applied on a 10,000-example test set with 107 active compounds and 9,893 inactive compounds and its performance of its prediction is summarized in the confusion matrix in Table 3.

	True Active	True Inactive
Predicted active	38	55
Predicted inactive	69	9838

Table 3: Confusion Matrix for SVM with Tanimoto Kernel

The Mondrian CP p-values for the Active class and the Inactive class were computed using SVM as underlying algorithm and they are illustrated in Fig.3.

Table 4 shows the results produced by the region predictor, for some of the significance values ϵ.

epsilon	Active predicted Active	Inactive predicted Active	Active predicted Inactive	Inactive predicted Inactive	Empty predictions	Uncertain predictions
0.01	15	13	0	22	0	9950
0.05	38	56	0	138	0	9768
0.10	50	113	1	402	9	9425
0.15	51	176	3	710	24	9036
0.20	54	232	3	1270	42	8399
0.25	55	266	5	2163	71	7440

Table 4: Transductive CP Region Prediction with Tanimoto Kernel

8.7 TCP using SVM with Tanimoto + RBF Kernel

The CascadeSVM whittled the 120,000 training set down to 17,459 examples (15,914 inactive + 1,545 active).

The resulting SVM was applied on a 10,000-example test set with 107 active compounds and 9,893 inactive compounds (same test set as for the other test) and the performance of its prediction is summarized in the confusion matrix in table 5.

The Mondrian CP p-values for the Active class and the Inactive class were computed and they are illustrated in Fig. 4.

The region prediction is reported in table 6

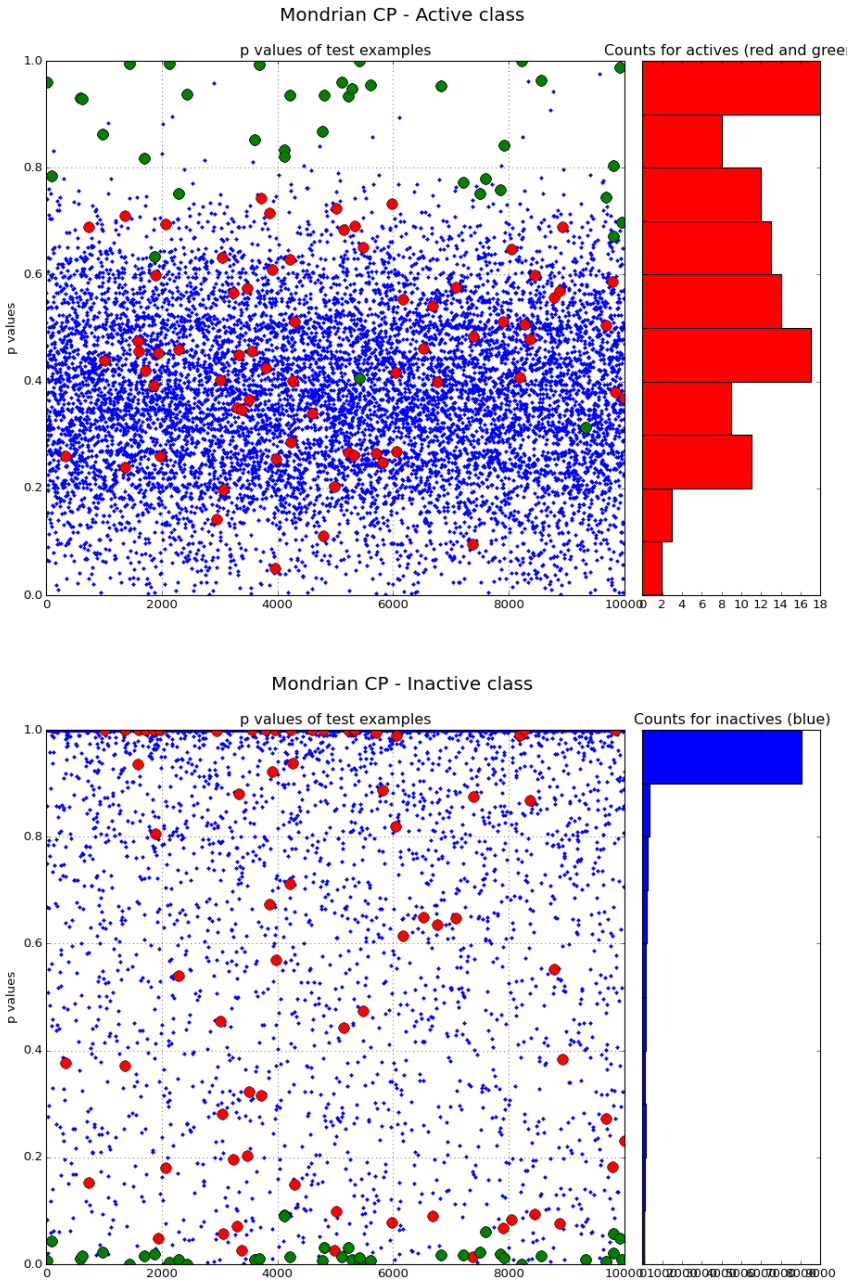


Figure 3: Mondrian p-values - Kernel: Tanimoto
The blue dots correspond to Inactive compounds. The larger dots are Active compounds, with the green colour denoting those that were correctly classified by the underlying SVM. Ideally, on the first plot the red and green dots would be uniformly distributed in $[0, 1]$, whereas the blue dots would be at the bottom; on the second plot the blue dots would be uniformly distributed in $[0, 1]$, whereas the red and green dots would be at the bottom.

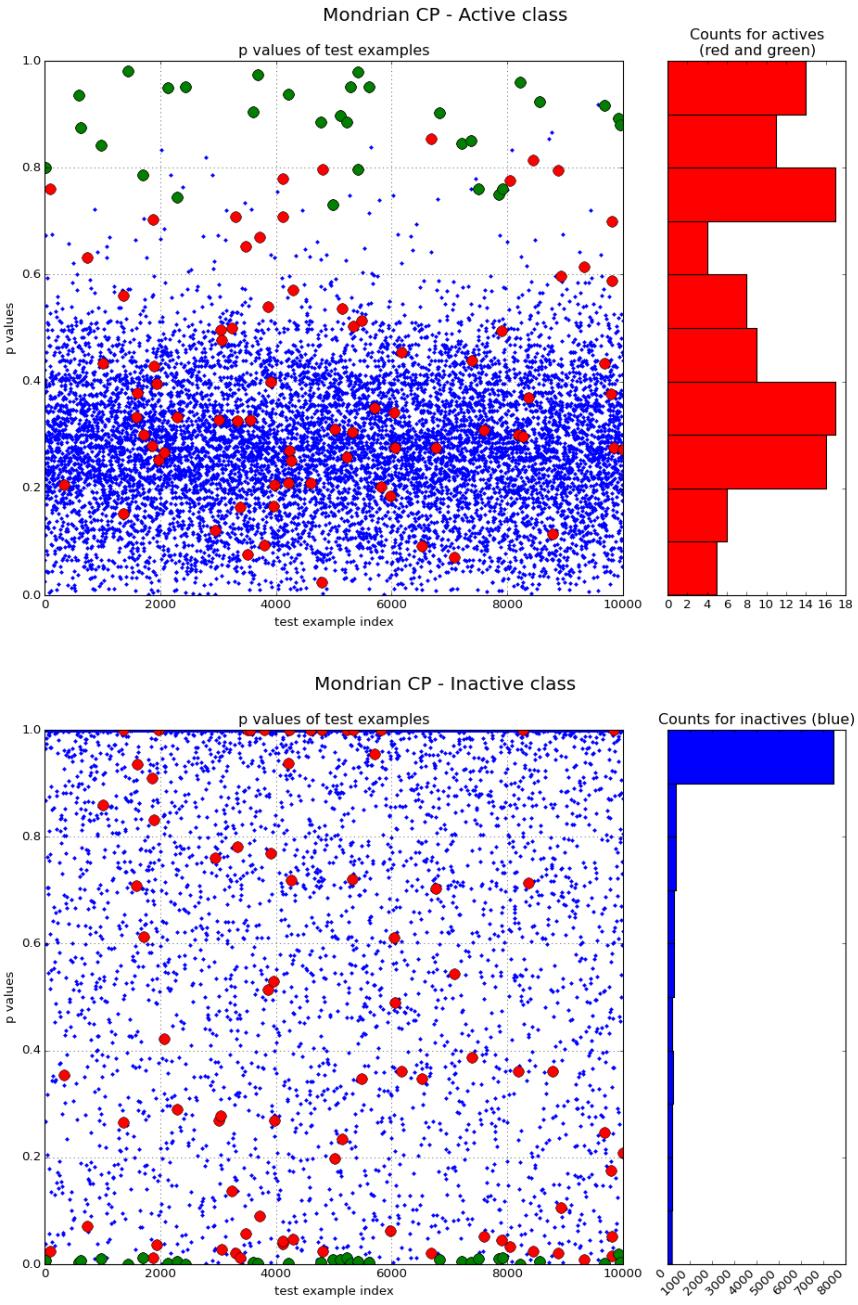


Figure 4: Mondrian p-values - Kernel: Tanimoto+RBF
The blue dots correspond to Inactive compounds. The larger dots are Active compounds, with the green colour denoting those that were correctly classified by the underlying SVM. Ideally, on the first plot the red and green dots would be uniformly distributed in $[0, 1]$, whereas the blue dots would be at the bottom; on the second plot the blue dots would be uniformly distributed in $[0, 1]$, whereas the red and green dots would be at the bottom.

	True Active	True Inactive
Predicted active	31	76
Predicted inactive	21	9872

Table 5: Confusion Matrix for SVM with Tanimoto+RBF Kernel

epsilon	Active predicted Active	Inactive predicted Active	Active predicted Inactive	Inactive predicted Inactive	Empty predictions	Uncertain predictions
0.01	26	21	0	25	0	9928
0.05	48	102	1	253	1	9595
0.10	54	202	5	806	10	8923
0.15	56	290	7	1575	32	8040
0.20	56	367	9	2533	65	6970
0.25	59	415	14	3909	114	5489

Table 6: Transductive CP Region Prediction with Tanimoto+RBF Kernel

9 Inductive Conformal Prediction

In this section, we report on some preliminary experiments using the Mondrian Inductive Conformal Prediction, introduced in sec. 4.3.

The Inductive Conformal Prediction has two advantages that are quite relevant for the problem we are tackling:

- it is less computationally demanding than the Transductive Conformal Prediction.
- it does not require the proper training set to satisfy the i.i.d. property (or to have the same distribution as the original training set).

The computational demand is smaller primarily because the calculation of the Non Conformity Measures is done on the calibration set rather than on the entire training set. This lower complexity allowed us to try out also other algorithms, namely k Nearest Neighbours, which would have been otherwise much harder in the Transductive setting.

The second advantage comes into play when the Cascade SVM is used. With our highly imbalanced data sets, the Cascade SVM extracts at each stage a set of Support Vectors that contains all the Active training examples but only a fraction (1 in 10 to 1 in 8) of the Inactive training examples. Consequently, the set of training examples that emerges from the Cascade SVM (and that we use for the underlying) no longer reflects the distribution of Actives and Inactives of the original training set. This affected the TCP using this underlying, but does not affect the ICP. In the latter case, we avoid the issue by ensuring that the proportion of Active and Inactive examples in the calibration set is the same as in the original data set.

We used 3 different ML algorithms as underlying algorithms for ICP, namely:

- SVM (Cascade) with Tanimoto+Gaussian RBF kernel
- k Nearest Neighbours with Tanimoto distance
- Naïve Bayes (Multinomial)

9.1 Experimental Results

As for the TCP experiments, we used the data set `827.svm`. However, the split between training and test set is different because of the requirement of a calibration set.

Total number of examples	=	138,287
Number of test examples	=	10,000
Number of training examples	=	90,000
Calibration set size	=	10,000 (23,520 for SVM)

The test examples were drawn with the same initialization of the pseudorandom number generator (which is also the same that was used in the tests previously reported).

The Non Conformity Measures for each of the three underlying algorithms are listed in Table 7.

Table 8 collects the performance of the region predictor on 10,000 test examples for each of the three underlying algorithms, for the best choice of parameters in each case.

It can be observed that there isn't much of a difference among these results in terms of precision (assuming we focus on the discovery of Actives), whereas the SVM-based ICP leads in terms of recall. If instead we want to consider the number of uncertain predictions as quality criterion, the 3NN mean ICP appears to outperform the other methods.

It should be noted that these results are preliminary because the run was done only on one random split. In general, a significant variance can be observed for different choices of test set and training set. However, the comparisons between algorithms are more often than not preserved, so it is in this sense (more than in the specific numerical values) that the performance tables should be read.

Tables 9, 10, and 11 show the performance of the region predictor in relation to the significance level ϵ for each of the three underlying algorithms. For completeness and similarly to what shown for the Transductive Conformal Predictors, we also provide the charts illustrating the distribution of the p values (see Figures 5,6,7)

9.1.1 Individual Predictions with confidence and credibility

Also, as an example of the calculation of confidence and credibility let's consider the examples identified respectively by the black and white arrows in Fig. 6.

Underlying	**Non Conformity Measure α_i**	**Comment**
SVM	$-y_i d(x_i)$	distance from separating hyperplane (in the "wrong" direction)
kNN	$\frac{agg_{j \neq i : y_j = y_i} d(x_j, x_i)}{agg_{j \neq i : y_j \neq y_i} d(x_j, x_i)}$	here agg is either mean of the k smallest values, or max of the k smallest values
Naïve Bayes	$-\log p(y_i = c \mid x_i)$	p is the posterior probability estimated by NB

Table 7: The Non Conformity Measures for the three underlying algorithms

Underlying	**A as A**	**NA as A**	**A as NA**	**NA as NA**	**Empty**	**Uncertain**
Casc SVM [Tani+RBF]	46	103	0	208	0	9643
3NN mean	36	83	4	785	0	9090
Multinomial Naïve Bayes	38	85	2	245	0	9630

Table 8: The performance of the region predictor

For the example pointed to by the black arrow, $p_{\text{active}} = 0.04$ and $p_{\text{inactive}} = 0.77$. The prediction is "Inactive", with confidence 0.96 (96%) and credibility 0.77 (77%).

For the example pointed to by the white arrow, $p_{\text{active}} = 0.70$ and $p_{\text{inactive}} = 0.002$. The prediction is "Active", with confidence 0.998 (99.8%) and credibility 0.70 (70%).

epsilon	Active predicted Active	Inactive predicted Active	Active predicted Inactive	Inactive predicted Inactive	Empty predictions	Uncertain predictions
0.01	46	103	0	208	0	9643
0.05	64	503	4	3247	0	6182
0.10	75	1001	6	5120	0	3798
0.15	80	1451	18	7285	0	1166
0.20	86	1886	21	7923	84	0
0.25	79	1408	18	7403	1092	0

Table 9: Region prediction vs. significance level for ICP with CascadeSVM with Tanimoto+RBF kernel

epsilon	Active predicted Active	Inactive predicted Active	Active predicted Inactive	Inactive predicted Inactive	Empty predictions	Uncertain predictions
0.01	38	83	4	785	0	9090
0.05	56	454	7	2405	0	7078
0.10	64	916	10	3557	0	5453
0.15	65	1400	13	4571	0	3951
0.20	69	1923	22	5693	0	2293
0.25	71	2424	33	6892	0	580

Table 10: Region prediction vs. significance level for ICP with Mean 3 Nearest Neighbours

9.2 Execution times

The execution times for the three ICP types (including the time for the training of the underlying) vary markedly.

In the case of the ICP with SVM, the 90,000 training examples were processed with the linear CascadeSVM approach (using the Tanimoto+RBF kernel) in blocks of 10,000 obtaining a reduced training set of 18,970 examples (1,266 Active + 17,704 Inactive). This training took 2 hrs 30 mins on 7 cores. The vast majority of the CPU time is spent

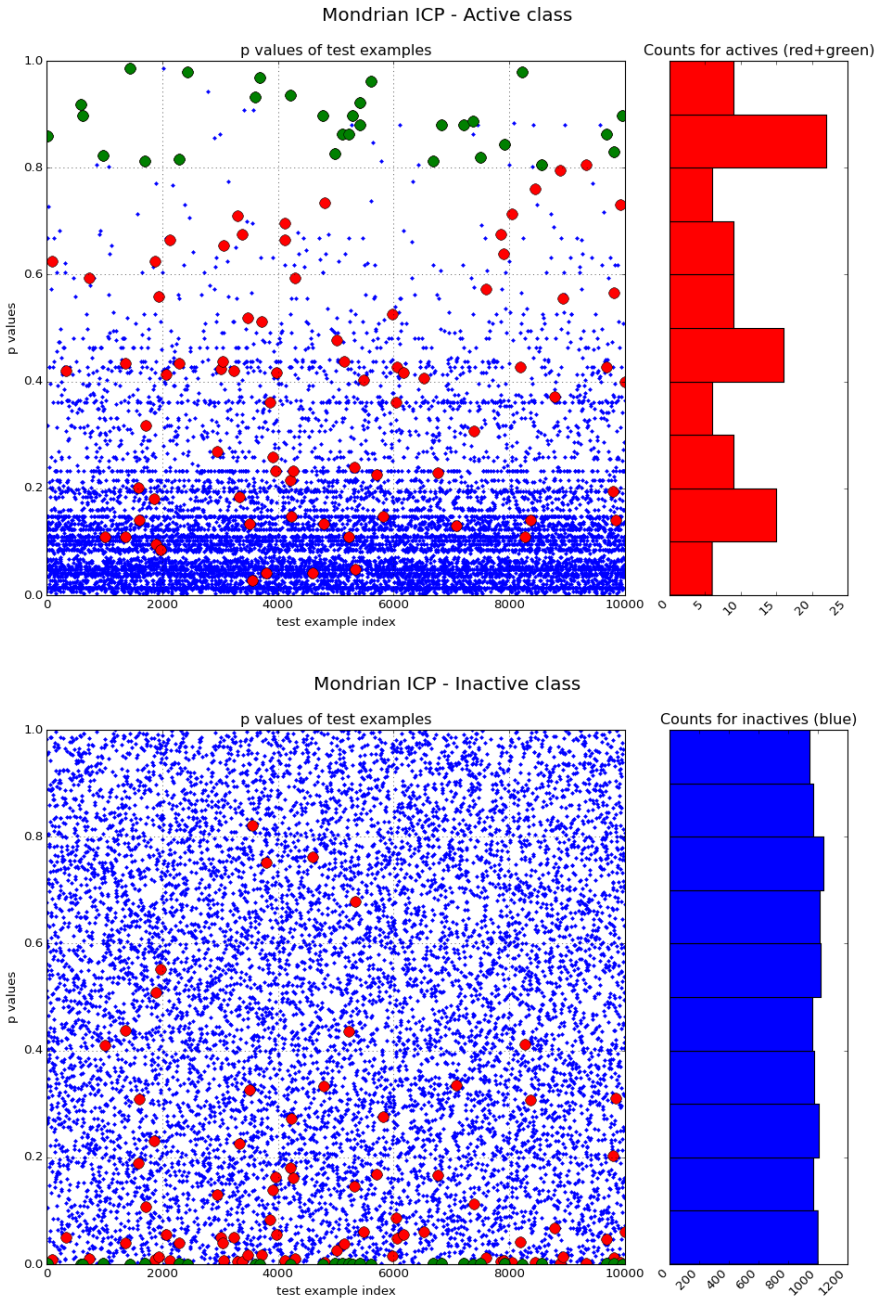


Figure 5: Mondrian ICP p-values - Kernel: Tanimoto+RBF
The blue dots correspond to Inactive compounds. The larger dots are Active compounds, with the green colour denoting those that were correctly classified by the underlying SVM. Ideally, on the first plot the red and green dots would be uniformly distributed in $[0, 1]$, whereas the blue dots would be at the bottom; on the second plot the blue dots would be uniformly distributed in $[0, 1]$, whereas the red and green dots would be at the bottom.

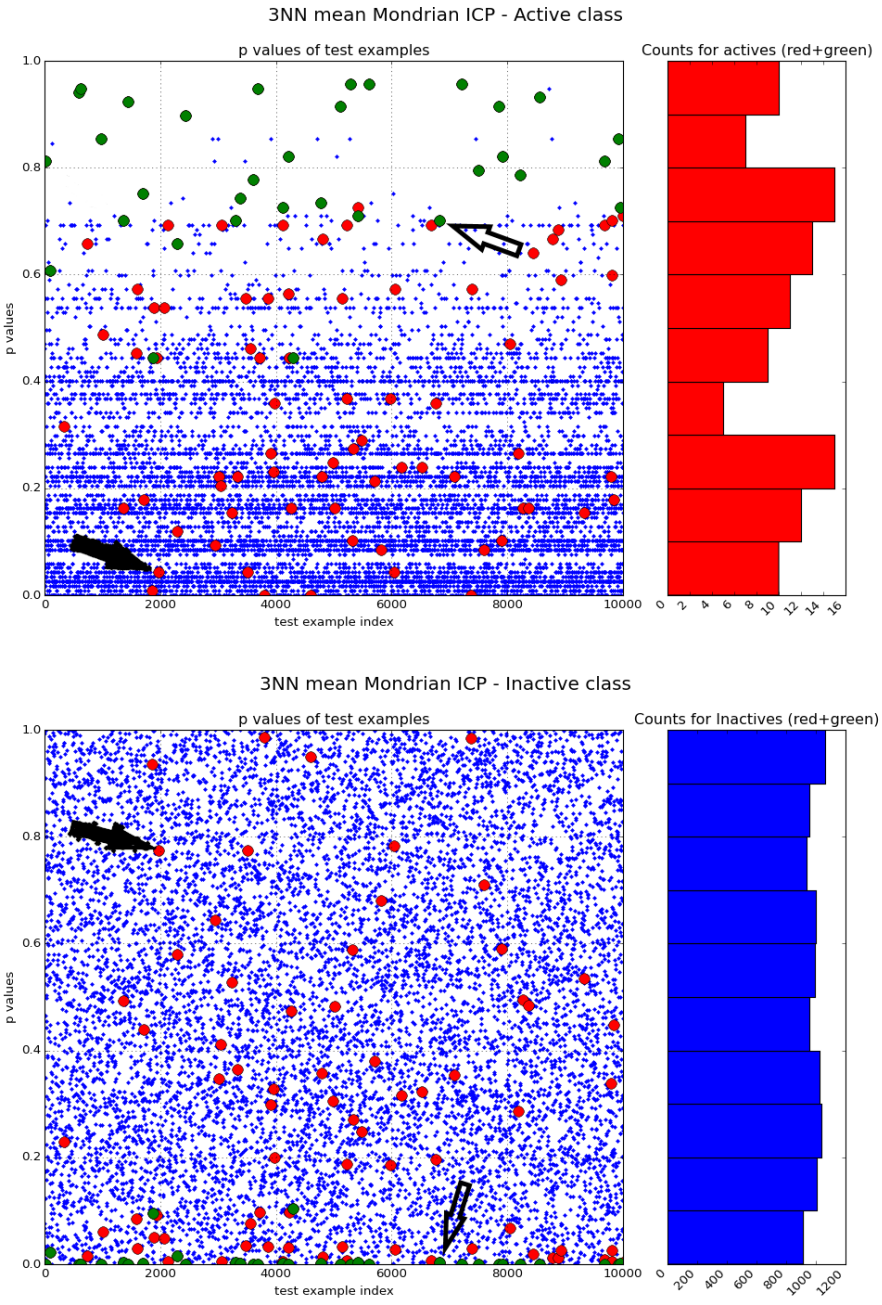


Figure 6: Mondrian ICP p-values - Underlying: Mean 3NN
The blue dots correspond to Inactive compounds. The larger dots are Active compounds, with the green colour denoting those that were correctly classified by the underlying SVM. Ideally, on the first plot the red and green dots would be uniformly distributed in $[0, 1]$, whereas the blue dots would be at the bottom; on the second plot the blue dots would be uniformly distributed in $[0, 1]$, whereas the red and green dots would be at the bottom.

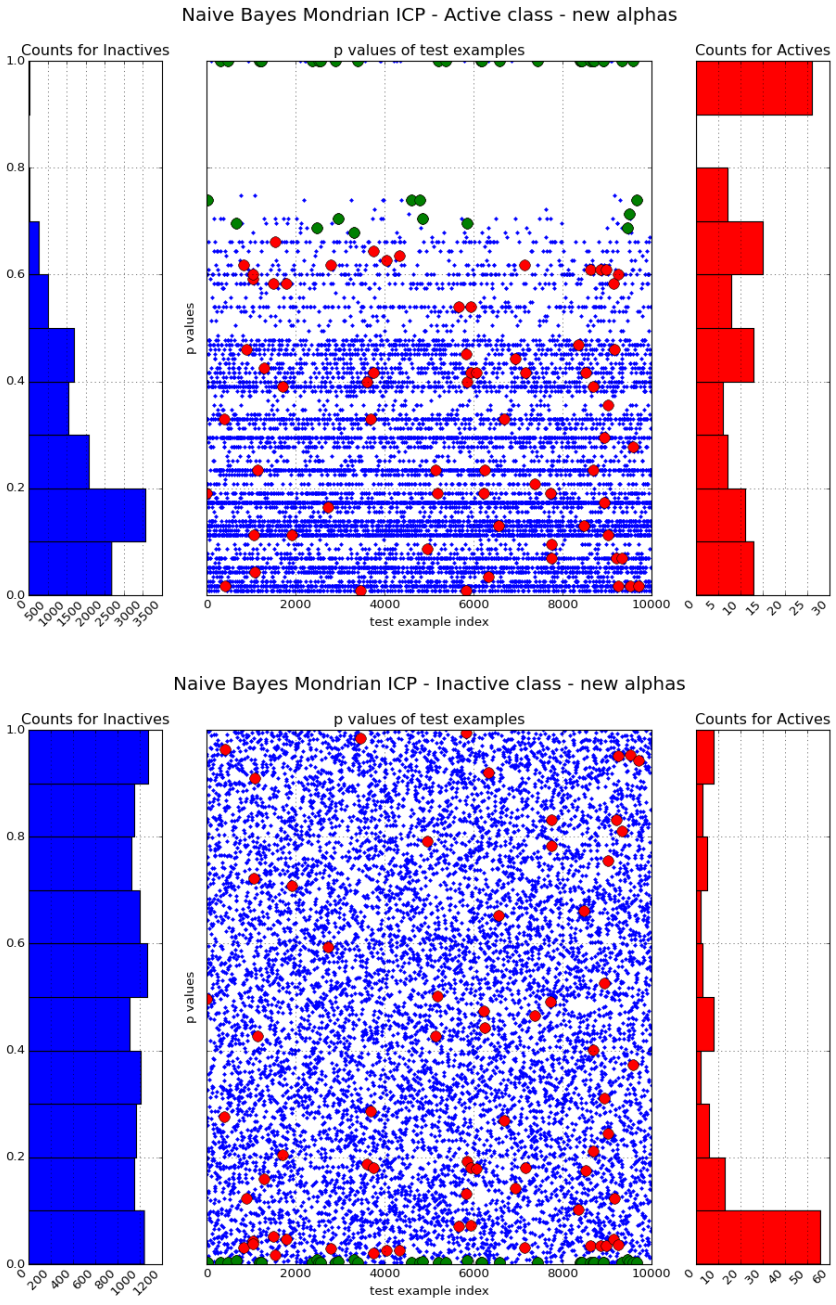


Figure 7: Mondrian ICP p-values - Underlying: Multinomial Naïve Bayes
The blue dots correspond to Inactive compounds. The larger dots are Active compounds, with the green colour denoting those that were correctly classified by the underlying SVM. Ideally, on the first plot the red and green dots would be uniformly distributed in $[0, 1]$, whereas the blue dots would be at the bottom; on the second plot the blue dots would be uniformly distributed in $[0, 1]$, whereas the red and green dots would be at the bottom.

epsilon	Active predicted Active	Inactive predicted Active	Active predicted Inactive	Inactive predicted Inactive	Empty predictions	Uncertain predictions
0.01	38	85	2	245	0	9630
0.05	53	491	8	1242	0	8206
0.10	56	1042	13	2182	0	6707
0.15	61	1503	18	3917	0	4501
0.20	69	1995	24	5246	0	2666
0.25	72	2504	29	6022	0	1373

Table 11: Region prediction vs. significance level for ICP with Multinomial Naïve Bayes

computing the Gram matrix, i.e., the kernel values for all the pairs of training examples. With this training set size, the training of the SVM in itself takes a few seconds. The values α_i of the NCM were computed on the calibration set and on the test set in 12 minutes and 5 minutes respectively on 7 cores. The calculation of the p-values was then very quick, of the order of 10 seconds.

The application of 3NN to ICP required approx. 4 mins for the calculation of the α_i for either calibration or test data, again distributing the computation across 7 cores.

Finally, the Naïve Bayes ICP method required less than 10s overall on one core (discounting the time it took to load the data set, which is around 10s too).

10 Open questions

To conclude this report, we list a number of questions we asked ourselves while carrying out the experiments described above.

1. What criteria should be used in this specific application to assess the desirability of models? Precision, recall, AUC, enhancement (a measure of the increase in the prevalence of Actives in the chosen set of compounds compared to that of random sampling)?

2. The Active/Inactive classification problem tackled here originates actually from thresholding a continuous Viability result. (In the specific example, the threshold was somewhat arbitrarily set to median minus 4 times the standard deviation.) Wouldn't the problem be better addressed as a regression problem?

3. There is some reason to believe that are gains to be made from feature selection. Would it make sense to try to use domain experts to steer this in the right direction?

4. The learning curves did not exhibit a levelling off of the test accuracy as more training examples were added. This can be informally interpreted as "the learning was not complete". This suggests that more training data would be useful. However in this specific context new data carries new features (new signature descriptors) and would increase the already huge dimensionality of the data and arguably result in ever "diminishing returns". Is there a point at which adding more data would not make economic sense?

References

[1] Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston. *Large-Scale Kernel Machines (Neural Information Processing)*. The MIT Press, 2007.

[2] Matthias Bussonnier. Interactive parallel computing in python. https://github.com/ipython/ipyparallel.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/c̃jlin/libsvm.

[4] Edward Y. Chang. Psvm: Parallelizing support vector machines on distributed computers. In *Foundations of Large-Scale Multimedia Information Management and Retrieval*, pages 213–230. Springer Berlin Heidelberg, 2011.

[5] Jean-Loup Faulon, Jr. Donald P. Visco, and Ramdas S. Pophale. The signature molecular descriptor. 1. using extended valence sequences in qsar and qspr studies. *Journal of Chemical Information and Computer Sciences*, 43(3):707–720, 2003. PMID: 12767129.

[6] Alexander Gammerman and Vladimir Vovk. Hedging predictions in machine learning. *Comput. J.*, 50(2):151–163, March 2007.

[7] Thomas Gärtner. *Kernels For Structured Data*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2009.

[8] Hans Peter Graf, Eric Cosatto, Leon Bottou, Igor Durdanovic, and Vladimir Vapnik. Parallel support vector machines: The cascade svm. In *In Advances in Neural Information Processing Systems*, pages 521–528. MIT Press, 2005.

[9] Ajay N. Jain and Anthony Nicholls. Recommendations for evaluation of computational methods. *Journal of Computer-Aided Molecular Design*, 22(3-4):133–139, 2008.

[10] Greg Landrum. Rdkit: Open-source cheminformatics. http://www.rdkit.org.

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

[12] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *J. Mach. Learn. Res.*, 9:371–421, June 2008.

[13] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[14] Derick C. Weis, Donald P. Visco Jr., and Jean-Loup Faulon. Data mining pubchem using a support vector machine with the signature molecular descriptor: Classification of factor {XIa} inhibitors. *Journal of Molecular Graphics and Modelling*, 27(4):466 – 475, 2008.

[15] Kristian Woodsend and Jacek Gondzio. Hybrid mpi/openmp parallel linear support vector machine training. *J. Mach. Learn. Res.*, 10:1937–1953, December 2009.

[16] Yang You, Haohuan Fu, Shuaiwen Leon Song, Amanda Randles, Darren Kerbyson, Andres Marquez, Guangwen Yang, and Adolfy Hoisie. Scaling support vector machines on modern hpc platforms. *J. Parallel Distrib. Comput.*, 76(C):16–31, February 2015.

11 Appendix

11.1 A simplified version of Cascade SVM

The sizes of the training sets considered here are too large to be handled comfortably by generally available SVM implementations such as `libsvm`. The approach we follow could be construed as a form of *training set editing*. Vapnik proved formally that it is possible to decompose the training into an n-ary tree of SVM trainings. The first layer of SVMs is trained on training sets obtained as a partition of the overall training set. Each SVMs in the first layer outputs its set of SVs (which is generally smaller than the training set). In the second layer, each SVM takes as training set the merging of n of the SVs sets found in the first layer. Each layer requires fewer SVMs. The process is repeated until a layer requires only one SVM. The set of SVs emerging from the last layer is not necessarily the same that would be obtained by training on the whole set (but it is often a good approximation). If one wants to obtain that set, the whole training tree should be executed again, but this time the SVs obtained at the last layer would be merged into each of the initial training blocks. A new set of SVs would then be obtained at the end of the tree of SVMs. If this new set is the same as the one in the previous iteration, this is the desired set. If not, the process is repeated once more. Vapnik proved that the process converges and that it converges to the same set of SVs that one would obtain by training on the whole training set in one go.

To give an intuitive justification, the fundamental observation is that the SVM decision function is entirely defined just by the Support Vectors. It is as if these examples contained all the information necessary for the classification. Moreover, had we had a training set composed only of the SVs, we would have obtained the same decision function. So, one might as well remove the non-SVs altogether from the training set.

In experiments discussed here, we followed a slightly different approach. Instead of a tree of SVMs, we opted for a linear arrangement as shown in Fig.8. This is arguably less efficient than the tree arrangement, but it was straightforward to implement and allowed us to monitor the effects of adding more training examples on the test accuracy resulting from the SVM as shown in sec. 8.4

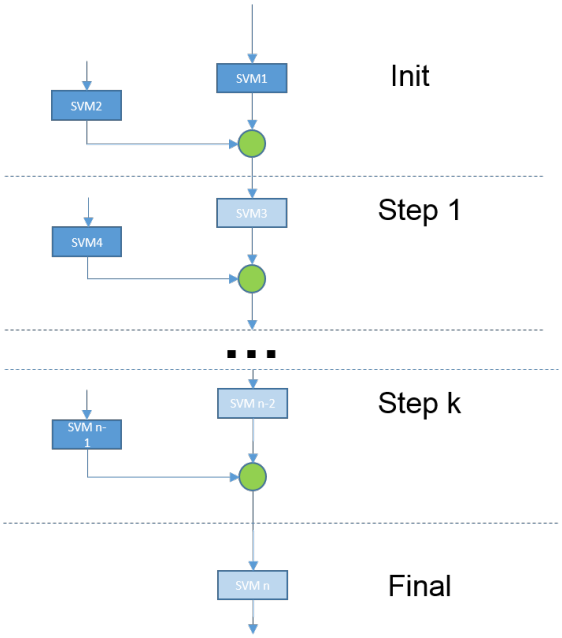


Figure 8: Linear Cascade SVM
At each step, the set of Support Vectors from the previous stage is merged with a block of training examples from the partition of the original training set. This is used as training set for an SVM, whose SVs are then fed to the next stage.