METHODOLOGY
European Journal of Research Methods for the Behavioral and Social Sciences

PsychOpen GOLD

Original Article

Check for updates

How to Use Independent Validation in Python

Thede von Oertzen [1], Hannes Diemerling [1,2], Timo von Oertzen [1]

[1] *Thomas Bayes Institute, Berlin, Germany.* **[2]** *Humboldt University, Berlin, Germany.*

Methodology, 2026, Vol. 22(2), 151–171, https://doi.org/10.5964/meth.18873

Received: 2025-07-16 • **Accepted:** 2026-02-15 • **Published (VoR):** 2026-06-30

Handling Editor: Pablo Nájera Álvarez, Universidad Pontificia Comillas, Madrid, Spain

Corresponding Author: Thede von Oertzen, Thomas Bayes Institute, Junker Jörg Str. 3, 10318 Berlin, Germany. E-mail: thede.vonoertzen@thomasbayes.de

Supplementary Materials: Code [see Index of Supplementary Materials]

Abstract

To statistically test whether two groups or models differ, classifier accuracy is compared. However, common accuracy estimates like cross-validation have unknown distributions, making them unsuitable for statistical inference. Alternatives like permutation tests or train-test splits are computationally expensive and limited to frequentist tests against chance. Independent Validation (IV) is a more flexible alternative providing a known estimate distribution. This enables both conventional hypothesis testing and Bayesian analysis of classifier performance. Although Python is most widely used for machine learning, a Python implementation of IV has been lacking so far. This article introduces such an implementation; beyond the core IV algorithm, the package allows to: (1) plot accuracy against training set size, (2) estimate the posterior distribution of the asymptotic accuracy, and (3) query the posterior for statistics and credible intervals. This makes it easy to apply IV when comparing accuracy posteriors across classes, datasets, or classifiers on the same data.

Keywords

machine learning, classification, Python implementation, independent validation, cross validation

Research in behavioral and social science frequently uses group comparisons, as evidenced by a plethora of research papers (e.g., Wee, 2000; Weisberg et al., 2011; Zhao et al., 2020). A variety of statistical tools are available for univariate comparisons, including parametric tests such as *z*-tests, *t*-tests, ANOVAs and Levene's test (Fisher, 1970; Levene, 1960; Pearson, 1900; Student, 1908). In fact, in a systematic review of major Canadian psychology journals, Counsell and Harlow (2017) found that 40% of analyses used simple

mean comparisons like ANOVA. A variety of methods is available for group comparisons for multidimensional purposes. However, all of those are predicated on assumptions regarding the model that the data is assumed to follow (Kim & von Oertzen, 2018). In instances where these assumptions are not well-founded, an alternative approach for comparing groups is necessary.

In this context, classifiers (Bay & Pazzani, 2001; Boucheron et al., 2005) have been proposed as a universal non-parametric alternative for group comparisons (Kim & von Oertzen, 2018). Group comparisons by classifier are done by training the classifier on the dataset to distinguish the groups. If the groups are identical, the classifier will be unable to classify better than by guessing. Therefore, if the classifier predicts better than guessing, it can be concluded that there is a group difference. In classical group comparison methods (e.g., ANOVA, *t*-tests, or χ^2-based tests), the group membership is an independent variable, and some dependent variables of the groups are compared. In group comparison by classifiers, group labels are generated by the classifier trained on the dependent variables, and the checks against the true group memberships (which are, again, independent variables) are then used to establish how strongly the groups differ in the dependent variables. Group comparison by classifiers is a generalization of non-parametric group comparisons, which is usually the most viable choice whenever no distributional assumptions about the dependent variables can be made. If the optimal classifier is chosen (i.e., one that asymptotically will reach a correctness exhausting all available shared entropy), the classifier group comparison will always be the most powerful group comparison.

In the domain of machine learning, classifiers are a prevalent instrument for distinguishing between groups. A variety of classifiers exist, including, for example, Decision Trees (Breiman et al., 2017), Random Forests (Ho, 1995), Support Vector Machines (Cortes & Vapnik, 1995), or *k*-Nearest Neighbor (Cover & Hart, 1967). To use classifiers for testing, it is necessary to estimate a classifier's prediction accuracy. That can be achieved in various ways; typical examples include a train-test split, Cross Validation (CV), or bootstrap (Kohavi, 1995).

Train-test split works by splitting the data into a train set and a test set. The training set is used to train the classifier, which then predicts the elements of the test set (Kohavi, 1995). The accuracy is then the number of correct classifications over the total number of classifications, which is binomially distributed and hence allows for all kinds of statistical tests. For instance, a binomial test against a null hypothesis of a guessing classifier can be used to determine if the groups differ significantly in a frequentistic analysis. However, since the amount of data is limited (Sahiner et al., 2008), splitting the data reduces the size of both the training set, giving less learning opportunities for the classifier, and the test set, adding more standard error in the binomial test. In consequence, this leads to a decline in the accuracy of the model (Santafe et al., 2015) and a less precise estimate of the accuracy. This also limits the statistical power, which is undesirable (Rossi, 2013).

Consequently, the train-test split is an uncommon method; usually, variants of CV are used instead (Devroye & Wagner, 1979; Geisser, 1975; Kohavi, 1995; Stone, 1974).

In the CV process, the dataset is split into k subsets called "folds." Then, a procedure analogous to the train-test split is repeated for each fold, in which the classifier is trained on $k-1$ folds and subsequently predicts the items of the remaining fold. In total, every element is tested, so that the test set size equals the size of the full dataset, and the size of the training set is $\frac{k-1}{k}$ times the size of the full dataset. This approach addresses the problem posed by the reduced dataset sizes, thereby eliminating the diminished statistical power inherent to the train-test split method. As the value of k increases, the training set size concomitantly increases. In the particular case where k is equal to the dataset size, this method is referred to as Leave-One-Out (LOO; Hastie et al., 2009), maximizing training-and test set size.

It is often believed that the accuracy of CV predictions also follows a binomial distribution (Salzberg, 1997). However, this is incorrect due to the dependency inherent in the repeated training and testing procedure, which results in an increased variance, as conjectured by Bouckaert (2003) and proven by Kim and von Oertzen (2017). This results in alpha inflation of frequentistic tests, as has already been observed by Dietterich (1998). It also makes Bayesian estimation of the accuracy impossible. This effect should not be confused with the alpha inflation caused by actual dependency between the samples of the dataset themselves (Kohavi, 1995). The alpha inflation in this case is caused by the repeated training and testing process, which creates dependencies of the probabilities that samples are classified correctly, even if the data samples themselves are independent. The accuracy distribution of CV is an unknown distribution that depends on the choice of classifier.

One approach to conducting a hypothesis test against the null hypothesis of no difference between the groups is to utilize permutation tests (Pesarin & Salmaso, 2010). Permutation tests repeatedly estimate the CV accuracy on the dataset while shuffling the group labels. This allows one to approximate the distribution of the accuracy under the assumption of no difference between the groups. Comparing the CV accuracy with the original labels to this distribution allows one to generate a significance test against the null hypothesis of no group difference. However, permutation tests are computationally expensive, as they have to repeat the CV procedure multiple times. In addition, they can only be used for frequentistic tests against the null hypothesis of no group differences, not for any other significance test (e.g., whether one classifier outperforms another) or any Bayesian analysis.

Another possibility is to repeatedly sample a training set by bootstrapping and then testing those samples that have not been sampled in the bootstrap (Kohavi, 1995). This emulates a Bayesian approach, as it generates a distribution around the point estimate on training set sizes identical to the full dataset size. However, the method introduces a dependency between the training and test data samples, as some of those are repeated,

which compromises the resulting distribution. In general, the distribution is not the posterior distribution, so that it cannot correctly be used for frequentistic or Bayesian testing.

As all of these methods are not suitable for Bayesian estimates and computationally expensive, Independent Validation (IV) was proposed by Kim and von Oertzen (2018). IV uses samples for training exclusively after they have been predicted, ensuring independence between the predictions. In this way, the training set size increases during the validation procedure, and tests on almost all samples of the dataset. This consequently leads to a binomial distribution of the accuracy. With this known distribution of the accuracy, it is possible to perform frequentistic tests against different null hypotheses and, more importantly, Bayesian estimation and testing of the accuracy.

However, for small datasets IV shows a tendency to underestimate the accuracy, since testing in the first few steps has a small training set size. The issue was addressed by Braun et al. (2023) by using an estimate of the asymptotic accuracy as the training set size approaches infinity. This utilizes the fact that IV provides correctness of items on differing training set sizes, which allows one to estimate the function mapping training set size to the accuracy distribution.

A preliminary version of this method was implemented in R (Braun et al., 2023; R Development Core Team., 2010). R is a useful tool for statistical computing, with a wide range of statistical libraries and packages. However, R is not explicitly built for machine learning applications. Instead, Mooney (2022) reports that the by far most common language for machine learning applications is Python (van Rossum, 1995). To make IV commonly available for the machine learning community, an implementation in Python is required. In addition, the R implementation of IV does not estimate the asymptotical accuracy in a Bayesian way, which is one of the most important advantages of IV.

Although the majority of the article can be understood without explicit programming skills, it is helpful to have some basic Python knowledge. We recommend the beginner-friendly Python course, "Python for Beginners" by Open HPI, which can be found freely available online at https://open.hpi.de/courses/python2025. It provides a practical introduction to Python programming. The official Python website https://www.python.org/ also offers comprehensive documentation and tutorials for getting started. As an IDE for running IV, we suggest using PyCharm, though alternatives such as Visual Studio Code or Jupyter Notebooks are equally suitable.

In this article, an implementation of IV in Python is introduced.[1] It applies a Bayesian algorithm that can compute the accuracy of the classifier within each class (e.g., the specificity and sensitivity for detecting a depression), on the whole dataset, and the Balanced Accuracy (BAC) for a weighted comparison of the classes. For each of these accuracies, the posterior distribution can be obtained for the asymptotic accuracy or

1) The package can be downloaded at von Oertzen (2026).

for every training set size. A real data example is provided to explain the usage of the package and to demonstrate the results. The article closes with a discussion of the package for the research field.

Background and Implementation

This section introduces the mathematical foundations of the IV process and a Markov Chain Monte Carlo simulation (MCMC; Metropolis et al., 1953) used for the improvements of the methods.

IV Process

To reiterate, Kim and von Oertzen (2018) showed that independence of the results is ensured if every tested point has not been used for training beforehand. To achieve this, IV starts with a small starting set as the initial training set, tests a batch of points, records the result, and then adds this batch to the training set. The classifier is subsequently retrained on the updated training set. This process continues iteratively until the entire dataset has been used. A flowchart illustrating this procedure is shown in Figure 1.

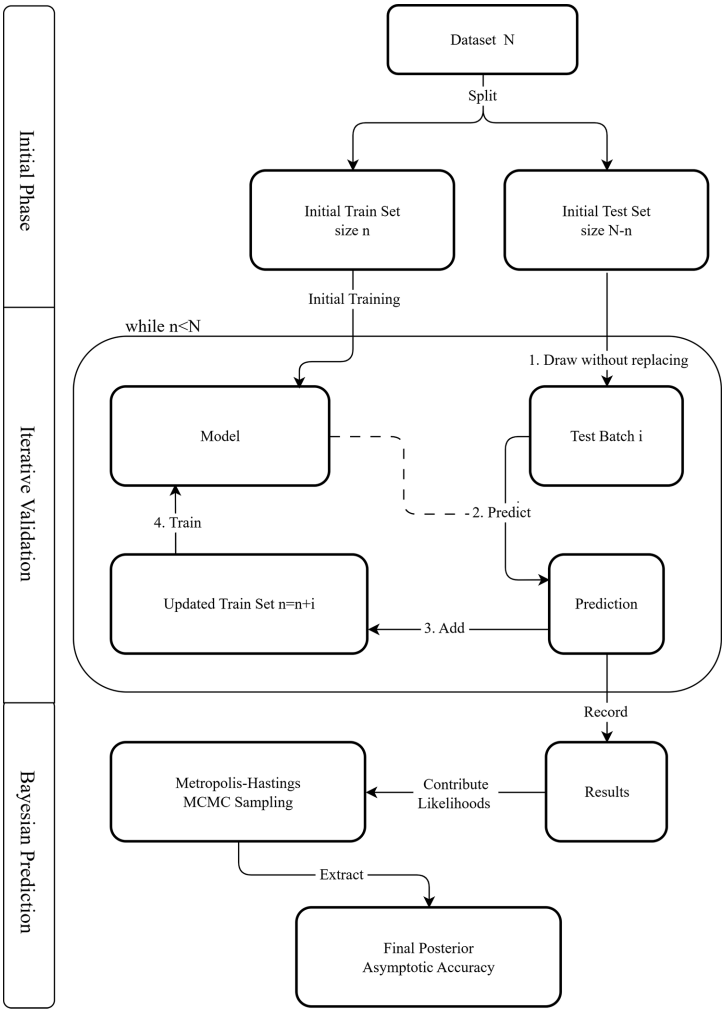
In theory, the initial training set size can be zero, which causes the classifier to begin by guessing the class of the first batch of samples before adding it to the training set. However, in practice, the training set is typically initialized with a small number of samples. This is because the first classifications provide no information while still introducing noise. Additionally, some classifiers require a certain minimum number of samples to function; for instance, a k-Nearest-Neighbor classifier needs at least k elements in the training set.

In the current implementation, the default initial training set size is two. For larger datasets, increasing this size to ten or more is recommended.

After being trained on the current training set, the classifier predicts the next batch of samples. The number of correct classifications in this batch is recorded for each class, along with the corresponding training set size. By default, the batch size is set to one. However, for larger datasets increasing the batch size is recommended. Larger batch sizes have minimal effect on the information for large datasets while significantly reducing computational costs.

Figure 1

Iterative IV Procedure That Maintains Predictive Independence by Testing Data Before Training Inclusion



Note. The algorithm partitions the total dataset into an initial training set and a test pool. Repeatedly, batches are sampled without replacement, predicted by the current model, and added to the training set until the test pool is exhausted. The test results are then used to compute the likelihood in the Metropolis-Hastings MCMC algorithm, which estimates the posterior distribution of the asymptotic accuracy.

Mathematical Model for the Likelihood

In the next step, the recorded training set sizes and corresponding correctness data are used to estimate the parameters of the function that maps training set size to correct-

ness probability. Braun et al. (2023) showed that the accuracy within each class can be approximated by:

$$P(\text{outcome} = 1) = a - \frac{b}{n} \tag{1}$$

where a is the asymptotic accuracy, that is, the theoretical accuracy as $n \to \infty$, and b is an offset factor that determines the deviation from this asymptote for finite training sample size n.

The likelihood for a correct classification at any value of (a, b) is given by Equation 1, while the likelihood of a misclassification is one minus this value. The overall likelihood of the complete set of classification results is the product of the likelihoods for each classification result. To prevent numerical underflow and improve computational stability, the current implementation uses the log likelihood.

Markov Chain Monte Carlo

An MCMC is then used to sample from the posterior distribution of a and b. A uniform prior between 0 and 1 is used for a, and a flat prior on the positive numbers for b. The MCMC implemented here is a Metropolis Hastings algorithm, chosen for its computational efficiency and robustness (Hastings, 1970).

The MCMC can be initialized with various parameters, including burn-in size (default 100), thinning (default 50), target number of samples (default 1000), and step size for the next proposed candidate (default 0.2 in both directions). Larger burn-in sizes, more samples, and greater thinning will improve sample quality, but at the cost of increased runtime. However, since the MCMC operates on precomputed classification results, computational costs remain manageable even with large numbers of samples.

The MCMC is performed separately for each class to obtain the posterior distribution for a and b, where a represents the asymptotical accuracy in that class, i.e., the accuracy the classifier would reach for an infinite amount of training data.

Output of the Analysis

The user of the package can access the distribution of a for each class directly as a list of samples. Additionally, they can request specific metrics, such as the Maximum A-Posteriori (MAP) accuracy for that class, the posterior mean, the standard deviation, the probability of exceeding a certain threshold (i.e., the cumulative distribution function), or the probability that the asymptotic accuracy exceeds the asymptotic accuracy of a second distribution (e.g., a different classifier that the user wishes to compare against).

In addition to the within-class accuracy, the package provides the accuracy and BAC for the entire dataset, as well as any other weighted combination of class accuracies. These metrics are computed as a weighted sum of the individual class accuracies. In

the case of the BAC, all classes are weighted equally, yielding an accuracy index that is independent of the class sizes in the dataset.

The combined class accuracies are computed by first multiplying the random variable for each class by its corresponding weight, then convolving the distributions to obtain the distribution for the weighted sum.

The package also combines the posteriors of a and b to provide a distribution of the expected accuracy for any finite sample size. Similar to the asymptotic accuracies, the expected accuracy for each class, for the entire dataset, or the expected BAC for the entire dataset can be accessed. Using the same methods as described above, the package provides metrics such as the MAP, mean, standard deviation, or any cumulative probability of the probability distributions. Table 1 provides a summary of the functionalities of the IV package that will be used throughout the illustrative example.

Table 1

Summary of Available Analysis Outputs

Description	Command
Distribution of the accuracy within that class	`iv.get_label_accuracy(label)`
Weighted combination of the class accuracies	`iv.get(key=weights)`
Class accuracies weighted by frequency in dataset	`iv.get_acc_dist()`
Class accuracies equally weighted	`iv.get_bacc_dist()`
Multiple metrics dependent on sample size	`iv.get_development(key)`
Distribution functions from the Scipy.stats package	
Cumulative distribution for the accuracy	`dist.cdf(x)`
Mean of the distribution	`dist.mean()`
Standard deviation of the distribution	`dist.std()`
Distribution functions from the IV package	
Maximum a-posteriori (MAP)	`dist.map()`
Comparison of two distributions	`dist1.is_greater_than(dist2)`

Note. Further functions for distribution from Scipy.stats can be found at https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.rv_continuous.html#scipy.stats.rv_continuou

Illustrative Example

This section will introduce an example of when and how to use the package in order to use machine learning in Bayesian hypothesis testing.

Is There a Group Difference?

Assume we want to find out if there is a difference between three types of wine called Barolo, Lugana, and Primitivo. The different types of wine could represent different

treatment groups in a psychological experiment, different gender identities in a social science study, or different conditions in a neurological study.

Our dataset consists of the chemical features for 59 different Barolo, 71 Lugana, and 48 Primitivo wines.[2] The dataset is structured as shown in Table 2.

Table 2

Structure of the Wine Data Set

Wine	Alcohol	Malicacid	Ash	Ash Alcalinity	Mg	Phenols	Flavanoids
Barolo	14.23	1.71	2.43	15.6	127	2.8	3.06
Barolo	13.2	1.78	2.14	11.2	100	2.65	2.76
Lugana	12.37	.94	1.36	10.6	88	1.98	.57
Lugana	12.33	1.1	2.28	16	101	2.05	1.09
Primitivo	12.86	1.35	2.32	18	122	1.51	1.25
Primitivo	12.88	2.99	2.4	20	104	1.3	1.22

Note. Wine origin and chemical features of the wine. Only seven columns and six lines of the data set are shown here.

Assuming all three types of wine are indistinguishable with respect to the chemical features under investigation, no classifier would be able to perform better than guessing, which corresponds to a BAC of $\frac{1}{3}$. Our research hypothesis, in terms of the BAC, is whether the BAC is above $\frac{1}{3}$ for a classifier that asymptotically should be able to detect the difference.

We choose a Support Vector Machine (SVM) as our classifier, import the IV package, and initialize the IV object with this classifier and our dataset. Then, we select an initial training set size of five and the default batch size of one since our dataset is relatively small. With this setup, we run the IV process. In the following we will illustrate the code we use in simplified code snippets. Since there are random processes involved the code will not always yield the exact same results.

```
# Numpy
import numpy as np
# Scikit-Learn
from sklearn.datasets import load_wine
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
# Independent validation
```

2) The dataset for this example is available as the 'Wine' dataset at https://archive.ics.uci.edu. The Python code for the examples presented here can be found at von Oertzen (2026).

```
from independent_validation import *

# Part 1 Group Difference
wine = load_wine()
X, y = wine.data, wine.target

# use Independent validation
iv_svm = IV(X, y, SVC(gamma='scale')) # initiating
iv_svm.run_iv(start_trainset_size=5) # classify samples
```

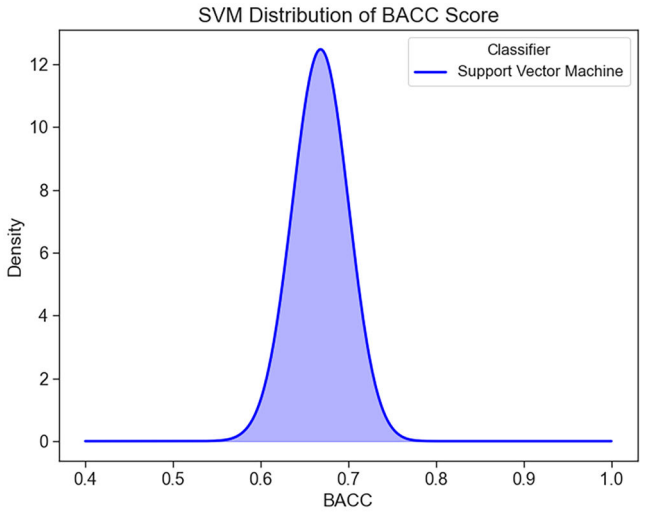
We then start the estimation process to obtain the posterior distribution for a and b. Using a burn-in phase of 1500 samples, a thinning factor of 10, and the default step size of 0.2, we generate 1000 samples from the posterior.

```
# use mcmc to compute posterior
iv_svm.compute_posterior(burn_in=1500, thin=10, step_size=0.2, num_samples=1000)
```

We can now extract various numerical values and distributions from this object; Figure 2 shows the corresponding plot. To investigate whether the model can predict the different wines, we calculate the posterior probability, assuming a flat prior, that the BAC exceeds $\frac{1}{3}$. We find that, within the limits of machine precision, the probability that the BAC is less than or equal to $\frac{1}{3}$ is effectively zero.

Figure 2

Posterior Distribution of the BAC for the SVM Classifier



```
bacc_svm_dist = iv_svm.get_bacc_dist() # get desired output

# MAP
print("Mode (MAP) value:", bacc_svm_dist.map())

# 95% CI
lower_bound = bacc_svm_dist.ppf(0.025)
upper_bound = bacc_svm_dist.ppf(0.975)
print("95% CI:", lower_bound, "-", upper_bound)

# calculate below 1/3
density_at_one_third = bacc_svm_dist.cdf(1/3) # this is density up to one third
print("Density up to 1/3:", density_at_one_third)

iv_svm.get_bacc_dist(plot=True) # (Note: The plot in the article has been adapted)
```

We extract the MAP of the distribution from the IV object, obtaining MAP = 65.46% with a 95% CI of [58.1%, 72.1%]. To further illustrate the results, we use the plot parameter of the distribution function to visualize the BAC. Since the distribution is precomputed, the function immediately generates the plot, allowing us to visually inspect the distribution around the MAP. The lower bound of the CI is at 58.1%, the guessing threshold of $\frac{1}{3}$ is not within that interval. This confirms that the prediction is better than chance. Based on these findings, we conclude that there are identifiable differences among the types of wine and that the model is capable of distinguishing between them.

We are now interested in determining whether two of our wine types, Barolo and Lugana, differ substantially in their chemical composition. To investigate this, we select a subset of our dataset containing only these two wines. We then create a new IV object and run the compute posterior method using the same default parameters as before, still using the SVM to distinguish between Barolo and Lugana. We are interested in the performance in terms of sensitivity and specificity of detecting Barolo wine. Again, this could also be the sensitivity and specificity of a screening tool in clinical psychology. We access those in the IV class by specifying the two wine labels in the accuracy method. The sensitivity, which reflects how well the model successfully identifies Barolo wines, is 76.92%. The specificity, i.e., the number of correct classifications of the other wine type, Lugana, is reported as 98.07%. The result suggests that although the classifier performs well overall, its ability to distinguish Barolo is lower than its ability to identify Lugana. Encouraged by these insights, we now turn our attention to a broader question: could another classifier perform even better?

```
selected_indices = np.where(y != 2)[0] # Select only Barolo and Lugana

X_bin = X[selected_indices]
y_bin = y[selected_indices]

```

```
6       iv_svm = IV(X_bin, y_bin, SVC(gamma='scale'))
7
8       iv_svm.run_iv(start_trainset_size=5)
9       iv_svm.compute_posterior(burn_in=1500, thin=10, step_size=0.2, num_samples=1000)
10
11      # get sensitivity
12      barolo = iv_svm.get(key=0)
13      print("Barolo (MAP) value:", barolo.map())
14
15      # get specificity
16      lugana = iv_svm.get(key=1)
17      print("Lugana (MAP) value:", lugana.map())
```

Which Classifier is Best?

We have already found a model that can distinguish between the wines, but we are not satisfied and want to know if there is a better model available. We create three more IV objects using the classifiers k-Nearest Neighbor (kNN), Random Forest (RF), and Logistic Regression (LR). We then run the IV and compute the posterior for each. To compare the different classifiers, we decide to evaluate them based on their BAC distributions, with higher values being better. See code in the next subsection (run with argument mode="bacc" in Line 10). Figure 3 shows the plotted distributions. As can be seen, the SVM and the kNN perform noticeably worse than the LR and the RF. The RF shows the best performance with a MAP of 95.64%, which is nearly perfect for our data. The LR has a MAP of 93.53% and thus is very similar. We therefore are curious about the probability that the RF outperforms the LR. A service function tests whether one random variable is greater than another, resulting in a probability of 77.59% for the hypothesis that the RF outperforms the LR on a latent level. We therefore change our model to the RF, now having a model that improves our ability to determine the wine type based on its chemical ingredients.

Which Classifier has the Best Global Accuracy?

Suppose we want to create a tool that can predict the type of wine based on its chemical ingredients. We are paid per correctly classified wine, so we are not interested in the BAC but in the global accuracy, which is the percentage of wines correctly classified. We now want to plot the accuracy distributions of our classifiers to find the best option. See code below (run with argument mode="acc"). Figure 4 shows the plot, and again the RF performs best with a MAP of 98.58% while the probability that it outperforms the LR is 98.38%. Therefore, we choose the RF.

Figure 3

Comparison of BAC Distributions for Different Classifiers

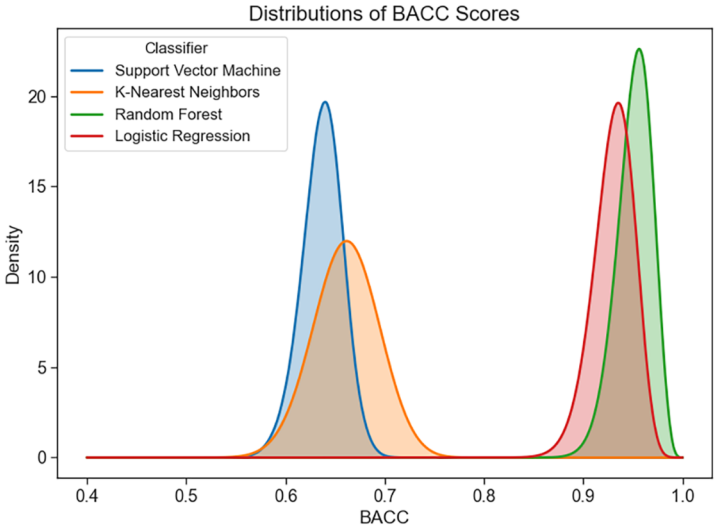
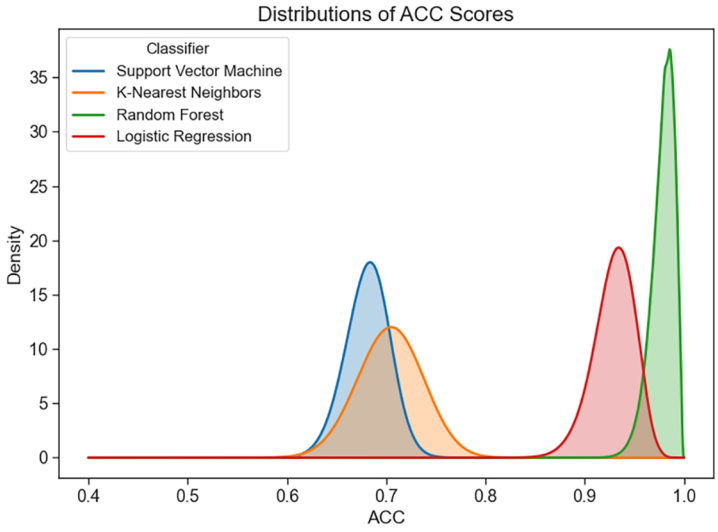


Figure 4

Comparison of Global Accuracy Distributions for Different Classifiers




```
1       # Define the classifiers.
```

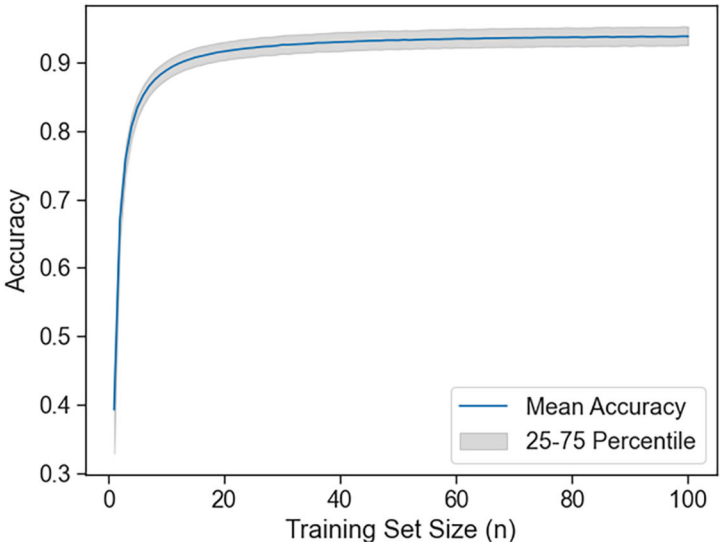
```
    classifiers = {
    'Logistic_Regression': LogisticRegression(solver='newton-cg'),
    'Support_Vector_Machine': SVC(gamma='scale'),
    'K_Nearest_Neighbors': KNeighborsClassifier(),
    'Random_Forest': RandomForestClassifier()
    }

    # Define the metric
    mode = "acc" # change this to "bacc" to get the balanced accuracy instead.
    assert mode in ['bacc', 'acc']
    bacc = mode == 'bacc'
    acc = mode == 'acc'

    # Dictionary to store the accuracy distributions.
    dists = {}
    for clf_name, clf in classifiers.items():
        print(f"\nRunning IV for classifier: {clf_name}")

        iv_instance = IV(X, y, clf)
        iv_instance.run_iv(start_trainset_size=5)
        iv_instance.compute_posterior(burn_in=1500, thin=10, step_size=0.2,num_samples=1000)

        if bacc:
            dist_clf = iv_instance.get_bacc_dist(plot=False)
        if acc:
            dist_clf = iv_instance.get_acc_dist(plot=False)
        dists[clf_name] = dist_clf

     # compare Random Forest and Logistic Regression
     prob_RF_gt_LR = dists['Random_Forest'].is_greater_than(dists['Logistic_Regression'])
     print('Probability that RF accuracy is greater than LR accuracy:', prob_RF_gt_LR)
```

What is the Expected Accuracy for Finite Sample Sizes?

There is one more thing we can do. To reduce costs, we want to use as few wines as possible for training, and we need to determine the optimal size. We investigate the impact of an increasing training set size on the expected accuracy by examining the development of the accuracy with respect to the training size. The IV object generates this posterior distribution for each possible test set size, and generates a plot showing the mean along with the first and third quartiles for every training set size from 1 to 100, which is the default value. This results in Figure 5. We look at the plot and determine that the first 20 samples are the most impactful, so we will purchase only 20 samples to save money.

Figure 5

Development of Expected Accuracy as a Function of Training Set Size




```
max_trainset_size = 100

iv_dev = IV(X, y, RandomForestClassifier())
iv_dev.run_iv(start_trainset_size=5)
iv_dev.compute_posterior(burn_in=1000, thin=10, step_size=0.1, num_samples=1000)
trainset_sizes = np.arange(1, max_trainset_size + 1)

mean_acc_list, lower_bound_list, upper_bound_list = iv_dev.get_development(key='acc', n=101, plot=True, confidence_range=0.5)
# (Note: The plot in the article has been adapted)
```

Discussion

IV is a method for evaluating classifier accuracy based on known data likelihood. Existing implementations are limited to R and are not widely available as packages (e.g., on CRAN; Braun et al., 2023). This article introduces a Python package, integrated with the scikit-learn library, which is currently the most widely used programming language in machine learning. The package employs a Metropolis-Hastings MCMC algorithm to estimate the posterior probability of classifier accuracy within each class and aggregates these estimates into multiple relevant metrics and distributions. An empirical example using the Wine dataset (Aeberhard et al., 1994) demonstrates the package's functionality and output.

Interpretations

To the best of the authors' knowledge, no other implementation exists that computes the Bayesian posterior of a classifier's asymptotic within-class accuracy. Existing methods that approximate at least frequentist baseline distributions for a purely guessing classifier have significant drawbacks: some are computationally expensive (e.g., permutation tests), others reduce available data (e.g., training-test set separation), and some do not work correctly to begin with (e.g., CV using a binomial null distribution). Moreover, none of these methods allow comparisons beyond anything but chance performance or support a Bayesian approach.

The Bayesian distributions of the asymptotic within-class accuracy serve as the foundation for computing the posterior of key indices describing classifier performance. They also enable Bayesian inference on empirical hypotheses, such as whether two groups differ or one classifier outperforms another.

These indices include the classifier's specificity and sensitivity (i.e., within-class accuracy for two classes), the asymptotic total classifier accuracy, the asymptotic balanced accuracy, and any other weighted sum of class accuracies. For all of these performance measures, the researcher can also use the package to compute the expectation for a final training set size (again represented as a Bayesian posterior). Each posterior distribution can be accessed either through the full set of samples or via summary descriptors, including the MAP, mean, standard deviation, credible intervals and percentile intervals of any size, and specific probabilities for defined thresholds. For example, researchers can determine the probability that the classifier performs better than chance or better than a chosen performance threshold.

Using these capabilities, the package enables researchers in the behavioral sciences to address various common research questions. For example, if clinical research wants to use a classifier — such as a threshold on a sum score — as a screening tool for a psychological disorder, the package provides Bayesian estimates of the sensitivity and specificity of the test. This allows one to test hypotheses about these, such as whether the sensitivity exceeds 90 percent.

Similarly, a sociologist may want to examine whether individuals with low socio-economic status (SES) perform differently from those with high SES on a diverse set of cognitive tasks that cannot be easily combined into a single score. By training a classifier on the two groups, they can use the Bayesian posterior of balanced accuracy (BAC) to determine the probability that the classifier performs better than chance, providing evidence of a meaningful group difference.

Analogously, a neurologist might investigate whether brain activities differ between two experimental conditions. Again, by training a classifier on imaging data, they can use the package to obtain a Bayesian posterior probability that the classification performance exceeds guessing.

Finally, an expert in machine learning may be interested in comparing a newly developed classifier to an existing one on a benchmark dataset. The package allows them to obtain the posterior distributions of both classifiers and visualize the posterior probability that one outperforms the other in terms of accuracy, BAC, or any other weighted accuracy measure.

Limitations

A limitation of IV is that the assumed functional form mapping training set size to classifier performance within a group is a good approximation for training set sizes around ten samples but less accurate on very low sample sizes. Additionally, testing is not entirely independent when sample sizes are very low, as misclassified items added to the training set can bias subsequent classifications. Overly small starting sets may lead to Heywood cases in the Bayesian estimation.

However, for moderate sample sizes, this issue largely disappears, as the initial training set size can be made sufficiently large. For small datasets, though, this approach results in some data loss, albeit less than in methods like train-test splitting. To mitigate this effect for small N, the Python package introduced constraints on the minimal expected success rate probability to the guessing probability. While this improves the approximation, it does not fully resolve the issue. Therefore, whenever possible, users are advised to use sufficiently large initial test sets.

The batch size used to add items to the training set in IV involves a tradeoff. From an estimation perspective, a batch size of one is optimal. However, for large datasets with thousands of samples, this approach is computationally expensive. For such large datasets, using larger batch sizes — in the order of magnitude of one tenth of the total dataset — results only in negligible loss of information, while significantly accelerating processing.

Future Directions and Conclusions

A promising future direction in this line of research is to mathematically investigate whether an improved model for expected accuracy at very low training set sizes can be developed, and possibly integrated with the existing model, which performs well for moderate sample sizes.

Another open question is to what extent IV can be applied to evaluate machine learning regression results or in the context of Large Language Models (Touvron et al., 2023). Similar to categorical classification, CV is commonly employed to assess regression performance, with various suggestions for loss functions (Picard & Cook, 1984). Accurately quantifying uncertainty in loss may require an independent validation technique analogous to the one proposed here for categorical classification.

Independent validation is the preferred method for optimizing statistically accurate accuracy estimation. With the Python implementation introduced here, it can now be seamlessly integrated with classifiers in empirical research. This will help researchers in nearly all empirical research fields to improve their research and strengthen the progress of science.

Funding: The authors have no funding to report.

Acknowledgments: The authors have no additional (i.e., non-financial) support to report.

Competing Interests: The authors have declared that no competing interests exist.

Data Availability: The Python code for examples presented in the study can be found at von Oertzen (2026).

Supplementary Materials

Type of supplementary material	Availability/Access
Data	
No study data available	—
Code	
Python code	von Oertzen (2026)
Material	
No study material available	—
Study/Analysis preregistration	
Study was not preregistered	—
Other	
No other study material available	—

References

Aeberhard, S., Coomans, D., & De Vel, O. (1994). Comparative analysis of statistical pattern recognition methods in high dimensional settings. *Pattern Recognition, 27*(8), 1065–1077.

Bay, S. D., & Pazzani, M. J. (2001). Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery, 5*(3), 213–246. https://doi.org/10.1023/A:1011429418057

Boucheron, S., Bousquet, O., & Lugosi, G. (2005). Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics, 9*, 323–375. https://doi.org/10.1051/ps:2005018

Bouckaert, R. R. (2003). Choosing between two learning algorithms based on calibrated tests. *Proceedings of the Twentieth International Conference on International Conference on Machine Learning* (pp. 51–58).

Braun, T., Eckert, H., & von Oertzen, T. (2023). Independent validation as a validation method for classification. *Quantitative and Computational Methods in Behavioral Sciences, 3*, 1–30. https://doi.org/10.5964/qcmb.12069

Breiman, L., Friedman, J., Olshen, R. A., & Stone, C. J. (2017, October). *Classification and regression trees*. Chapman Hall/CRC. https://doi.org/10.1201/9781315139470

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297. https://doi.org/10.1007/BF00994018

Counsell, A., & Harlow, L. L. (2017). Reporting practices and use of quantitative methods in Canadian journal articles in psychology. *Canadian Psychology / Psychologie canadienne, 58*(2), 140–147. https://doi.org/10.1037/cap0000074

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21–27. https://doi.org/10.1109/TIT.1967.1053964

Devroye, L., & Wagner, T. (1979). Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory, 25*(5), 601–604. https://doi.org/10.1109/TIT.1979.1056087

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation, 10*(7), 1895–1923. https://doi.org/10.1162/089976698300017197

Fisher, R. A. (1970). Statistical methods for research workers. *Breakthroughs in statistics: Methodology and distribution* (pp. 66–70). Oliver and Boyd.

Geisser, S. (1975). The predictive sample reuse method with applications. *Journal of the American Statistical Association, 70*(350), 320–328. https://doi.org/10.1080/01621459.1975.10479865

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika, 57*(1), 97–109.

Ho, T. K. (1995). Random decision forests. *Proceedings of the 3rd International Conference on Document Analysis and Recognition* (Vol. 1, pp. 278–282). https://doi.org/10.1109/ICDAR.1995.598994

Kim, B., & von Oertzen, T. (2017). *Independent validation remedies alpha inflation in classifier accuracy testing* (Technical Report). Department of Psychology, University of Virginia-Charlottesville.

Kim, B., & von Oertzen, T. (2018). Classifiers as a model-free group comparison test. *Behavior Research Methods, 50*(1), 416–426. https://doi.org/10.3758/s13428-017-0880-z

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (Vol. 2, pp. 1137–1143).

Levene, H. (1960). Robust tests for equality of variances. *Journal of the American Statistical Association, 69*(346), 364–367.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics, 21*(6), 1087–1092.

Mooney, P. (2022). *The 2022 Kaggle Machine Learning Data Science Survey*. Kaggle.

Pearson, K. (1900). X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 50*(302), 157–175.

Pesarin, F., & Salmaso, L. (2010). The permutation testing approach: A review. *Statistica, 70*(4), 481–509. https://doi.org/10.6092/issn.1973-2201/3599

Picard, R. R., & Cook, R. D. (1984). Cross-validation of regression models. *Journal of the American Statistical Association, 79*(387), 575–583. https://doi.org/10.1080/01621459.1984.10478083

R Development Core Team. (2010). *R: A language and environment for statistical computing: Reference index* [OCLC: 1120300286]. R Foundation for Statistical Computing.

Rossi, J. S. (2013). Statistical power analysis. In J. A. Schinka & W. F. Velicer (Eds.) *Handbook of psychology: Research methods in psychology, 2nd ed.* (Vol. 2, pp. 71–108). John Wiley & Sons.

Sahiner, B., Chan, H.-P., & Hadjiiski, L. (2008). Classifier performance estimation under the constraint of a finite sample size: Resampling schemes applied to neural network classifiers. *Neural Networks, 21*(2), 476–483. https://doi.org/10.1016/j.neunet.2007.12.012

Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery, 1*(3), 317–328. https://doi.org/10.1023/A:1009752403260

Santafe, G., Inza, I., & Lozano, J. A. (2015). Dealing with the evaluation of supervised classification algorithms. *Artificial Intelligence Review, 44*(4), 467–508. https://doi.org/10.1007/s10462-015-9433-y

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society Series B: Statistical Methodology, 36*(2), 111–133. https://doi.org/10.1111/j.2517-6161.1974.tb00994.x

Student. (1908). The probable error of a mean. *Biometrika, 6*(1), 1–25.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023, February). *LLaMA: Open and efficient foundation language models* [arXiv:2302.13971]. arXiv. https://doi.org/10.48550/arXiv.2302.13971

van Rossum, G. (1995). Python tutorial. In *Department of Computer Science [CS]* (R 9526). CWI. https://ir.cwi.nl/pub/5007

von Oertzen, J. T. (2026). *Implementation of independent validation in Python* [GitHub project page containing Python code for study examples]. GitHub. https://github.com/jonasthedevonoertzen/IV

Wee, A. G. (2000). Comparison of impression materials for direct multi-implant impressions. *Journal of Prosthetic Dentistry, 83*(3), 323–331. https://doi.org/10.1016/S0022-3913(00)70136-3

Weisberg, Y. J., DeYoung, C. G., & Hirsh, J. B. (2011). Gender differences in personality across the ten aspects of the Big Five. *Frontiers in Psychology, 2*, Article 178. https://doi.org/10.3389/fpsyg.2011.00178

Zhao, M., Wang, M., Zhang, J., Gu, J., Zhang, P., Xu, Y., Ye, J., Wang, Z., Ye, D., Pan, W., Shen, B., He, H., Liu, M., Liu, M., Luo, Z., Li, D., Liu, J., & Wan, J. (2020). Comparison of clinical characteristics and outcomes of patients with coronavirus disease 2019 at different ages. *Aging, 12*(11), 10070–10086. https://doi.org/10.18632/aging.103298