Algorithm primitives

Master Thesis

L.D. Stooker, 0819041

Supervisor: Joaquin Vanschoren
May 19, 2018

Abstract

By researching the python library which is gaining popularity we give a few insights into the qualitative properties of these machine learning algorithms. For accuracy GradientBoostingClassifier is a solid pick which outperforms with default settings on nearly all cases. on equal footing in accuracy is RandomForestClassifier an easier and quicker solution. For noisy data KNeighborsClassifier is most robust and Naive Bayes classifiers 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 Eindhoven University of Technology in the Data mining department. In this project we investigated annotations of primitives, more specifically primitives in the scikit-learn library. To elaborate on this we will outline the research questions and thesis structure further in this introduction.

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 classifiers 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 This figure however is outdated as new algorithms are added to the library over time.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[25].

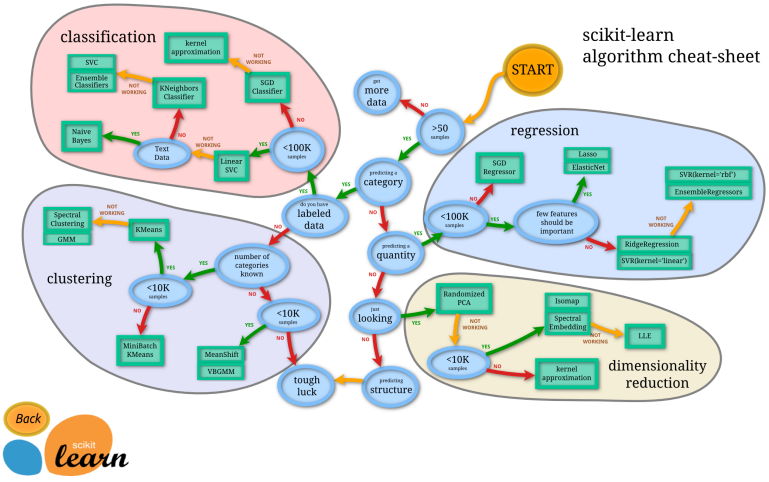


Figure 1: FlowChartML scikit-learn for any dataset which machine learning algorithm to choose

1.2 research question

We base our research question on the work of Joaquin to give properties to classifiers[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]

- What is the impact on runtime of the amount of features for classification machine learning classifiers?
- What is the impact on runtime of the amount of instances for classification machine learning classifiers?
- What is the impact on accuracy of irrelevant or redundant features for classification machine learning classifiers?
- What is the impact on accuracy of noisy features for classification machine learning classifiers?
- What is the percentage of Bias-error or variance-error for classification machine learning classifiers?

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 classifiers on their performance(predictive accuracy), robustness and scalability 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. The fifth chapter is a conclusion and discussion of the obtained results. This chapter also explains an outlook on future work and different roads taken. In chapter six the references are outlined. The last chapter is the appendix with some additional information which help with reproducing and validating the results.

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 libraries hosting machine learning algorithms in python but scikit-learn was the first to make a standard guideline which makes the classifier easier to interchange. By having default functions for fitting and predicting.

2.1.1 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 uses the highest probabilistic prediction for the prediction. 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. As RandomForestClassifier is made up of trees you can use the ordering of the trees, more specifically the top most layer of the tree to find feature importance. The first split in a decision tree has the most impact on the selection and might indicate a deciding feature. By looking at all the tree in a random forest this gives an average result of important features in a trained dataset.

2.1.2 KNeighborsClassifier

In the scikit-learn library KNeighborsClassifier is an implementation of the k-nearest neighbors classification algorithm(kNN)[29]. KNN 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. The k-NN uses as it names tells the nearest neighbors for calculation. For prediction the inputted feature set is traversed to find the nearest k points. Depending on the majority 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 should not influence predictive accuracy. The previously mentioned k parameter is an influencer for prediction quality.[13]

2.1.3 SGDClassifier

SGDClassifier is an incremental function to stochastically approximate the gradient descent of a loss or cost function [17]. The default classifier to optimize is a linear SVM on its loss function. To fit the classifier it expects continuous features with a mean of zero in a sparse setting. This makes the classifier sensitive to categorical data as it performs optimally with continuous features. The iterative steps for calculation gradient descent are bounded by the inverse of the learning rate and a threshold value. The threshold value indicates what degree of slope indicates a near minimal or maximal. The learning rate is used to update the model in each iteration. As the fitted functions are linear, if the input has mutliple classes a classifier predicts one class versus all other classes.

2.1.4 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 classifier 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.

2.1.5 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]. To optimize this method there are two main parameters C and gamma. The parameter C trades off misclassification of training examples against simplicity of the decision surface, a low C indicates a simple decision surface and lenient misclassification. The gamma parameter is a measure of influence for a single training example. The larger gamma is, the less influence a single instances has.

2.1.6 GaussianNB

GaussianNB is a naïve Bayes classifier implementation with the assumption that 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 classifiers. The only parameter of this classifier specifies the prior probabilities of the classes, which will when specified not be adjusted to the given input.

2.1.7 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.

2.1.8 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. The default loss function which is optimized in a stage-wise fashion is a logistic regression for classification with probabilistic outputs[19].

2.2 Definitions and abbreviations

In this section definitions and abbreviations are explained and set in context. Common synonyms are mentioned to avoid some confusion in text

2.2.1 Definitions

algorithm	A process with a specified in and output that solves a problem in a step by step case.
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 classifiers, 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 a dataset like injecting std deviation or inserting random categories. Adding or removing; of features or instances to the dataset also counts as it also influences the structure of a dataset.
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.
estimator	An estimator is part of an ensemble classifier. It is a single instance of a classifier in the vote of an ensemble classifier.
features	part of a whole, Consider a flower it has a color, size, amount of branches, amount of leaves and age. Features describe someone or something in this context it describes something, in this context it is the input to a machine learning for predicting a target , a synonym is variable.
fit	To fit the data, synonym with inputting the training data, preparing the classifier 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 classifier	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 classifier to cope with changes in features. An classifier is more robust if it deteriorates less than another.
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.

slope	a slope value is the value to go from one point to another as a vector. For example from point (1,1) to point (2,3) there is a slope of 1/2.
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.
algorithm	

2.2.2 Abbreviations

adaBoost	adaptive boosting classifier
biasVar	bias and variance
did	dataset identifier
SGD	Stochastic gradient descent
std	standard deviation
TU/e	Eindhoven University of Technology
SVC-rbf	Support vector classification with a radial based function kernel

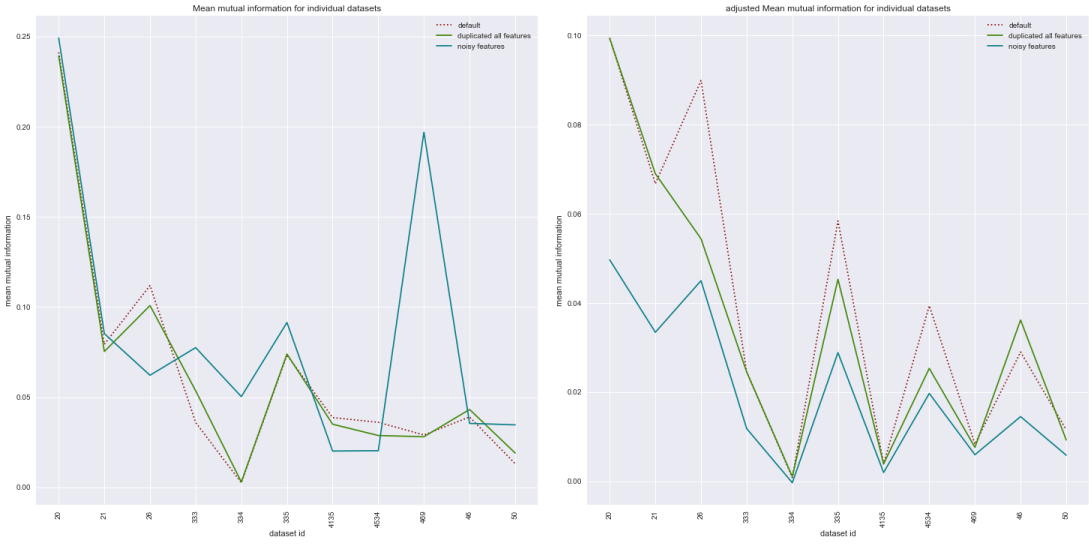
3 Experimental setup

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 MetaFeatures

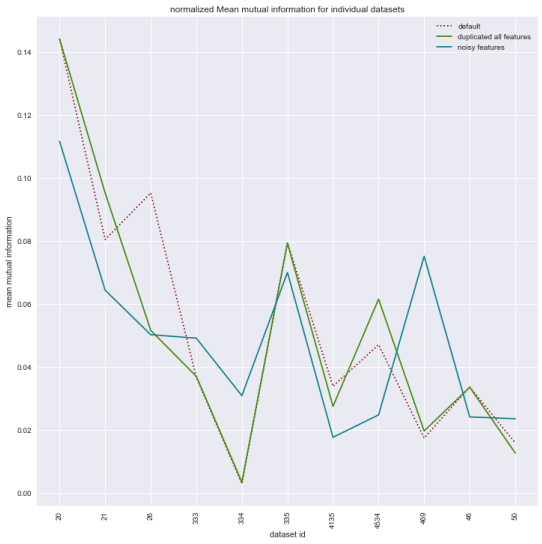
3.1.1 Mean Mutual Information

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, seen in figure 2a. However if we take the adjusted mean mutual information we can see a more clearer distinction between the permutations of the datasets, figure 2b. With those values the noisy dataset can be more recognized as being worse than before. The normalized seen in figure 2c is a combination of both which shows depending on the dataset a different ranking of all three datasets.
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 mutations of datasets



(c) normalized Mean Mutual Information for mutations of datasets

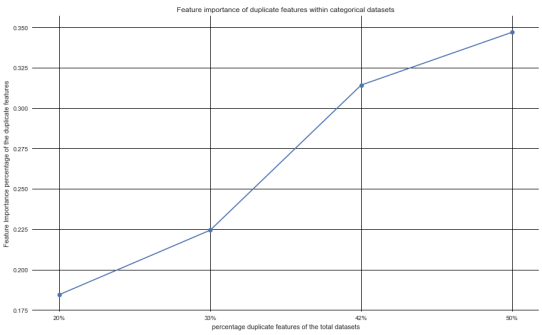
Figure 2: Mutual information difference between measure

3.1.2 Feature Importance

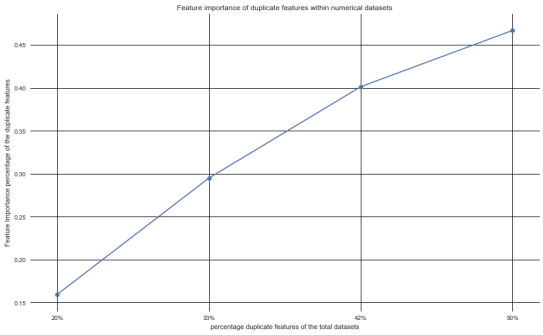
Feature importance calculated with a RandomForestClassifer gives an indication what features are important for the decision trees. In figures 3b, 3a, 4b, 4a, the feature importance of the added random features are shown. This makes it clear that RandomForestClassifier does not recognize these features as unimportant. There is a difference in the perceived importance of random categorical and numerical features. Even more so the distribution of random categorical features. The slope of random categorical with k=3, features importance is also steeper as it more than doubles and for numerical features it only nearly doubles. This is however not true for random categorical features with k= 100 which has an equal absolute increase but starts higher. This can be explained as more perceived variance in k=100 which might indicate better probability of differentiating between the target classes. The lack of increase can be attributed to the cap on the importance at 100%. The original features might give some better results in predicting but the added features still consists of some percentage of the total amount of features. Random Forest Classifier has a limit on the amount of features considered for each split which is capped

at a percentage of the total set. This is partly the reason that these random features can be considered as important in the split together with their variance. The variance gives perceived information as you look at a subset of instances the high variability shows that for some target class this feature value is this number and for the other classes the odds of different values for that feature is high.

In the figures 3,5 the feature importance of duplicates can be seen. This shows that the order of features is important as the duplicate features have less importance than their percentage of the dataset. It also means that they are not recognized as totally redundant for prediction. The difference between the feature importance for numerical and categorical is also shown. This is partly discussed earlier as the variation in the features and feature order. This has the effect of duplicate numerical features having around 47% feature importance when the dataset is half duplicates. For categorical features this is only 39%. The feature importance of the GradientBoostingClassifier seen in figure 6 shows a bit less importance for random and duplicate numerical features. The random features even start with more feature importance than the duplicate mostly due to the high variation within the duplicates. With 50% additional features the duplicates have more importance than the random features on average.

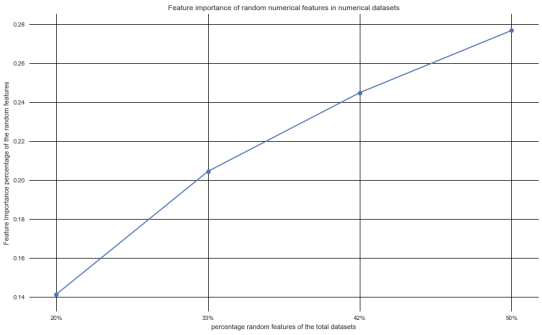


(a) Feature importance for random categorical features with k = 3.

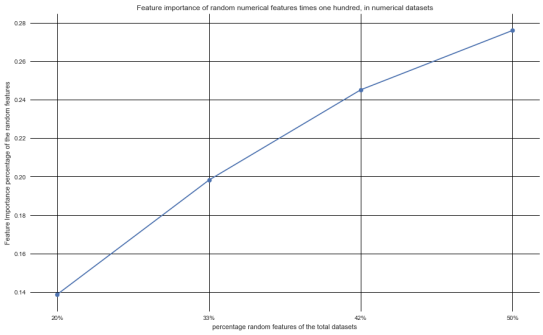


(b) Feature importance for random categorical features with k = 100.

Figure 3: Feature importance of random categorical features. The k value has an impact on the perceived importance of the features.

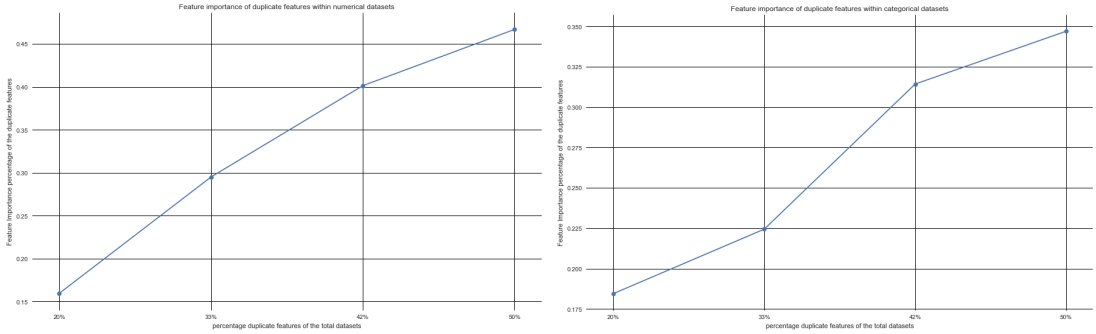


(a) Feature importance for random numerical features with the default uniform spread between 0 and 1.



(b) Feature importance for random numerical features which are multiplied by a 100.

Figure 4: Feature importance of random numerical features. The average mean does not influence the importance of random numerical features



(a) Feature importance for duplicate features for numerical datasets

(b) Feature importance for duplicate features for categorical datasets.

Figure 5: Feature importance of duplicate features and their effect on categorical or numerical datasets

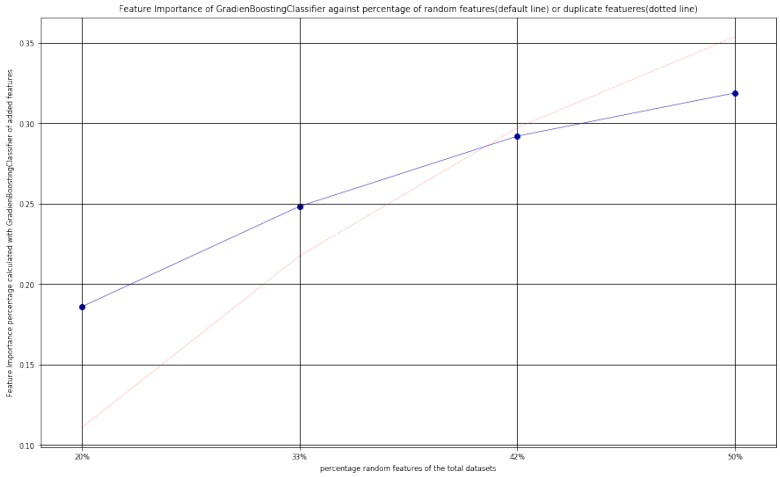


Figure 6: Feature importance for duplicate features and random features calculated with a GradientBoostingClassifier instead of the RandomForestClassifier on a numerical dataset

3.2 Model Validation strategies

What strategies do we use to split the datasets into training and test to measure predictive accuracy, duration, bias and variance.

3.2.1 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 as standard as it is perceived to be a well rounded amount of folds for decent results[28].

3.2.2 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 error.

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. This is 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 more suitable for a 2 classes target.

3.3.2 Categorical datasets

Categorical datasets hold only features with categorical values. These features are useful for a RandomForestClassifeir to make decisisons with the internal decision tree but for a KNeighborsClassifier it is harder to use these features as input as the structure of the translation to numbers also has an effect.

3.3.3 Numerical datasets

Numerical datasets hold only features with numerical values. These features are hard to use by classifiers like BernoulliNB which translates the values in their own way by uniqueness. For classifiers like KNeighborsClassifier it is easier to use the uniquess of numerical values for predictions.

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 scikit-learn 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 dictionaries 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 do 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 original 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 random 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 Redundant duplicate features

Redundant duplicate features experiments are datasets with duplicate features appended in training and different features appended to the test set. These features so appear to have some predictive quality similar to the features already in the dataset. These features are similar to the original dataset, so we can better measure the impact of scalability of features similar to the duplicate features. The comparison can be made to the randomly added features datasets in terms of scalability and predictive accuracy

3.5.5 Noisy data

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

3.5.5.1 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** data point x in feature k
. **if** random()*amount > 0.5
. x = random.choice(distribution_X feature k)

3.5.5.2 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 data points 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** data point 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)

3.5.6 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.

3.5.7 PreProcessing

Preprocessing is a neccesary job for most machine learning classifiers as they are dependent on the input structure to classify. For example KNeighborsClassifier is dependent on distance between instances, if we look at categorical data the distance between instances is not always relevant. That is why experiments with specifically categorical datasets are repeated with some preprocessing for at least KNeighborsClassifier, SGDClassifier and SVC-rbf. For these classifiers a translation of categorical features seems to be necessary [26][27]. The proposed preprocessing steps are OneHotEncoder and Standardscaler in that order. OneHotEncoder translates the categorical features in features corresponding to the amount of classes. Each feature is then a boolean value of being the class or not. The StandardScaler removes the mean and scales on the standard deviation. Storing the mean and std of the training set to use it to transform the test set balances the dataset accordingly.

4 Experimental Results

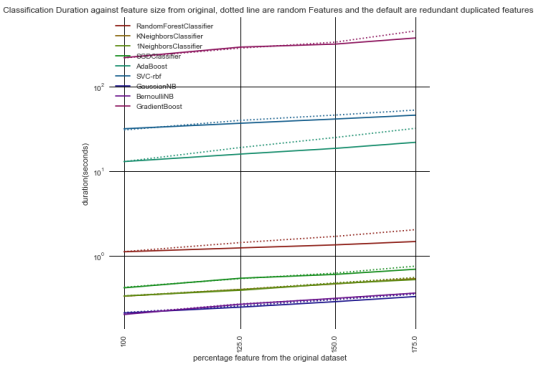
4.1 Scalability

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. In total fitting over 9 times the dataset size and once predicting the whole dataset.

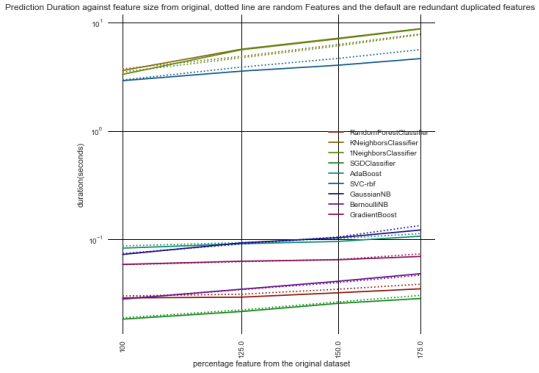
4.1.1 features

Due to the robustness experiments a lot of results change the composition of a dataset in the feature dimension. In this subsection all results changing the composition of a dataset are included with a duration y axis.

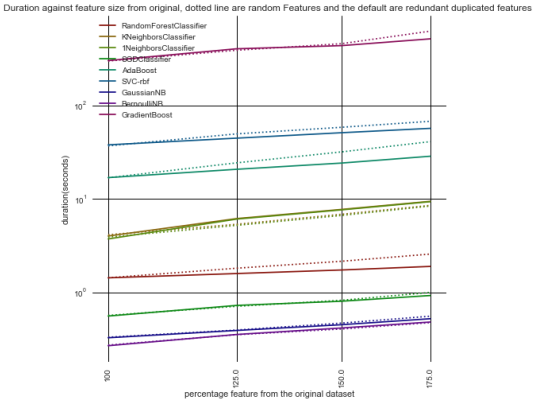
4.1.1.1 Added features In figures 7a,7b,7c the durations of adding redundant duplicate or random features are shown. In figures 8a,8b,8c the durations of adding duplicate or random features are shown In figure 7a the classification times are shown for a growing feature size, these durations are a great part of most classifiers.In figure 7b the prediction time is shown, here most classifier take only a short while. The combined time can be seen in figure 7c which shows the total difference. The averaged size of this dataset is 3797 instances and 200 features. and the slope of these added figures is seen in figure 9.



(a) only the time needed to predict

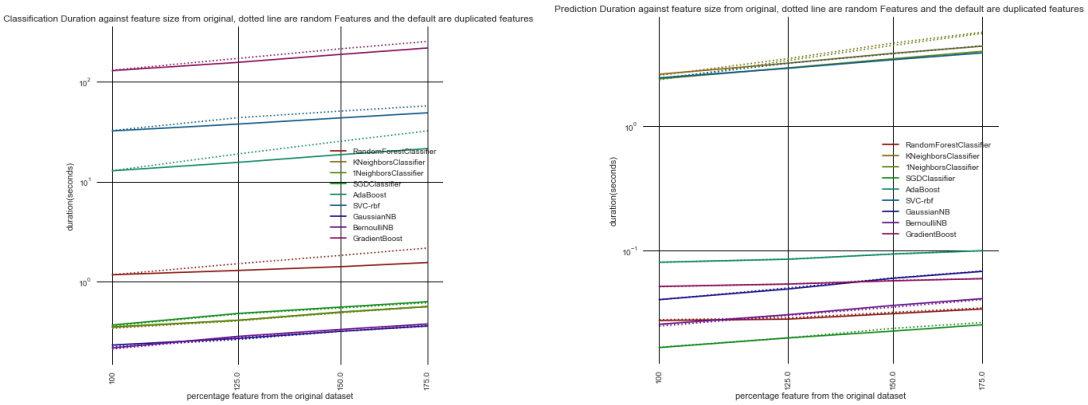


(b) only the time needed to make the predictions



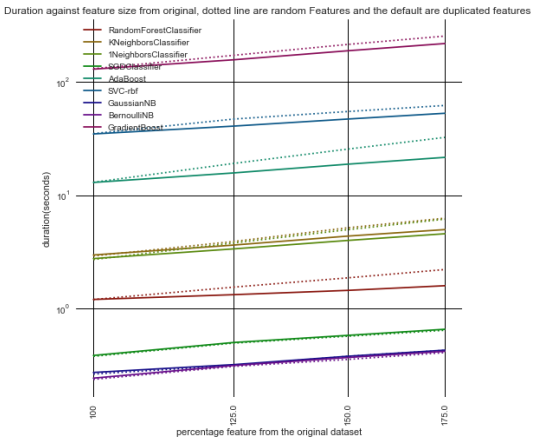
(c) The combined time of predicting and classifying on a dataset with added random and redundant duplicated features

Figure 7: Adding redundant duplicate and random features to a dataset plotted against the duration needed.



(a) only the time needed to predict

(b) only the time needed to make the predictions



(c) The combined time of predicting and classifying on a dataset with added random and duplicated features

Figure 8: Adding duplicate and random features to a dataset plotted against the duration needed.

RandomForestClassifier The duration of the RandomForestClassifier is fairly below average in comparison. The difference between random and redundant duplicate is noticeable as the random feature take more time mostly for classification as classification takes the longest of the two. This longer duration can be attributed to the at. As seen in section3.1.2 the features are given some importance which will take some time. There is also hardly any difference between the redundant or normal duplicate features.

KNeighborsClassifier The duration of KNeighborsClassifier is average in comparison. Most of the time is needed for prediction as there is only minimal effort during classification. The difference in prediction time between the three variants is surprisingly large. The smallest prediction duration is for the duplicate features which seems obvious considering they are most similar to the features in the training set. The redundant duplicate features take longer than the random which can be explained as slightly more variation between the features. On average the std of a random numerical feature is 0.28, the std of a numerical redundant features is higher on average in for each dataset. A difference between 1NN and kNN is also only minimal as the largest part of the prediction is making for example a kd-tree and the search on such a data structure is far smaller than the making of.

SGDClassifier SGDClassifier has one of the shortest durations of the machine learning process. The prediction is even the smallest. There is also hardly any difference between the input of features. The random features have take only a slightly longer than any sort of duplicate feature. The short prediction time is due to only filling in made linear regressions lines based on the inputted features.

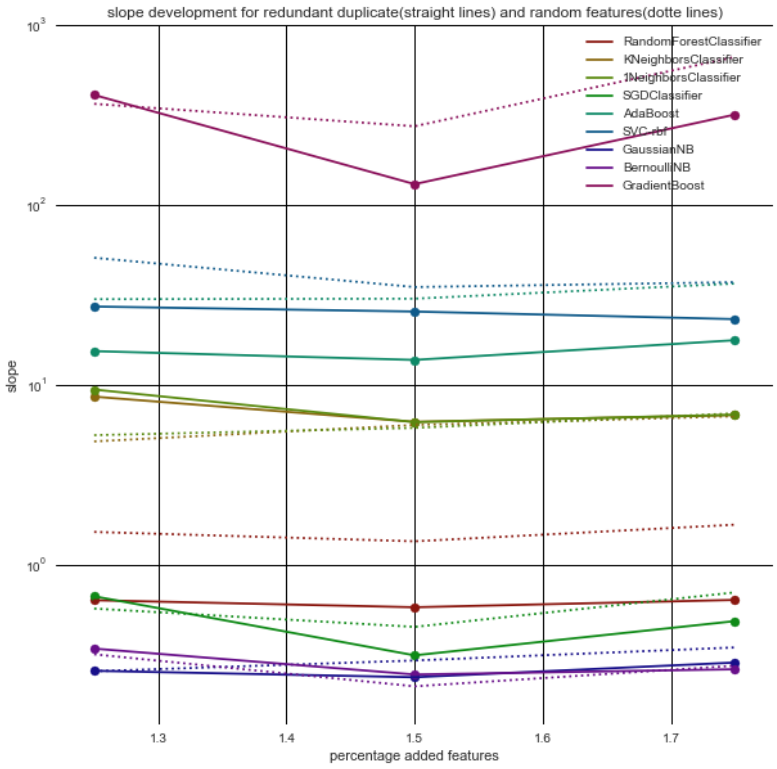


Figure 9: Slope for 7c with the dotted line for random added features and the straight line for redundant duplicate features

AdaBoost AdaBoost has one of the longest classification times with default 5 times more estimators it is bound to take some more time as RandomForestClassifier. The impact of random features compared to any sort of duplicate features is large with a steady increase in duration. This is mostly due to the variation in the added features and the more effort the Decision Trees need to filter out any information. The effect of the adaptive boost is unnoticeable between random or duplicate features compared to the standard RandomForestClassifier.

SVC-rbf SVC-rbf has one of the slowest classification and prediction time. With the calculation of the decision surface taking the most time. The duration of prediction is also slow in comparison to the other classifiers. The impact of the random features is noticeable with the variation present in those features SVC takes a bit longer to classify and predict.

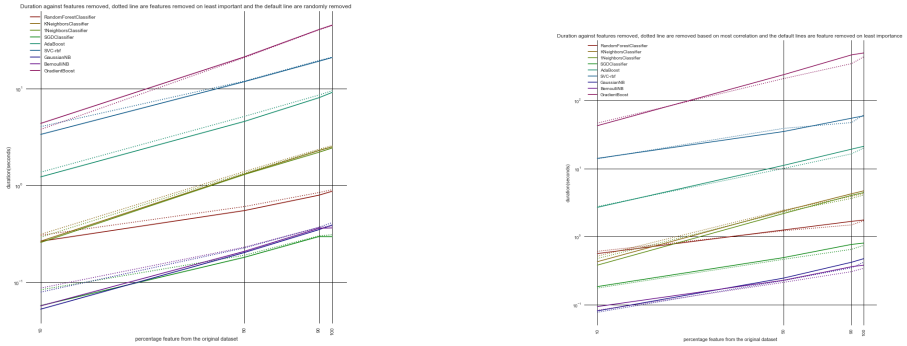
GaussianNB GaussianNB has one of the quickest prediction and classification time. The calculation needed for the mean and the variance is easy and there seems no significant impact of the different added features. That is reasonable as it should have little impact on the calculation of the mean and variance of a single feature.

BernoulliNB BernoulliNB is also one of the quickest in prediction and classification time. With a similar approach as BernoulliNB there is little calculation in both prediction and classification. The only added calculation for BernoulliNB is a binarization.

GradientBoostingClassifier The GradientBoostingClassifier takes the longest amount of time of all classifiers. With 10 times the estimators of RandomForestClassifier it takes on average 250 times the total duration. With the calculation in multiple stages the duration time ramps up to fit the Decision trees. The impact of random features is more than either of the duplicate features. This is also due to more variation in the features with also using DecisionTrees, GradientBoostingClassifier

The slopes of the different classifiers are close to a linear scale and vary a bit between addition(figure 9). This is mostly an the starting impact of the added features compared to the clean dataset.

4.1.1.2 Removed features In figure 10 the duration against features removed from a dataset are shown. Three different techniques of feature removing are shown. In figure 10a features are randomly remove or a RandomForestClassifier is fitted on the training set and the least important feature is removed. In figure 10b the features are removed by their importance or by having the most correlation with another feature in the training set. In figure 10a the average dataset is 2230 instances with 307 features and in figure 10b the average dataset is 3329 instances with 258 features.



(a) features randomly removed or removed by importance

(b) features removed by least importance or by most correlation with other features

Figure 10: Duration of classification and prediction of the classifiers against removing features from clean datasets.

RandomForestClassifier A small difference in duration can be seen between the randomly removed feature and feature removed by lack of importance. On the smallest dataset the importance removed feature dataset takes longer. This can be explained as the feature set needing more time for the quality of the split. Between highest correlation removing and importance removing the difference is minimal

KNeighborsClassifier For KNeighborsClassifer there is also more time needed for a small importance removed dataset and even more for highest correlation based removing. The longest duration for correlation based removing can come from the lack of correlation giving more variation between features as they are least similar to each other. This means that the features are different and more calculation is needed to find neighbors. For important features this is also the case but to a lesser extend than the least correlated features, which indicates that some features can still be closely correlated but still have some perceived importance.

SGDClassifier The SGDClassifier can be seen as one of the quickest algorithms in figure 10a but a bit more time consuming in figure 10b. There is even a big time difference between randomly removing features and removing feature by least importance. The more important the features the longer the classifier needs. This can be explained as the features less likely being standard for the input of the classifier. This means that the important features are likely dense features.

AdaBoost AdaBoost classifier needs more time for least important removed feature dataset than randomly removed features. This seems similar to the RandomForestClassifier. With the important features left AdaBoost has a harder time to classify outliers which the removed features were more likely to be used for.

SVC-rbf The SVC-rbf classifier needs more time for important feature in line with most other classifier, this means that with the less important feature removed, more time is needed for the decision surface calculation. This is also in line with the similarity of duration between the feature importance and correlation features being removed duration. With less correlated features being important features the

odds are higher of a less linear relation between the targets.

GaussianNB Removing the least important features from a dataset has less of an impact of decreasing the duration than removing random features. The effect of calculating means and variances should not have this effect as that calculation is more dependent on instances than values. The only reason can be the calculation of the maximum likelihood needing slightly more workload. The Difference between feature importance and correlation feature removing is negligible

BernoulliNB The same behavior seen in the GaussianNB can be seen for the BernoulliNB. This can be due to the randomly remaining features being easier to binarize. As that is the only factor that might be influenced by values of the features.

GradientBoostingClassifier As the only classifier GradientBoostingClassifier takes less time with only the most important features. This can be the nature of the stage wise steps of GradientBoostingClassfier needing less time with more important features than random features who's importance can be questionable. The dataset with the least correlated features hardly as any difference with the least important removed feature dataset.

4.1.1.3 combined in figure 11 adding and removing features is shown. in figure 12 the slopes are shown for adding and removing the features. The two different calculation styles show what the difference in measuring has for influence. Figure 12b has the most balanced approach as basing the calculation on the clean dataset reduces the influence of outliers. The slopes in figure 9 are also considered in these results.
For most classifiers the results paint a clear picture. Removing features has more impact on duration as adding additional features, the general trend line is mostly linear for adding features and less linear for feature removing. In figure 12b there is a decrease of the duration slope from the minimal to the clean dataset. Averaging these results gives a slope close to the slope found in only adding features. Another strange behavior is around the full dataset results there is a drop in the slope. For SGDClassifier there is nearly no difference in duration between 90% and 100% features of the dataset seen as the slope dropping outside of the picture. For most classifiers there is a similar but less drastic drop. The drop can be seen in figure 12b as being low. This is because the difference between instances is small and the duration calculation is inaccurate. With the short duration needed needed for training and prediction in the SGDClassifier the difference can be larger between the results.

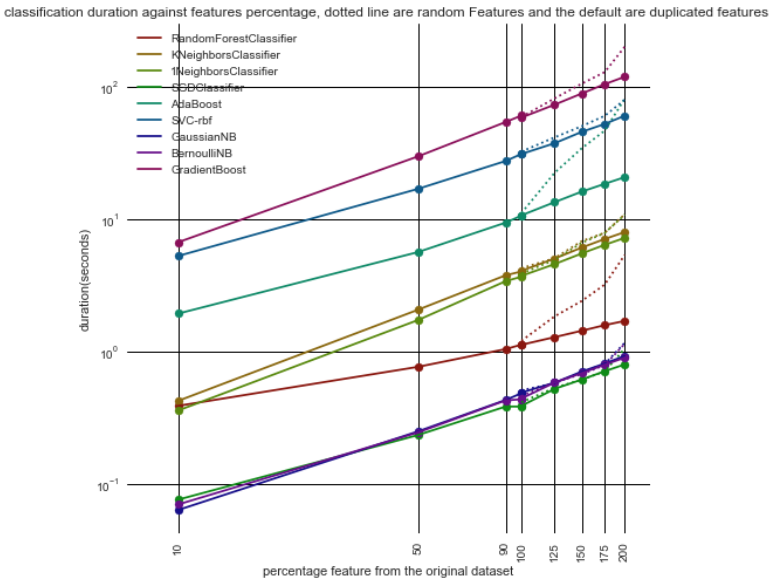
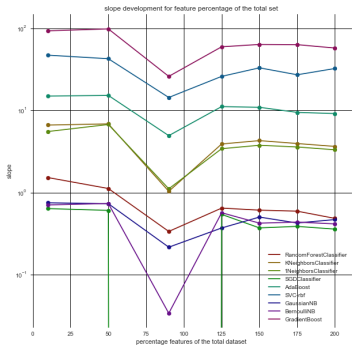
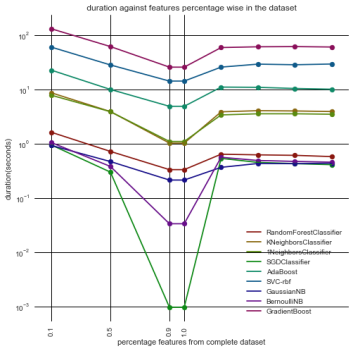


Figure 11: features removed randomly and features redundantly duplicated slope



(a) features removed randomly and features duplicated with a log scale

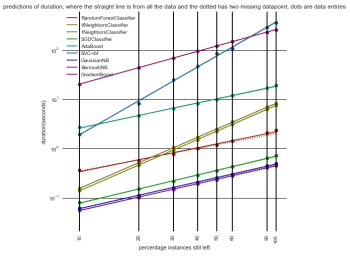


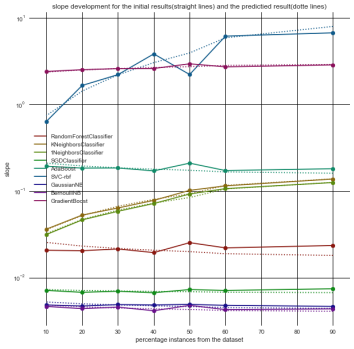
(b) features removed randomly and features duplicated slope based on the average point of a clean dataset, with a log scale

Figure 12: Slopes of Feature percentage total duration of the original dataset calculated from figure 11.

4.1.2 Instances

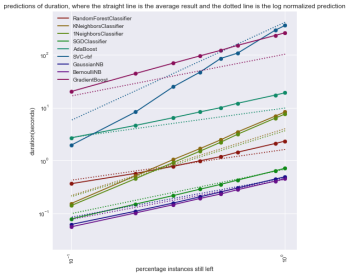
In this sections datasets are split reduced in the instance dimension. First the set is shuffled and then a percentage of the dataset is cross validated. In figure 13 the results can be seen. In these results the scalability in the instance dimension of the classifiers is shown. The averaged dataset is 7786 instances with 154 features for figure 13
Comparing the lines visible in figure 13a with the actually slope in figure 13b the classifiers GradientBoostingClassifer, RandomForestClassifier, AdaBoostClassifier, SGDClassifier, BernoulliNB and GaussianNB have a near linear ascension. Expect for the bump in duration around 50 % is the duration increase stable for these classifiers. This is also expected from classifier depending on standard calculations of mean, variance, linear fit and decision trees. Those calculation are linear in the amount of input. For the remaining classifiers SVC-rbf and the kNN variants the algorithms show a non linear trend. For the kNN classifier this is due to the prediction using a kdTree or ball tree which needs longer linear time for construction and searching. For SVC-rbf the implementation estimates a quadratic time complexity for the number of samples.
The averaged result can be easily captured with a linear fit with taking the log of the duration and instances. In figure 13a the predictions lines for most classifiers, line up with the actually results. Together with the slope seen in figure 13b this seems reasonable. However this is the average result on around 80 datasets. Can we actually predict for a single dataset the duration it takes for classification and predicting. The result is shown in figure 13c as the dotted line, that line clearly does not line up with the averaged result. Visisble in the appendix in chapter 7.4.2 the duration lines per datasets are less linear than the averaged result. The prediction is made by adding some additional information to a Kernel Ridge regression algorithm namely the number of categorical, numerical and total features, the number of instances and the number of classes in the target feature. These are chosen based on the configuration of the chosen classifiers and early results of duration experiments.





(a) The percentage of the initial daaset against the duration needed. The dots are the results and the dotted line is a prediction line with figure lines are fitted on these dots. The scale is log based 13a as input.

(b) The default line is the slope of the experiments and the and the dotted line is a prediction line with figure 13a as input.



(c) The default line are the result of experiments done, the same as the ones in figure 13a. The dotted line is prediciting by a KernelRidge for each inputted dataset the trend line of the duration.

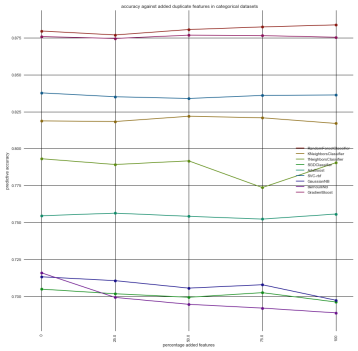
Figure 13: Instances scalability, the slope and the prediction attempts.

4.2 Redundant features

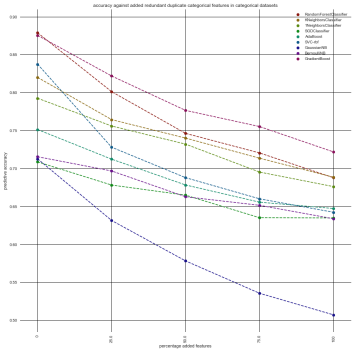
This section focusses on feature manipulation of clean datasets and the accuracy impact. By either removing or adding features the dataset changes and maybe also the workings of classifiers. By comparing the results of a clean dataset to a manipulation in the feature dimension we see how classifiers handle these feature changes.

4.2.1 Adding features

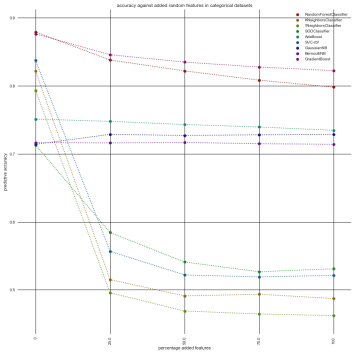
We consider here random, duplicate and redundant duplicate features as they are all redundant. In section 3.1.2 the importance and in section 3.1.1 the mutual information is shown which changes with the addition of certain features.



(a) Adding increasingly more duplicate features to categorical datasets

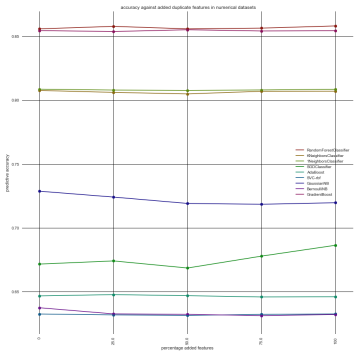


(b) Adding increasingly more redundant duplicate features to categorical datasets

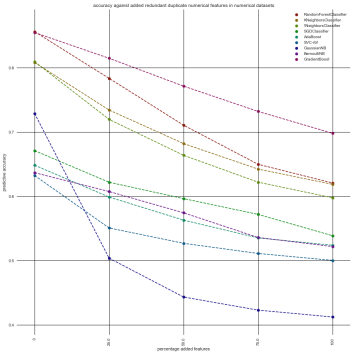


(c) Adding increasingly more random features(k = 100) to categorical datasets

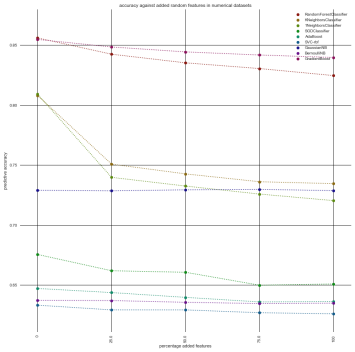
Figure 14: Predictive accuracy of categorical datasets injected with irrelevant or redundant features



(a) Adding increasingly more duplicate features to numerical datasets



(b) Adding increasingly more redundant duplicate features to numerical datasets



(c) Adding increasingly more random features to numerical datasets

Figure 15: Predictive accuracy of numerical datasets injected with irrelevant or redundant features

RandomForestClassifier RandomForestClassifier is robust against default duplicate features with even gaining slightly better accuracy in both numerical and categorical datasets. The decrease in accuracy with adding redundant duplicate features is one of the hardest. As seen in chapter 3.1.2 RandomForestClassifier gives these added duplicate features less than its share of the dataset importance. The redundant duplciate feature have slightly more importance than the random numerical features and the result is far more decrease of accuracy for the redundant duplicate features. RandomForestClassifier is a ensemble method and therefor the performance does not drop dramatically on some redundant features but with a larger share the impact becomes more devastating.

KNeighborsClassifier The kNN classifier the robustness against duplciate feature is good for the default setting but for the 1NN there is more variance of performance for the categorical datasets. Adding the redundant features the kNN classifier is not so robust and the accuracy drops quickly. Against the random features the most impact can be seen when introducing the features. Upon introducing of the random categorical features the accuracy drops hard 30 % accuracy. For the numerical random features this is less but in comparison it is one of the largest drops in accuracy. This is due the numbers associated with these random features. For the categorical features with k=100 the absolute difference between features become larger than with the default classes which are incremental categories. With a class feature value in the training set of 10 and a similar class with feature value 90 the absolute difference is large between features. Without preprocessing the kNN has no robustness against these irrelevant features.

SGDClassifier SGDClassifier is robust against duplicates with the random nature of the classifier. With added duplicates with numerical datasets the classifiers fluctuates a lot. With a doubled feature size the accuracy goes up a few percentages in accuracy. For categorical datasets the performance is less positive. Mostly for adding random categorical features the accuracy drops harshly. This is mostly because the SGDClassifier cannot handle categorical data nicely without preprocessing. The increase in number of classes and so more value variation makes is harder to use those features. The robustness for numerical

datasets is stronger with average robustness for redundant duplicate features.

AdaBoost The accuracy is hardly influence by duplicating features only for categorical datasets there is some slight variation between datasets. The robustness against redundant features is one of the greatest with subpar starting performance the drop in accuracy is fairly low. With similar reasons as RandomForestClassifier does the AdaBoost also give priority to the redundant duplicate features. For the random features AdaBoost is a robust classifier, with the focus on outliers AdaBoost can discredit these features as limiting the impact on the predictive accuracy.

SVC-rbf The robustness of SVC-rbf against duplicate features is similar to most classifiers. The lack of robustness against categorical datasets is mostly with the impact of these features. SVC-rbf builds on all features the decision surface and with these features being irrelevant the SVC has a harder time to correctly classify. Similar behavior can be seen with the introduction of categorical random features. There are not misleading like redundant duplicate features but with the variation of 100 classes the SVC will fail to make a good decision surface without tuning of the C parameter to simplify the decision surface. There is more robustness for SVC with numerical random features. Handling of numerical redundant duplicate features is also not the strong suit of SVC but with a low accuracy the SVC is already under fitted on the data.

GaussianNB GaussianNB is one of the only once to lose some accuracy with duplicate categorical features. With the added features the likelihood function has a decreased performance. The dependance on all the feature is also visible with the redundant feature addition for both categorical and numerical datasets the performance drops significantly with the introduction. The classifier depends on all features to fall in the distribution with a mean and variance and the added features fall outside these values and accuracy drops. For random categorical features the GaussianNB is even the most robust with an increase in accuracy with the introduction. For numerical the accuracy stays the same. This is by virtue of the random features having a clear structure of uniform randomness which the classifiers gives no importance to in classification accuracy.

BernoulliNB Similair to GaussianNB does BernoulliNB lose accuracy with introduction of duplicate features. As a trait of a Naïve Bayes classifier all features are given some importance and these duplicate feature have equal perfomance as the original features. As this is mostly categorical datasets the binarization of these feature decreases the accuracy. The redundant duplicate feature addition to the dataset decreases the accuracy only slightly as seen with the introduction of duplicates the accuracy can decrease with more irrelevant features. The decrease of accuracy of the BernoulliNB is the lowest which indicates that the classifier has hardly classified correctly because of under fitting. The result is that the accuracy drops only because it depends on all features useful for prediction calculated during fitting. The addition of random feature has a similar effect as on GaussianNB namely no effect.

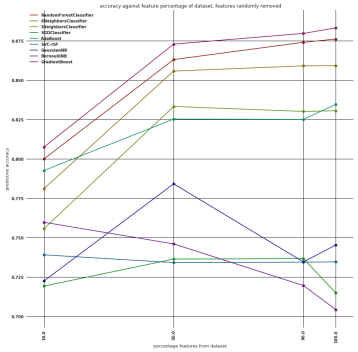
GradientBoostingClassifier The effect of duplicate features has a similar story to both AdaBoost and RandomForestClassifier in other words no effect. The decision trees handle the duplicate features as a bit less important than the original dataset(figure fig:FIGBC). This explains a less decrease in accuracy compared to the RandomForestClassifier, the classifier is less fitted to those added features and so more robust. For the random features the same is true, as it shows better performance over the RandomForestClassifier.

	duplicate features	redundant duplicates features	random features
RandomForestClassifier	0.0	-0.23	-0.04
KNeighborsClassifier	-0.0	-0.18	-0.11
1NeighborsClassifier	-0.0	-0.2	-0.12
SGDClassifier	0.01	-0.13	-0.04
AdaBoost	-0.0	-0.12	-0.01
SVC-rbf	-0.0	-0.14	-0.05
GaussianNB	-0.01	-0.3	0.0
BernoulliNB	-0.01	-0.11	-0.0
GradientBoost	-0.0	-0.16	-0.02

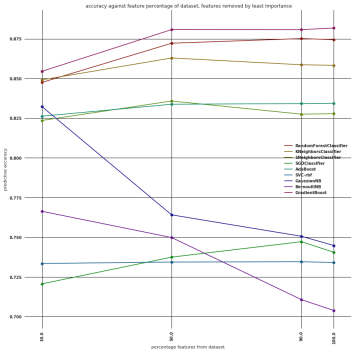
Table 1: all changes in predictive accuracy in adding both kind of features(results from figures 15 and 14)

4.2.2 Removing Features

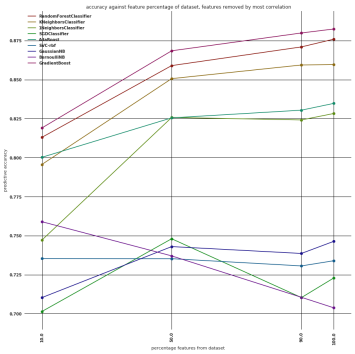
Randomly removing features, based on least importance or based on most correlation. The accuracy against the feature left is shown.



(a) Features randomly removed from a dataset.



(b) Features removed based on the least feature importance in the training set done by a RandomForestClassifier



(c) Removing the feature most correlated to another feature and also most correlated to all other features of the two.

Figure 16: Predictive accuracy of classifiers with feature removed based on certain preconceptions.

RandomForestClassifier The predictive accuracy of RandomForestClassifier has a normal behavior considering we are randomly removing features, the accuracy gradually declines. For the features removed by least importance the accuracy stays nearly the same there is little gain of removing the features, so RandomForestClassifier had no problem classifying before. Removing features based on most correlation has a similair effect as randomly removing.

KNeighborsClassifier The predictive accuracy of the kNN classifiers has a steady flow with the first 50 % features removed randomly as the accuracy stays nearly equal and the drop off in accuracy is from then on. For the features removed by least importance a similar pattern is visible but with a lite increase in accuracy for the default and a bump for the 1NN version. The decrease in accuracy is also less for the final drop off. This is mostly due to the nature of the kNN of distance between points, the RandomForestClassifier finds importance in the features and so these points distinguish for the classifier what will influence which target it will be. The correlation removing has a bad effect on the accuracy as it goes down hill similar to random features.

SGDClassifier The SGDClassifier has a strange behavior off accuracy as the highest accuracy is attained with either 10 % or 5 % randomly removed features. This is most likely the result of removing a random feature that fitted less in the linear regression or a split off the data that was better considering the accuracy only changed a percentage. The importance of features matter less for fitting a SGDClassifier as now the increase in accuracy noticeable at 50 % features randomly removed is gone. This means that the important features are not likely to fit an SGDClassifier. Removing by most correlation has more effect on the accuracy as with only 50 % of the features left the highest accuracy is measured, matching the previous highest of removing by least importance. The accuracy in that experiment has also more variance as there is no clear straight line of decline or decrease but a fluctuation of the result.

AdaBoost The accuracy of adaBoost remains rather robust on randomly removed features. With the focus on outliers the remaining dataset is a good example of the strength of AdaBoost even without configuration. Similair behavior is found with removing the least important features, the accuracy stays nearly the same overall. Removing the most correlated features gives a more gradual decline of each step losing some accuracy.

SVC-rbf The accuracy of the SVC-rbf classifier has a steady pattern over all the methods of removing features. The only noticable difference is even a slight increase in accuracy for the smallest set made by randomly removing. This means that the feature remaining still have a lot of information in them and that nearly no features were hindering the accuracy of SVC-rbf for theses datasets. The accuracy gain of most features in those dataset is nearly equal for a default SVC-rbf.

GaussianNB The accuracy of the GaussianNB classifier shows interseting behavior of fluctuating between improving or deteriorating for randomly removed features. This means that similair for SVC-rbf that most features have an similair impact for GaussianNB, when looking at them randomly. Only when looking at the feature removed by least importance there is a steady increase in accuracy by removing more features. This explains that there due exist more fitting features for the GaussianNB in these datasets. For the most correlated removed features a steady decline is seen. Which shows that those feature that are somewhat correlated may be the most important featueres in comparison for GaussianNB.

BernoulliNB The most interesting behavior of the accuracy can be seen with the BernoulliNB as with all feature removing methods give an increase in accuracy. Least feature importance removing only by a slight margin gaining the best result. This means that the binarization and fitting to the Bernoulli distribution can be best realized with only the most important features of a dataset. For this classifier most features are an obstacle for performance.

GradientBoostingClassifier GradientBoostingCLassifier shows a similair pattern as the RandomForestClassifier in all cases showing their similarity. The default accuracy is only slightly higher of the GradientBoostingClassifier compared to the RandomForestClassifier.

4.3 Noisy data

Datasets in these experiments get increasingly more random by injecting the features with noise. In figures 17, 18 and 19 the accuracy against the noise are displayed. The noise consists of either flipping categories for categorical data or adding a random amount of standard deviation.

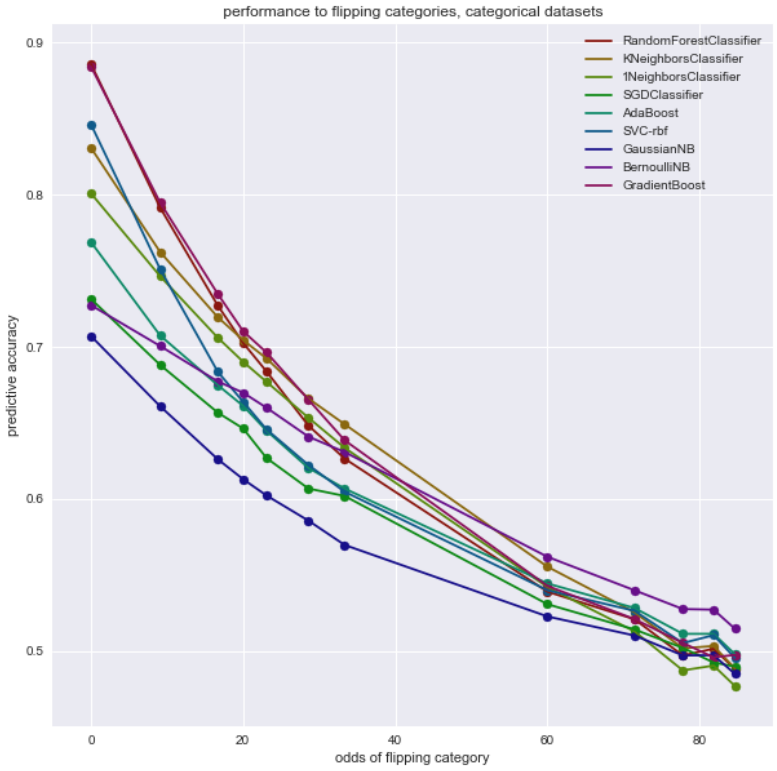


Figure 17: flipping categories for categorical features

RandomForestClassifier: The accuracy of RandomForestClassifier is one of the best without optimization or cleaning the data. The decline in predictive accuracy is however far greater. This is because of the nature of a decision tree. The decision tree has numerical boundaries or categorical boundaries made on the original data set to set apart the target classes. By adding the noise or flipping the categories these boundaries are blurred and the RandomForestClassifier has a higher chance to choose a different class, if in the original case it was straightforward. This makes the RandomForestClassifier not that robust against noisy data.

KNeighborsClassifier: KNeighborsClassifier has a decent initial predictive accuracy and it is only slightly increased by the preprocessing. The downward trend of the influence of the noise is one of the least decending. The default 5 neighbors or 1 neighbor also has only a slight influence on accuracy. For the numerical datasets the initial accuracy on the clean dataset is equal and only after the added noise a clear difference is noticeable which does not expand that much after 1 std.

SGDClassifier: SGDClassifier is not performing that well initially on either categorical or numerical datasets. The average accuracy on a clean dataset is one of the lowest of the 8 classifiers. In Robustness the SGDClassifier scores better. With the clean dataset the decline is only below KNeighborsClassifier and SVC. With the preprocessing however it beats the SVC in robustness and has a greatly improved initial accuracy.

AdaBoost: AdaBoost does not perform that well on the datasets without preprocessing. The classifier optimizes on edge cases and without the noisy instances in the training set it has a hard time optimizing on things it has not seen yet. The decline in accuracy is in this case not that bad and on par with SGD-

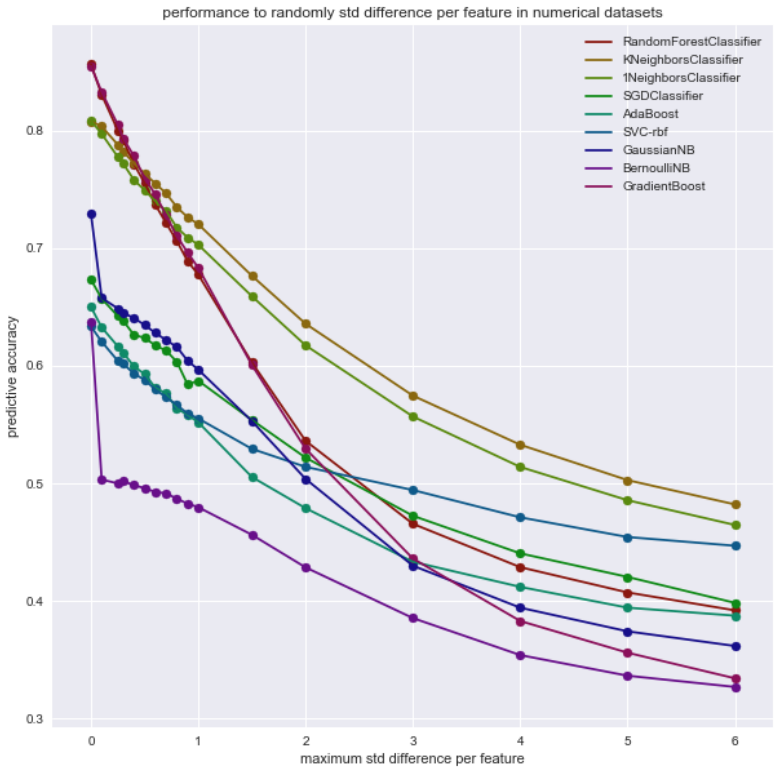


Figure 18: adding or removing uniformly random std to numerical features

Classifier. Considering that AdaBoost is not optimized and that the datasets are not all particularly relevant for AdaBoost the performance is too be expected.

SVC-rbf SVC-rbf has great predictive accuracy on the categorical dataset but not so much with the numerical dataset. The robustness of SVC-rbf with the numerical dataset is pretty strong as it ends up as one of the best classifiers. This does not translate well with more cleaned data. With the preprocessing steps SVC-rbf has a high predictive accuracy but the decline is far greater. This can be attributed to a sort of overfitting on the initial clean data as the difference in decline is easily noticeable between the two results.

GaussianNB: GaussianNB has the worst performance on the categorical datasets and a decent performance on the numerical datasets. The robustness is however pretty bad as there is a large initial drop when adding the noise for the numerical datasets. The decline there after is also very high which indicate a lack of robustness. This is largely due to the nature of the GaussianNB. It assumes a normal distribution with a mean and variance but a slight shift in the numbers changes a lot in the distribution of the datasets. In figure 38 the big drop can be noticed for a few datasets. A different trend can also be noticed of increased accuracy, those results are on unbalanced datasets.

BernoulliNB BernoulliNB has one of the worst predictive accuracy on both categorical and numerical datasets. The robustness is also really bad on the numerical datasets. The accuracy has a large drop when a little bit of noise is added and only a decrease equal to the drop for the remaining added noise. For the categorical dataset a better robustness can be observed as it beats all others from the point that the datasets are 60% noise. This can be explained by the conversion the classifier does with the input. For categorical data this means that it depends on multiple features for its prediction. For numerical features it means that a slight change in data from the training to the prediction means the transformation does not work anymore. For some of the numerical datasets there are still only a few unique values and those are used like the categorical features. This explains the large drop for the numerical datasets.

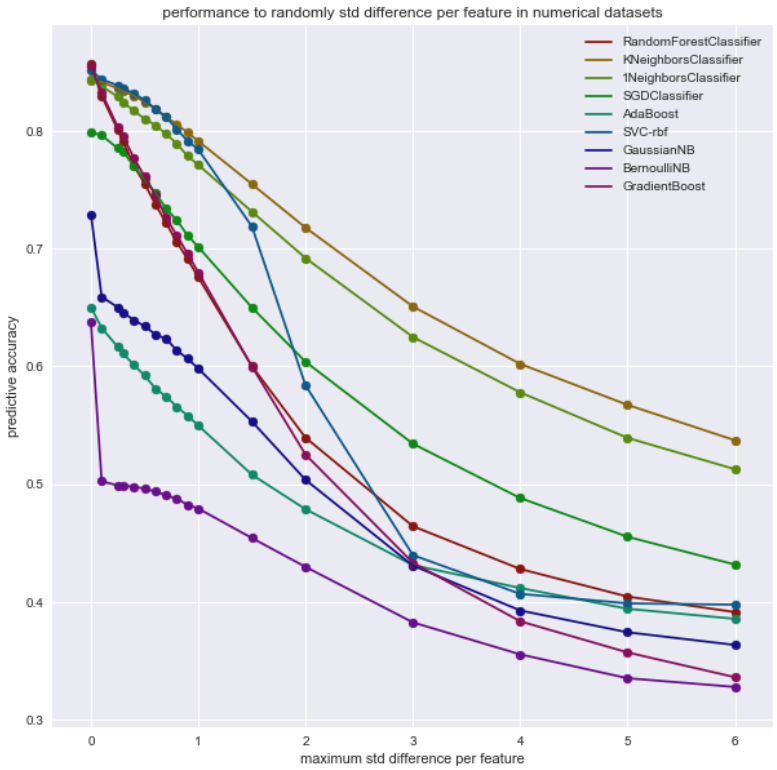


Figure 19: adding or removing uniformly random std to numerical features and preprocessing for some classifiers.

GradientBoostingClassifier GradientBoostingClassifier has one of the best predictive accuracy on both sorts of datasets. Performing near equal to RandomForestClassifier. In robustness it loses to RandomForestClassifier on numerical datasets and there is only a slight increase over the results on categorical datasets. The results of GradientBoostingClassifier copy those of RandomForestClassifier as they both use Decision trees for their ensembles. The lack of robustness on numerical datasets can be regarded as over fitting compared to the RandomForestClassifier. The GBC uses more time to fit the classifiers for an additive model rather than pure random voting.

4.4 Bias variance

In figure 20 the bias percentage is given against the instance percentage of a dataset. The averaged dataset has a number of instance of 3186 and a number of features of 160.

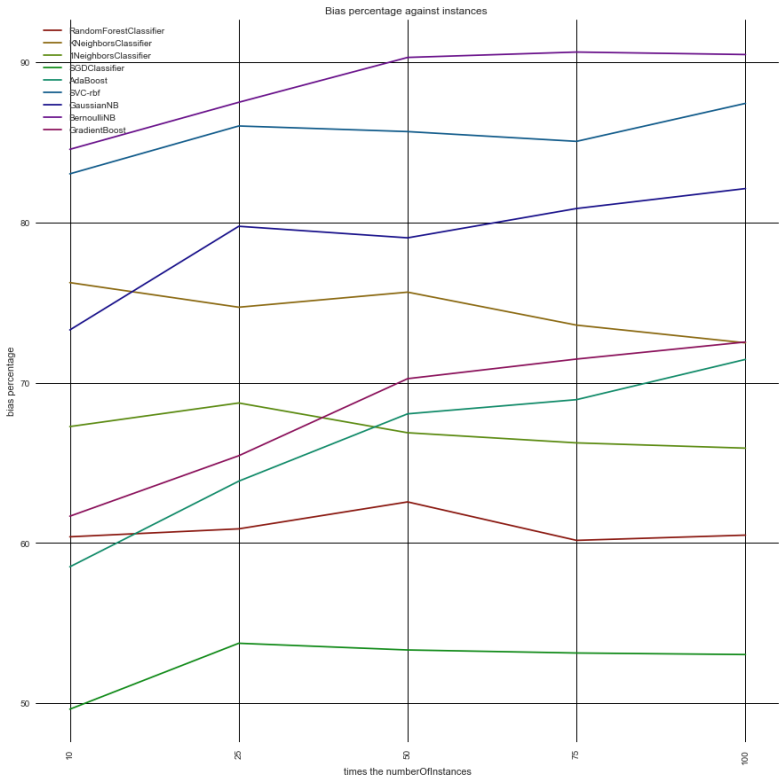


Figure 20: Bias percentage against percentage of instances from a dataset.

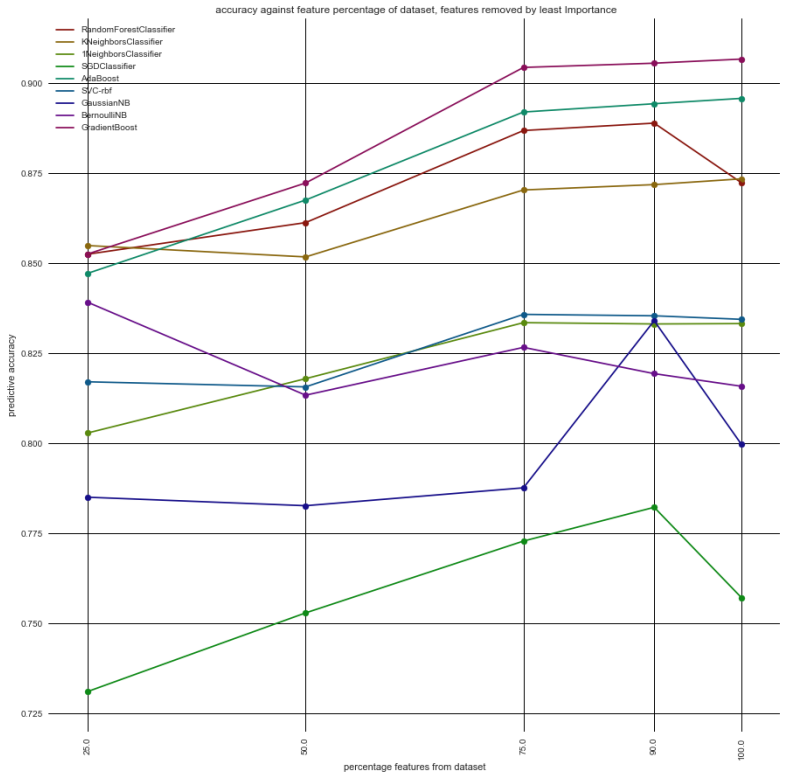


Figure 21: Bias accuracy against percentage of instances from a dataset.

4.5 Ontology

To present the results from all the experiments, this ranks the different classifiers 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 classifiers on different scales. The scales are:

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

Table 2: Comparison of Elements in Air on the Space Station and sea level on Earth

	dur Feat	dur Inst	robust Noise	robust Rand
RandomForestClassifier	4	4	5	5
KNeighborsClassifier	5	5	1	8
SGDClassifier	1	3	2	6
AdaBoost	6	6	3	3
SVC-rbf	7	8	4	7
GaussianNB	3	2	7	1
BernoulliNB	2	1	8	2
GradientBoostingClassifier	8	7	6	4

	dur Feat	dur Inst	Robust Noise	Robust	Robust	pre-	Bias
			Noise	Rand	ReFeat	process	Manage
RandomForestClassifier	4	4	5	5	7		+/-
KNeighborsClassifier	5	5	1	8	6	+	+/-
SGDClassifier	1	3	2	6	4	+	-
AdaBoost	6	6	3	3	3		+
SVC-rbf	7	8	4	7	5	+	++
GaussianNB	3	2	7	1	8	+	++
BernoulliNB	2	1	8	2	1	+	++
GradientBoostingClassifier	8	7	6	4	2	+	+

Figure 22: Comparison between the different algorithms.

RandomForestClassifier The strenght of RandomForestClassifier is the easy and quick solution, solid solution. With a high un proccessed predictive accuracy, quick estimates of predictive accuracy on a dataset can be achieved. The classifier does lack some robustness against features and noisy data. For a more supervised approach the feature can be more preproccessed for better results. Lastly some bias managements is needed as a large part of the error is due to bias error on average.

KNeighborsClassifier The strenght of kNN is in the robustness against noisy numerical features. With a dependence on numerical relations between instances a shift in that dimension is not the downfall. Categorical values and random features are a harder issue. With a robustness againt noisy data, random featuers may be non obvious to be not robust against. The problem with the random data is the scale of these features in comparison to the other features and the difference this can make. If the random features have larger value differences than the important features, those features get rained under.

SGDClassifier The strenght of the SGDClassifier is in the quickness and the robustness against noise. With a default dependence on linear SVC the noise is only variance in the fit.

AdaBoost The strenght of adaBoost is the average robustness, AdaBoost has not the most resilience against the introduced noise and irrelevance. The strenght lies more in fitting to outliers found in a training set. With more redundant features, outliers remain to exist
SVC-rbf
GaussianNB
BernoulliNB
GradientBoostingClassifier

5 Conclusion and discussion

5.1 Discussion

5.1.1 Missed opportunities

Within saving as many information obtainable from the algorithms a missed opportunity is the probability prediction of all the test sets.
Another missed collected data is the prediction of the training data. This data might not give a good indication of predictive accuracy but can give some valuable information of overfitting on the input data. This can be calculated as a difference in prediction results of the training and test set. For some classifiers this holds more than others. For example SGDClassifier which fits the inputted data as differentiable functions. Opposed to KNeighborsClassifier which will look up all the inputted data for neighbors which will match the training data for prediction.

5.1.2 Duration

For the duration pictures some results are removed as they do not fall in line with the other results at all. By using the standard of the clean dataset which is calculated with each run we throw out any results changing more than 5% from another average. The reason for these differences in duration is because of multiple used systems and other processes running during computation.

5.1.3 Bias Variance

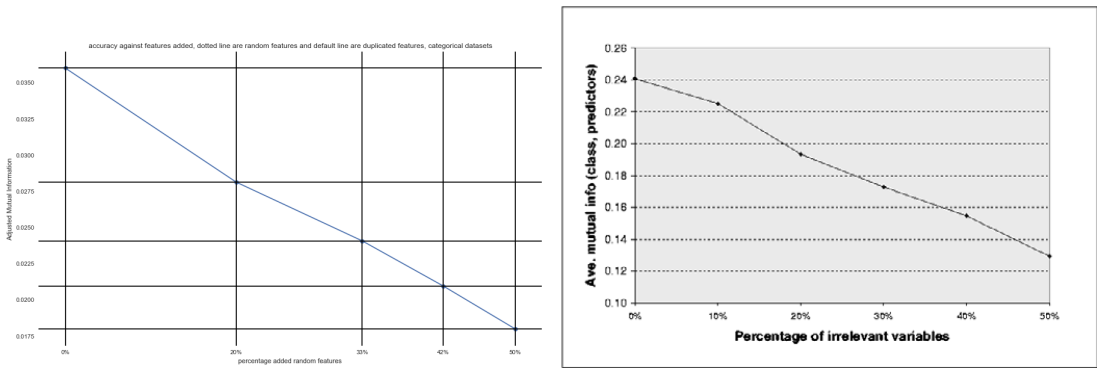
The datasets used by Jaoquin in [20] were UCI datasets, some of these are also published on the Openml under the dataset ids (1370-1377)

5.2 Resilience to noise

5.2.1 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 similar 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.

5.3 replication of previous work



(a) adjusted Mean Mutual Information for added random features.

(b) average Mutual Information as a measure of relevance

Figure 23: Mutual information decline between previous work and this work

In figure 23 the experiment setup of Hilario and the experiment setup of this thesis are shown[6]. For Hilario it was to show that with the added random features the mutual information decreases an indicator that the added features could not improve the accuracy. In the experiment of this thesis the added features also show a decrease in mutual information but the mean mutual information scale is 10 times smaller. This indicates that the datasets used are less valuable in contrast to the ones used by Hilario. The change of x-scale compared to previous experiments is to able to compare it more to the work of Hilario. Comparing the results is harder as the versions of the classifiers have been changed over time.

5.4 Conclusion

As recommendation the RandomForestClassifier is a strong pick for most problems with little tuning. Within a sort amount of time a strong result can be noticed. A direct improvement when robustness is needed, the GradientBoostingClassifier scores equal to the RandomForestClassifier with more robustness for adding features. If a dataset is more likely to get noise in the features GradientBoostingClassifier is a worse choice. The most robustness against adding irrelevant features is found in the BernoulliNB while also being one of the fastest classification and prediction. The downsize of BernoulliNB is that the input must be close to the Bernoulli distribution to achieve any competing accuracy. For noise in numerical features kNN is the best option and with some preprocessing the classifier outshines all others.

5.5 Future work

Adding more experimental runs to this topic can further solidify the results, break the result or alter only slightly the results. As benchmark datasets have a hard time to be a universal standard for testing. The results of a golden standard benchmark may also limit results because of the no free lunch theorem A source of attention can be the use of that only datasets available on Openml are used. Openml has a certain community and the datasets available on Openml may not translate to other areas of interest. Another party to consider then is a platform like Kaggle which share a lot of datasets already but the commercial side to Kaggle may be more of a counterpart to Openml.
A partly solution for the previous mentioned problem of datasets is the splitting of datasets in certain categories. In this thesis there was a split for categorical and numerical features possible by the classification in Openml of all inputted features. Different splitting of dataset can be done in categories like sparse/dense or cleaned and uncleaned. Further distinguishing in categorical and numerical features the different values of discrete, continuous or ordinal values. Each can need different preprocessing steps for different classifiers.

6 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] Investigating the performance of Naive-bayes classifiers and k-nearest neighbor classifiers, M. J. Islam, Q. M. J. Wu, M. Ahmadi and M. A. Sid-Ahmed, International conference on convergence information technology(ICCIT) Gyeongju pages 1541-1546,(2007)

[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)

[26] A Weighted Nearest Neighbor Algorithm for Learning with symbolic features, S. Cost, S. Salzberg, Machine Learning Volume 10, number 1 pages 57-78, (1993)

[27] A practical guide to support vector classification, Chih-wei Hsu, Chich-Chung Chang, Chich-Jen Lin, 101. 1396-1400, (2003)

[28] A study of cross-validation and bootstrap for accuracy estimation and model selection, R. Kohavi, (1995)

[29] Nearest neighbor pattern classification, T. Coverm P. Hart, IEEE Transaction on Information Theory 13:21-27,(1967)

7 Appendix

7.1 datasets

7.1.1 datasets per figure

openml dataset identifiers per figure
figure 26 - 10,12,18
figure 30 - 1038,1043,1049,1050,1176,12,1466,1468,1475,1476,1478,1479,1485,1487,1491,1492,1493,1494,14,1501,1504,1510,
figure 11 - 22, 1176, 4134, 1063, 1067, 1068, 44, 53, 4538, 458
figure 10b - 12, 1038, 14, 16, 1043, 22, 1176, 1049, 1050, 28, 1063, 1067, 1068, 300, 1466, 1467, 1468, 1475, 1476
figure 10a - 1038, 1043, 22, 1176, 1049, 1050, 1063, 1067, 1068
figure 15 - 11, 12, 1038,14, 16, 18, 1043, 1046, 22,1176, 1049, 1050, 28,30, 32, 1570, 36, 37, 4134, 1063, 39, 40, 1067, 1068, 300, 41, 44, 1459, 40499 , 53, 1462, 1464, 54, 1466, 1467, 1468, 40509, 4538, 1475, 1476, 1478, 1479, 458,1485, 1487, 1489, 1491, 1492, 1493, 1494, 1497, 1501, 1504, 1510, 1515, 375
figure 14 - 20, 21, 26, 333, 334, 335, 40668, 4135, 4534, 469, 46, 50
figure 16 - 1038, 1043, 22, 1176, 1049, 1050, 1063, 1067, 1068
figure 13 - 6, 1036, 12, 1038, 14, 16, 18, 1043, 20, 21, 1046, 22, 23, 1049, 1050, 26, 28, 30, 32, 1570, 36, 4134, 4135, 1067, 1068, 44, 46, 40499, 60, 1120, 151, 1176, 182, 184, 40645, 40646, 40647, 40648, 40649, 40650, 40651, 40652, 40653, 40654, 40655, 40656, 40664, 40668, 40670, 40677, 40678, 40680, 40687, 40695, 40696, 40697, 40702, 40706, 300, 179000, 312, 375, 1459, 1461, 1462, 4534, 953, 1466, 4538, 1468, 959, 1475, 1476, 1478, 1479, 1485, 1486, 1487, 1489, 1491, 1492, 1493, 1494, 1497, 1501, 991, 1504, 1505, 1022
figures 21,20 - 1025, 1067, 1036, 1068, 1038, 1043, 1014, 1049, 1050, 1022

7.1.2 Distribution

The number of features and instances for datasets

7.2 different approach

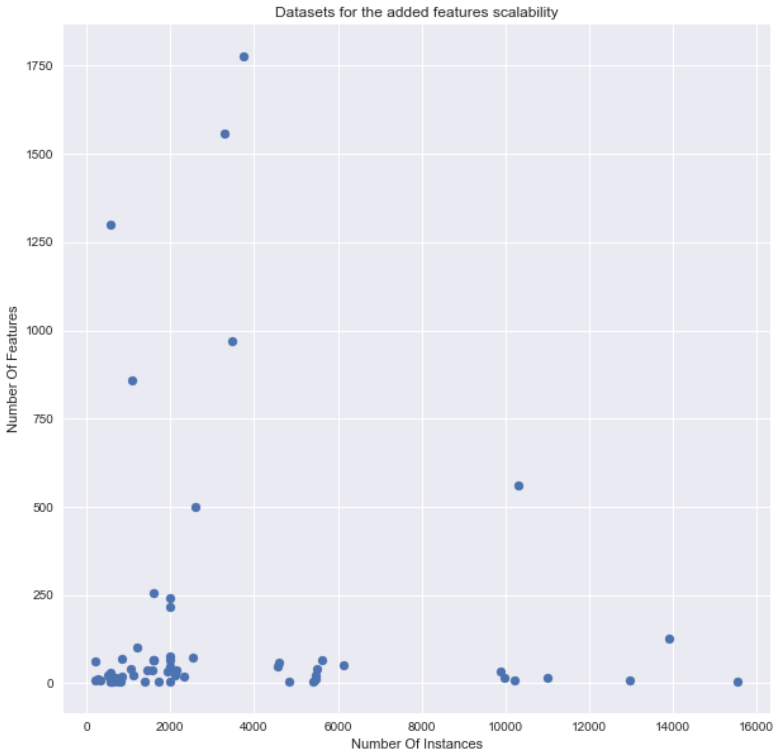


Figure 24: Dataset metafeatures for figure 7

7.3 Duration variance

In this section duration examples are shown which show behaviour that should not happen.

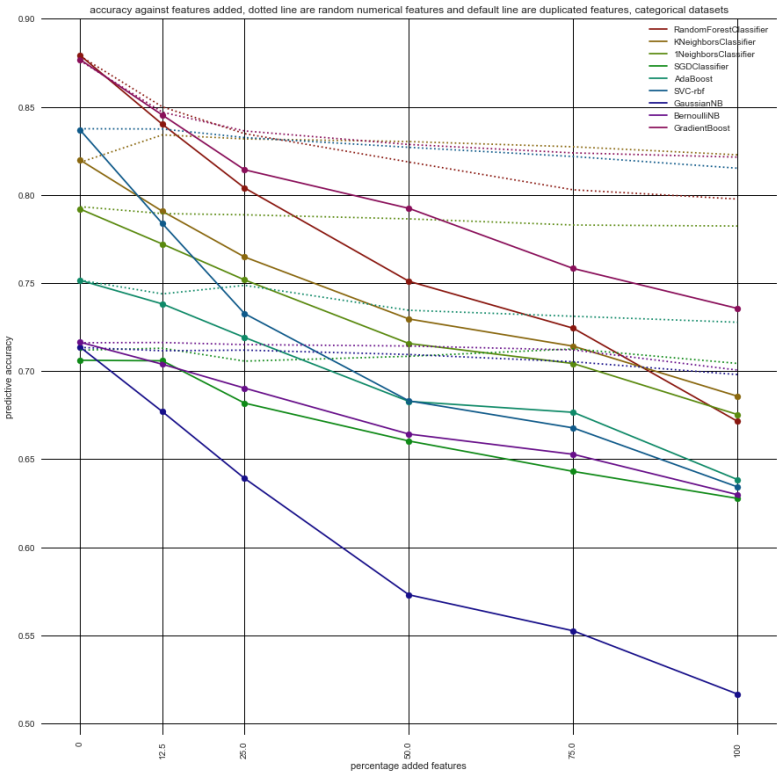


Figure 25: adding or removing uniformly random std to numerical features for GaussianNB.

7.4 Results per dataset per classifier

results for individual datasets grouped per classifier instead of averaged.

7.4.1 Noisy data

injection of std into numerical datasets figure 18.

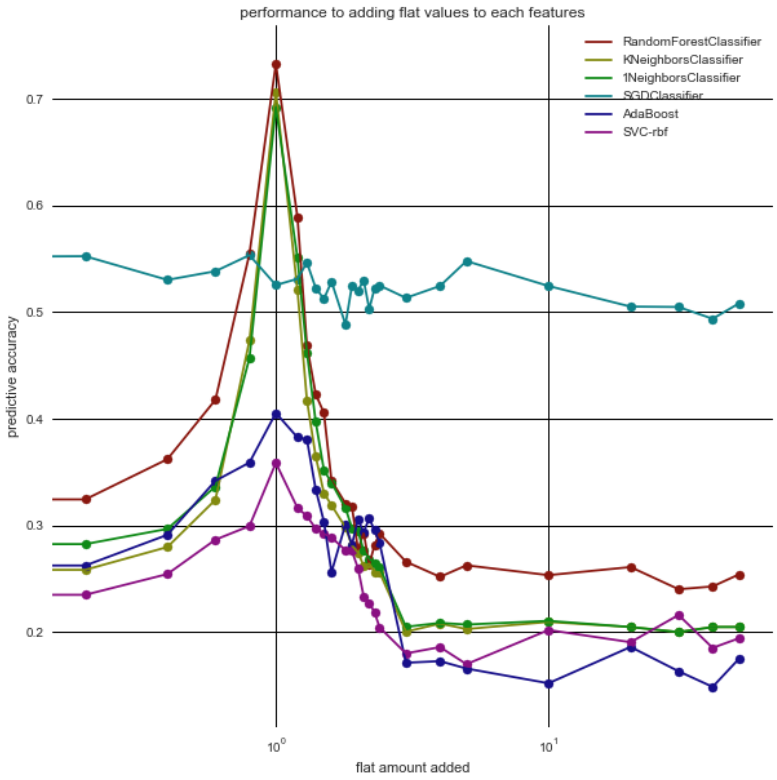


Figure 26: adding flat values to each feature

7.4.2 Instance duration

Percentage of instances remaing of a dataset duration per individual dataset reference to figure 13

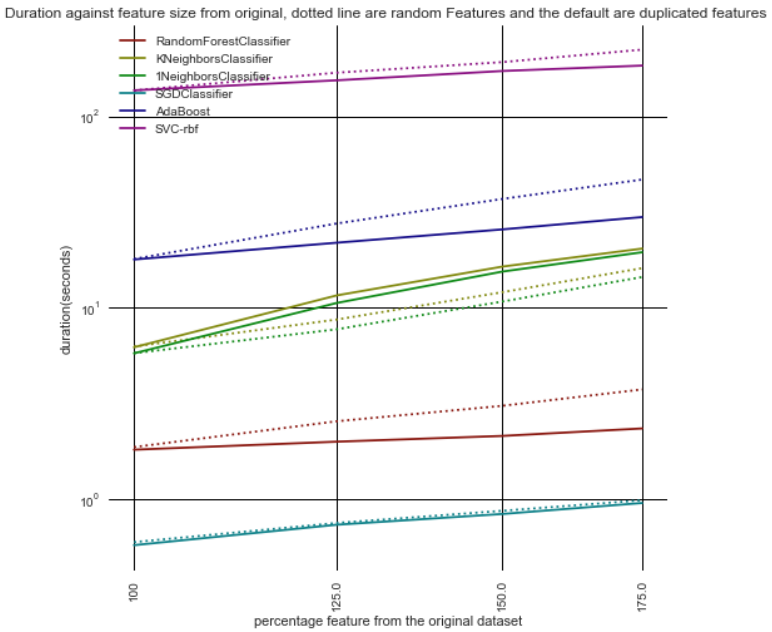


Figure 27: Duration of the oldest version of adding random features

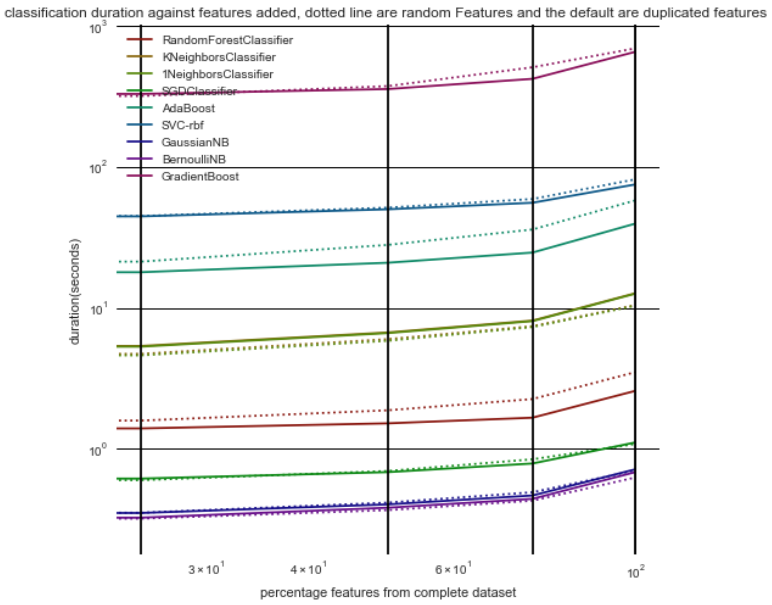


Figure 28: Duration of the oldest version of adding random features

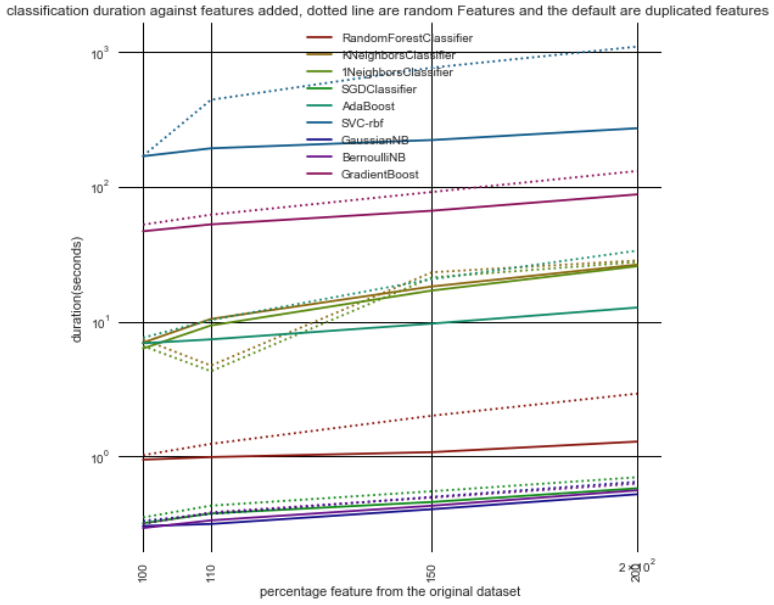


Figure 29: Duration of the newest version of adding random features

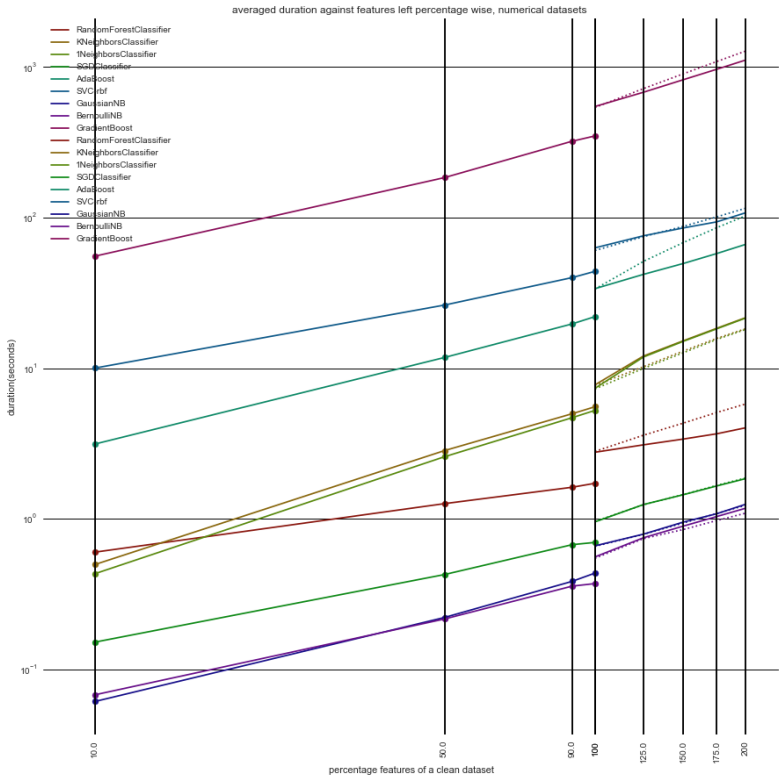


Figure 30: Duration of adding and removing features combined result

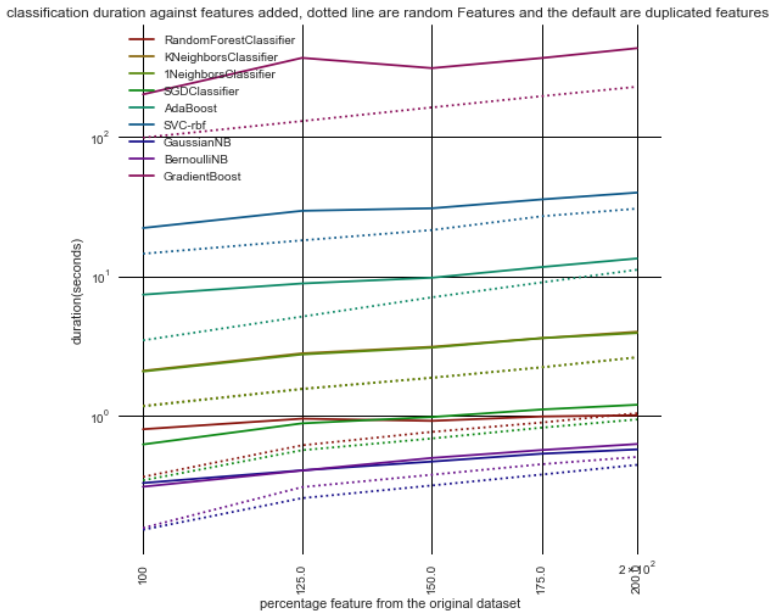


Figure 31: Duration of duplicated features against random features with clear spikes in duration registering

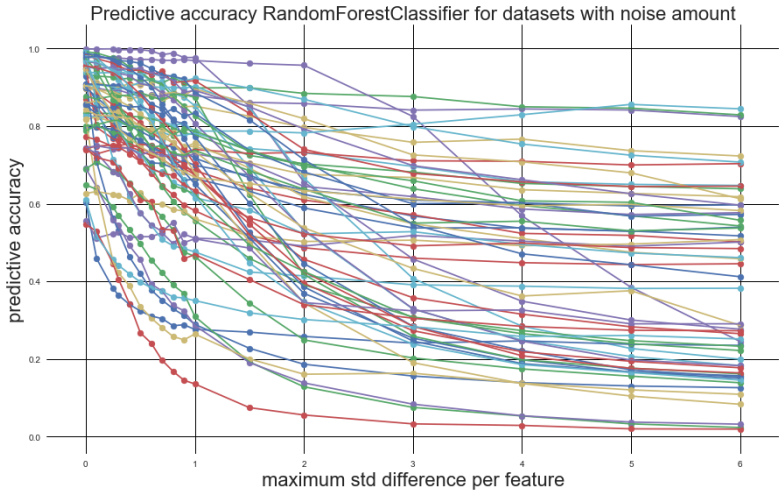


Figure 32: adding or removing uniformly random std to numerical features for RandomForestClassifier.

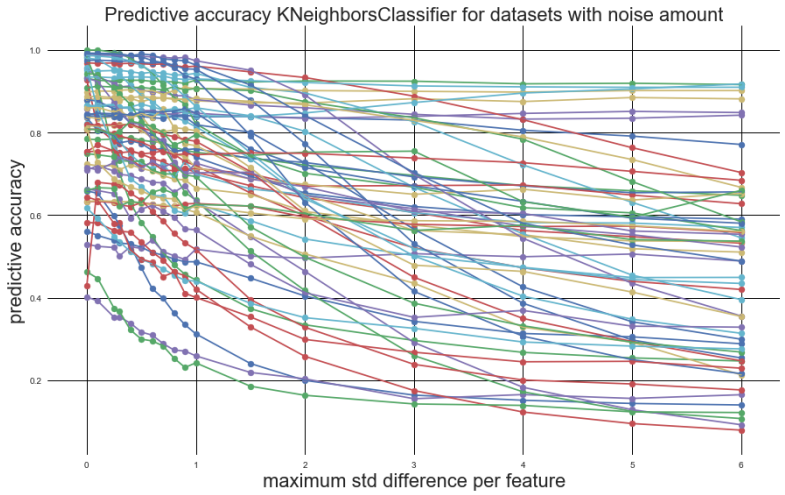


Figure 33: adding or removing uniformly random std to numerical features for KNeighborsClassifier.

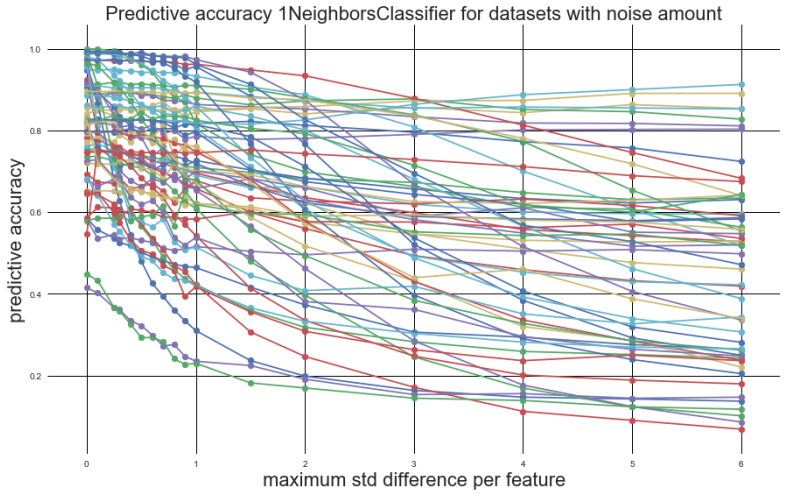


Figure 34: adding or removing uniformly random std to numerical features for 1NeighborClassifier.

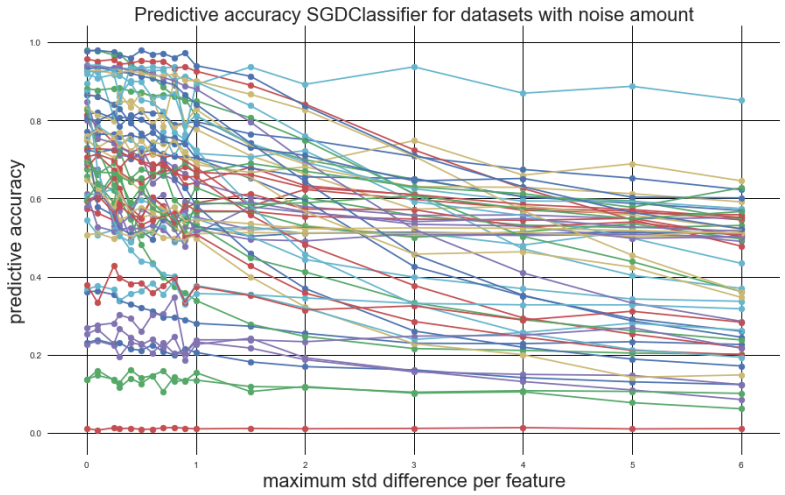


Figure 35: adding or removing uniformly random std to numerical features for SGDClassifier.

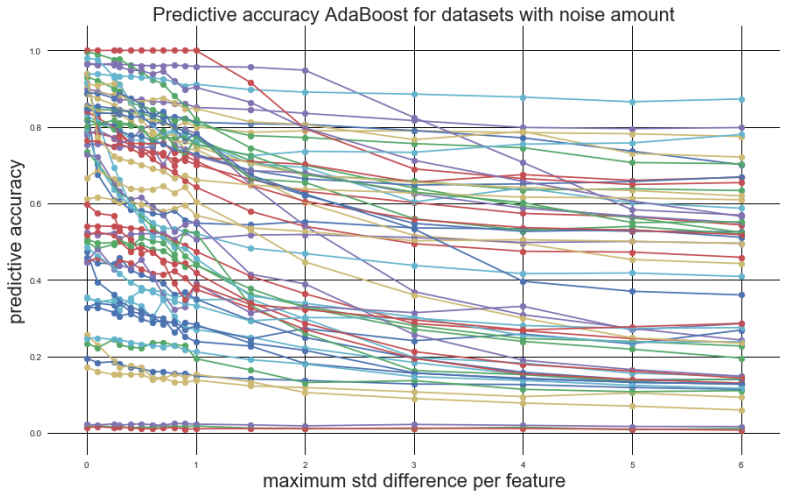


Figure 36: adding or removing uniformly random std to numerical features for AdaBoost.

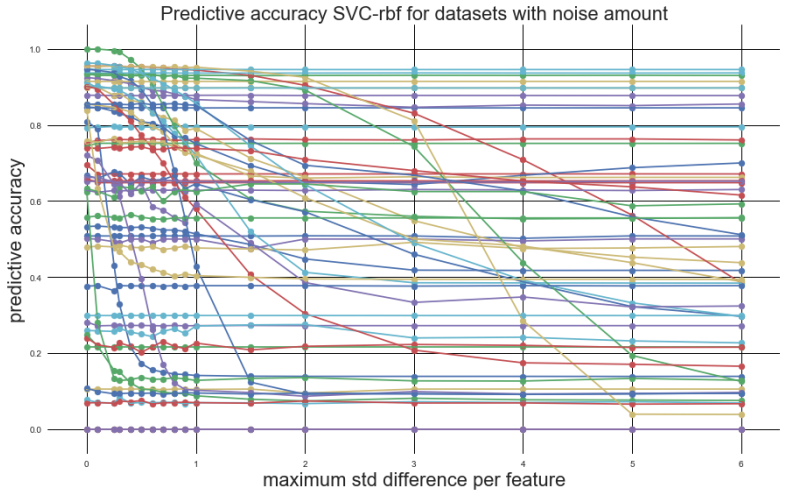


Figure 37: adding or removing uniformly random std to numerical features for SVC-rbf.

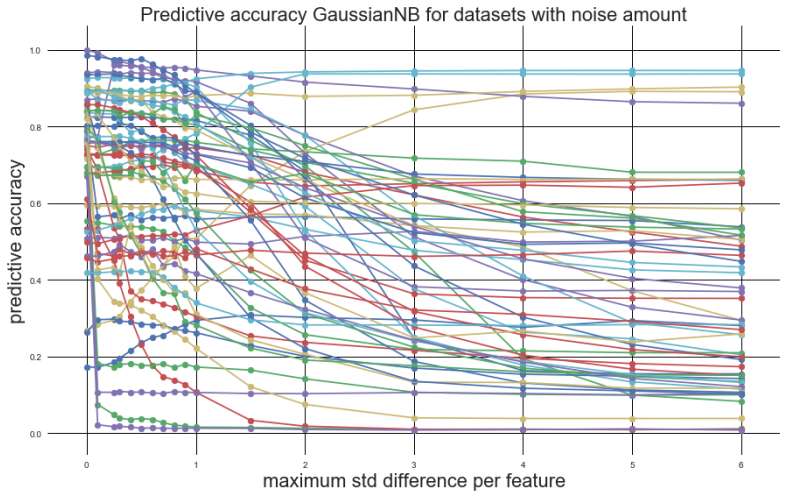


Figure 38: adding or removing uniformly random std to numerical features for GaussianNB.

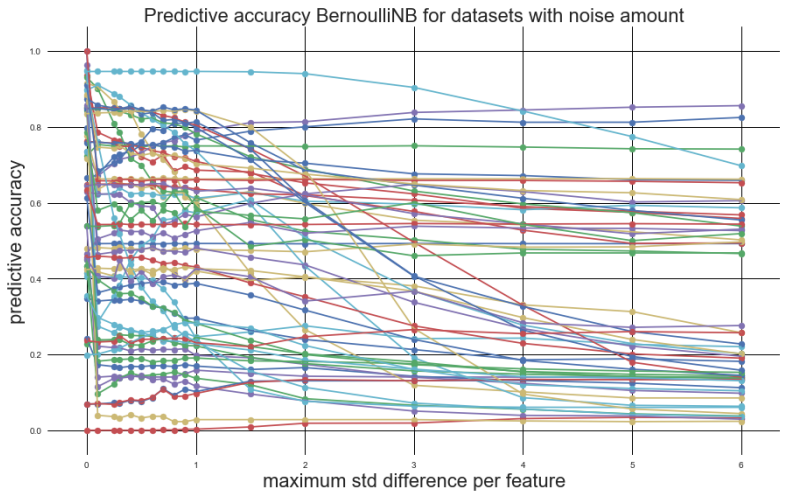


Figure 39: adding or removing uniformly random std to numerical features for BernoulliNB.

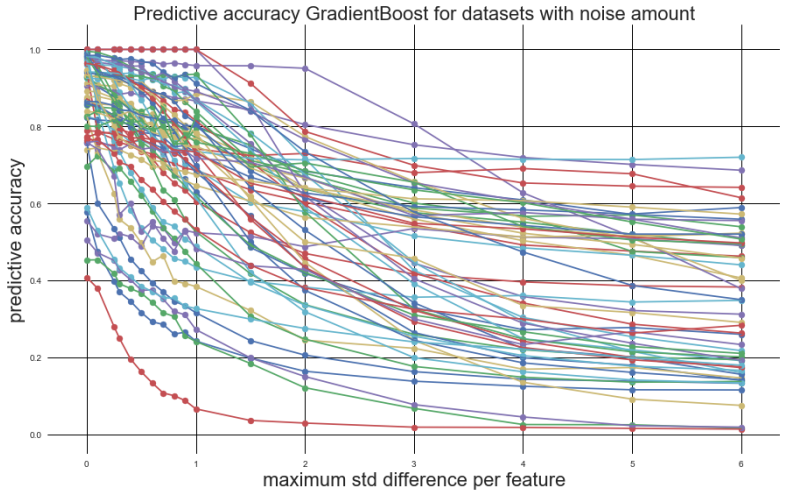


Figure 40: adding or removing uniformly random std to numerical features for GradientBoostingAlgorithm.

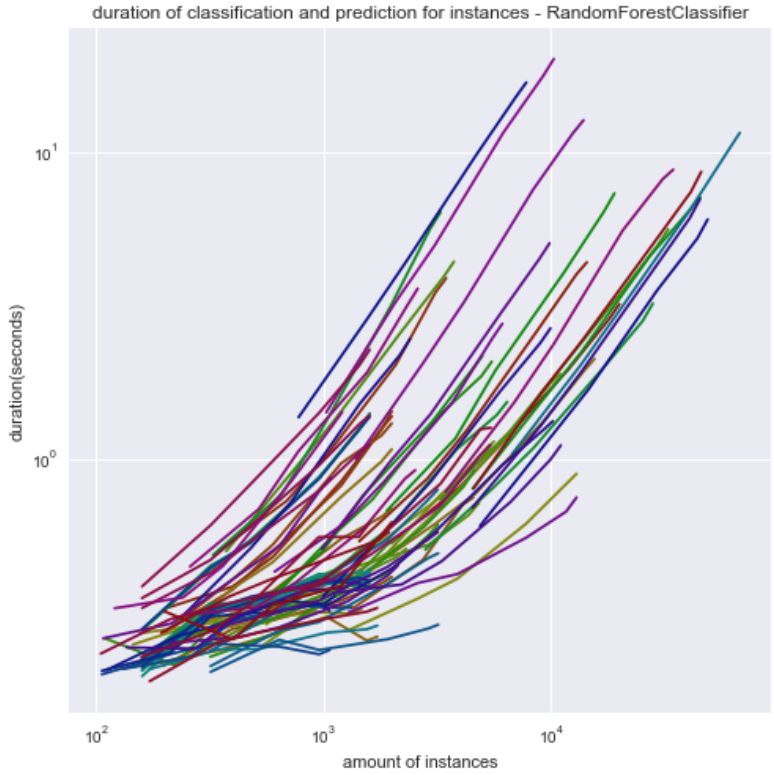


Figure 41: number of instances against duration for RandomForestClassifier

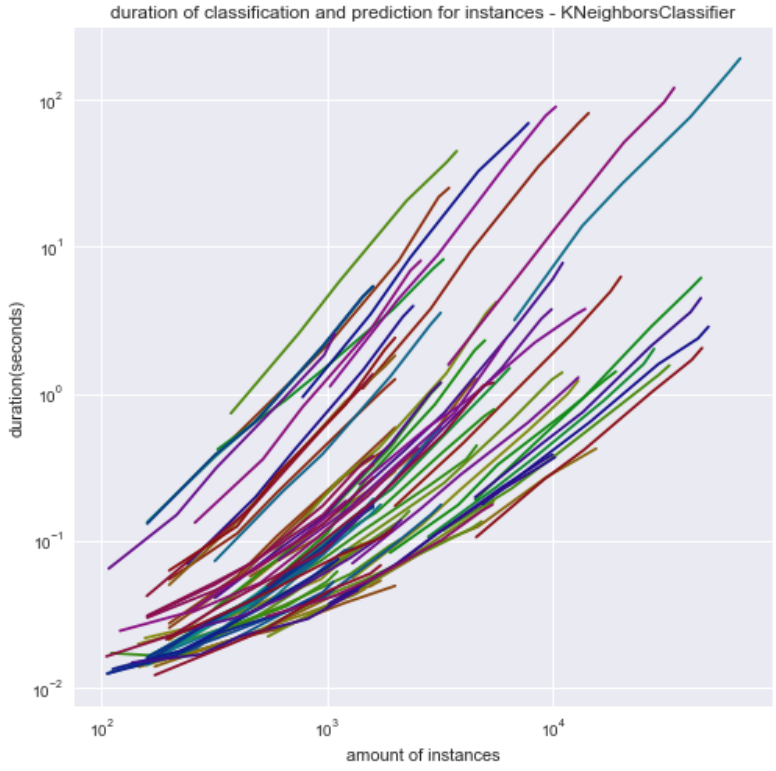


Figure 42: number of instances against duration for KNeighborsClassifier

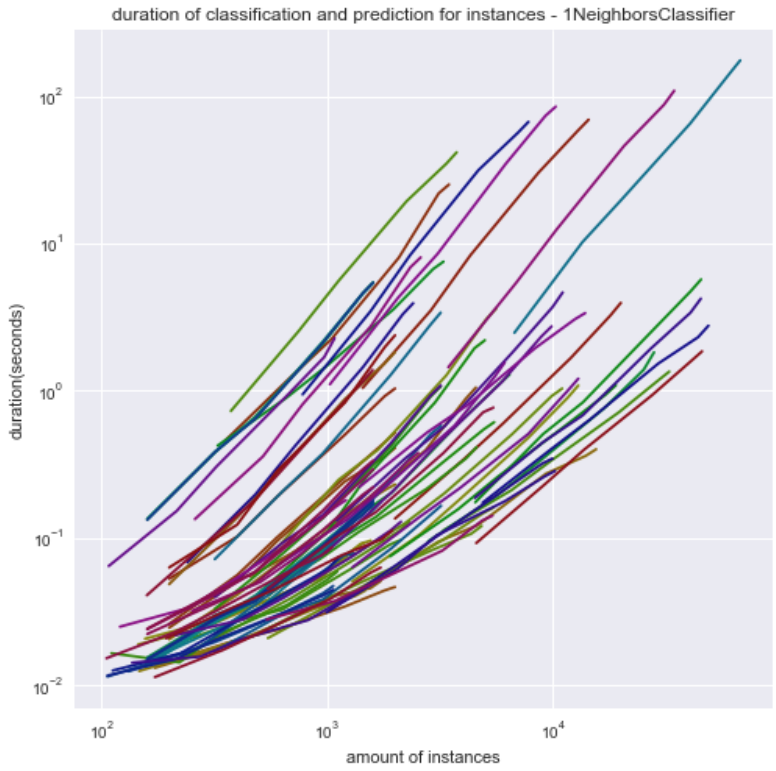


Figure 43: number of instances against duration for 1NeighborsClassifier

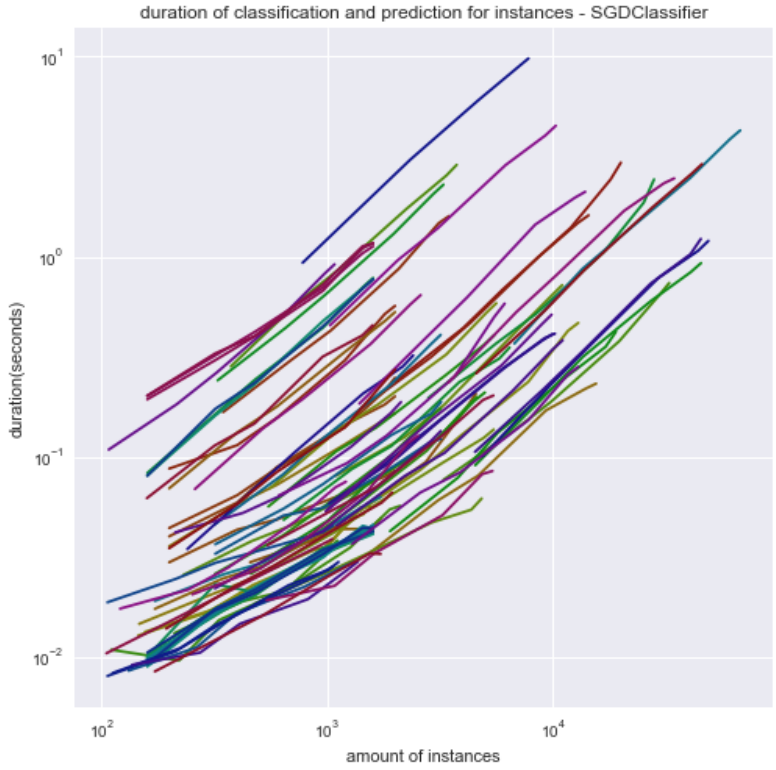


Figure 44: number of instances against duration for SGDClassifier

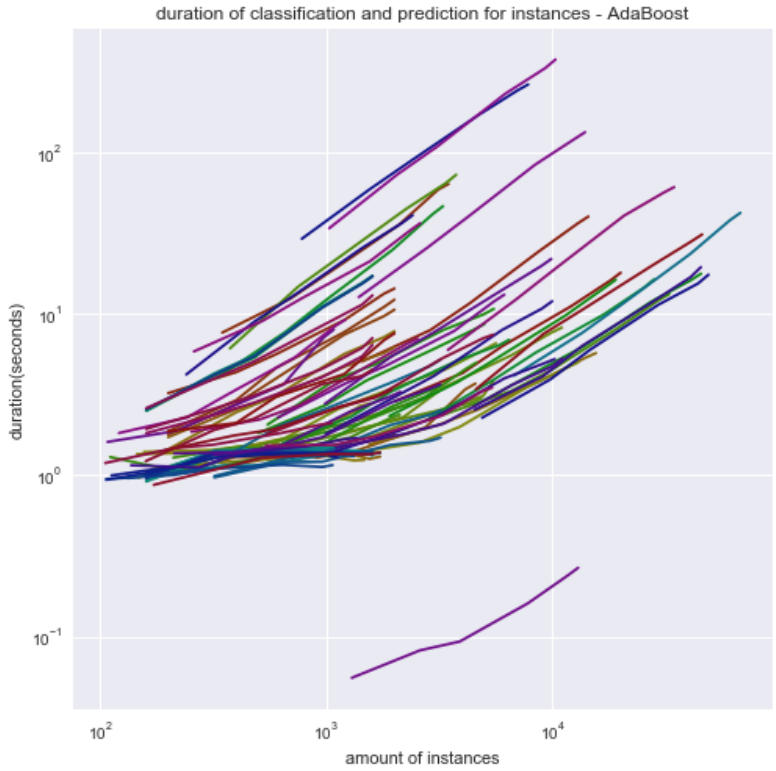


Figure 45: number of instances against duration for AdaBoostingClassifier

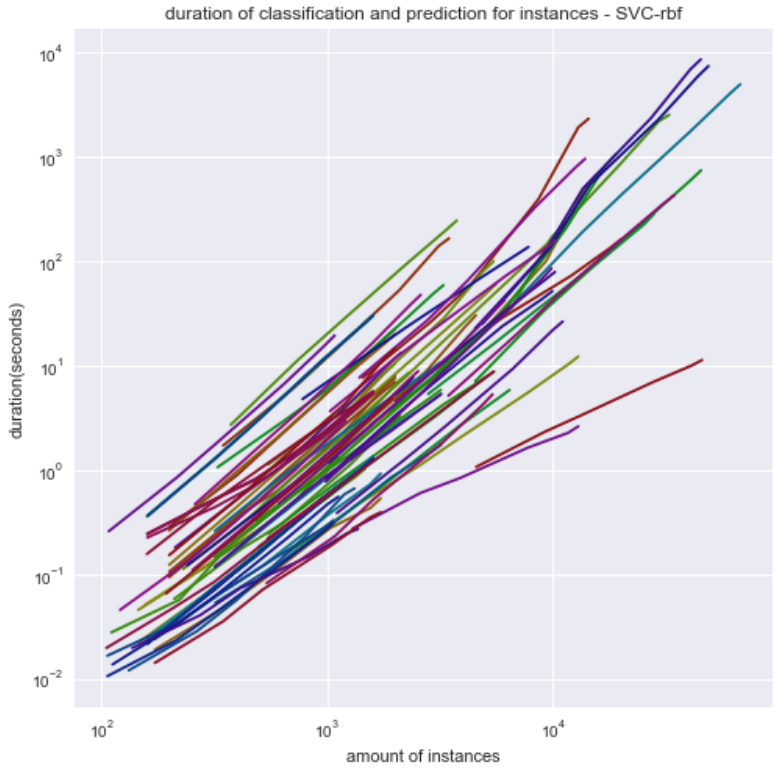


Figure 46: number of instances against duration for SVC-rbf

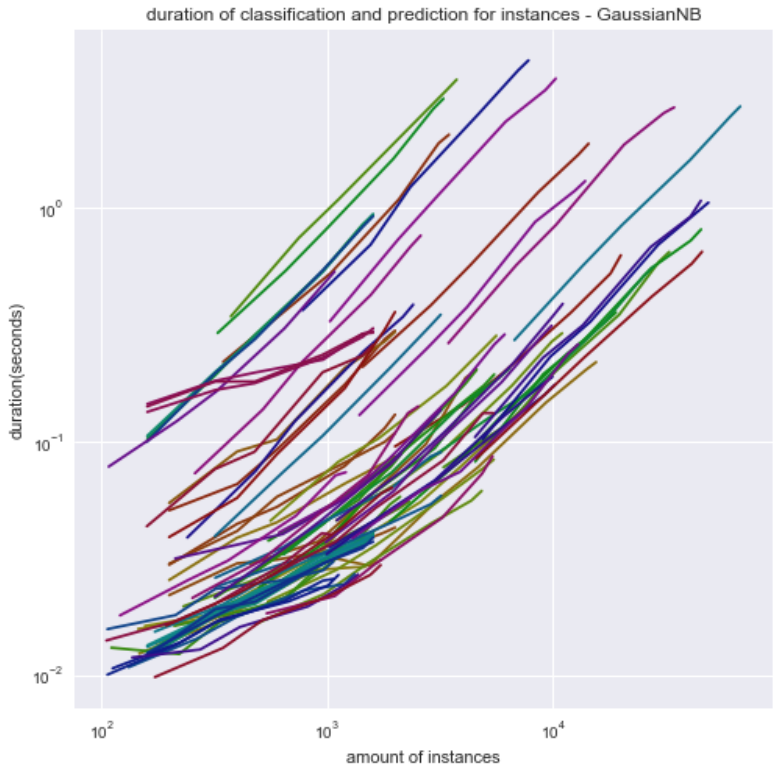


Figure 47: number of instances against duration for GaussianNB

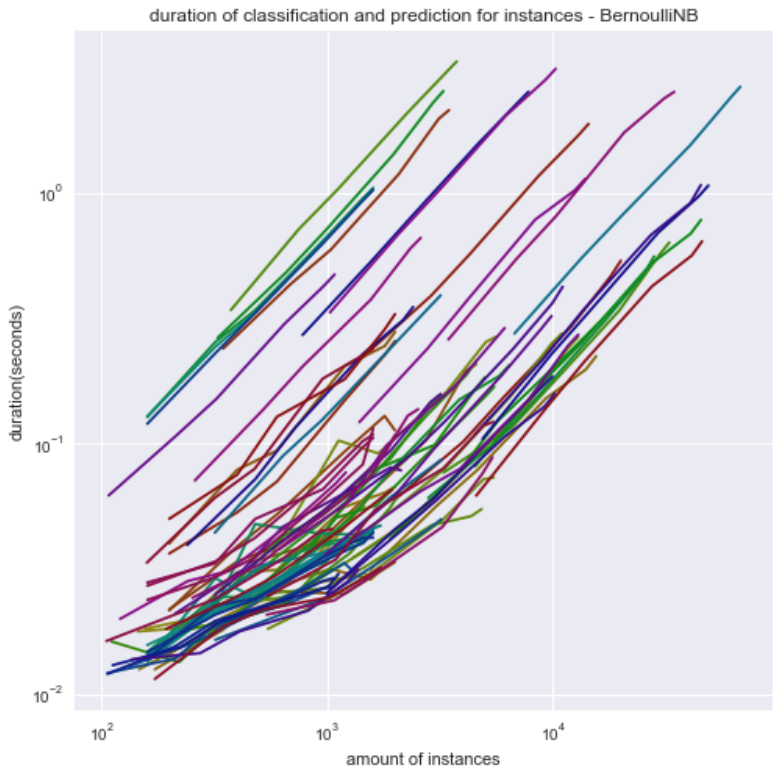


Figure 48: number of instances against duration for BernoulliNB

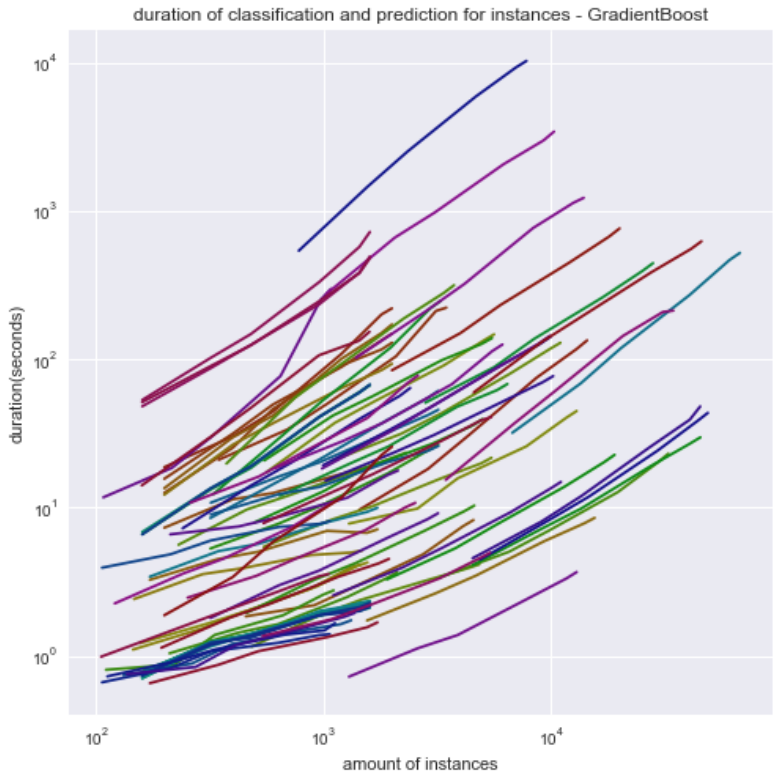


Figure 49: number of instances against duration for GradientBoostingClassifier