# Predicting hyperparameters from meta-features in binary classification problems

**Eleni Nisioti** ELENNISIOTI@GMAIL.COM
**Kyriakos C. Chatzidimitriou** KYRCHA@ISSEL.EE.AUTH.GR
**Andreas L. Symeonidis** ASYMEON@ENG.AUTH.GR
*Electrical and Computer Engineering Department,*
*Aristotle University of Thessaloniki,*
*Thessaloniki, Greece*

## Abstract

The presence of computationally demanding problems and the current inability to automatically transfer experience from the application of past experiments to new ones delays the evolution of knowledge itself. In this paper we present the Automated Data Scientist [1], a system that employs meta-learning for hyperparameter selection and builds a rich ensemble of models through forward model selection in order to automate binary classification tasks. Preliminary evaluation shows that the system is capable of coping with classification problems of medium complexity.

**Keywords:** meta-features, hyperparameter selection, automl, binary classification

## 1. Introduction

Recent concerns (Caruana, 2015; Feurer et al., 2015; Rostislav, 2009) about the dependence of applied machine learning (ML) on human experts and the lack of transferability (leveraging knowledge from past experiments) in the current approach, have set the ML community in a quest of automation. The latter refers to off-the-self methods that could automate (previously) manually-performed tasks when experimenting. *AutoML* has emerged out of this necessity, acknowledging the importance of the *meta-learning* concept (Rostislav, 2009) that incorporates experience into the application phase of ML tasks.

According to the *No Free Lunch Theorem* (Wolpert, 1996), all ML algorithms exhibit the same mean performance when modelling all possible data generation distributions. Nevertheless, the datasets ML deals with, are not randomly generated, but usually exhibit some common characteristics. In this context, we argue on the usefulness of an AutoML tool that can explore, train and gain experience from a variety of classification tasks.

While other approaches use meta-learning for warm starting optimization algorithms (Feurer et al., 2014) and ranking predetermined hyperparameter sets (Soares et al., 2004), we propose the use of meta-learning to *directly* predict the optimal hyperparameters of various ML algorithms for training models in binary classification tasks. In order to circumvent any prediction weaknessess, we incorporate prediction intervals in our models. Following the work of Feurer et al. (2014), we applied our approach on a large library of models

1. https://github.com/issel-ml-squad/ads

through forward model selection ensembles (Caruana et al., 2004). Our research led to the development of an end-to-end, automated tool that employs meta-learning and makes opinionated choices, in order to close the gap between the input dataset and the end result.

## 2. Methodology

The architecture of the Automated Data Scientist (ADS) is depicted in Figure 1.

### 2.1. Overview

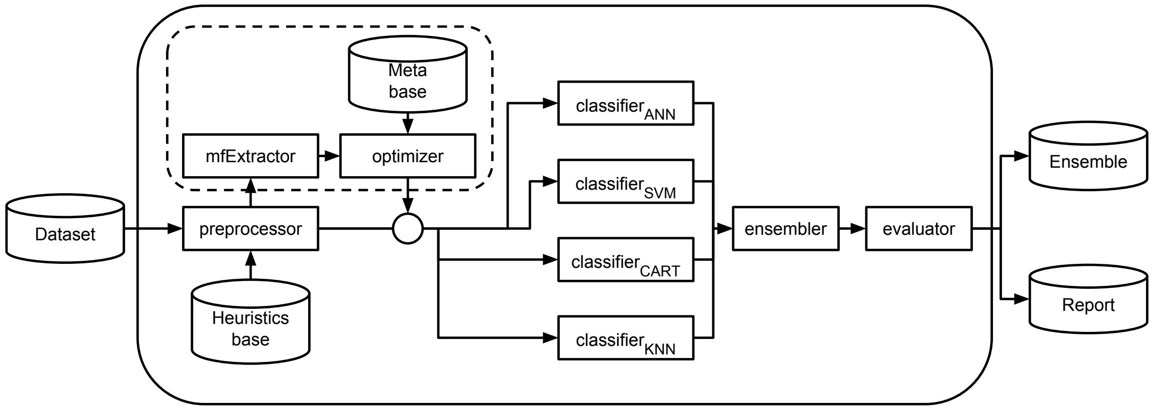


Figure 1: ADS architecture diagram

An experiment begins when the user inserts a classification dataset to our system. The *preprocessor* module is responsible for the tasks of data cleaning (inappropriate value removal, data type recognition, compression), data preprocessing (normalization, compression, feature engineering) and data splitting. Decision making at this stage is based on heuristic rules, as extracted from the relevant literature (Tukey, 1977) [2]. The *mfExtractor* module is assigned the task of extracting meta-features, while the *optimizer* is responsible for selecting the optimal hyperparameters for a variety of ML algorithms. Hyperparameter selection is performed using prediction models, henceforth called HPP (HyperParameter Prediction) models. HPPs are trained on data consisting of meta-features and optimal hyperparameters, which are produced by employing Bayesian optimization on a repository of binary classification datasets (as described in Section 2.2). In order to benefit from the generation of a large and versatile library of models, forward model selection ensembles (Caruana et al., 2004) are employed. An overall description of our model creation process is provided in Section 2.3. Finally, module *evaluator* provides the functionalities of evaluating a model through metrics and performance visualization plots, as well as comparing different techniques through statistical testing and performance profile plots. The output of our system comprises a reusable ensemble of trained models and a human-readable report of the performed experiment.

2. Also courses like `http://work.caltech.edu/telecourse.html` and `https://www.coursera.org/learn/machine-learning` offer useful heuristic rules

### 2.2. Hyperparameters prediction system

Employing meta-learning for optimal hyperparameter prediction constitutes a novel approach to model tuning. Traditional methods, such as searching in predetermined spaces, entail the drawbacks of slow execution and ad-hoc implementation. Furthermore, a research on the a priori generally best optimization algorithm would contradict the intuition of the No Free Lunch Theorem. In order to find the class for each HPP model, namely the optimal hyperparameter value, tuning has to be performed for each training dataset in a repository of gathered datasets. Our method of choice was Bayesian optimization, and in particular the Tree Parzen Estimator (TPE). According to Bergstra and Bengio (2012) TPE outperforms traditional methods. An important characteristic of our HPP models is the embedding of prediction intervals. This trait was added to mitigate the already observed weakness of such models to accurately predict the optimal hyperparameters (Feurer et al., 2014). Hyperparameter optimization of all datasets was performed using HPOlib (Eggensperger et al., 2013). Extensive research was conducted in order to select the appropriate meta-features that encompass all valuable information residing in a dataset; these meta-features are displayed in Table 1. Linearly correlated meta-features were removed and the remaining, uncorrelated meta-features are provided in Table 2 of the on-line Appendix [3]. In order to generate the prediction intervals for the HPP training process, bootstrapping was employed, (Stine, 1985), since it ensures no bias in favor of any specific ML algorithm.

Table 1: List of meta-features used to train HPP models

| **Simple** | **Statistical numeric** | **Information-theoretic** |
|---|---|---|
| Number of features | Summation | Class entropy |
| Logarithm of number of features | Mean | |
| Number of instances | Standard deviation | **meta2-** |
| Logarithm of number of instances | Minimum | Summation |
| Number of features with unknown values | Maximum | Mean |
| Percentage of number of features with unknown values | Kurtosis | Standard deviation |
| Number of instances with unknown values | Skewness | Minimum |
| Percentage of number of instances with unknown values | Percentage of PCs for 95% pertained variance | Maximum |
| Number of unknown values | Kurtosis of first PC | Kurtosis |
| Logarithm of number of unknown values | Skewness of first PC | Skewness |
| Number of numeric features | | |
| Number of categorical features | **Statistical categorical** | |
| Class probabilities | Number of levels | |
| Minimum class probability | | |
| Maximum class probability | | |
| Mean class probability | | |
| Standard deviation of class probabilities | | |

**Outliers dataset detection** The ability of our system to optimize when a new dataset is presented depends on its experience, which manifests itself in the form of the repository of gathered datasets, used for training the HPP models. In order to provide the user with a rating for the readiness of our tool to optimize a given problem, we have built a metric that quantifies it.

To do so, we attempt to detect outliers in the instances of the new dataset against the datasets of a repository by exploiting the meta-feature extraction mechanism. We adopt the distance-based approach, which has been proven suitable for multidimensional data

3. The on-line Appendix can be found in https://github.com/issel-ml-squad/ads/blob/master/ads-with-appendix.pdf

and does not make assumptions about data distributions (Knorr et al., 2000). The on-line Appendix offers more information upon the procedure.

### 2.3. Model creation system

The process of building an ensemble using forward model selection follows the approach discussed in Caruana et al. (2004), where an ensemble formation technique is proposed, that gradually picks the best-performing model to add from a heterogeneous library of trained models. The attractiveness of this technique lies in its efficiency of adding a new model by averaging the outputs, its ability to avoid overfitting and the importance it continusouly lays on the performance of the final ensemble.

The following parameters allow the tuning of the building process: total number of models included, initial sample size, number of bootstrap samples, probability of inclusion in each bootstrap sample. Note that in our experiments a constant configuration was preferred due to the induced computational complexity.

## 3. Results and Discussion

We have evaluated separately the hyperparameters prediction system and the overall performance of ADS. Caruana et al. (2004) offer an evaluation framework for forward model selection ensembles. We collected 125 datasets (on-line Appendix, Section D) and used 80% to train the HPP models of ADS. The remaining datasets were used for evaluating ADS for all compared techniques using 10-fold cross-validation for each dataset. The results are presented in Section 3.2.

### 3.1. Hyperparameters prediction system results

Regression models were trained using the Root Mean Squared Error (RMSE) as a performance metric and prediction intervals using bootstrapping were computed. We are interested in both predictive accuracy of the regression models, as well as the improvement caused by the adoption of prediction intervals, measured in Coverage, defined in the on-line Appendix, Section C.

Figure 2 depicts the performance of all HPP models trained during our experiment. One may argue that, despite the fact that the HPP models fail to always accurately predict the optimal value, the embedding of prediction intervals helps to circumvent this problem. We assume that if the optimal value lies in the prediction interval, then the ensemble building technique will incorporate the corresponding model in the final ensemble. Note that testing datasets classified as outliers for a particular HPP model are omitted from Figure 2.

### 3.2. Model generation system results

Since hyperparameter optimization was the main focus of our work, we use grid search and Bayesian optimization (TPE) as benchmark methods. We experimented with individual models, as well as with forward model selection ensembles.

Comparisons were performed using performance profile plots (Dolan and Moré, 2002) and statistical testing, in particular Friedman rank sum test for the rejection of the equiva-

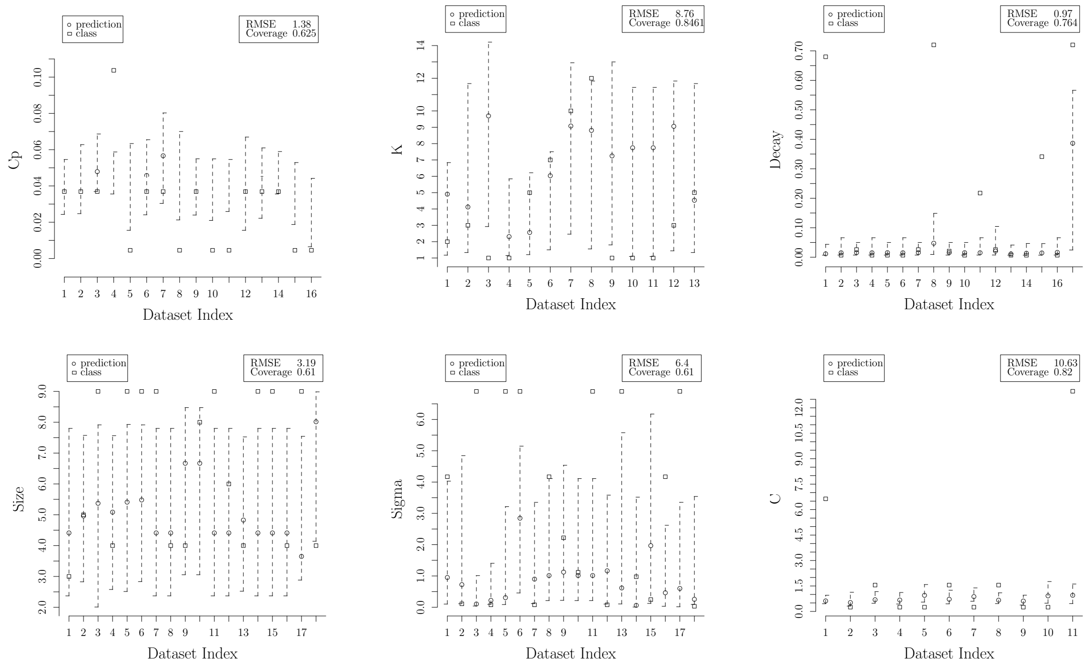


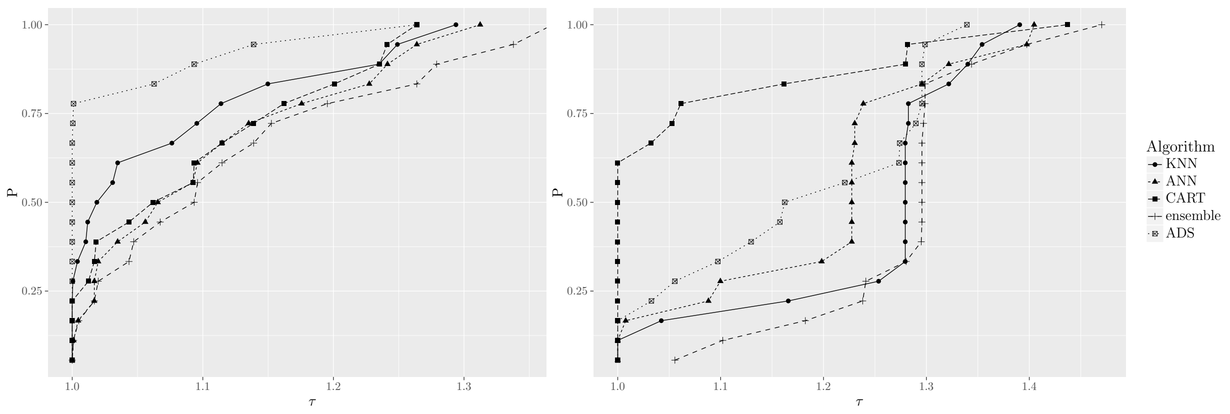
Figure 2: Evaluation of HPP models: for each dataset, the prediction, class, RMSE and confidence intervals are provided. From left to right: cp-HPP predicts the cp parameter of *rpart**, k-HPP predicts the parameter k of *knn**, Decay-HPP and size-HPP predict the decay and size of *nnet**, Sigma-HPP and C-HPP predict the sigma and C parameters of *svmRadial**.
* Names refer to R methods, as provided by package *caret*.

lence hypothesis and post-hoc Nemenyi test to detect pairwise differences, as suggested by Demšar (2006).

The left plot of Figure 3 suggests that ADS is by 77% the best-performing algorithm for all datasets, with CART (Classification and Regression Trees) models being the second method of choice. The right plot, places CART models at the first place with 67.5% confidence. In this scenario, ADS exhibits a constantly good behavior deviating a factor of $\tau = 1.33$ from the best algorithm for all datasets. Statistical testing, presented in Figure 3, outrules any important statistical difference between ADS and the single model benchmark methods at a 95% confidence level. We thus conclude, that ADS offers performance equivalence to both well-established and state-of-the-art hyperparameter optimization techniques, while bringing the additional benefits of generation of meta-knowledge and speed, as the time-consuming search was replaced by a simple prediction.

## 4. Related Work

Understanding that the well-established grid search is worse than random search (Bergstra and Bengio, 2012) motivated the search for tuning algorithms and acknowledged Bayesian optimization as a promising optimization approach. AutoWEKA addresses the CASH problem (Thornton et al., 2012) through Bayesian optimization, while Feurer et al. (2014) de-



| | automl | ensembleGrid | ensembleTpe | knnGrid | knnTpe | nnetGrid | nnetTpe | treeGrid |
|---|---|---|---|---|---|---|---|---|
| ensembleGrid | **0.0021** | — | — | — | — | — | — | — |
| ensembleTpe | **0.0021** | 1 | — | — | — | — | — | — |
| knnGrid | 0.9213 | 0.1593 | 0.1593 | — | — | — | — | — |
| knnTpe | 0.9213 | 0.1593 | 0.1593 | 1 | — | — | — | — |
| nnetGrid | 0.2337 | 0.8531 | 0.8531 | 0.9643 | 0.9643 | — | — | — |
| nnetTpe | 0.7854 | 0.3024 | 0.3024 | 1 | 1 | 0.9949 | — | — |
| treeGrid | 0.8321 | 0.2554 | 0.2554 | 1 | 1 | 0.9902 | 1 | — |
| treeTpe | 1 | **0.0067** | **0.0067** | 0.9828 | 0.9828 | 0.409 | 0.9213 | 0.9457 |

Figure 3: Evaluation of system's performance: (Left) performance profile plot for grid-search, (Right) performance profile plot for TPE, (Bottom) Statistical testing of performances. Pairwise-comparisons were performed using the post-hoc Nemenyi test and statistically significant differences are indicated with bold.

signed a system that embeds meta-learning in optimal hyperparameter selection. Other attempts to incorporate meta-learning to the tuning process include the works of Feurer et al. (2014); Kuba et al. (2002); Soares et al. (2004), who dealt with the weakness of their meta-learning models in the task of predicting the optimal hyperparameters in different ways: by using the predicted values for warm-starting Bayesian optimization, applying local search around predicted values, or exploiting the top-N performing values respectively. Rostislav (2009) enriched meta-learning by introducing the concept of meta2-features, laying the ground for more successful application of meta-learning in optimization tasks.

## 5. Conclusions and Future Work

Our contribution to the AutoML domain includes our experimental research on HPP models with embedded prediction intervals and the developed ADS tool. By exploiting the robust building mechanisms of forward model selection ensembles, we managed to tune our models using solely meta-learning and achieved performance equivalent to Bayesian optimization. ADS leverages the findings of our work and, additionally, places great importance into expandability, re-usability and intuitiveness of the experiment, traits that, to our point of view, are quintessential for an AutoML tool. We recognize the potential of improving our HPP-models by further experimenting with meta-features, as well as the parameterization of prediction intervals and ensemble model building.

## References

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, February 2012. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2188385.2188395.

Rich Caruana. Research opportunities in automl. *Automl Workshop, ICML 2015*, 2015.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 18–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015432. URL http://doi.acm.org/10.1145/1015330.1015432.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1248547.1248548.

Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. ISSN 1436-4646. doi: 10.1007/s101070100263. URL http://dx.doi.org/10.1007/s101070100263.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000. ISBN 0471056693.

K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, 2013.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Using meta-learning to initialize bayesian optimization of hyperparameters. In *Proceedings of the 2014 International Conference on Meta-learning and Algorithm Selection - Volume 1201*, MLAS'14, pages 3–10, Aachen, Germany, Germany, 2014. CEUR-WS.org. ISBN 1613-0073. URL http://dl.acm.org/citation.cfm?id=3015544.3015549.

Matthias Feurer, Jost Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *AAAI Conference on Artificial Intelligence*, 2015. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/10029.

Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-based outliers: Algorithms and applications. *The VLDB Journal*, 8(3-4):237–253, February 2000. ISSN 1066-8888. doi: 10.1007/s007780050006. URL http://dx.doi.org/10.1007/s007780050006.

Petr Kuba, Pavel Brazdil, Carlos Soares, and Adam Woznica. Exploiting sampling and meta-learning for parameter setting support vector machines. In *Proceedings of the IBERAMIA*, volume 2002, pages 217–225, 2002.

Striz Rostislav. *Extending Metalearning to Data Mining and KDD*, pages 61–72. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-73263-1. doi: 10.1007/978-3-540-73263-1_4. URL http://dx.doi.org/10.1007/978-3-540-73263-1_4.

Carlos Soares, Pavel B. Brazdil, and Petr Kuba. A meta-learning method to select the kernel width in support vector regression. *Machine Learning*, 54(3):195–209, 2004. ISSN 1573-0565. doi: 10.1023/B:MACH.0000015879.28004.9b. URL http://dx.doi.org/10.1023/B:MACH.0000015879.28004.9b.

Robert A. Stine. Bootstrap prediction intervals for regression. *Journal of the American Statistical Association*, 80(392):1026–1031, 1985. doi: 10.1080/01621459.1985.10478220. URL http://amstat.tandfonline.com/doi/abs/10.1080/01621459.1985.10478220.

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms. *CoRR*, abs/1208.3719, 2012. URL http://arxiv.org/abs/1208.3719.

John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Comput.*, 8(7):1341–1390, October 1996. ISSN 0899-7667. doi: 10.1162/neco.1996.8.7.1341. URL http://dx.doi.org/10.1162/neco.1996.8.7.1341.

## Appendix A.

Regarding HPP models, we trained one for each hyperparameter of each machine learning algorithm we included in the ensemble, namely: k-HPP for knn, cp-HPP for rpart, size-HPP and decay-HPP for nnet and finally, C-HPP and sigma-HPP for svmRadial. [4]

The meta-features presented in Figure 1form an extensive list of 77 meta-features that aim at capturing all information necessary for training the HPP models. In order to drop correlated information, we applied filtering on the meta-features, based on the linear correlation, which resulted in choosing the meta-features in Figure 2.

Table 2: List of meta-features after filtering

| | |
|---|---|
| Summation of summation | Standard deviation of number of levels |
| Summation of maximum values | Kurtosis of number of levels |
| Mean of standard deviation values | Skewness of number of levels |
| Mean of minimum values | Number of features |
| Mean of kurtosis values | Logarithm of number of features |
| Mean of skewness values | Number of instances |
| Standard deviation of minimum values | Logarithm of number of instancces |
| Minimum of mean values | Percentage of unknown values |
| Minimum of standard deviation values | Number of numeric features |
| Minimum of minimum values | Number of categorical features |
| Minimum of maximum values | Maximum of class probabilities |
| Minimum of skewness values | Mean of class probabilities |
| Kurtosis of minimum values | Percentage of PC for 95% pertained variance |
| Kurtosis of maximum values | Kurtosis of first PC |
| Skewness of skewnesss values | Skewness of first PC |
| Summation of number of levels | |

## Appendix B.

In our approach the mean distance from each dataset's 9 neighbors (number of neighbors was chosen using the heuristic rule of Duda et al. (2000)) is computed and a distribution is fit in order to determine a threshold for positive outliers. Algorithm 1 offers a more detailed version of the outlier detection mechanism.

## Appendix C.

Metric *Coverage* was engineered in order to evaluate the performance of the HPP models taking prediction intervals into consideration:

$$Coverage_{HPP} = \frac{\sum_{i=1}^{i=N}(Class_i \in interval_i)}{N} \tag{1}$$

where $N$ is the number of testing instances and $Class_i$ and $interval_i$ are the class and prediction interval for a particular example.

This metric thus captures the percentage of examples for which the class lies inside the prediction intervals.

4. These are models offered by package caret.

```
**Data:** meta-features of each HPP model
**Result:** bound for dataset-outlier detection
for $j \in metafeatures$ do
    calculate $kOpt_j$
    for $i \in j$ do
        $sum_i \leftarrow sum_i + dist_{kOpt_j} i$
    end
end
$meanDistances \leftarrow \frac{sum}{nModels}$
fit distribution to meanDistances
apply Tukey test to get upper bound for outlier detection
```

**Algorithm 1:** Detection of outliers in datasets. *metafeatures* is a list of the metafeatures dataset for each HPP model and *kOpt* is the optimal number of neighbors. Since each HPP model requires its own subset of the meta-features, the distances of each dataset were computed for each subset and then averaged. The distribution is then fit using a gaussian kernel and the Tukey test is used to compute the upper bound, using the third-quartile and inter-quartile range of the fitted distribution.

## Appendix D.

Citations for all the datasets used for training and evaluating our tool are provided here: `https://github.com/issel-ml-squad/ads/blob/master/DATASETS.md`