Implementation of independent validation in Python.

Thede von Oertzen Hannes Diemerling Timo von Oertzen

To statistically test whether two groups are different or whether two models differ, the accuracy of classifiers can be estimated and compared statistically. However, the distribution of common accuracy estimates like cross validation (CV) is unknown and depends on the classifier used, which makes it inapplicable for statistical inference. Independent validation (IV) has been suggested as an alternative, as the distribution of the IV estimate is always a binomial distribution, allowing both for conventional tests on the accuracy and the Bayesian analysis of classifier performance. Although Python is the most frequently used tool for machine learning methods, so far an implementation of IV in Python is missing. This article introduces a new Python package that implements IV. In addition to the core IV algorithm, the package includes the option to (1) plot the accuracy in dependence of the training set size, (2) to estimate the asymptotical classifier accuracy including its posterior distribution, and (3) to query the posterior distribution for confidence intervals. The package also allows to compare different accuracy posterior distributions of each class of the dataset, different classifiers on the same dataset, or of a single classifier on different datasets. An introduction is provided how to use the package for some examples, and a short simulation that shows how the package works in practice. Using this new Python package will allow empirical scientists to easily use IV in their research, for example to analyze group differences or compare group differences between different data sets.

Introduction

Group comparisons are a common task in the social sciences, as evidenced by the plethora of research papers that have been published (Wee 2000; Weisberg, DeYoung, and Hirsh 2011; Zhao et al. 2020). In a systematic review of major Canadian psychology journals, Counsell and Harlow (2017) found that 40% of analyses used univariate mean comparison. In order to compare groups, a variety of statistical tools are available, including parametric tests such as z-tests (Fisher 1915), t-tests and Levene's test. These tests are designed to identify group

differences; however, they are predicated on certain assumptions regarding model specifications (Kim and Oertzen 2018). In instances where these assumptions are not well-founded, an alternative approach for comparing groups becomes necessary.

In this context, classifiers (Boucheron, Bousquet, and Lugosi 2005; Bay and Pazzani 2001) have been proposed as a non-parametric, universal alternative for group comparisons (Kim and Oertzen 2018). In the domain of machine learning, classifiers are a prevalent instrument for distinguishing between groups.There exists a variety of classifiers, including Decision Trees (Breiman et al. 2017), Random Forests (Ho 1995), Support Vector Machines (Cortes and Vapnik 1995), and K-nearest neighbor (Cover and Hart 1967). To compare groups with a classifier the classifier is being trained to distinguish between the groups. Then the classifier attempts to predict group membership for unlabeled samples of data. If the classifier predicts better than guessing it can be concluded that there is a group difference. The estimation of the classifier's prediction accuracy is, therefore, of relevance (Kohavi 1995). This estimation can be achieved through various ways, like a train-test split, cross-validation, or bootstrap (Kohavi 1995).

Train-test split works by splitting the data into a train set and a test set. The training set is then used to train a classifier, which then predicts the test set (Kohavi 1995). The accuracy of the predictions on the test set is binomially distributed, implying that tests can be performed unproblematically. For instance, a hypothesis test against a null hypothesis of a guessing classifier can be used to determine if the groups differ significantly. However, splitting the data reduces the size of both subsets as the data is usually limited (Sahiner, Chan, and Hadjiiski 2008). This reduction in training data leads to a decline in the accuracy of the model (Santafe, Inza, and Lozano 2015). Additionally, less testing data leads to a less accurate estimation of the accuracy. A reduced sample size diminishes statistical power, which is undesirable (Rossi 2013). Consequently, the train-test split is an uncommon method; instead, most often cross validation (CV) is used (Kohavi 1995; Devroye and Wagner 1979; Geisser 1975; Stone 1974).

In the process of cross-validation, the dataset is split into k subsets, or "folds." Then, a procedure analogous to the train-test split is repeated k times. The classifier is trained on a "trainset" consisting of k-1 folds and subsequently predicts the remaining fold. This process is repeated for each fold, resulting in a total testset size equal to the full dataset and a trainset size equal to (k-1)/k times 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 statistical power of CV concomitantly rises. In the particular case where k is equivalent to the dataset size, this method is referred to as leave-one-out (LOO), a procedure that attains maximum statistical power. It is often believed that the accuracy of CV predictions follows a binomial distribution (Salzberg 1997). However, this assertion is not accurate 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 & von Oertzen (2017). The phenomenon of Alpha inflation in the context of cross-validation (CV) has been documented by Dietterich (Dietterich 1998). In the event that dependencies exist between the samples in the dataset, Alpha inflation is also caused by said

dependencies (Kohavi 1995). However, this is a separate problem from the one described herein. The Alpha inflation in this case is caused by the repeated training and testing progress, which creates dependencies of the probabilities that samples are classified correctly. The accuracy distribution of CV is an unknown distribution that depends on the choice of classifier and dataset. One approach to conducting a hypothesis test against the null hypothesis of no difference between the groups and, consequently, a guessing classifier is to utilize permutation tests (Pesarin and Salmaso 2010).Permutation tests identify the distribution of accuracy under the assumption of no difference between the groups, thereby enabling tests to be performed against this assumption as a null hypothesis. It is important to note that permutation tests are computationally intensive and can only be used to test against this specific null hypothesis.

An alternative validation method was proposed by Kim and von Oertzen (2018) called Independent Validation. This method utilizes samples for training exclusively after they have been predicted (Kim and Oertzen 2018). This approach ensures independence between the predictions and consequently leads to a binomial distribution of the accuracy (Kim and Oertzen 2018). With a known distribution of the accuracy independent validation enables testing on the accuracy of a classifier.

Independent Validation exhibited a tendency to be biased in cases of small datasets. This issue was addressed by Braun, Eckert, and von Oertzen (Braun, Eckert, and Oertzen 2023) through the development of an estimation for the theoretical accuracy of an infinite dataset (asymptotical accuracy) based on the predictions of the limited dataset. The implementation of independent validation and this estimation method was originally conducted in R (Kim and Oertzen 2018; Braun, Eckert, and Oertzen 2023; R Development Core Team. 2010). However, for machine learning, Python is the by far more common language (Mooney 2022). This may be part of the reason why Independent Validation is not yet used frequently.

An implementation of independent validation in Python follows.

Methods – Independent Validation (IV)

This section details the Independent Validation (IV) process implemented in our Python package, its demonstration through example scripts and visualization, the testing procedures ensuring its correctness, and a fictional example application.

IV Process Description

TODO: This part can be shortened, introduction again ain't necessary.

The Independent Validation (IV) process is an incremental evaluation method to assess the evolving performance of a classifier. Unlike standard approaches such as cross-validation, IV ensures statistical independence between predictions and training data by predicting each new sample *before* adding it to the training set. This independence guarantees that the resulting accuracy estimates follow a binomial distribution, thereby enabling standard statistical tests and Bayesian inference.

The IV process comprises the following core steps:

TODO: Explanation can be improved, though is fine (and universal).

1. **Initialization:**
   A small subset of the available data (e.g., the first few samples) is used to train a copy of the classifier. This initial training set must be smaller than the complete dataset.

2. **Incremental Prediction & Recording:**

   - The remaining samples are processed in batches.

   - For each batch, the classifier makes predictions on the unseen samples.
   - For every sample in the batch, the outcome (1 for a correct prediction or 0 for an incorrect one) is recorded along with the current training set size.

3. **Training Set Expansion:**
   After recording the prediction outcomes, the new samples are added to the training set, and the classifier is retrained. This iterative process continues until all samples have been processed.

4. **Posterior Distribution Computation:**
   Based on the recorded IV outcomes, the probability to classify a new sample correctly for a given training set size (n) is modeled as:

   $$p_n(outcome = 1) = \text{asymptote} - \frac{\text{offset\_factor}}{n}$$

   Here, the *asymptote* represents the classifier's theoretical accuracy as (n $\to\infty$), and the *offset factor* controls the decline from the asymptote for finite (n).
   Using the Metropolis-Hastings algorithm (an MCMC sampler), we compute the posterior distribution for these model parameters. This posterior is computed separately for each class (label) in the dataset, enabling both class-specific accuracy assessments and aggregated metrics.

5. **Aggregation to Overall Metrics:**
   Two principal aggregated accuracy metrics are derived:

- **Balanced Accuracy (bacc):** Formed by convolving the per-label accuracy distributions with equal weights.
- **Overall Accuracy (acc):** Derived from a weighted sum of the per-label accuracies, with weights given by the prevalence (frequency) of each label in the dataset.

Demo

The implementation is demonstrated through a collection of example scripts (e.g., `demo.py` and `services.py`) which exercise different facets of the IV process:

1. **Artificial Data Validation:**
   - **Binary Data Demo:**
     A synthetic dataset is generated with 2D features where the binary target is defined by the condition (X_0 + X_1 > 1). A k-Nearest Neighbors (kNN) classifier is used to run the IV process.
     Output:
     - Mean balanced accuracy is computed via repeated IV calls.
     - The overall accuracy distribution is assessed.
     - A comparison is made by computing the probability that the balanced accuracy exceeds the overall accuracy.
   - **Multi-class Data Demo:**
     An artificial dataset with three classes is processed. The IV class (`IV` in `iv8.py`) is used to log outcomes, compute the posterior using MCMC, and generate distributions for per-label and aggregated accuracies.
   - **Graphical Outputs:**
     The demo routines create several plots using Matplotlib:
     - **Histogram Plots:**
       For example, a histogram of MCMC samples (generated in `demo_mcmc()`) is overlaid with the analytic standard normal probability density function.
     - **Development Curves:**
       Line plots showcase the evolution of accuracy (mean and interquartile ranges) as a function of training set size. These plots help visualize how performance converges to its asymptotic value.
     - **Distribution Plots:**
       Density curves for balanced accuracy (e.g., saved as `demo/synthetic/bacc50_1`) and overall accuracy are generated.
2. **Real Data Validation (Titanic Demo):**
   The IV process is also demonstrated on a real-world dataset (the Titanic dataset). After necessary pre-processing and encoding, a Logistic Regression classifier is evaluated:

- The IV process records predictions over increasing training set sizes.
- Posterior distributions for the classifier's asymptotic and finite-sample performance are computed.
- Plots are generated for balanced and overall accuracy distributions as well as for development trends.

Figure placeholders in the demo include: - **Figure 1:** Histogram of MCMC samples vs. the analytical standard normal. - **Figure 2:** Density plots for balanced accuracy with varying training sizes. - **Figure 3:** Development curves showing mean accuracy and quartile bounds over increasing training set sizes.

Testing Procedures

Robustness and correctness of the IV implementation are ensured via a comprehensive battery of unit tests that cover:

1. **MCMC Sampling (`mcmc.py`):**
   - **1D and Multi-dimensional Sampling:** Tests confirm that the Metropolis-Hastings algorithm produces samples with approximately the correct mean and variance (e.g., resembling a standard normal distribution).
   - **Acceptance Rate:** The proportion of accepted proposals is verified to be within sensible bounds.
2. **Weighted Sum Distribution (`weighted_sum_distribution.py`):**
   - **Normalization:** Tests ensure that the combination of individual probability distributions yields a final density that integrates to one.
   - **Weighting Options:** Both default (uniform) and user-specified weights are tested for consistency.
3. **IV Core Functionalities (`iv8.py` and `services.py`):**
   - **IV Record Logging:** The `run_iv` method correctly logs training set sizes and prediction outcomes.
   - **Posterior Computation:** The `compute_posterior` method populates the posterior dictionary with valid MCMC samples for each label.
   - **Distribution Retrieval and Caching:** Methods such as `get_label_accuracy`, `get_bacc_dist`, and `get_acc_dist` are tested to return valid frozen distribution objects (e.g., instances of `rv_histogram`), with caching ensuring repeated calls yield consistent results.

- **Service Interface:** The high-level `independent_validation` function is tested for correct behavior under different parameter settings (e.g., when returning only the mean or the full distribution).

These tests are implemented using Python's `unittest` framework and verify the statistical properties and consistency of the simulation outcomes.

Example

To illustrate the practical application of IV, consider a fictional scenario in biomedical diagnostics.

Fictional Scenario: Diagnostic Accuracy of a Biomarker

Objective:
Assess the performance of a classifier that predicts the presence of a disease based on a single biomarker.

Fictional Dataset:

- **Features (Biomarker Level):**
  [X = {0.5,; 1.2,; 1.8,; 2.1,; 2.5,; 3.0,; 3.3,; 3.8,; 4.2,; 4.8}]
- **Labels (Disease Status):**
  [y = {0,; 0,; 0,; 1,; 0,; 1,; 1,; 1,; 1,; 1}] Here, a label of 1 indicates disease presence and 0 indicates absence.

IV Application

1. **Initialization:**
   A simple k-Nearest Neighbors classifier is initialized. The first three samples, for instance, ((0.5, 1.2, 1.8)) (all with label 0), are used to train the classifier initially.
2. **Incremental Prediction & Training:**
   - **Batch Processing:**
     The next sample, (2.1) (with true label 1), is evaluated. Assume the classifier incorrectly predicts 0.
     The outcome ((0 , with training set size 3)) is recorded.
   - The sample (2.1) is then added to the training set and the classifier is retrained.

- This process continues for the remaining samples, with each prediction's correctness recorded along with the training size.

3. **Posterior Computation:**
   After the entire dataset is processed, the IV records (comprising prediction outcomes and the respective training set sizes) are used to compute the posterior distribution of the model parameters via MCMC.

   - For example, the posterior for label 1 (disease present) might indicate an asymptotic accuracy mean of 0.87 with a credible interval of [0.82, 0.91].

4. **Aggregated Metrics:**

   - **Balanced Accuracy (bacc):**
     By convolving the separate per-label accuracy distributions (with equal weight for each class), a balanced accuracy distribution is obtained. Suppose this yields a mean balanced accuracy of 0.84 at a training set size of 50.
   - **Overall Accuracy (acc):**
     The overall accuracy is determined via a weighted average of the per-label accuracies. This metric can be used to compare the diagnostic performance against a baseline (e.g., chance-level performance).

Interpretation

- The **posterior distributions** derived from the IV process offer a transparent picture of uncertainty in classifier performance, facilitating rigorous statistical tests (e.g., testing against a null hypothesis of no predictive power).
- **Development curves** (accuracy as a function of training set size) help determine how quickly the classifier's performance converges and whether additional data could significantly improve accuracy.
- The derived plots (similar to those generated in our demos) serve as both a validation tool and a means for comparative analysis in applied research.

By combining a rigorous, incremental validation process with robust posterior inference, the described methods enable the practical application of Independent Validation in varied research settings—from artificial simulations to real data scenarios such as biomedical diagnostics.

Results

Synthetic data

To estimate the quality of a KNN classifier run on some synthetic data. For the data the features are normally distributed with different means and standard deviations for the different groups. The data consists of three different groups that are not balanced.

TODO: Add Details about data and classifier.

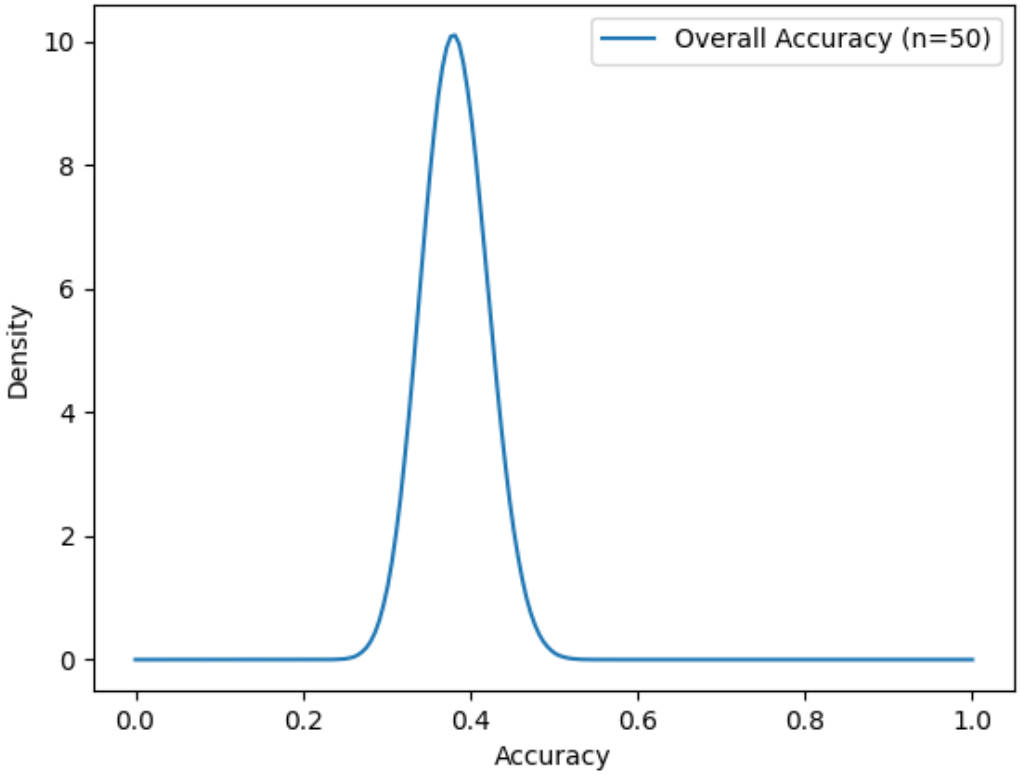

Figure 1: Estimated Accuracy Distribution of a Classifier Trained on 50 samples of the synthetic dataset.

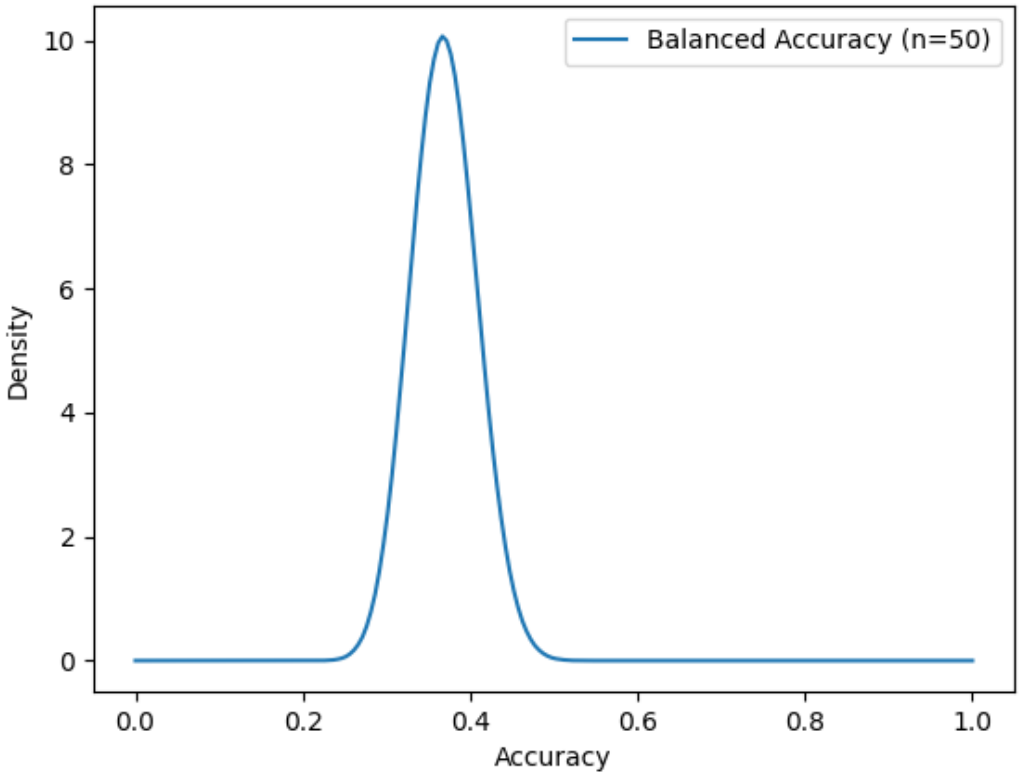

Figure 2: Estimated Balanced Accuracy Distribution of a Classifier Trained on 50 samples of the synthetic dataset.

And now the improved variant. Though this needs to be rechecked whether this is realistic.

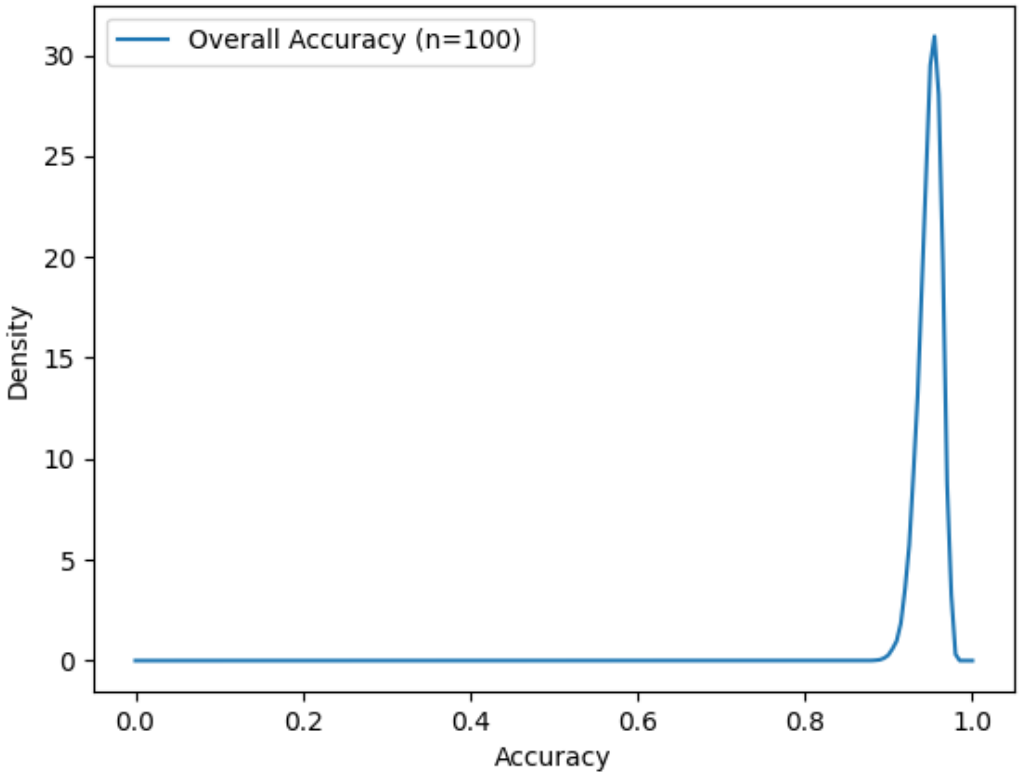

Figure 3: Estimated Accuracy Distribution of a Classifier Trained on 100 samples of the synthetic dataset.

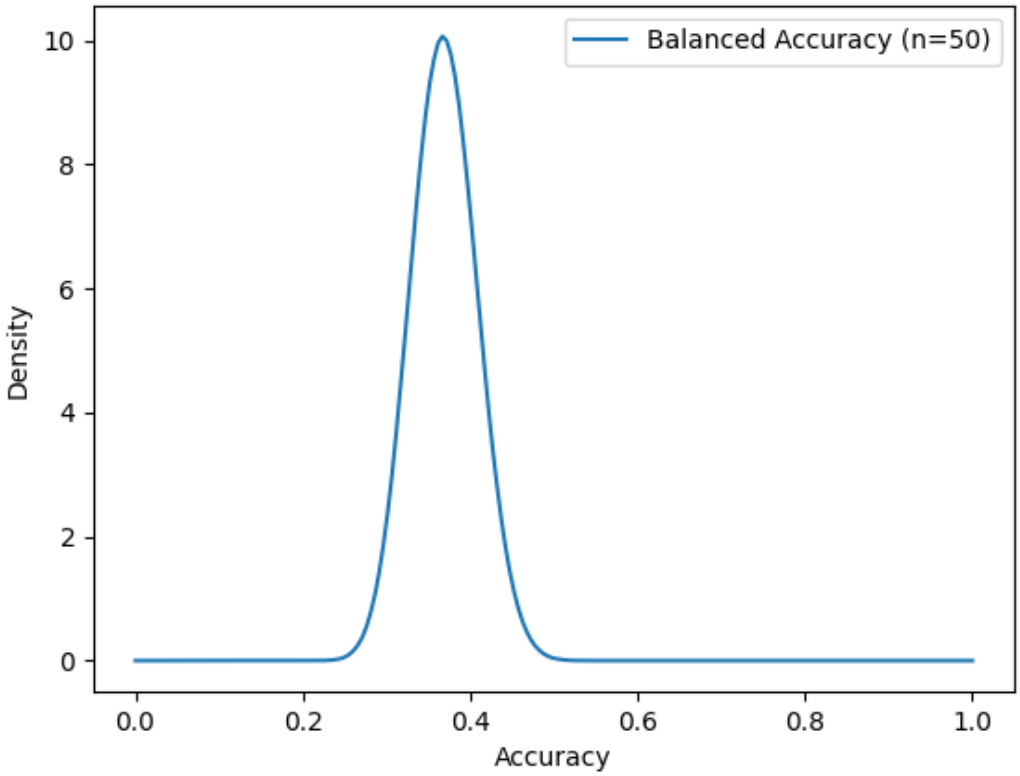

Figure 4: Estimated Balanced Accuracy Distribution of a Classifier Trained on 100 samples of the synthetic dataset.

If you are really interested in the development these metrics over increasing sample size you can plot this as well.

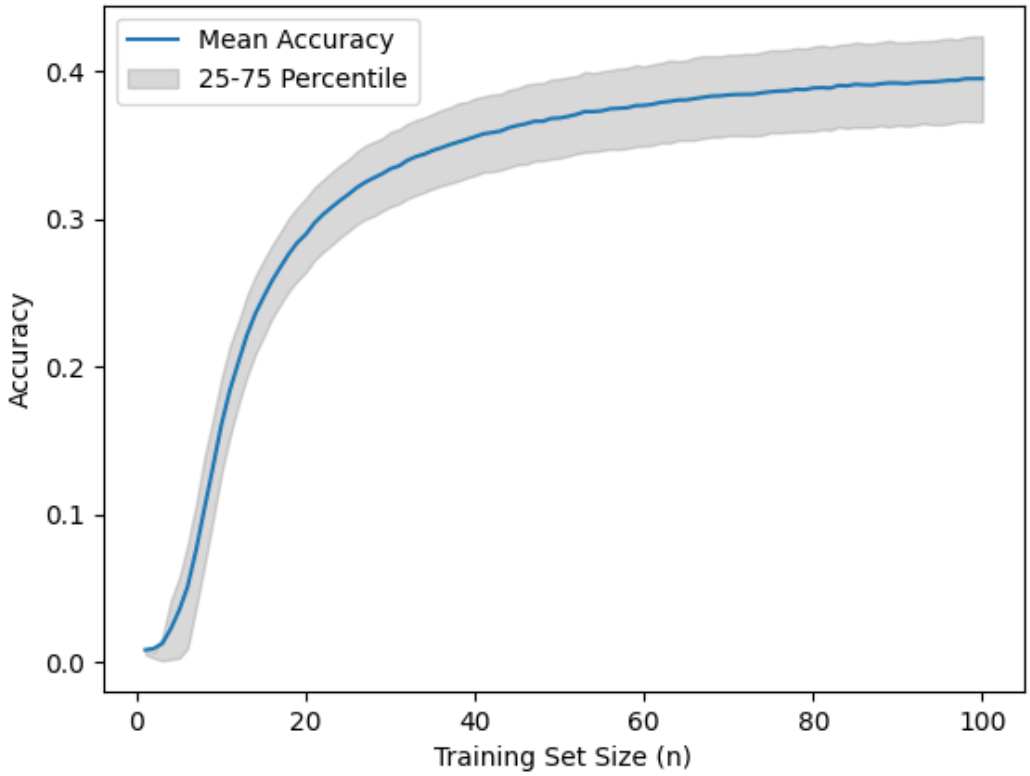

Figure 5: Development of Balanced Accuracy Distribution of a Classifier on the synthetic dataset.

The Titanic

The tragic sinking of the Titanic in 1912, which claimed the lives of over 1,500 passengers and crew, remains one of the most infamous maritime disasters in history; through the lens of modern data analysis, this historical event provides a unique opportunity to classify and predict survival outcomes based on passenger demographics and circumstances.

Is it just random chance who survived and who didn't? Or is it dependent on some of the features recorded in this dataset? If the classifier is capable of predicting the outcomes of the passengers significantly better than chance there must be a difference between the group that the classifier detects.

For this it is interesting if the classifier would be capable of finding a difference when trained on infinite amounts of data. Therefore we observe the asymptotical accuracy.

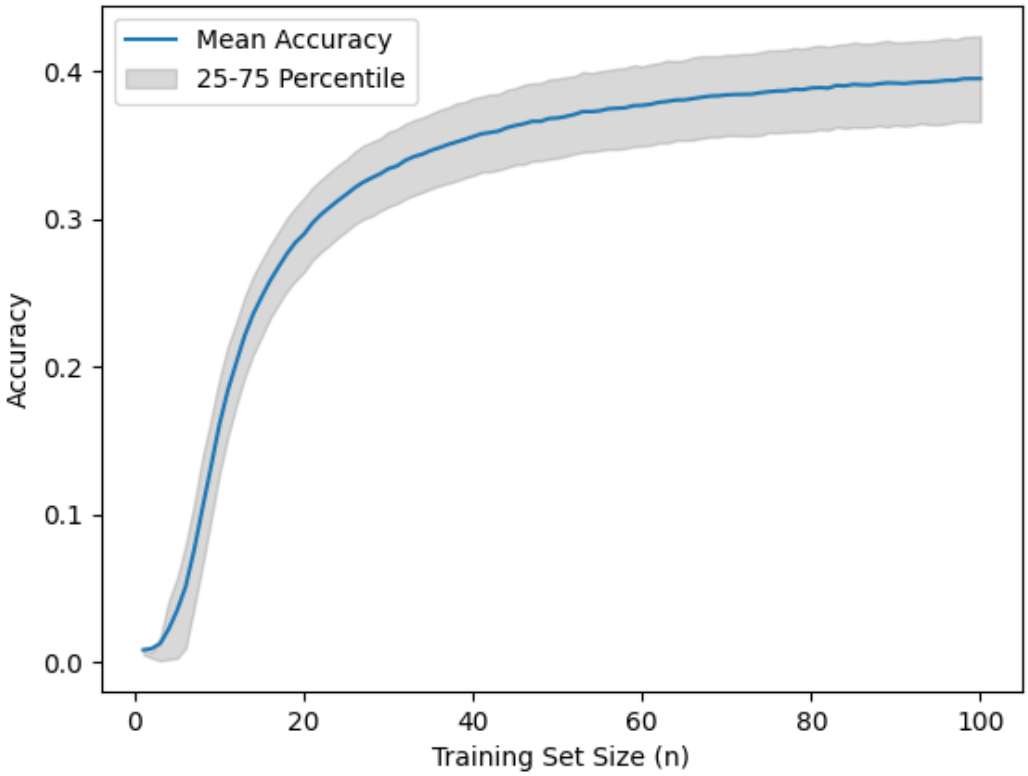

Figure 6: Asymnptotical Accuracy Distribution of a Classifier on the Titanic Dataset. TODO: Does not exist yet and needs to be generated.

Alternatively if we want to use it for inference to predict if somebody on board the titanic is going to drown and we want to know how accurate our own predictions are, we can observe the accuracy for a training set of the size of our actual training set.

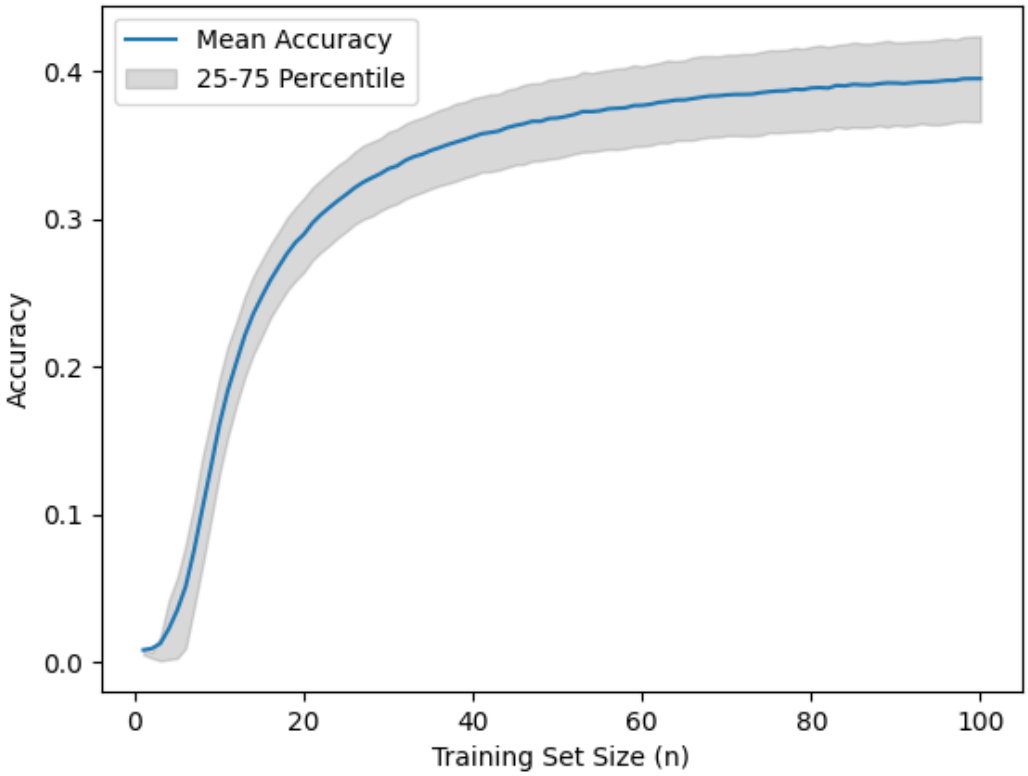

Figure 7: Accuracy Distribution of a Classifier trained on ??? Samples from the Titanic Dataset. TODO: Does not exist yet and needs to be generated.

Or lets assume we have different classification algorithms and want to find out which is best for our scenario. In that case it would be interesting to look at the development of the accuracy over sample size because some classifiers perform better for small sample sizes but scale less than other classifiers which then outperform the first ones as soon as a certain threshold is reached.

TODO: Can this work in accordance with the assumed formular? Or does the formular imply that all classifiers scale similarly? I fear its the latter.

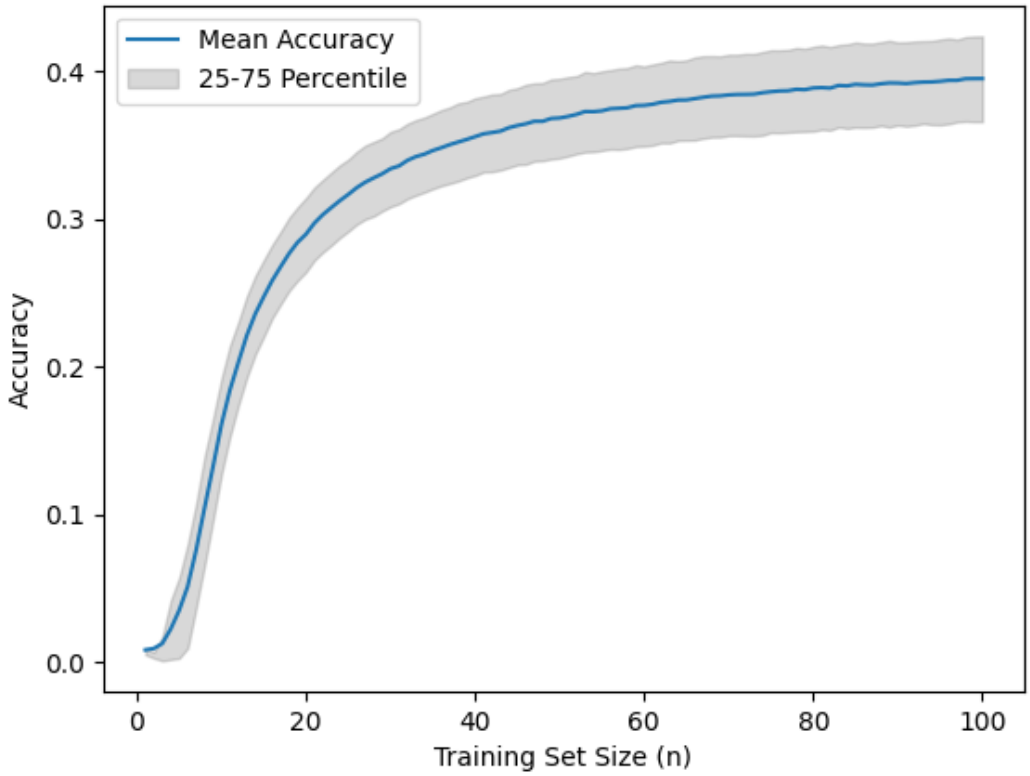

Figure 8: Accuracy Development of a Classifier on the Titanic Dataset. TODO: Does not exist yet and needs to be generated.

Random Guesses

If all groups have the same distribution for the features, the results of IV look like this:

Discussion

The last paragraph

Independent validation is the method of choice to optimize for statistically accurate results. With this Python implementation it can now easily be used in combination with classifiers from the sklearn library.

References

Bay, Stephen D., and Michael J. Pazzani. 2001. "Detecting Group Differences: Mining Contrast Sets." *Data Mining and Knowledge Discovery* 5 (3): 213–46. https://doi.org/10.1023/A:1011429418057.

Boucheron, Stéphane, Olivier Bousquet, and Gábor Lugosi. 2005. "Theory of Classification: A Survey of Some Recent Advances." *ESAIM: Probability and Statistics* 9 (November): 323–75. https://doi.org/10.1051/ps:2005018.

Bouckaert, Remco R. 2003. "Choosing Between Two Learning Algorithms Based on Calibrated Tests." In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, 51–58. ICML'03. Washington, DC, USA: AAAI Press.

Braun, Tina, Hannes Eckert, and Timo von Oertzen. 2023. "Independent Validation as a Validation Method for Classification." *Quantitative and Computational Methods in Behavioral Sciences*, December, 1–30. https://doi.org/10.5964/qcmb.12069.

Breiman, Leo, Jerome Friedman, R. A. Olshen, and Charles J. Stone. 2017. *Classification and Regression Trees*. New York: Chapman; Hall/CRC. https://doi.org/10.1201/9781315139470.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-Vector Networks." *Machine Learning* 20 (3): 273–97. https://doi.org/10.1007/BF00994018.

Counsell, Alyssa, and Lisa. L. Harlow. 2017. "Reporting Practices and Use of Quantitative Methods in Canadian Journal Articles in Psychology." *Canadian Psychology / Psychologie Canadienne* 58 (2): 140–47. https://doi.org/10.1037/cap0000074.

Cover, T., and P. Hart. 1967. "Nearest Neighbor Pattern Classification." *IEEE Transactions on Information Theory* 13 (1): 21–27. https://doi.org/10.1109/TIT.1967.1053964.

Devroye, L., and T. Wagner. 1979. "Distribution-Free Performance Bounds for Potential Function Rules." *IEEE Transactions on Information Theory* 25 (5): 601–4. 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. 1915. "Frequency Distribution of the Values of the Correlation Coefficient in Samples from an Indefinitely Large Population." *Biometrika* 10 (4): 507. https://doi.org/10.2307/2331838.

Geisser, Seymour. 1975. "The Predictive Sample Reuse Method with Applications." *Journal of the American Statistical Association* 70 (350): 320–28. https://doi.org/10.1080/01621459.1975.10479865.

Ho, Tin Kam. 1995. "Random Decision Forests." In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1:278–282 vol.1. https://doi.org/10.1109/ICDAR.1995.598994.

Kim, Bommae, and Timo von Oertzen. 2017. "Independent Validation Remedies Inflation in Classifier Accuracy Testing." Technical Report Report number. Timo von Oertzen, Department of Psychology, University of Virginia, 1023 Millmont Street, VA22903 Charlottesville.:

Department of Psychology, University of Virginia, Charlottesville.

Kim, Bommae, and Timo von Oertzen. 2018. "Classifiers as a Model-Free Group Comparison Test." *Behavior Research Methods* 50 (1): 416–26. https://doi.org/10.3758/s13428-017-0880-z.

Kohavi, Ron. 1995. "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection." In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, 1137–43. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Mooney, Paul. 2022. "2022 Kaggle Machine Learning & Data Science Survey." https://kaggle.com/competitions/kaggle-survey-2022.

Pesarin, Fortunato, and Luigi Salmaso. 2010. "The Permutation Testing Approach: A Review." *Statistica* 70 (4): 481–509. https://doi.org/10.6092/issn.1973-2201/3599.

R Development Core Team. 2010. *R a Language and Environment for Statistical Computing: Reference Index*. [Vienna]: R Foundation for Statistical Computing.

Rossi, Joseph S. 2013. "Statistical Power Analysis." In *Handbook of Psychology: Research Methods in Psychology, Vol. 2, 2nd Ed*, 71–108. Hoboken, NJ, US: John Wiley & Sons, Inc.

Sahiner, Berkman, Heang-Ping Chan, and Lubomir Hadjiiski. 2008. "Classifier Performance Estimation Under the Constraint of a Finite Sample Size: Resampling Schemes Applied to Neural Network Classifiers." *Neural Networks*, Advances in Neural Networks Research: IJCNN '07, 21 (2): 476–83. https://doi.org/10.1016/j.neunet.2007.12.012.

Salzberg, Steven L. 1997. "On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach." *Data Mining and Knowledge Discovery* 1 (3): 317–28. https://doi.org/10.1023/A:1009752403260.

Santafe, Guzman, Iñaki Inza, and Jose A. Lozano. 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): 111–33. https://doi.org/10.1111/j.2517-6161.1974.tb00994.x.

Wee, Alvin G. 2000. "Comparison of Impression Materials for Direct Multi-Implant Impressions." *The Journal of Prosthetic Dentistry* 83 (3): 323–31. https://doi.org/10.1016/S0022-3913(00)70136-3.

Weisberg, Yanna J., Colin G. DeYoung, and Jacob B. Hirsh. 2011. "Gender Differences in Personality Across the Ten Aspects of the Big Five." *Frontiers in Psychology* 2 (August). https://doi.org/10.3389/fpsyg.2011.00178.

Zhao, Mengmeng, Menglong Wang, Jishou Zhang, Jian Gu, Pingan Zhang, Yao Xu, Jing Ye, et al. 2020. "Comparison of Clinical Characteristics and Outcomes of Patients with Coronavirus Disease 2019 at Different Ages." *Aging (Albany NY)* 12 (11): 10070–86. https://doi.org/10.18632/aging.103298.