Algorithm primitives

Master Thesis

L.D. Stooker, 0819041

Supervisor: Joaquin Vanschoren
May 4, 2018

Abstract

To improve machine learning selection based on robustness and scalability in the scikit-learn library. By researching the python library which is gaining popularity we give a few insights into the qualitive properties of these machine learning algoritms. For accuracy GradientBoostingClassifier is a solid pick which outperforms with default settings on nearly all cases. A runner up on accuracy is RandomForestClassifier an easier and quicker solution. For noisy data KNeighborsClassifier is most robust and Naive Bayes algorithms are least robust. For the more unpredictable cases of noisy data in a categorical setting Gaussian Naive Bayes is however a more robust solution.

Contents

1 Introduction

This report is the result of my graduation project which completes my Business Information Systems study at Eindhoven University of Technology. The project was performed internally at the University in the Data mining department. In this project we investigated annotations of primitives, more specifically primitives in the scikit-learn library. In the section 1.1 we will briefly explain more about primitives and the annotations. To elaborate on this we will outline the research questions and thesis structure

1.1 problem description

Machine learning is a growing field that can help process the increase of available data[4][3]. Python is a language which holds premade machine learning algorithms in libraries like scikit-learn[**?**]. In recent years python is also increasing in so called market share for machine learning[2]. To help people choose machine learners in the scikit-learn library a model was made to indicate what algorithm to use for what problem. In figure **??** you can see that depending on size of the data and early results, different algorithms are recommend. This gives a quick overview of available algorithms and when to pick which. This figure however is outdated as new algorithms are added to the library over time. Such a model is based on the concept of no free lunch which explains that there cannot be one algorithm to solve all optimization problems[**?**].

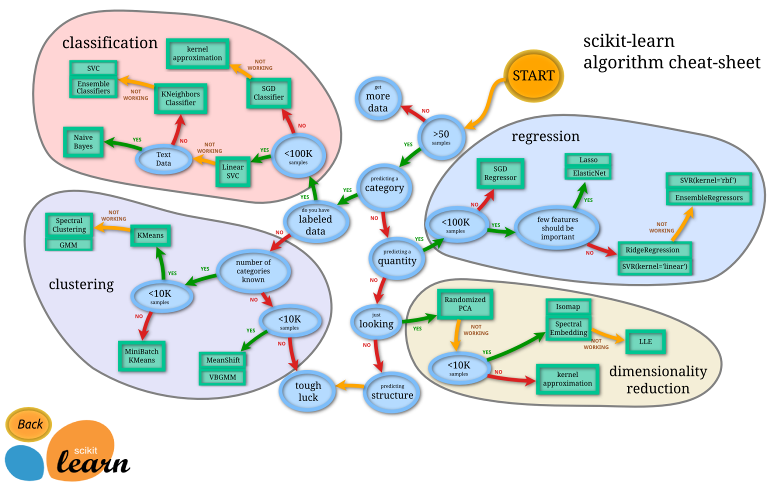


Figure 1: FlowChartML scikit-learn for any dataset which machine learning algortihm to choose

1.2 research question

We base our research question on the work of Joaquin to give properties to algorithms[5]. More specifically we look more closely to the resilience properties and the bias-variance profile. Earlier research has been done on scalability and resilience to irrelevant variables[6]

- scalability: runtime is measured with either a varying sample size or varying feature space
- robustness: robustness against irrelevant or redudant features
- robustness: robustness against noisy features
- What is the impact on runtime of the amount of features for classification machine learning algorithms?
- What is the impact on runtime of the amount of instances for classification machine learning algorithms?
- What is the impact on accuracy of irrelevant or redundant features for classification machine learning algorithms?

- What is the impact on accuracy of noisy features for classification machine learning algorithms?
- What is the percentage of Bias-error or variance-error for classification machine learning algorithms?

The question for scalability is how varying sample size and feature space influences the runtime. Is there a measure to match this.
We rank the algorithms on their performance(predictive accuracy) on all datasets to find a ranking like done by Quan Sun and Bernhard Phahringer[23].

1.3 Outline

In the first chapter the research question is introduced. In the second chapter background information is discussed relevant for the research. In the third chapter the setup of the experiments is outlined with the input and output of executed experiments. The fourth chapter shows the results of the experiments

2 Preliminaries

Before we discuss in detail the solutions for the steps of our approach, this chapter provides some background knowledge and definitions which are required for a good understanding of the remainder of this thesis.

2.1 Sklearn/scikit-learn library

The scikit-learn library is based in Python and is made to make machine learning in python accessible and organized. All resources are open source and hosted on Github. Before scikit-learn there were already other libaries hosting machine learning algorithsm in python but scikit-learn was the first to make a standard guideline which makes the algorithm easier to interchange. By having default functions for fitting and training

2.1.1 KNeighborsClassifier

In the scikit-learn library KNeighborsClassifier is an implementation of the k-nearest neighbors algorithm(k-NN). K-NN uses instance-based learning or non-generalizing learning. This means that during fitting no complete model is made but only the given feature set is stored in order of appearing in a tree for example. The k-NN uses as it names tells the nearest neighbors for calculation. The inputted feature set is traversed to find the nearest points and depending on the parameter k, an amount of k points is searched for. Depending on the class of the k neighbors the classification is vote is decided. The default metric for distance measuring is Euclidean distance another option is the Manhattan distance which is less accurate but needs less computing. To find the nearest points an option can be made between a ball tree, a kd-tree or a brute search. This can heavily influence the search time, depending on the amount and size of the input (features and instances) this can influence the prediction time heavily but will not influence predictive accuracy. The previously mentioned k parameter is an influencer for prediction quality—find source—

2.1.2 GaussianNB

GaussianNB is a naïve Bayes classifier implementation assuming the feature set is gaussian distributed [11]. For fitting the data, a partial fit function is used based on the work of Chan, Golub and LeVeque [7]. This calculates the assumed means and variances of a gaussian distribution of the inputted feature set. Based on this distribution the prediction is made by filling in the maximum likelihood. The limited calculation needed for classification and prediction makes this one of the fastest algorithms. The only parameter of this classifier specifies the prior probabilities of the classes, which will when specified not be adjusted to the given input.

2.1.3 BernoulliNB

BernoulliNB is a naïve Bayes classifier implementation assuming a Bernoulli distribution with Boolean like values [12]. The first step of this implementation is checking if the features are binary-valued, if any other data is found this input will be binarized. This setting can be disabled or reduced by a threshold for the input. Based on this Boolean model a smoothed version of the maximum likelihood is used for prediction. This classifier is mostly used in document classification as it can binary store occurrence useful for prediction class probability.

2.1.4 SVC-rbf

SVC-rbf is a support vector classifier(SVC) implementation with a radial basis function. The radial basis function(rbf) is used to handle a large feature dimension, since the standard support vector machine splits the spaces with linearly lines computation grows too large for a large feature space [10]. The fit time is already quadratic with the number of samples based on the implementation of libsvm[8]. The fitting of a SVC will assign each example to one of two categories and will represent them in a dimension space mapped so there is a clear separation between the two categories. With the radial basis function this is with the distant from the points indicated by a separation area. Classifying a point is finding in which class area this point falls. For a multiclass problem this is done in pairs of two for all categories and then the most voted class is picked [9].

2.1.5 RandomForestClassifier

RandomForestClassifier is an ensemble method of directive trees [14]. During fitting a random forest classifier constructs directive trees on subsamples of the input data and averages the results for the result. These sub-samples are chosen randomly so the results can vary between runs on the same input. A directive tree is a decision tree classifier which splits the features on certain thresholds to decide on the type of class. This splitting of the data is either randomly or choosing the best split, to measure this split a criterion is used like Gini or entropy. The amount of splits, features and samples are also considered and can be inputted.

2.1.6 AdaBoost

AdaBoost is an ensemble classifier that fits other classifiers and outputs the weighted results of those classifiers [15]. AdaBoost trains these other classifiers on previously misclassified results by increasing their influence this makes it heavily subjected to noisy data and outliers. The scikit-learn library uses the multi class AdaBoost-SAMME implementation from J. Zhu et al [16]. The solution of J. Zhu also solves the lack of multi-class solution of the weak learners (other classifiers) by extending the initial AdaBoost algorithm with a forward stage wise additive step. In this step a continual calculation of a loss function will output the prediction and in a two class case it reduces to the initial solution.

2.1.7 SDGClassifier

SDGClassifier is an incremental function to stochastic approximate the gradient descent of the input [17]. It will iteratively minimize or maximize a set of differentiable functions, the input must fit these differentiable functions and this makes than an optimal input is with a mean of zero. This makes the classifier sensitive to raw data as it performs optimally with sparse features. The iterative steps needed are bound by the inverse of the learning rate and a threshold value. The threshold value indicates what degree of slope indicates a near minimal or maxima. The learning rate is used to update the model in each iteration.

2.1.8 GradientBoostingClassifier

GradientBoost is an ensemble classifier that builds from weaker classifiers [18]. Like AdaboostClassifier it builds an additive model in a forward stage-wise fashion. The weak classifiers used in the scikit-learn implementation are decision trees.

2.2 Definitions and abbreviations

2.2.1 Definitions

algorithm	The process with a specified in and output that solves a problem
amount	a value that indicates the change of the feature set either in dimension or in value. For
annotations	Adjectives of something like a machine learning algorithms, examples can be robust or biased
categorical	property of the features distribution and content of the feature in this case meaning seperate distinc classes, which have no numerical meaning, a unique number for each distinguished class
class	categorical featueres consist of at least 2 classes
datapoint	a datapoint is a single value of a feature. For example an instance has for all feature of a dataset a single datapoint.
dataset	a dataset are values in a matrix format, where each row represents a single Instance and each collum represent all the values of a feature.
dataset manipulation	like the word suggested is the manipulation of dataset like injecting std deviation or inserting random categories. Adding or removing; of features or instances to the dataset also counts as it also influence duration and/or accuracy.
dense matrix	most values in a matrix are different and fluctuate with each row or collumn.(non-zero)
distribution	A distribution of a dataset is the probability distribution of that dataset. So considering the dataset what are the odss of picking a specific value.
features	part of a whole, Consider a flower it can be the color, size, amount of branches, amount of leaves, age. Features describe someone or something in this context it describes something so you can make a prediction of the Target.
fit	To fit the data, synonym with inputting the training data, preparing the algorithm for prediction
Github	an online platfrom to host data. It uses git commands and is mostly used with programming project to organize a common project which each member can locally alter and centeralizly share updates or modifications.
machine learning algorithm	An algorithm that will will learn something and may adopt to the input to better fit the learned instance. To goal of learning can be mostly to predict a Target value, this can be part of an initial input.
numerical	exact values, more uniquely than a category. For example a temperature value or time value. Such a value can be subtracted or divided
profiling	To sketch information of something in a category, so you can relate it to other things in the same category
robustness The ability of an algorithm to cope with changes in features. An algorithm is more robust if it deteriorates less than another.[1ex] sample	the size of the to be predicted target set. So all distinct features once matched with a target feature
scalability	is the capability of a system to handle growing amount of work.
sparse matrix	a matrix with lots of zero values, the counterpart of dense were all values fluctuate a lot.
target	In a classificication the value or classes that needs to be predicted, where it mostly about a single feature with at least 2 classes. This can be a feature of something
weight	The influence or power of a value, function or object. It can be expressed as a fraction of 1 to inidicate its factor from other weights.

2.2.2 Abbreviations

adaBoost	adaptive boosting algorithm
biasVar	bias and variance
did	dataset identifier
SDG	Stochastic gradient descent
std	standard deviation
TU/e	Eindhoven University of Technology

3 Experimental setup

For different parts of the research different datasets are chosen. There is overlap between these datasets but the chosen datasets can have a large impact on shown results. Most results are shown as an average result of the datasets involved.

3.1 Cross validation

To easily test a single dataset once we use cross validation as it does this k times to get k splits of equal size. We can observe k classifications, to gain insight in duration time. We pick k=10

3.2 Bootstrapping

For the bias variance calculation it is import to gain multiple predictions for the same instance and with bootstrapping this can be easily achieved. The downside however can be an increased bias and reduced variance.

3.3 Datasets

Depending on the property we are going to study we need different datasets or can make assumption. Classification features are not an optimal input for a nearest neighbor classifier as the distant between converted numbers does not tell much about the relation between the features. Multiple target classes datasets are inconvenient for the bias-variance calculation, so we focus on 2 class target datasets, which there are plenty enough on openml.

3.3.1 Bias-variance datasets

An important part of a dataset to be viable for bias variance analysis is it having 2 classes as target due to calculation we use to find the bias and variance error. The bias and variance error is in this way like recall or precision error.

3.3.2 Categorical datasets

Categorical datasets hold only features with categorical values

3.3.3 Numerical datasets

Numerical datasets hold only features with numerical values

3.4 Collected data

For each experiment data is saved to give insight to what the results are. Depending on the experiment different data is important or stored.

- predictive accuracy for measuring the predictive accuracy we store the default sklearn scoring calculation
- duration instances for the time needed to calculate
- control data like predictions and real target values. Summarizing data of the predictions to make a faster observation.

3.4.1 Duration

For each classification instance and for each prediction the time is added to indicate how the classification took.

3.4.2 Predictions

For each predictions the outputted target value. Multiple files indicate multiple predictions. One of the files is also the true prediction of the inputted test set.

3.4.3 scores

Gives the predictive accuracy of all the made classifiers. There can be multiple lists for each configuration of classifier or test input. The score is a value between 0 and 1 indicating the fraction of rightly predicted values

3.4.4 SummaryGuesses

SummaryGuesses give a quick overview of the obtained results. It stores in python dictonaries the total amount of predictions for each class. The results is that you can easily observe if a classifier has picked a class exclusively and you can compare the balance to the inputted dataset to see if the classifier does find a difference between classes. This data can also be generated from the predictions 3.4.2.

3.4.5 BiasVar

When bootstrapping is done a bias and variance error is calculated together with the total error value. These are stored for easy lookup to the bias and variance error part of a classifier.

3.4.6 Identifier

The data input is shuffled as the saved datasets are sorted by class. The identifier can be used to match prediction results to a specific instance in the dataset. This way of saving is used to reduce space needed to save potential useful information. Odd behavior on small datasets can be explained by an off balanced dataset for training. The split of the data can be realistic but may affect averaged result significantly.

3.4.7 RemovedFeatures

In the case we use metafeatures like feature importance or correlation to remove features we save the removed features per fold of the cross validation. Comparing the removed features of each fold we can find if there are multiple irrelevant features or the features are likely to be randomly more important than another.

3.5 Experiments

Experiments are grouped by all mentioned classifier with some initial settings on the dataset and/or classifiers. Experiments are defined as functions in python with input values indicating the way the experiment is done and on which dataset.

3.5.1 Scalability

Scalability experiments can be split up in instance based or features based. To measure the effect of features we take datasets with lots of features and remove features in steps to find the impact of these lost features. There is a disadvantage with this strategy as some classifiers calculate values like feature importance which depend on a somewhat complete dataset of features. The removed features are randomly chosen and can be defining features for the accuracy of the dataset.
To combat this we also so feature removing based on feature importance of a RandomForestClassifier and on correlation between features. For feature importance we train a RandomForestClassifier to find the most important features. We then remove the least important features and remember the features we removed to also remove them in the test set. For correlation we find the feature the 2 most correlated features and then remove the feature that is most correlated to all features of the 2. Based on the amount we are going to remove we repeat the process to remove more.

Another option to measure scalability is to measure the impact of number of instances. For most classifiers each instance is considered during training and we measure an average calculation for each feature. The duration is measured over the whole dataset by doing a 10 fold cross validation.

3.5.2 Duplicate features

Duplicate features experiments have multiple goals in mind. By adding existing feature we can measure scalability of datasets with some amount of features. These duplicate features can also be identified as adding little to the dataset or the same features can overrule existing important features. The accuracy on these modified dataset can teach us about the impact of features on accuracy and how classifiers handle these irrelevant features. The method to add these duplicate features is by randomly picking and adding. This can result in features being multiple times in the manipulated dataset even though it is only twice the orignal dataset size. The accuracy is measured over the whole dataset by doing a 10 fold cross validation.

3.5.3 Random Features

Random features experiments have similar goals in mind as duplicate features. By adding the random features we can measure scalability of datasets with randomish features. These random features can deceivingly have information as there is much variability. By measuring the accuracy we can observe what the impact is for different classifiers. There are two sorts of random features we add; Numerical and categorical features. We add either the categorical or numerical depending on the distribution of the dataset. The odds of either a categorical or numerical feature being added is the distribution of the initial dataset. The accuracy is measured over the whole dataset by doing a 10 fold cross validation.

3.5.3.1 categorical random features

The categorical random feature is a uniform value between 0 and k. The value of k can influence how a classifier perceive this random feature. For all the instances in the set a uniform random number between 0 and 1 is multiplied by k and then rounded.

3.5.3.2 numerical random features

The numerical random feature is a uniform random value between 0 and 1. This feature has in this case all unique values.

3.5.4 Noisy data

Datasets can get more noisy over time. During training of a machine learning algorithm the dataset is clean and over time new data can change. By measuring the predictive accuracy off the algorithms on more noisy datasets the robustness can be measured. The accuracy is measured over the whole dataset by doing a 10 fold cross validation.

3.5.4.1 categorical features

To explain the implementation of our noisy data for a categorical feature we present this snippet of pseudo code. THe input is the dataset X and the amount of noise. The amount can be converted to a percentage of features being flipped by this formula $()1 - 1/(amount + 0.5)) * 100$. The distribution$_X$ is derived from the dataset X as the probability distribution of all categorical classes in a feature. The random.choice function uses this probability function to pick a value in the range of the feature. The default random function picks a uniform random value between 0 and 1.

Initiliaze distribution$_X$ for all features in dataset X
for numerical feature k in dataset X
.   **for** datapoint x in feature k
.   **if** random()*amount $>$ 0.5
.   x = random.choice(distribution$_X$ feature k)

3.5.4.2 numerical features

The input is the dataset X and the amount of noise. The amount is multiplied by the standard deviation to give the maximum deviation of the feature. The calculation for the std_X is done beforehand to control the deviation for all datapoints in the feature set. The random function is like mentioned before producing a uniform random value between 0 and 1.

Calculate std_X for all features in dataset X
for numerical feature k in dataset X
. **for** datapoint x in feature k
. **if** random() > 0.5
. x = x + random() *amount*(std_X for feature k)
. **else**
. x = x - random() *amount*(std_X for feature k)

3.5.5 Bias Variance

To measure the bias and variance error factor we use the calculation of Kohavi and Wolpert[21]. This is the same measure used in the work of Joaquin et al.[20]. This experiment is made to reproduce that experiment with the scicikit-learn library. The input for the bias and variance calculation is done by doing 40 bootstraps.

4 Experimental Results

4.1 Scalability

All results with a duration axis. The duration axis is in seconds and shows a 10 fold cross validiation. This means that the duration encompasses 10 times fitting over 90% of the data and 10 times predicting 10% of the data.

4.1.1 features

4.1.1.1 Categorical features

the averaged results are shown in figure 2

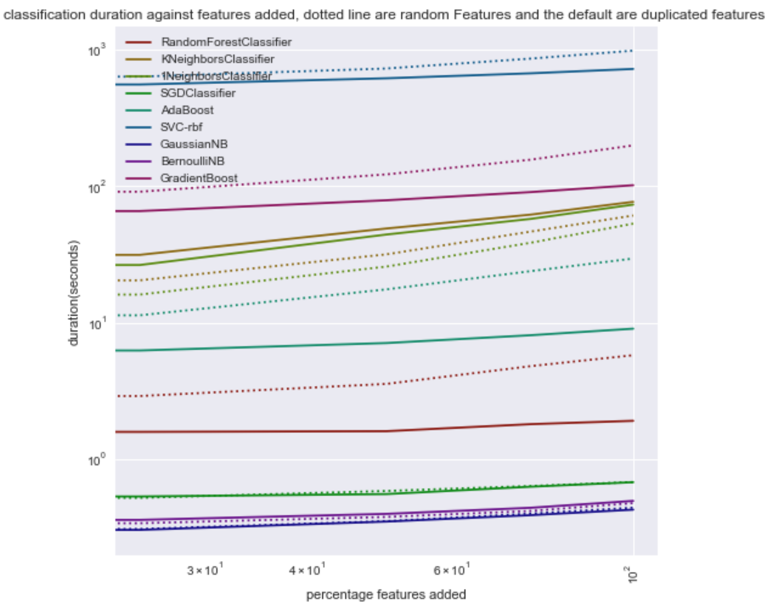


Figure 2: Features added straight line represent duplicate features and dotted line are random categorical features between 0 and 2

4.1.1.2 Numerical features in figure 3 and the slope in figure 4

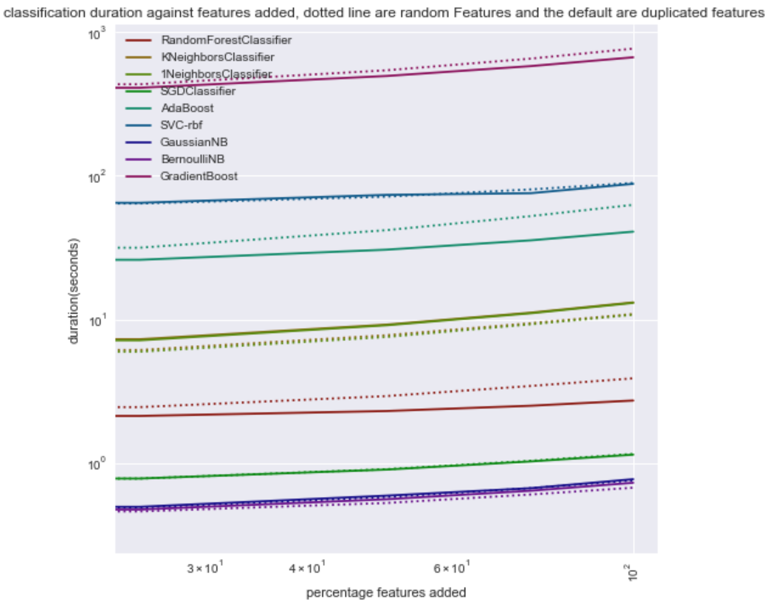


Figure 3: Features added straight line represent duplicate features and dotted line are random numerical features

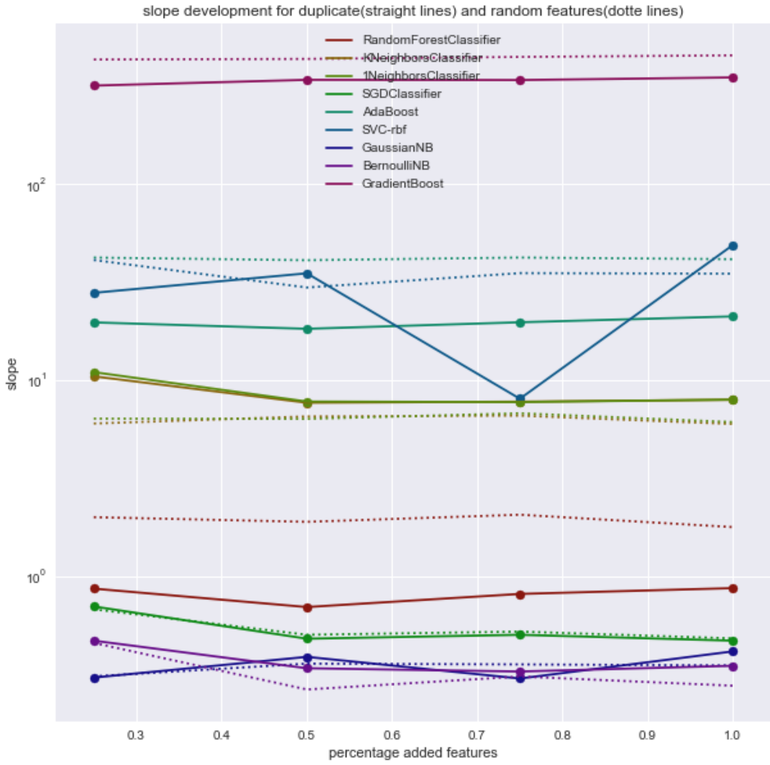


Figure 4: Features added straight line represent duplicate features and dotted line are random numerical features

4.1.1.3 combined in figures 5 and 6 adding and removing features is shown. in figures 9,8 and 7

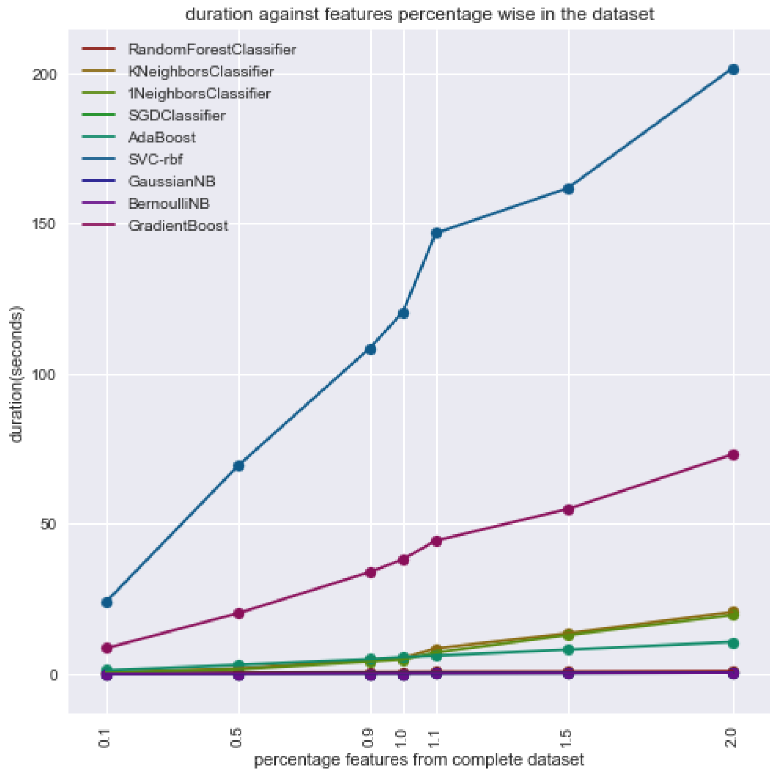


Figure 5: features randomly removed and feature randomly duplicated

4.1.2 Instances

4.1.2.1 Prediction For scalability it is easier to predict then for accuracy. The lines for the duration increase are on a log scale nearly linear. This makes for an easy fit with a linear line with log preprocessing. In [?] you can see that the averaged results of an instance scalability has a good fit. However if we try to predict on an individual bases per dataset the results are less promising [?]. Here we tried fitting a KernelRidge with some additional meta features. The meta features include the number of instances,features,numeric features ,categorical featueres and number of classes.

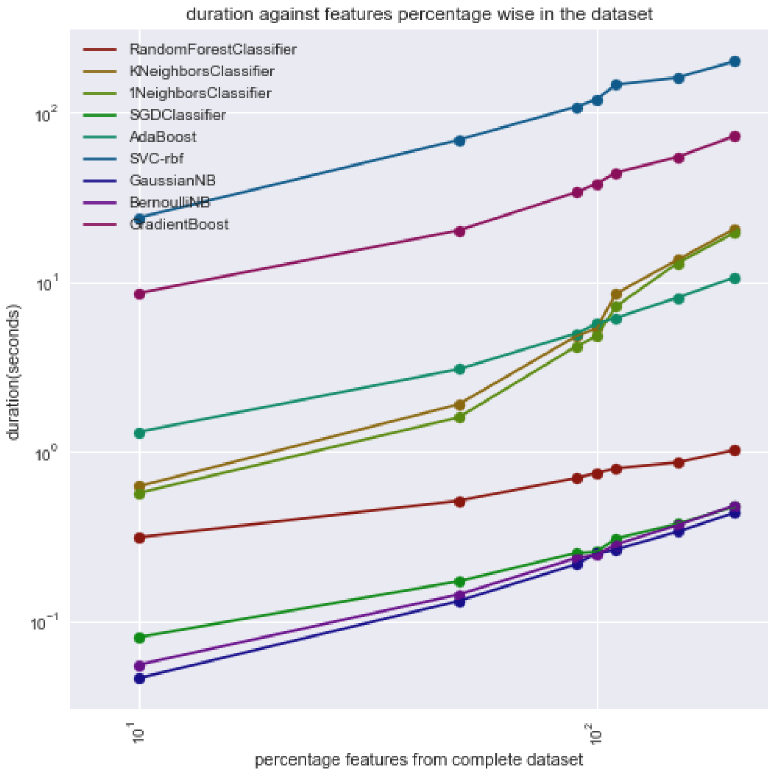


Figure 6: features randomly removed and feature randomly duplicated

4.2 Redundant features

We consider here both random and duplcite features as they are both redundant.

4.3 Noisy data

Datasets in these experiments get increasingly more random by injecting the features with noise.

4.4 Bias variance

4.5 Ontology

To present the results from all the experiments, this ranks the different algorithms on different aspects discussed. These rankings can be compared to results from a review from 2007 of classification techniques [22] The result will be a ranking of algorithms on different scales. The scales are:

- training and prediction duration
- robustness to noise
- preprocessing needed
- initial prediction accuracy

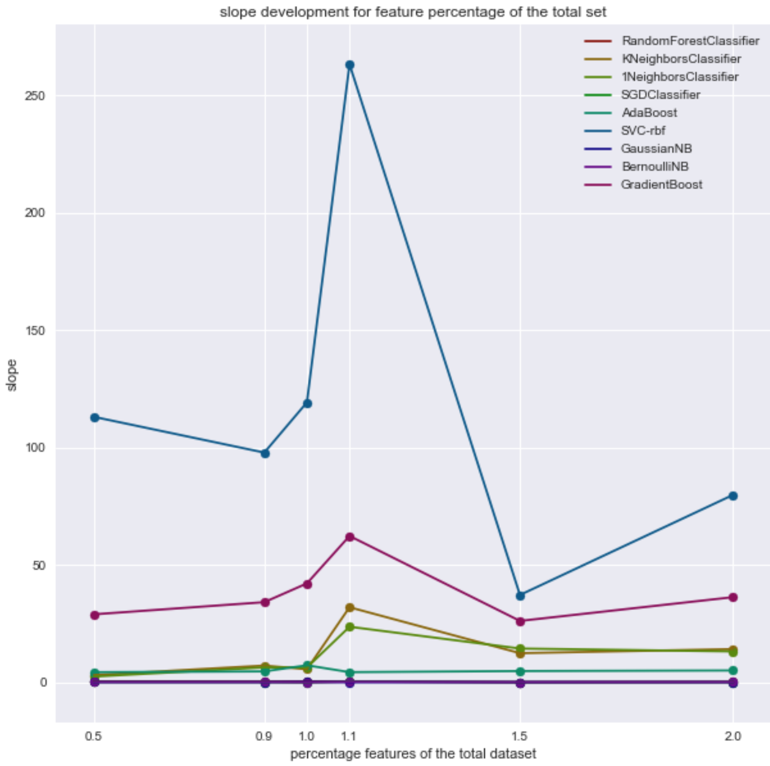


Figure 7: features randomly removed and feature randomly duplicated

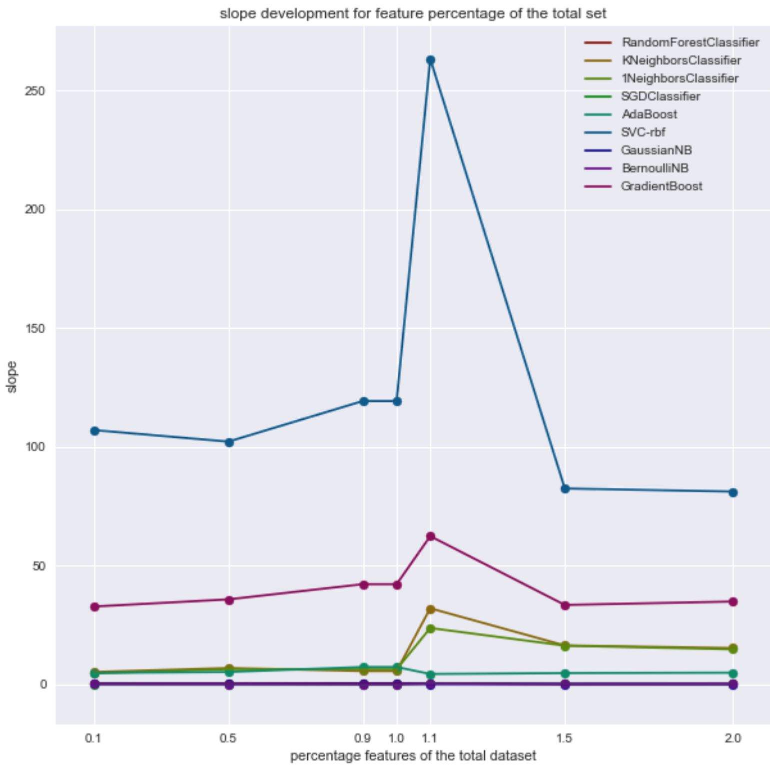


Figure 8: features randomly removed and feature randomly duplicated

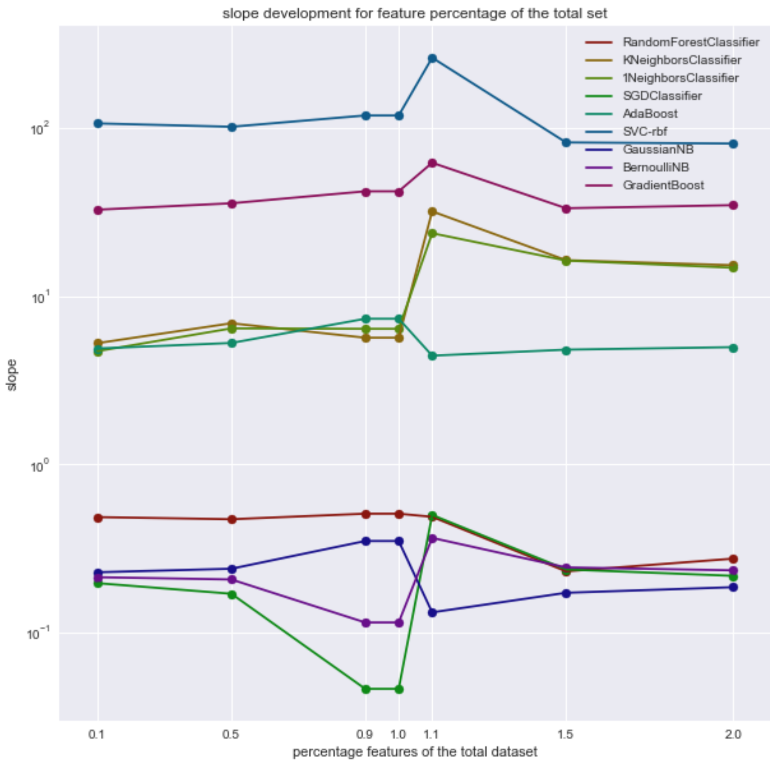


Figure 9: features randomly removed and feature randomly duplicated

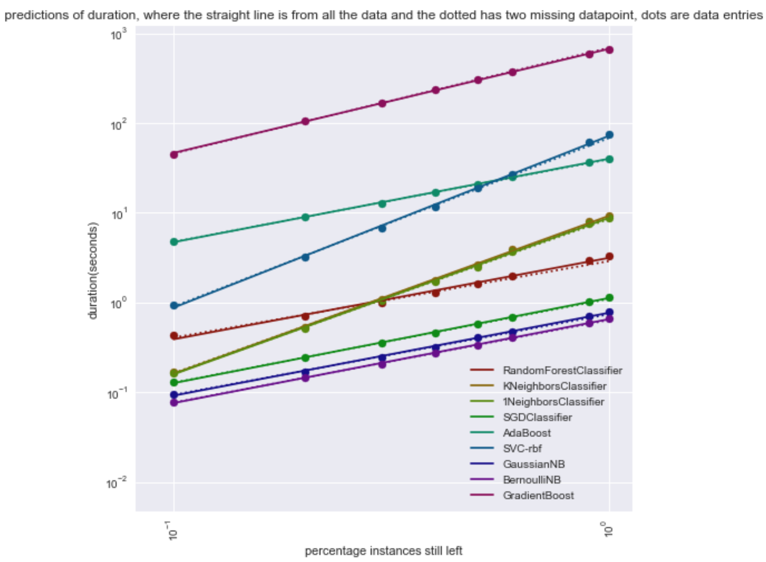


Figure 10: The dataset is reduced to a split of the initial dataset and then cross validation is done on the reduced dataset. The lines are predictions

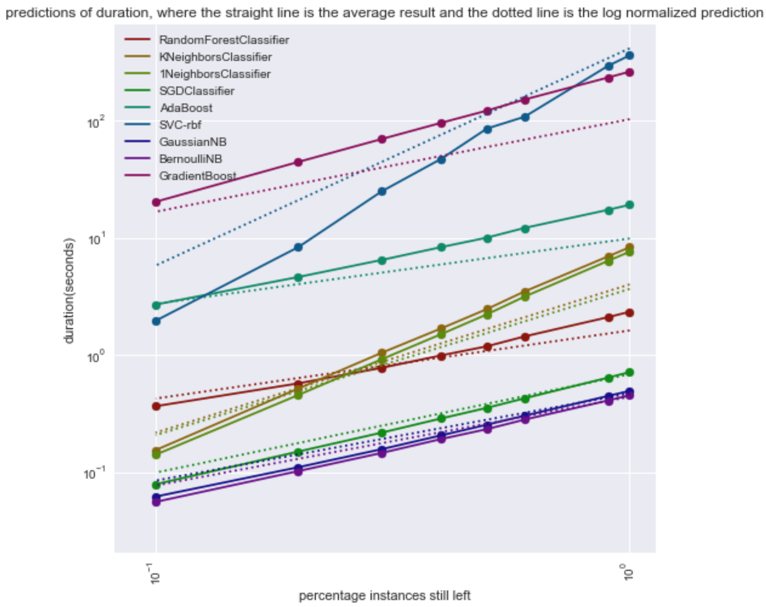


Figure 11: the dataset is reduced to a fraction of the initial dataset. A prediction is made of the duration of classification and prediction per dataset, indicated by the dotted line

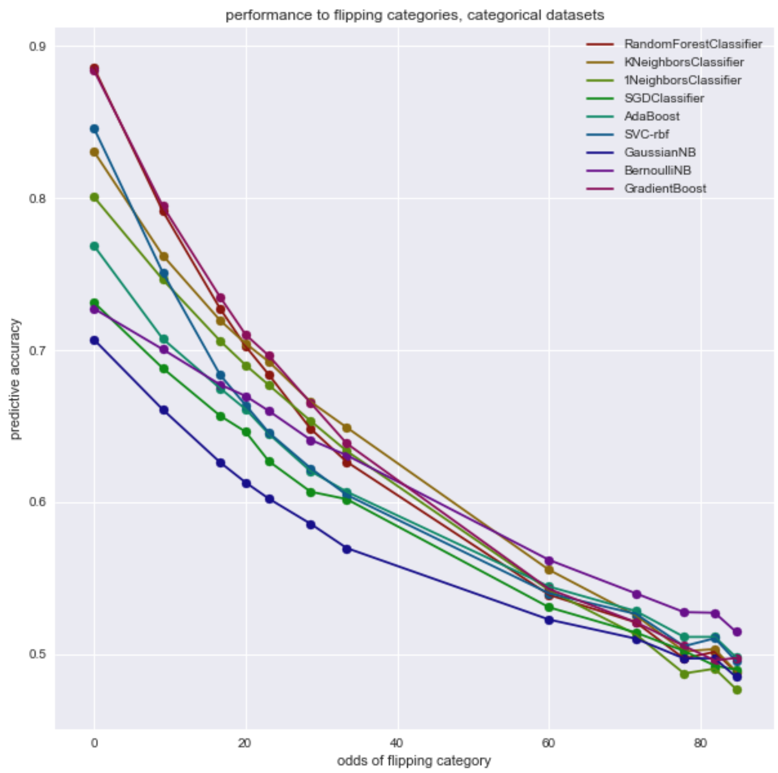


Figure 12: flipping categories for categorical features

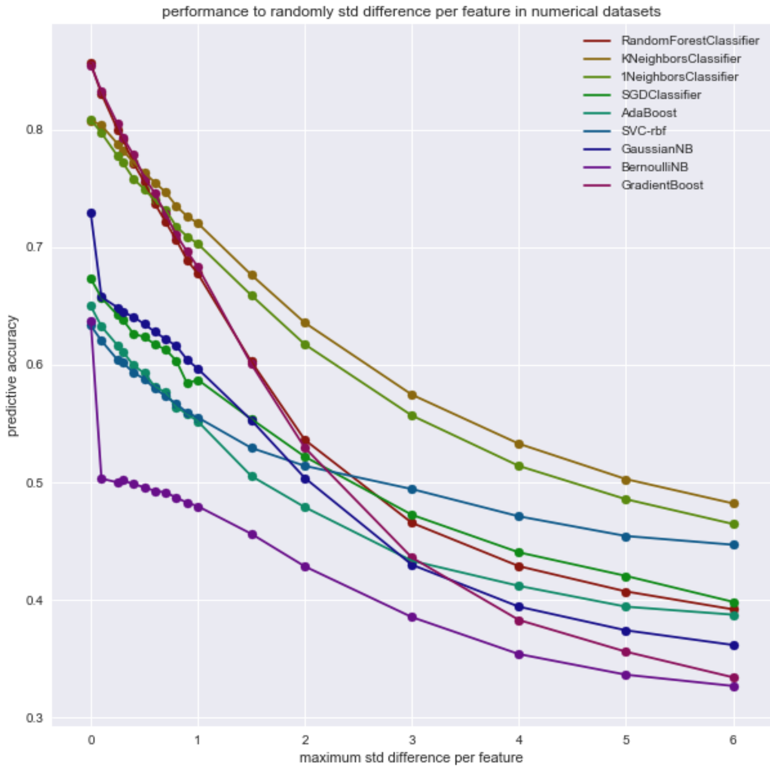


Figure 13: adding or removing uniformly random std to numerical features

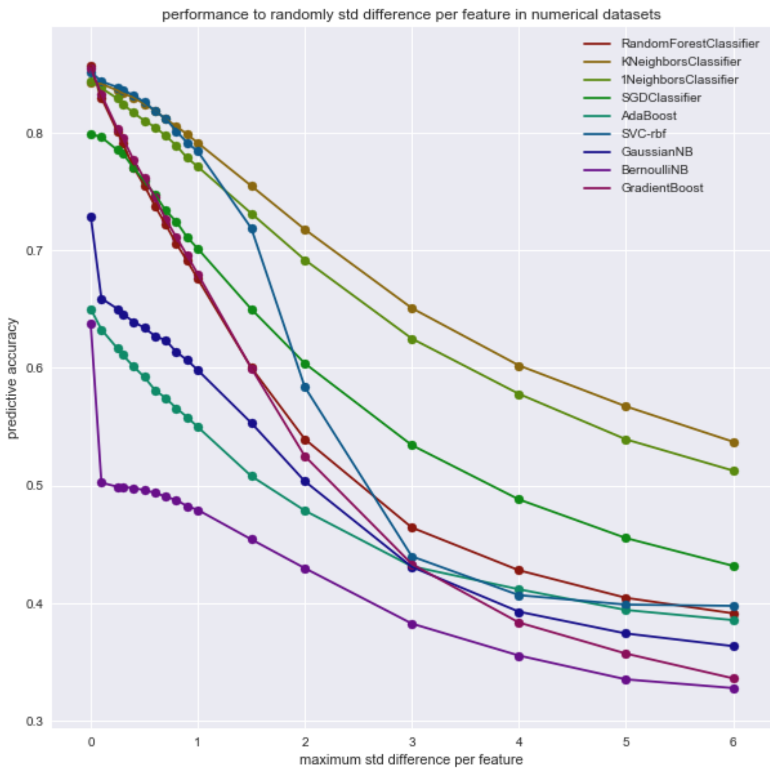


Figure 14: adding or removing uniformly random std to numerical features and preprocessing for some algorithms.

5 Discussion

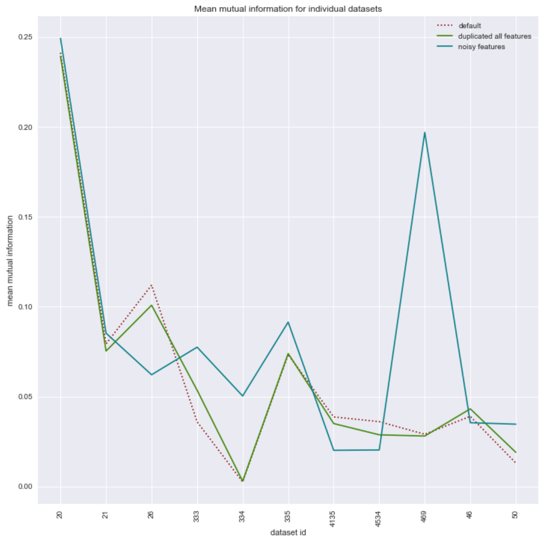
5.1 Resilience to noise

5.1.1 different approach

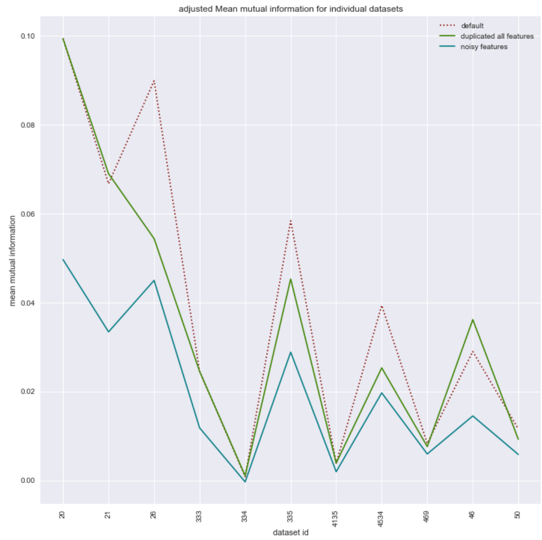
Instead of measuring the influence of standard deviation, we tried to add standard values to all features, the problem with that is features have different deviation and some classifiers prefer a zero mean. The results were similair to injection std but more abrupt. Just changing(moving/word) all features for some datasets mend a drop in accuracy but the continuation of upping the number mend little to change accuracy further.

5.2 MetaFeatures

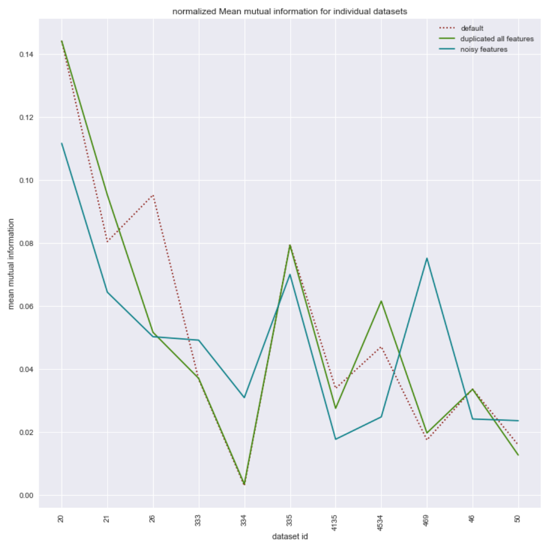
Meta features like mean mutual information or entropy for categorical features are calculated for our enhanced dataset with duplicate feature and random features. The result is that they do little to change these values and so do not indicate reduced information even tough commonly the results deteriorate when these features are added. However if we take the adjusted mean mutual information we can see a more clearer distinction between the permutations of the datasets. With those values the noisy dataset can be more recognized as being worse than before. The normalized is combination of both which has all three kinds in some case the best or the worsed.
The calculation of the normalized: $sqrt(H(labels_true) * H(labels_pred))$
the calculation for two cluster for adjusted mean mutual information: $AMI(U, V) = [MI(U, V) - E(MI(U, V))]/[max(H(U), H(V)) - E(MI(U, V))]$
The default mutual information is: $MI(U, V) = \sum_{i=1}^{I} U \mid \sum_{j=1}^{I} V \mid \frac{\mid U_i \cap V_j \mid}{N} \log \frac{N \mid U_i \cap V_j \mid}{\mid U_i \mid\mid V_j \mid}$



(a) Mean Mutual Information for mututations of datasets



(b) adjusted Mean Mutual Information for mututations of datasets



(c) normalized Mean Mutual Information for mututations of datasets

Figure 15: Check-out times

6 Conclusion

6.1 replication of previous work

6.2 New discoveries

6.3 Future work

6.3.1 More computation

with more computation time we can further solidify the results or see a different approach
Use datasets outside of openml like keggleto illistrate that openml is not the factor, openml can be focused on a certain subgroup more than another
splitting datasets for finer results. By focusing on a certain dataset type you can find why that dataset type is robust, think about sparse/dense feature rich, feature less. A combination.

7 References

References

[1] The Popularity of Data Science Software, Robert A. Muenchen, r4stats.com, (2017)

[2] Most Popular Programming Languages For Machine Learning And Data Science,Adarsh Verma, fossbytes.com, (2016)

[3] Machine learning: Trends, perspectives, and prospects, M. I. Jordan, T. M. Mitchell, Science Volume 349 issue 6245 pages 255-260, (2015)

[4] Storage predictions: Will the explosion of data in 2017 be repeated in 2018?, Nick Ismail, www.information-age.com/, (2017)

[5] Understanding Machine Learning Performance with experiment databases, Joaquin Vanschoren, KU Leuven, (2010)

[6] Quantifying the resilience of inductive classification algorithms, M. Hilario, A. Kalousis, Proceedings of the 4th European Conference on Principles of data mining and knowledge discovery, pages 106-115, (2000)

[7] Updating Formulae and a Pairwise Algorithm for Computing Sample Variances Tony F. Chan* Gene H. Golub'* Randall J. LeVeque, Stanford CS tech report STAN-CS-79-773(1979)

[8] LIBSVM: A library for support vector machines, Chang, Chih-Chung, Lin, Chih-Jen, ACM Transactions on Intelligent Systems and Technology (2011)

[9] Probability estimates for multi-class classification by pairwise coupling, Wu, Lin, Weng, Journal of Machine Learning Research 5 (2004)

[10] Support-Vector networks, Cortes, Corinna, Vapnik, Vladimir, Machine Learning Volume 20 issue 3 (1995)

[11] Idiot's Bayes—Not So Stupid After All?, David J. Hand, Keming Yu, International Statistical Review Volume 69 Number 3(2001)

[12] A Comparison of event models for naïve Bayes text classification, Andrew McCallum, Kamal Nigam, AAAI-98 workshops on learning for text categorization (1998)

[13] An introduction to kernel and nearest-neighbor nonparametric regression, N. S. Altman, American Statistician 46(3): 175-185 (1992)

[14] Random Decision Forests, Tin Kam Ho, Proceedings of the 3rd Iternational Conference on Document analysis and recognition (1995)

[15] A short introduction to Boosting, Yoav Freund, Robert E. Shapire, Journal of Japanse Society for artificial Intelligence 14(5):771-780 (1999)

[16] Multi-class AdaBoost, J. Zhu, S. Rosset, H. Zou, T. Hastie, Statistics and its Interface volume 2, pages 349-360 (2009)

[17] Solving Large Scale Linear Prediction problems using stochastic gradient descent algorithms, T. Zhang, ICML Proceedings of the 21 International conference on machine learning (2004)

[18] Stochastic Gradient Boosting, J. H. Friedman, Computational Statistics & Data analysis – Nonlinear methods and data mining volume 38 issue 4 pages 367-378,(2002)

[19] Greedy function approximation: A gradient boosting machine, J. H. Friedman, The annals of statistics volume 29 issue 5, pages 1189-1232,(2001) Previous bias-variance research has shown a trend of larger dataset increasing the bias component but still fluctuating with less than 10000 instances.

[20] Experiment databases, J. Vanschoren, H. Blockeel, B. Pfahringer, G. Holmes, Machine Learning Volume 82 issue 2 pages 127-158, (2012)

[21] Bias plus variance decomposition for zero-one loss functions,R. Kohavi, D. Wolpert, Proceedings of the international conference on machine learning pages 275-283,(1996)

[22] Supervised Machine Learning: A review of classification techniques, S.B. Kotsiantis, Informatica 31, (2007)

[23] Pairwise meta-rules for better meta-learning-based algorithm ranking, Quan Sun, Bernhard Pfahringer, Machine Learning Volume 93 Issue 1 pages 141-161,(2013)

[24] Scikit-learn: Machine Learning in python, F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion et al. Journal of Machine Learning Research 12, (2011)

[25] No free lunch theorems for optimization, D.H. Wolpert, W.G. Macready, IEEE Transactions on Evolutionary Computation Volume 1 Issue 1, (1997)

8 Appendix

8.1 datasets per figure