Algorithm primitives

Master Thesis

L.D. Stooker, 0819041

Supervisor: Joaquin Vanschoren
May 15, 2018

Abstract

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

Contents

1 Introduction

This report is the result of my graduation project which completes my Business Information Systems study at Eindhoven University of Technology. The project was performed internally at the 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 algorithms in libraries like scikit-learn[**?**]. In recent years python is also increasing in so called market share for machine learning[2]. To help people choose machine learners in the scikit-learn library a model was made to indicate what algorithm to use for what problem. In figure **??** you can see that depending on size of the data and early results, different algorithms are recommend. This gives a quick overview of available algorithms and when to pick which. This figure however is outdated as new algorithms are added to the library over time. Such a model is based on the concept of no free lunch which explains that there cannot be one algorithm to solve all optimization problems[25].

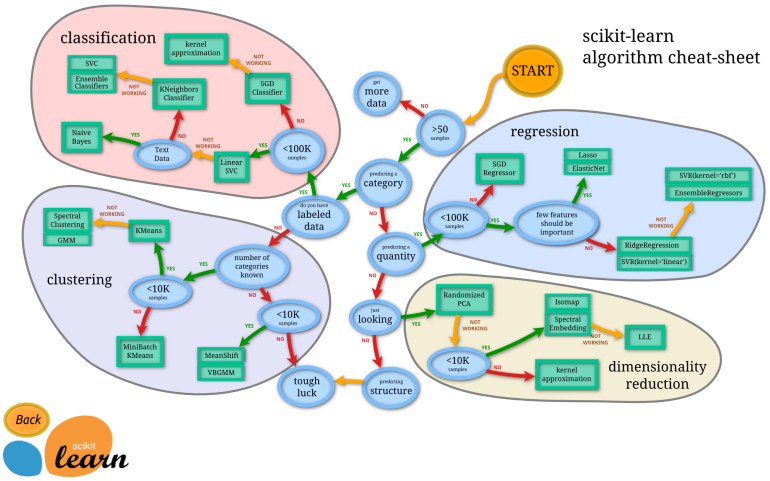


Figure 1: FlowChartML scikit-learn for any dataset which machine learning algortihm to choose

1.2 research question

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

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

The question for scalability is how varying sample size and feature space influences the runtime. Is there a measure to match this.
We rank the algorithms on their performance(predictive accuracy), 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 libaries hosting machine learning algorithsm in python but scikit-learn was the first to make a standard guideline which makes the algorithm easier to interchange. By having default functions for fitting and 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 averages the results for the result. These sub-samples are chosen randomly so the results can vary between runs on the same input. A directive tree is a decision tree classifier which splits the features on certain thresholds to decide on the type of class. This splitting of the data is either randomly or choosing the best split, to measure this split a criterion is used like Gini or entropy. The amount of splits, features and samples are also considered and can be inputted. 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 SDGClassifier

SDGClassifier 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 continious features with a mean of zero in a sparse setting. This makes the classifier sensitive to categorical data as it performs optimally with continious 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 maxima. 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 algorithm with a forward stage wise additive step. In this step a continual calculation of a loss function will output the prediction and in a two class case it reduces to the initial solution.

2.1.5 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 gainst 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 algorithms. The only parameter of this classifier specifies the prior probabilities of the classes, which will when specified not be adjusted to the given input.

2.1.7 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	The process with a specified in and output that solves a problem
amount	a value that indicates the change of the feature set either in dimension or in value. For
annotations	Adjectives of something like a machine learning algorithms, examples can be robust or biased
categorical	property of the features distribution and content of the feature in this case meaning seperate distinc classes, which have no numerical meaning, a unique number for each distinguished class
class	categorical featueres consist of at least 2 classes
datapoint	a datapoint is a single value of a feature. For example an instance has for all feature of a dataset a single datapoint.
dataset	a dataset are values in a matrix format, where each row represents a single Instance and each collum represent all the values of a feature.
dataset manipulation	like the word suggested is the manipulation of 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.
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 algorithm for prediction
Github	an online platfrom to host data. It uses git commands and is mostly used with programming project to organize a common project which each member can locally alter and centeralizly share updates or modifications.
machine learning algorithm	An algorithm that will will learn something and may adopt to the input to better fit the learned instance. To goal of learning can be mostly to predict a Target value, this can be part of an initial input.
numerical	exact values, more uniquely than a category. For example a temperature value or time value. Such a value can be subtracted or divided
profiling	To sketch information of something in a category, so you can relate it to other things in the same category
robustness	The ability of an algorithm to cope with changes in features. An algorithm is more robust if it deteriorates less than another.
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 algorithm
biasVar	bias and variance
did	dataset identifier
SDG	Stochastic gradient descent
std	standard deviation
TU/e	Eindhoven University of Technology
SVC-rbf	Support vector classification with a radial based function kernel

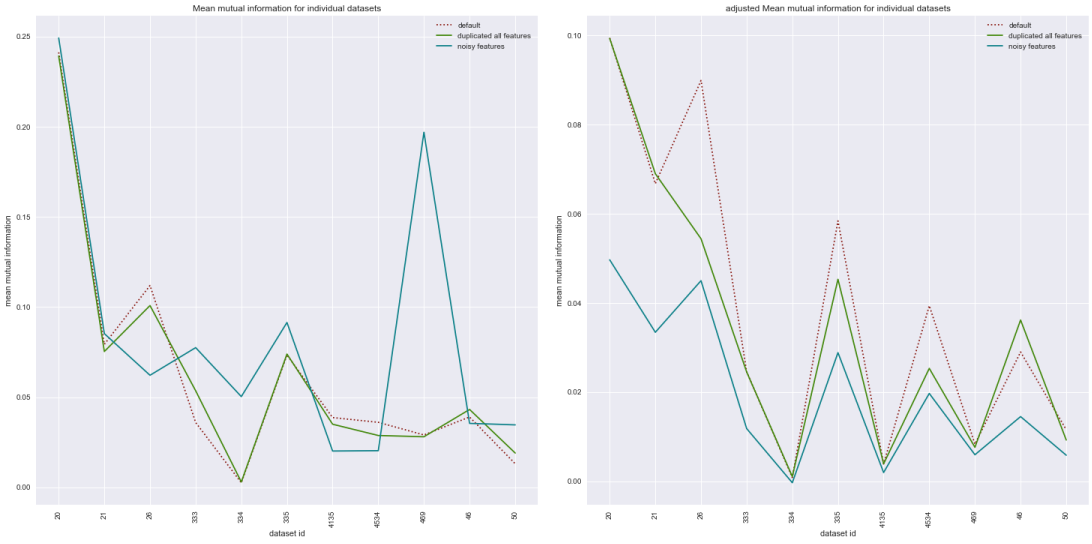
3 Experimental setup

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

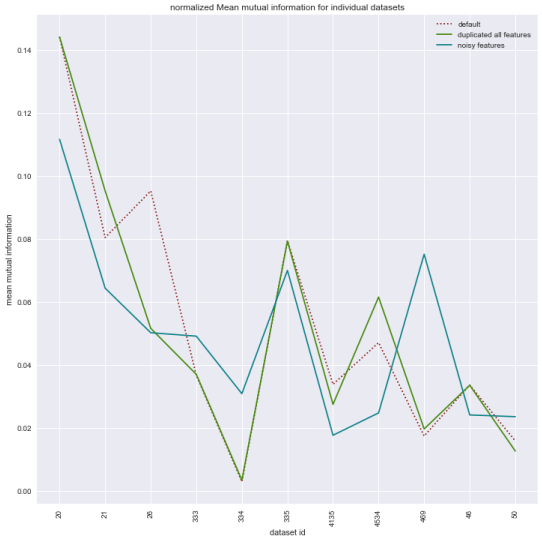
3.1.1 Mean Mutual Information

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 mututations of datasets



(c) normalized Mean Mutual Information for mututations of datasets

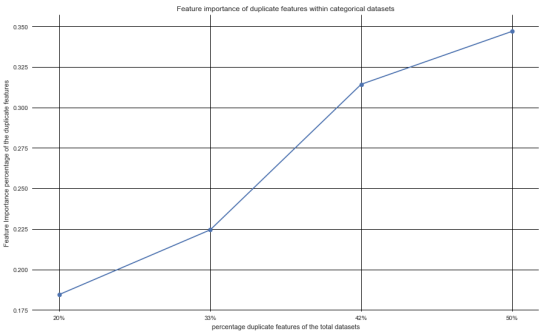
Figure 2: Mutual information difference between measure

3.1.2 Feature Importance

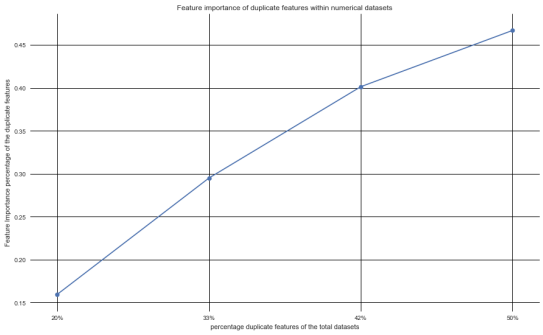
Feature importance calculated with a RandomForestClassifer gives an indication what features are important for the ensembled decision trees. In figures 3b, 3a, 4b, 4a, the feature importance of the added random features are shown. This makes it clear that RandomForestClassifier does not recognize these 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 redundant for predicting. The difference between the feature importance for numerical and categorical is also shown. This is partly discussed earlier as the variance in the features and feature order. This has only a small effect

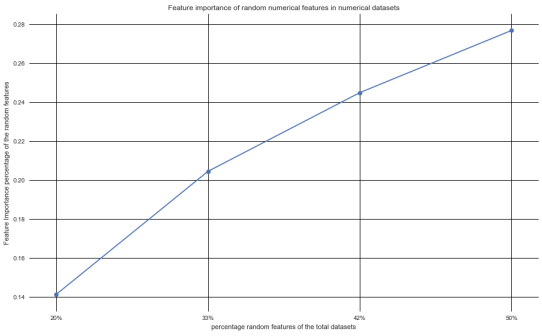


(a) Feature importance for random categorical features with k = 3.

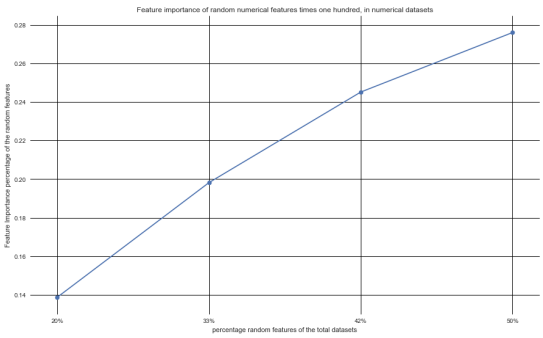


(b) Feature importance for random categorical features with k = 100.

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

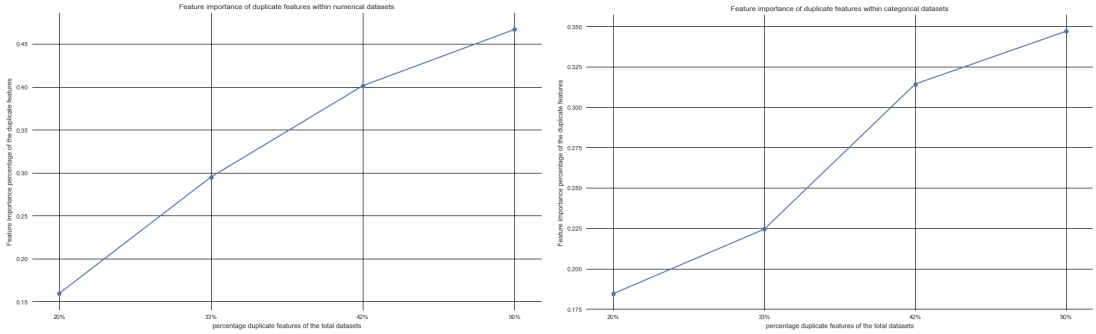


(a) Feature importance for random numerical features with the default uniform spread between 0 and 1.



(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 featueres for categorical datasets.

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

3.2 Model Validation strategies

What strategies do we use to split the datasets into training and test to measure predictive accuracy, 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 sklearn scoring calculation
- duration instances for the time needed to calculate
- control data like predictions and real target values. Summarizing data of the predictions to make a faster observation.

3.4.1 Duration

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

3.4.2 Predictions

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

3.4.3 scores

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

3.4.4 SummaryGuesses

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

3.4.5 BiasVar

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

3.4.6 Identifier

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

3.4.7 RemovedFeatures

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

3.5 Experiments

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

3.5.1 Scalability

Scalability experiments can be split up in instance based or features based. To measure the effect of features we take datasets with lots of features and remove features in steps to find the impact of these lost features. There is a disadvantage with this strategy as some classifiers calculate values like feature importance which depend on a somewhat complete dataset of features. The removed features are randomly chosen and can be defining features for the accuracy of the dataset.
To combat this we also 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 orignal dataset size. The accuracy is measured over the whole dataset by doing a 10 fold cross validation.

3.5.3 Random Features

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

3.5.3.1 categorical random features

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

3.5.3.2 numerical random features

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

3.5.4 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 similair to the features already in the dataset. These features are similair to the original dataset, so we can better measure the impact of scalability of features similair 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 algorithm the dataset is clean and over time new data can change. By measuring the predictive accuracy off the algorithms on more noisy datasets the robustness can be measured. The accuracy is measured over the whole dataset by doing a 10 fold cross validation.

3.5.5.1 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** datapoint 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 datapoints in the feature set. The random function is like mentioned before producing a uniform random value between 0 and 1.

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

3.5.6 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 algorithms 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 algorithms a translation of categorical features seems to be neccesary [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

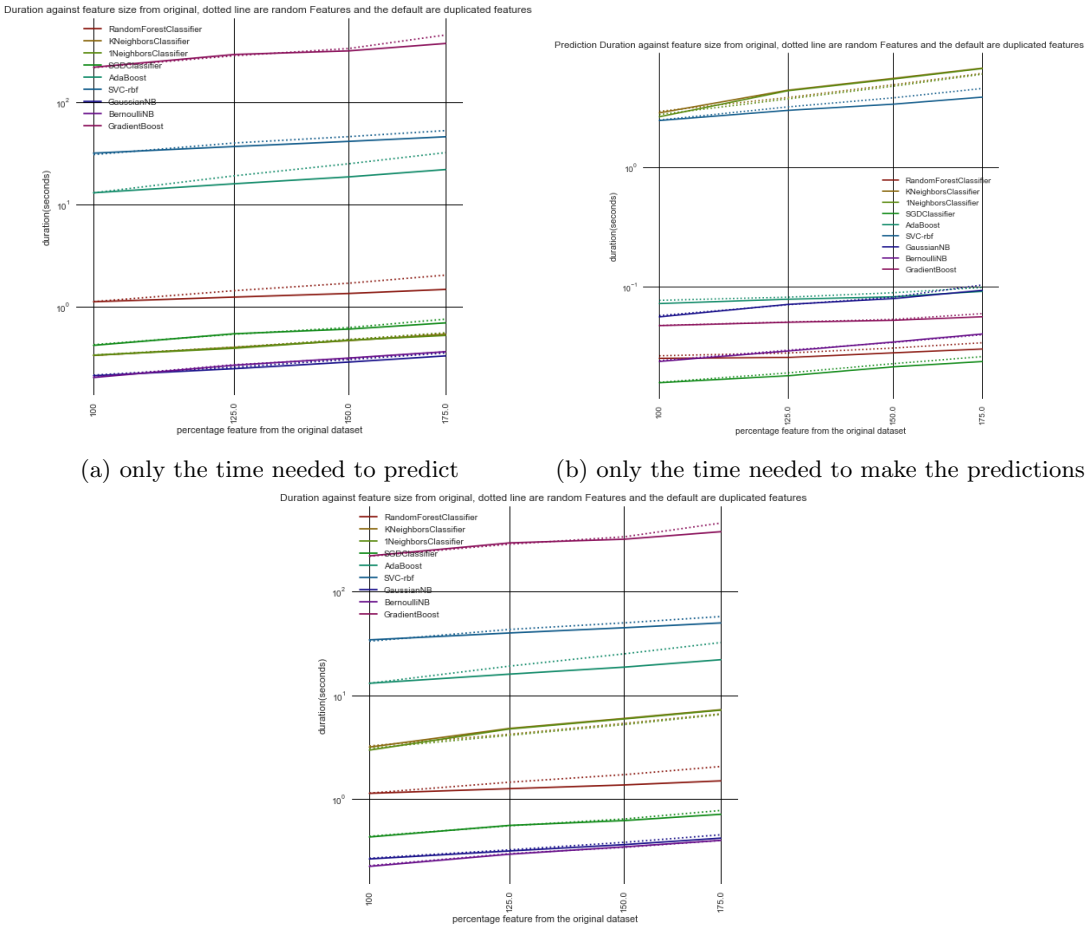
All results with a duration axis. The duration axis is in seconds and shows a 10 fold cross validiation. This means that the duration encompasses 10 times fitting over 90% of the data and 10 times predicting 10% of the data.

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 figure **??** and the slope in figure **??**.



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

RandomForestClassifier
KNeighborsClassifier
SGDClassifier
AdaBoost

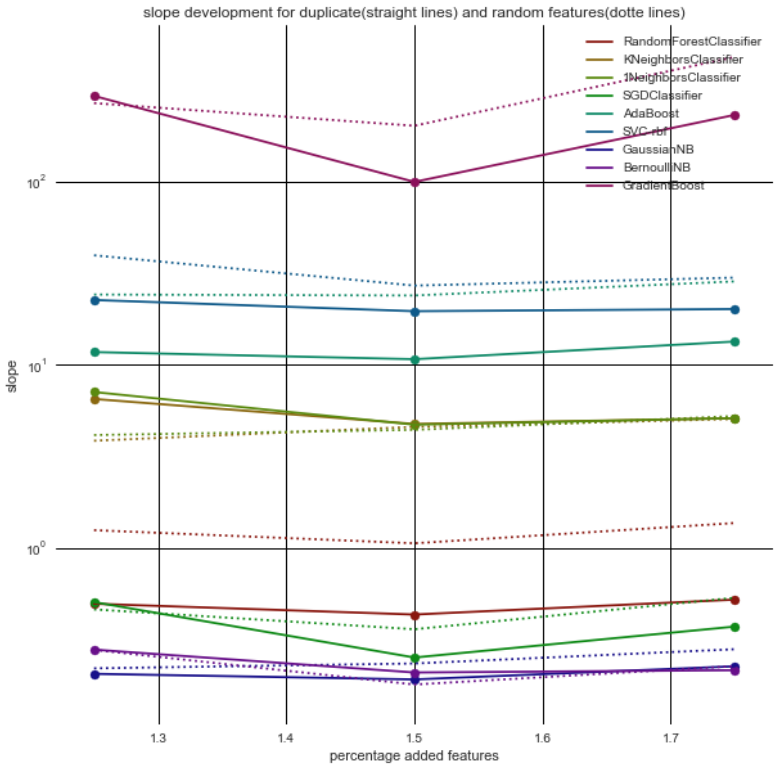


Figure 7: Slope for 6c with the dotted line for random added features and the straight line for redundant duplicate features

SVC-rbf
GaussianNB
BernoulliNB
GradientBoostingClassifier

4.1.1.2 combined in figures 8 and 9 adding and removing features is shown. in figures 12,11 and 10

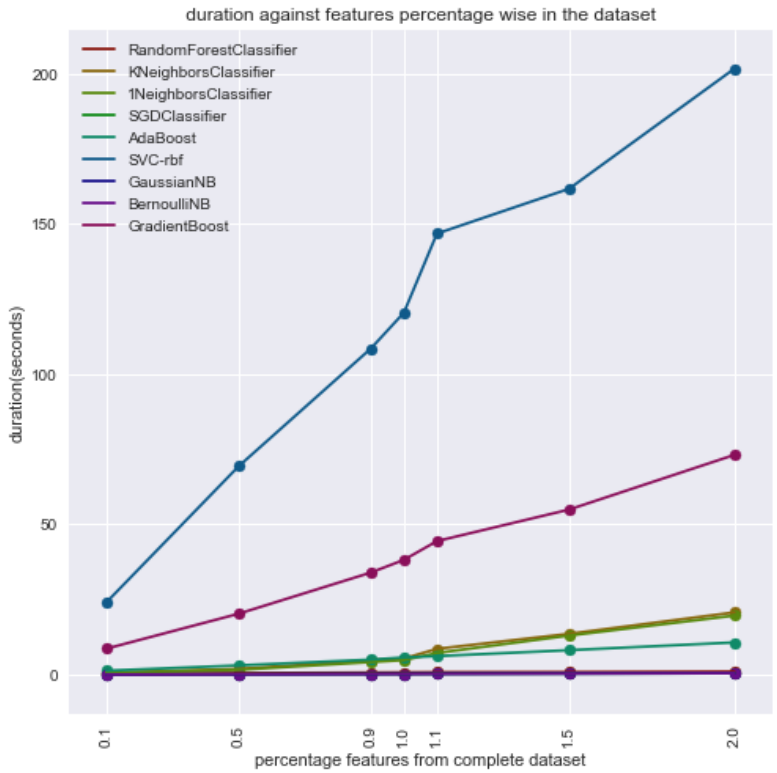


Figure 8: features removed randomly and features redundantly duplicated

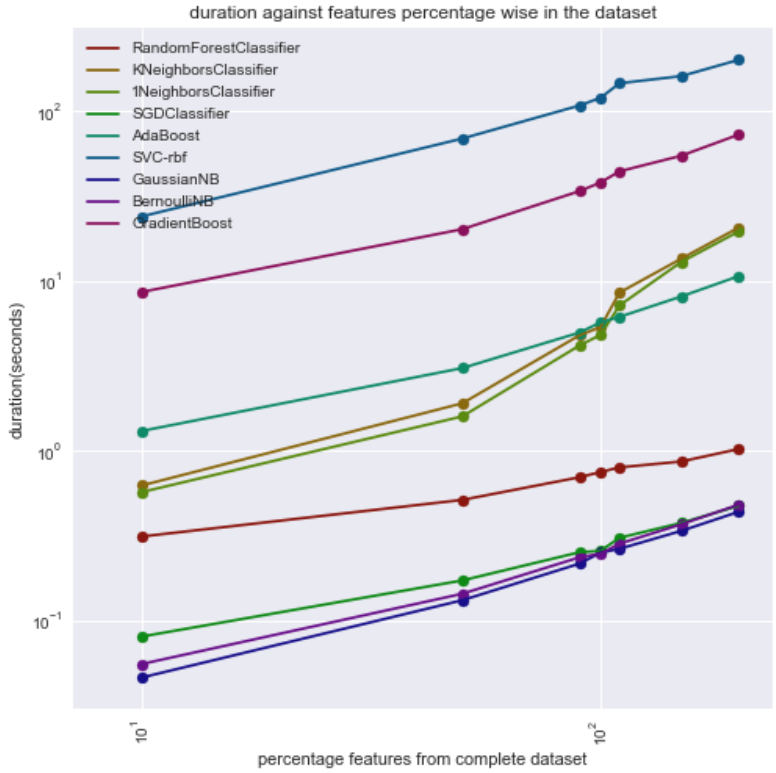


Figure 9: features removed randomly and features redundantly duplicatedd with a log scale

4.1.2 Instances

RandomForestClassifier
KNeighborsClassifier
SGDClassifier
AdaBoost
SVC-rbf
GaussianNB
BernoulliNB
GradientBoostingClassifier

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

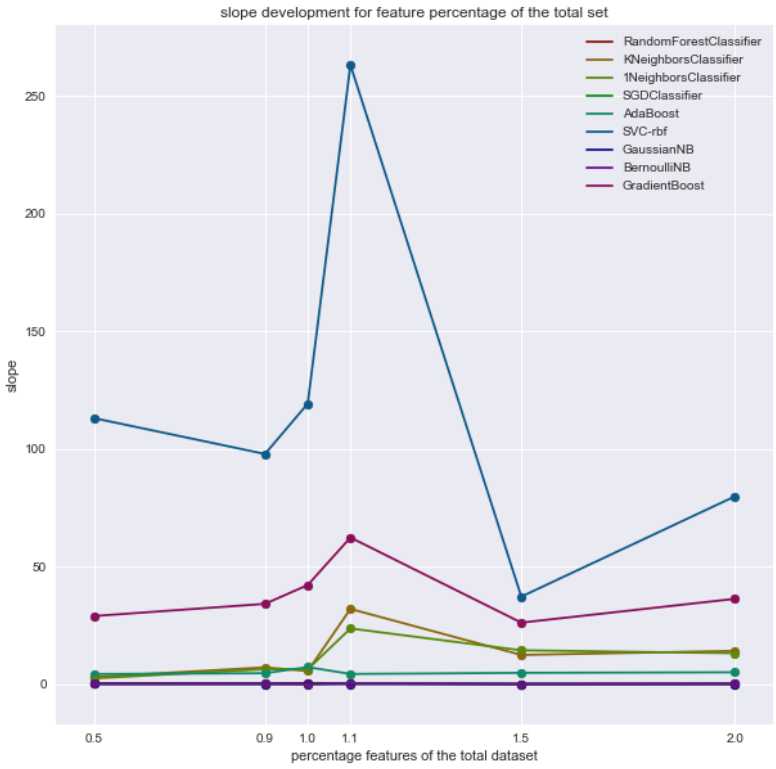


Figure 10: features removed randomly and features redundantly duplicated slope of 8

4.2 Redundant features

We consider here both random and duplcite features as they are both redundant.
RandomForestClassifier RandomForestClassifier has one of the best predictive accuracies on the clean data sets. The decline however with added features is noticable steep. With duplicate features the decline is pretty steep. The added random features have less of an impact on the accuracy. A large part of the impact can be seen with the initial addition as that has the largest impact in comparison. As indicated in 3.1.2 RandomForestClassifier indicates that the random features have some importance in the predictions which only increases with more features, but less than the importance it indicates. The same holds for duplciate features which also gains feature importance with more duplicates.

KNeighborsClassifier The predictive accuracy of Knn is decent on the categorical datasets for both values of k. The downfall in quality however with random categorical values is hard, which is mostly due to inability to handle categorical features. If we look at the same datasets with added numerical categories there is even an increase in predictive accuracy.22. The duplicate features have in both cases of numerical and categorical datasets a near equal impact considering the different datasets. The random numerical features in the numerical datasets have a clear impact when they are added but only a slight decrease afterwards. This can be explained as the variability of the random features. A previous instances would have a chance to be close to its equal class neighbors but with the added dimensions of random features this closeness is shifted. This variances does not increase that much with more features as the odds of extreme cases still staying the same. An extreme case being a previous neighbors instance having the minimal value for all added value and the neighbors of equal class having all maximal values of the added class.

SGDClassifier The predictive accuracy of the SGDClassifier is the worst on the categorical datasets and one of the worst on the numerical datasets. This lack of initial performance is because of the clean datasets, of which it can hardly handle the categorical features. The robustness to random categorical features is also really bad and is only saved as it can hardly handle the normal categorical features. The

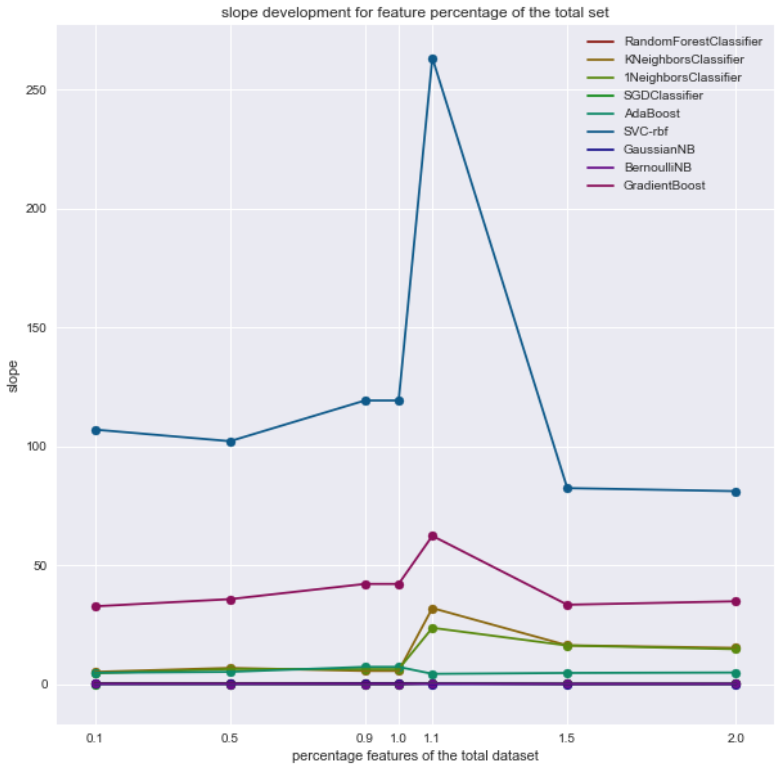


Figure 11: features removed randomly and features redundantly duplicated absolute slope calculated of the clean dataset (values from 8)

duplicate categorical features have a less of an impact as they are more of the same the accuracy has less to lose. The robustness for numerical features is better, for duplicate features there is a slight decline similair to most of the other algorithms. The random numerical features however have nearly no impact on the predictive accuracy. With the stagewise steps of optimizing a cost function the SGDClassifier is really robust against random numerical features. The added random numerical features are clearly continious values which the SGDClassifier expects for its input.

AdaBoost The predictive accuracy of unoptimized AdaBoost is below average on both datasets. The handling of added features is however strong as it can handle both random numerical and categorical features with only a slight dip in accuracy with the introduction of these features. This behavior can be explained by the optimization of outliers Adaboost is focussed on. The added random features are so uniform that there is a clear lack of outliers to optimize on. The adaBoost has a harder time disregarding the duplicate features with an average robustness compared to the others. These duplicates have the same effect as the below average prediction of the original set. AdaBoost struggles to foucs on the outliers and so overfits on these duplicate features.

SVC-rbf The predictive accuracy of SVC-rbf is great on the categorical dataset and average on the numerical dataset. The handling of the clean dataset with default settings is not fruitful. The robustness to the random categorical features is also bad. The variance within these categories is high and without tweaking the C parameter it can be expected to overfit to these features. For duplicate features the robustness

GaussianNB

BernoulliNB

GradientBoostingClassifier

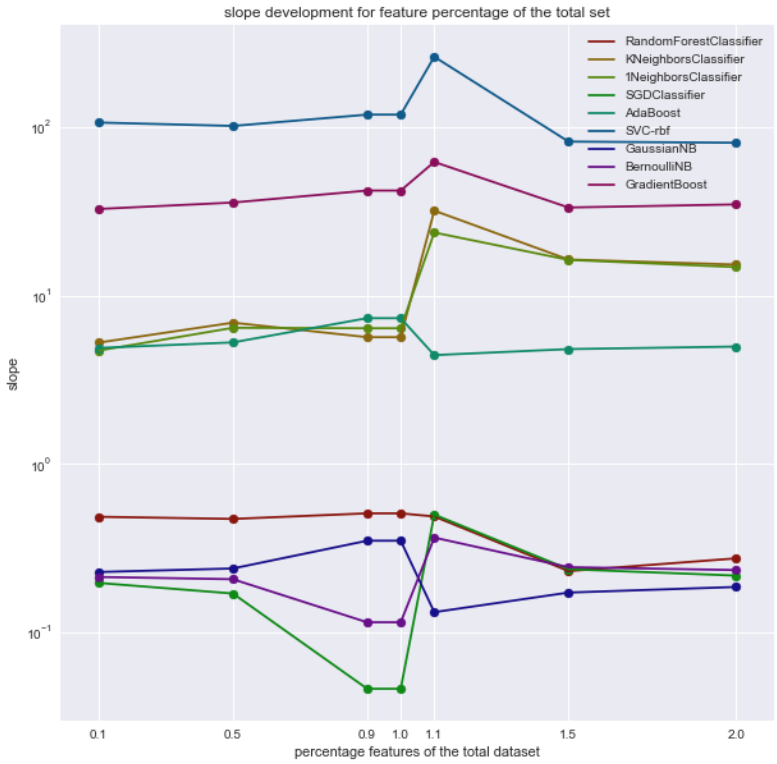


Figure 12: features removed randomly and features redundantly duplicated absolute slope calculated of the clean dataset with a log scale (values from 8)

	duplicate features	random features	redundant duplicates features
RandomForestClassifier	0.0	-0.19	-0.08
KNeighborsClassifier	-0.0	-0.13	-0.33
1NeighborsClassifier	-0.0	-0.12	-0.33
SGDClassifier	-0.01	-0.07	-0.18
AdaBoost	0.0	-0.1	-0.02
SVC-rbf	-0.0	-0.2	-0.32
GaussianNB	-0.02	-0.21	0.01
BernoulliNB	-0.03	-0.08	-0.0
GradientBoost	-0.0	-0.15	-0.05

Table 1: all change in predictive accuracy in adding certain kind of features(results from 16)

4.2.1 Combined feature manipulation

RandomForestClassifier
KNeighborsClassifier
SGDClassifier
AdaBoost
SVC-rbf
GaussianNB
BernoulliNB
GradientBoostingClassifier

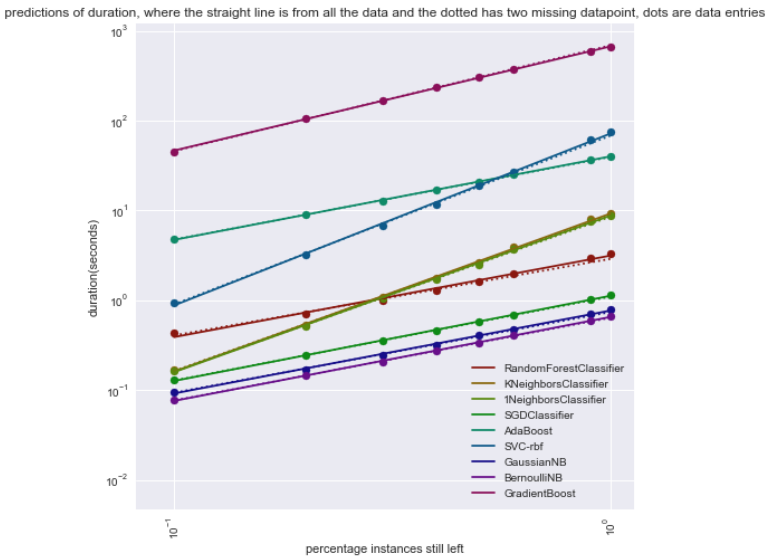


Figure 13: The dataset is reduced to a split of the initial dataset and then cross validation is done on the reduced dataset. The lines are predictions

4.3 Noisy data

Datasets in these experiments get increasingly more random by injecting the features with noise. In figures 18, 19 and 20 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.
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 noticable 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 algorithms. 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 algorithm 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 SGDClassifier. Considering that AdaBoost is not optimized and that the datasets are not all particularlly 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 algorithms. This does not translate well with more cleaned data. With the preprocessing

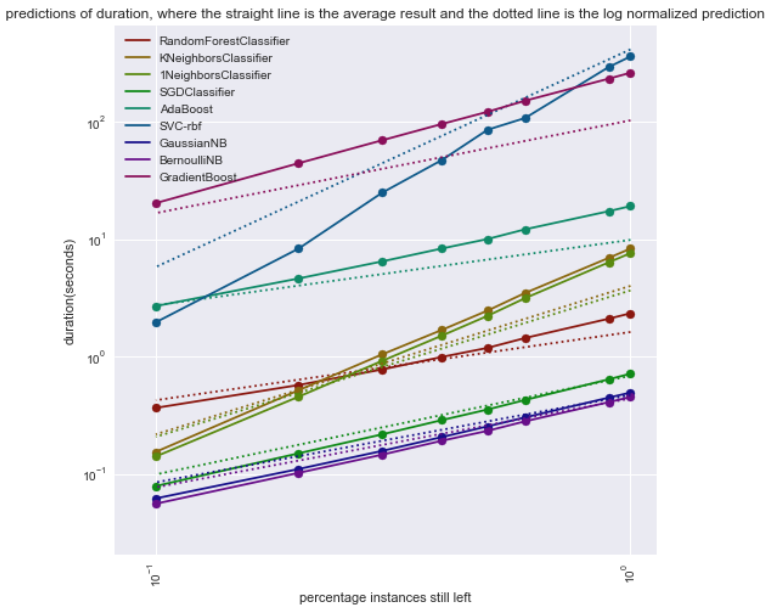


Figure 14: the dataset is reduced to a fraction of the initial dataset. A prediction is made of the duration of classification and prediction per dataset, indicated by the dotted line

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 noticable 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 36 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 remainging 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 of the algorithm of 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.

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 overfitting compared to the RandomForestClassifier. The GBC uses more time to fit the classifiers for an additive model rather than pure random voting.

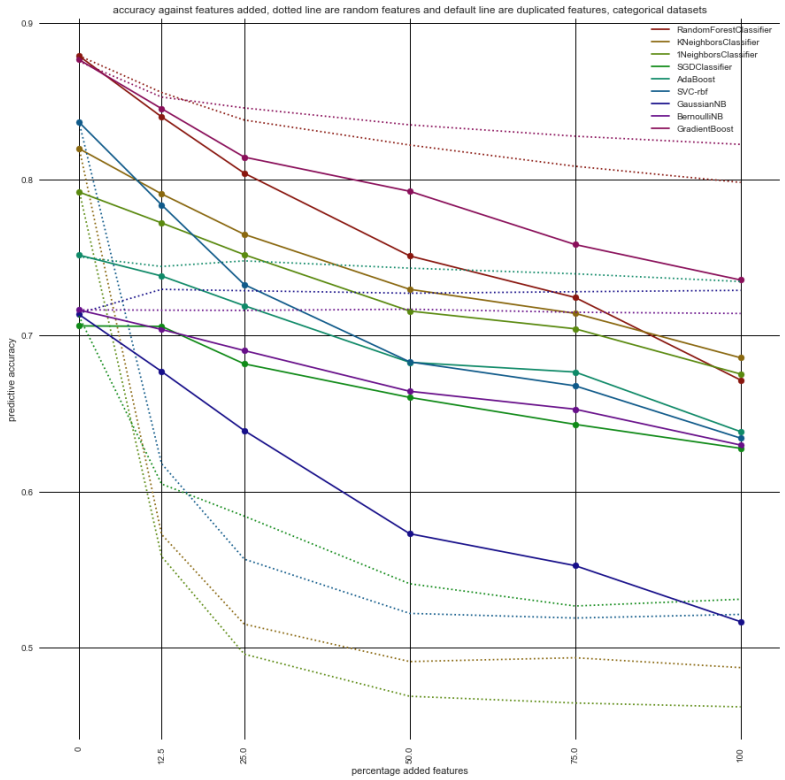


Figure 15: predictive accuracy for categorical datasets with added redundant features

4.4 Bias variance

4.5 Ontology

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

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

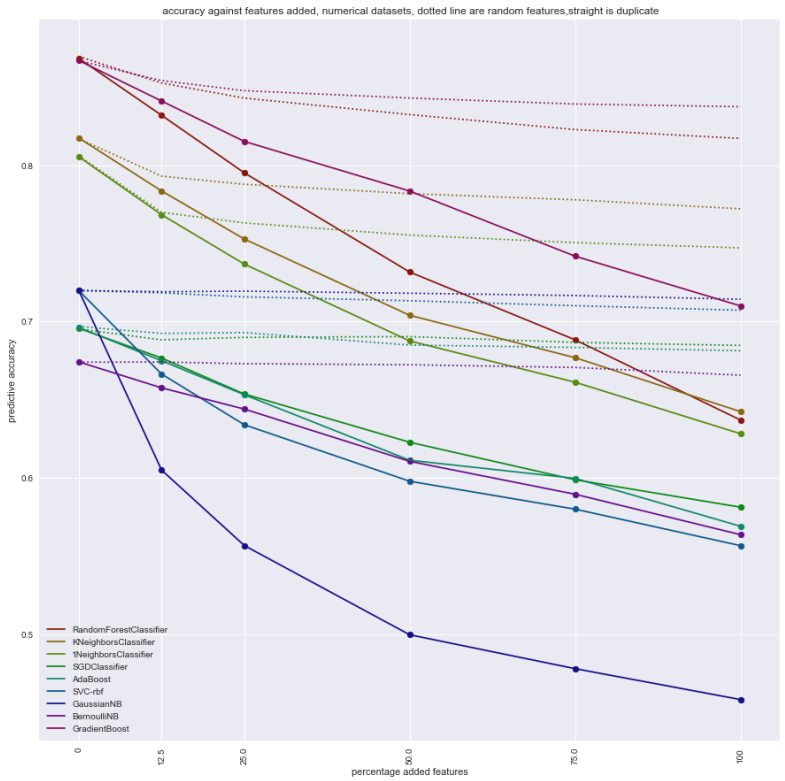


Figure 16: predictive accuracy for numerical datasets with added redundant features

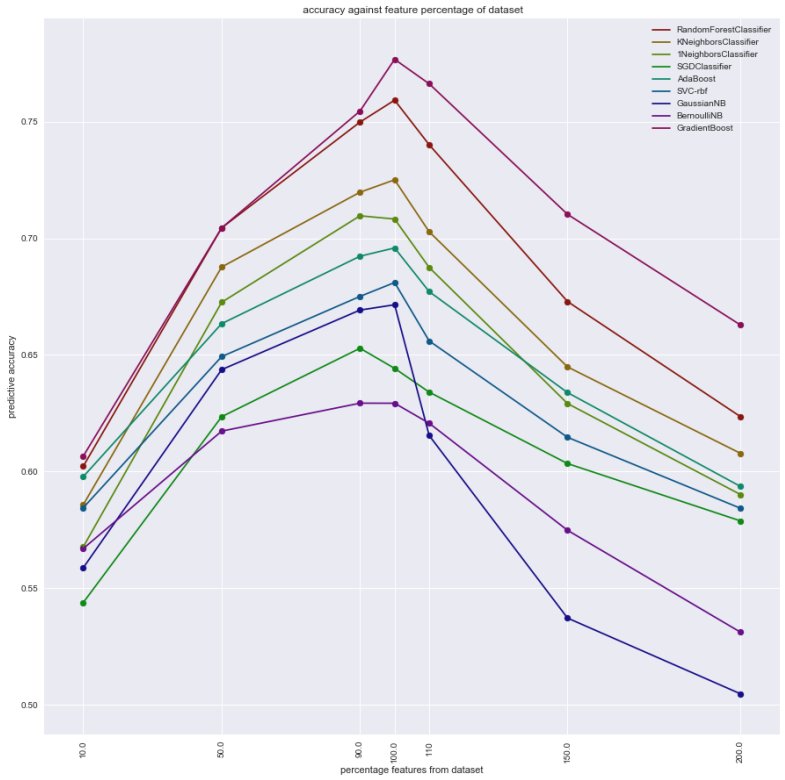


Figure 17: predictive accuracy for datasets with added or removed redundant features

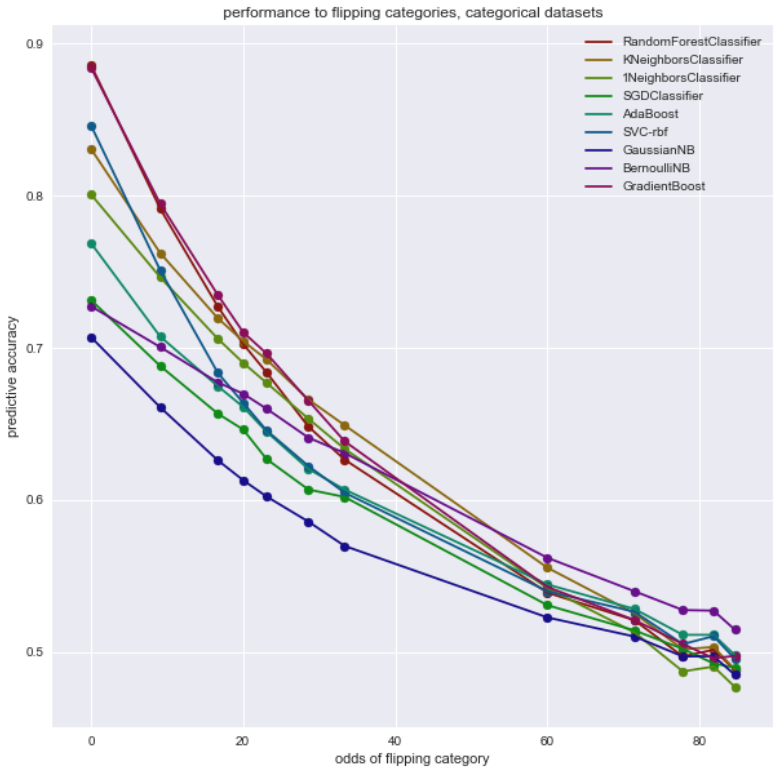


Figure 18: flipping categories for categorical features

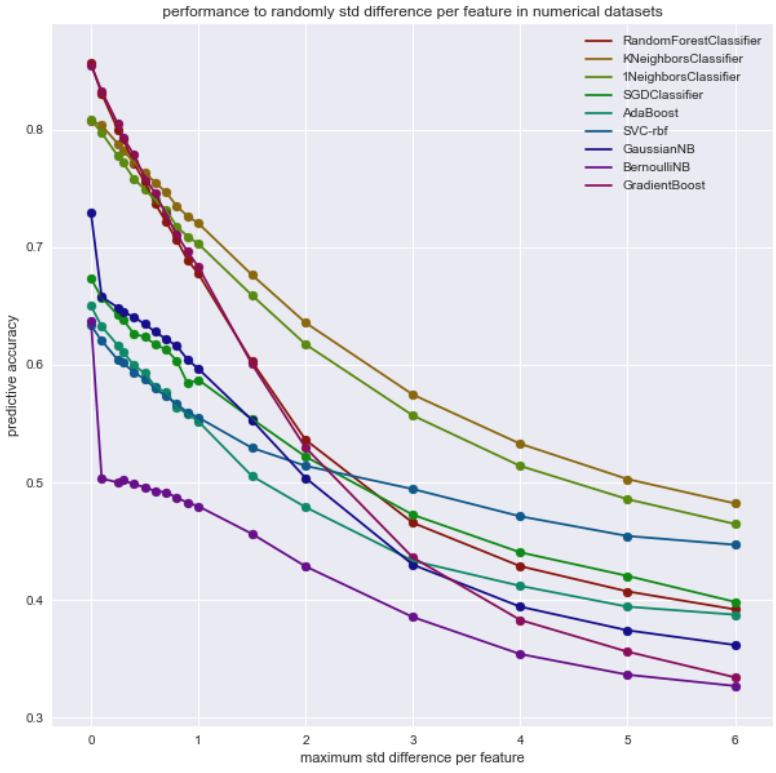


Figure 19: adding or removing uniformly random std to numerical features

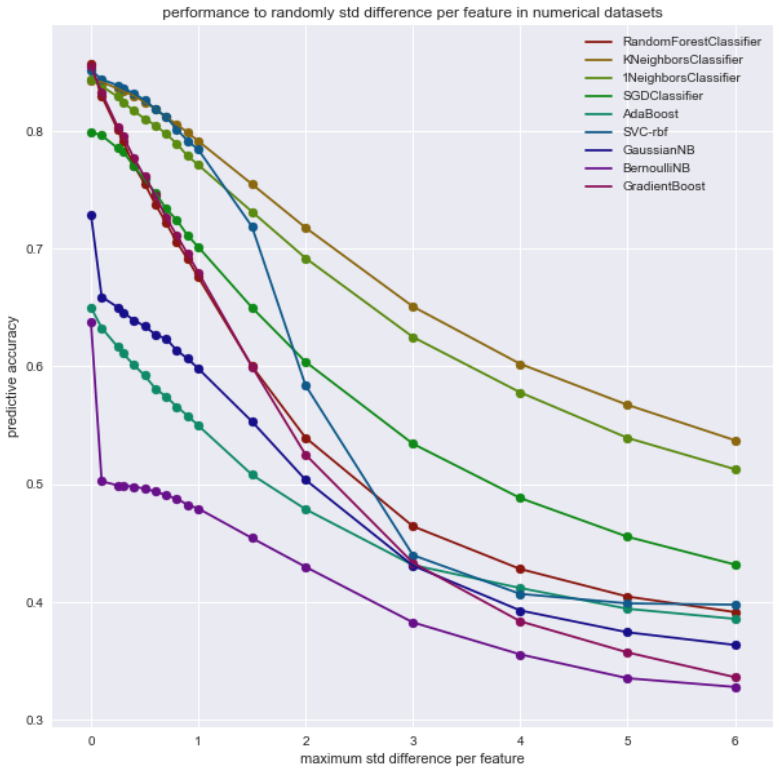


Figure 20: adding or removing uniformly random std to numerical features and preprocessing for some algorithms.

5 Conclusion and discussion

5.1 Discussion

5.1.1 Missed opportunities

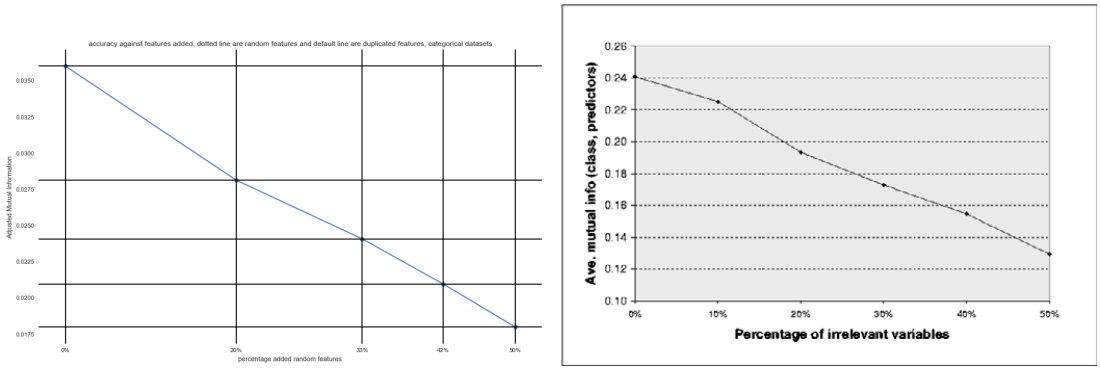
Within saving as many information obtainable from the algoritms a missed oppertunity is the probability prediction of all the test sets.
Another missed collected data is the prediction of the training data. This data might not give a good indication of predictive accuracy but can give some valuable information of overfitting on the input data. This can be calculated as a difference in prediction results of the training and test set. For some algorithms this holds more than others. For example SDGClassifier 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.

5.2 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 similair to injection std but more abrupt. Just changing(moving/word) all features for some datasets mend a drop in accuracy but the continuation of upping the number mend little to change accuracy further.

5.3 replication of previous work



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

(b) average Mutual Information as a measure of relevance

Figure 21: Mutual information decline between previous work and this work

In figure 21 the experiment setup of Hilario and the expriment 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 valueable 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.

5.4 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 featueres the different values of discrete, continous or ordinal values. Each can need different preproccessing steps for different algorithms.

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 per figure

figure 23 - 10,12,18 figure 28 - 1038,1043,1049,1050,1176,12,1466,1468,1475,1476,1478,1479,1485,1487,1491,1492,1493,1494,

7.2 different approach

7.3 Duration variance

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

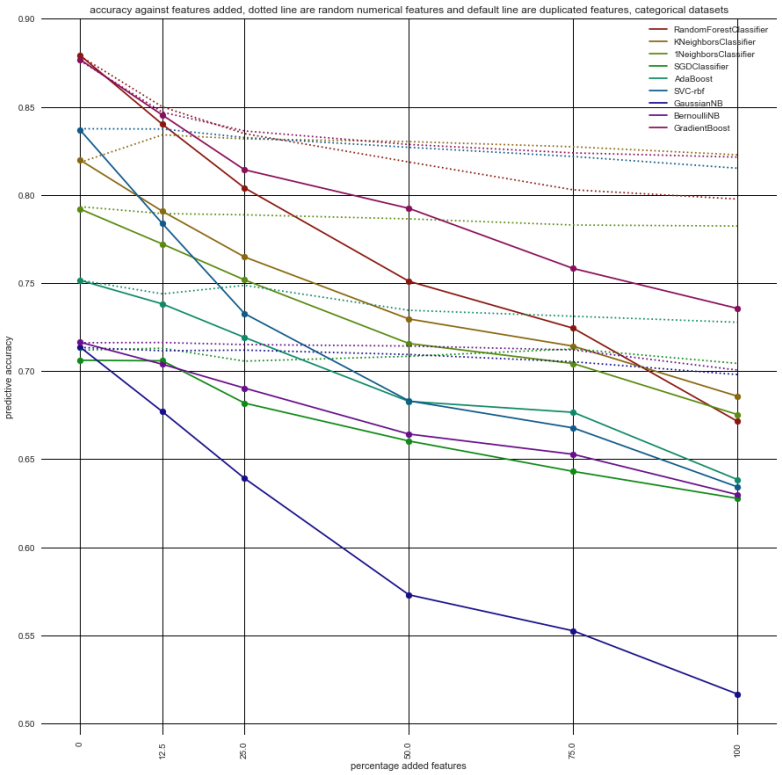


Figure 22: adding or removing uniformly random std to numerical features for GaussianNB.

7.4 Results per dataset per classifier

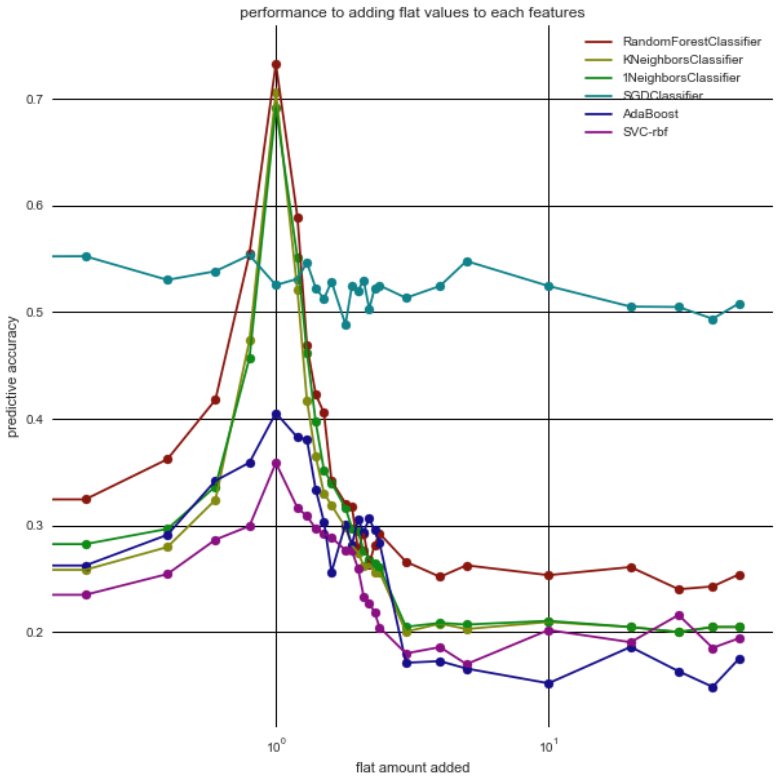


Figure 23: adding flat values to each feature

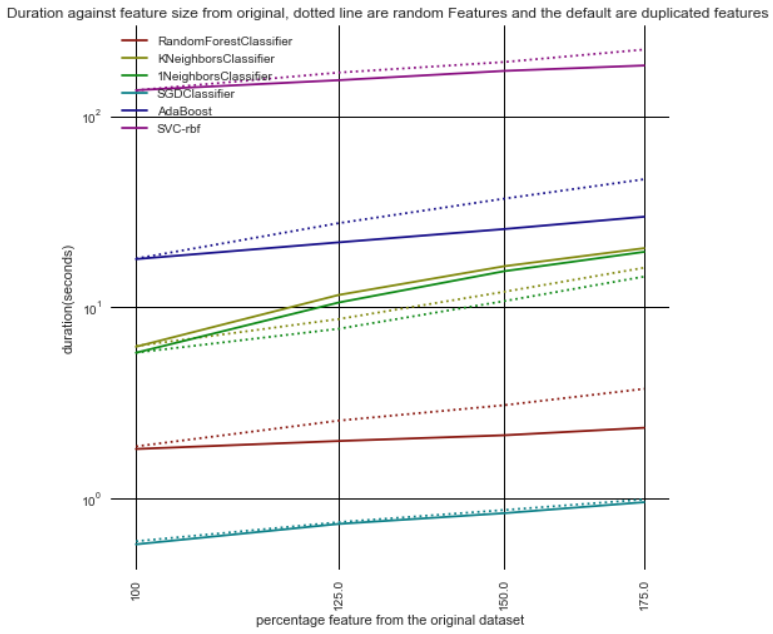


Figure 24: Duration of the oldest version of adding random features

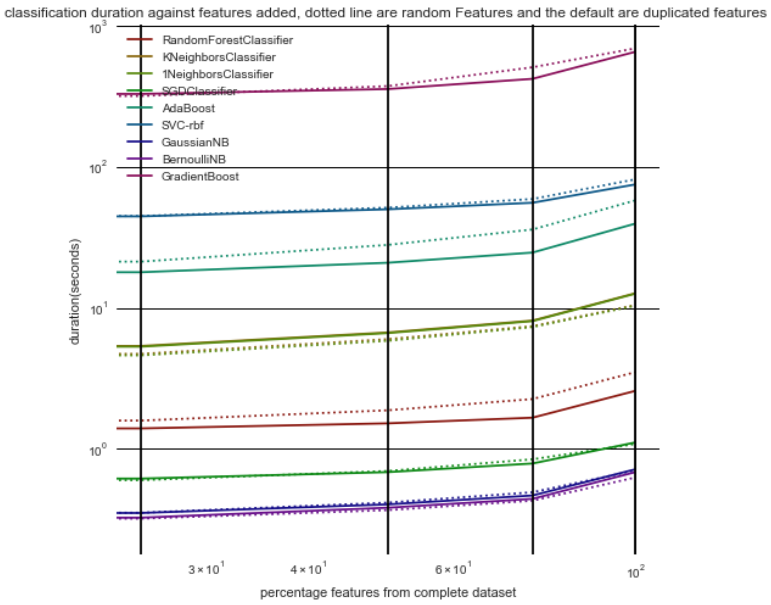


Figure 25: Duration of the oldest version of adding random features

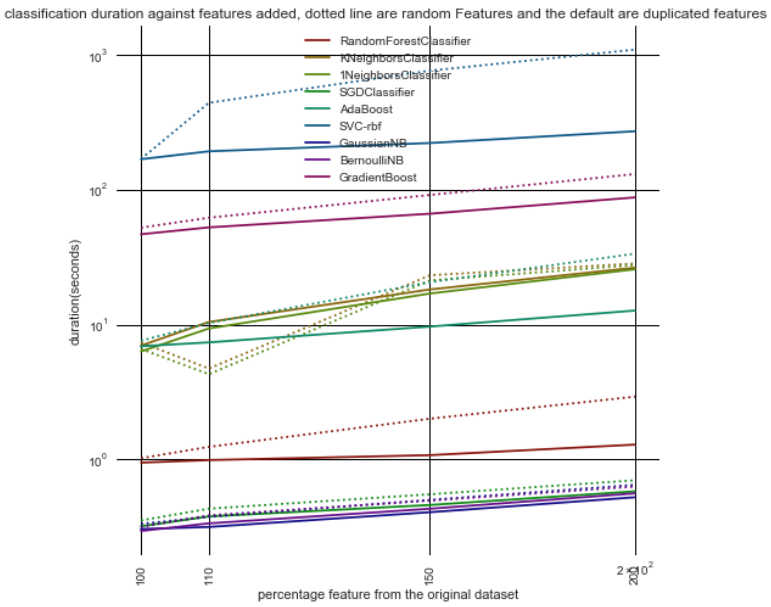


Figure 26: Duration of the newest version of adding random features

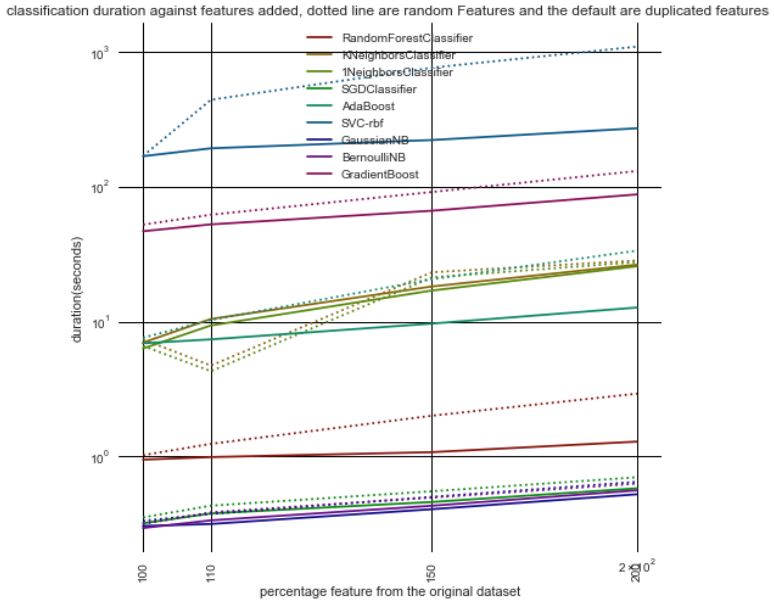


Figure 27: Duration of the newest version of adding random features

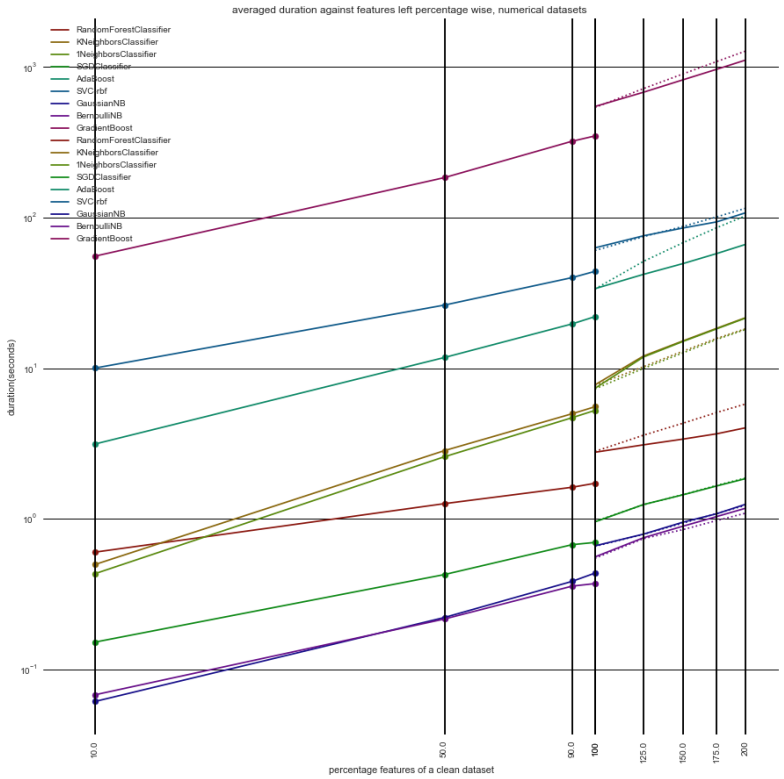


Figure 28: Duration of adding and removing features combined result

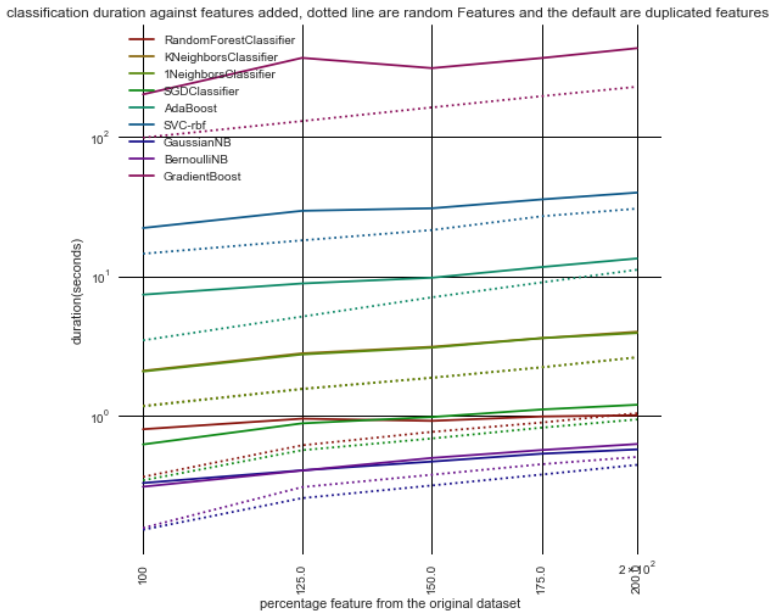


Figure 29: Duration of duplicated features against random features with clear spikes in duration registering

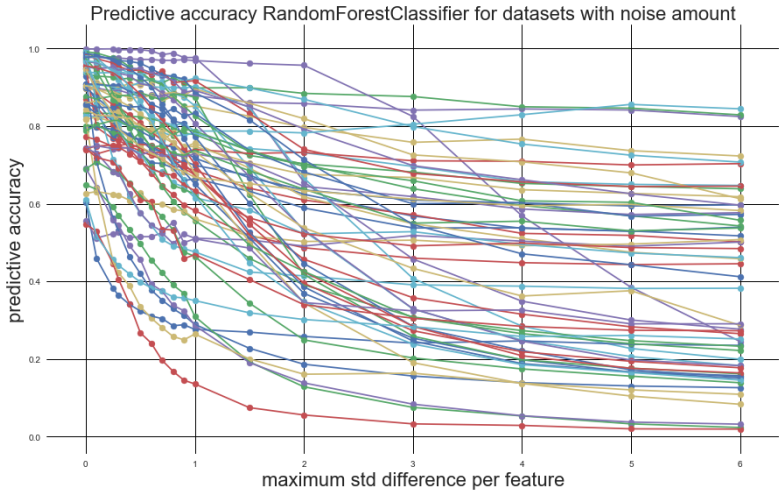


Figure 30: adding or removing uniformly random std to numerical features for RandomForestClassifier.

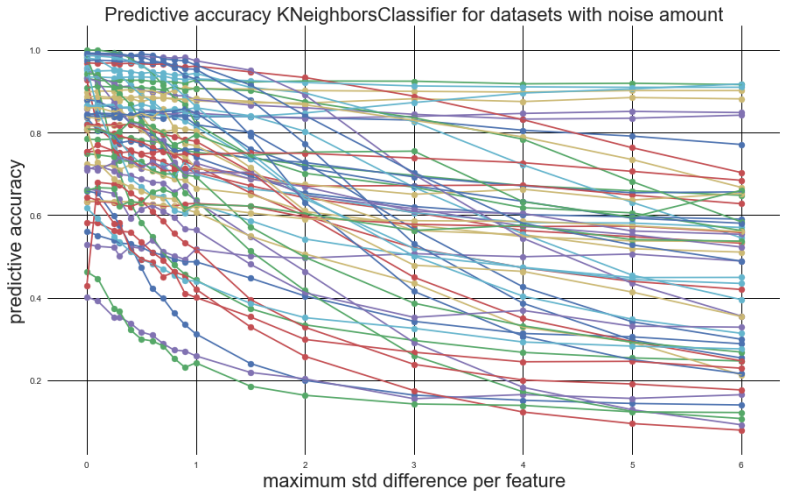


Figure 31: adding or removing uniformly random std to numerical features for KNeighborsClassifier.

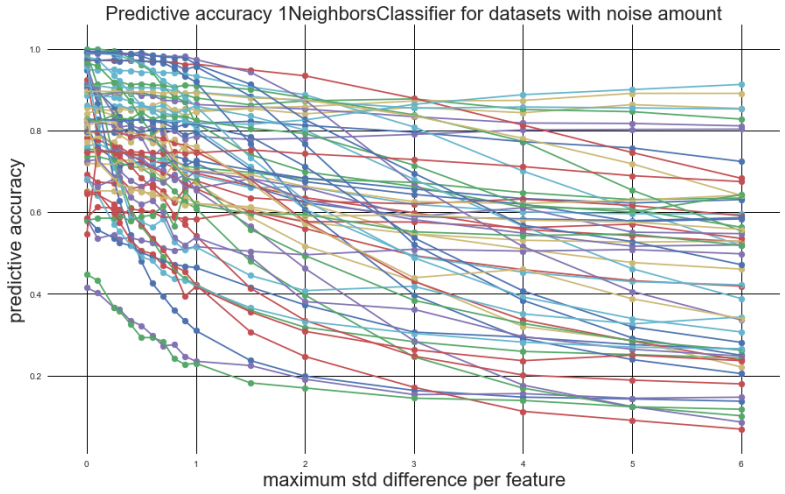


Figure 32: adding or removing uniformly random std to numerical features for 1NeighborClassifier.

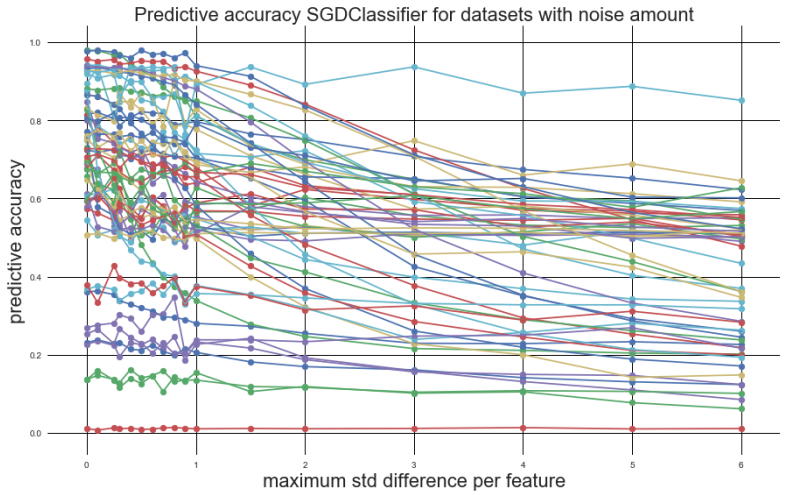


Figure 33: adding or removing uniformly random std to numerical features for SGDClassifier.

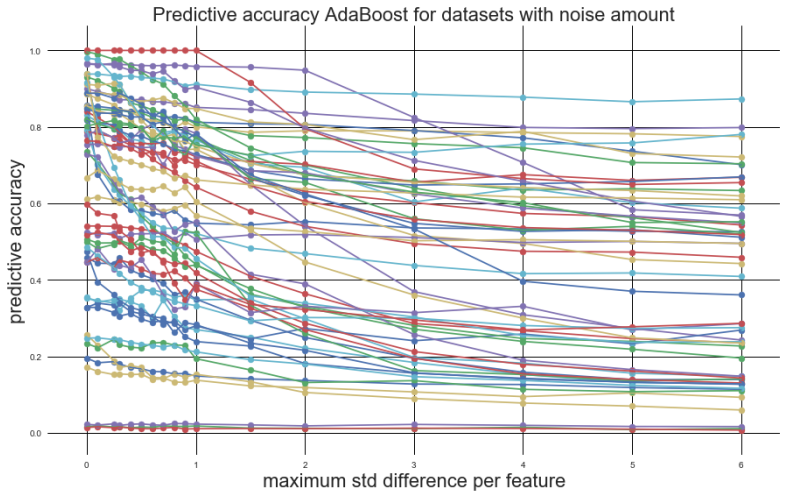


Figure 34: adding or removing uniformly random std to numerical features for AdaBoost.

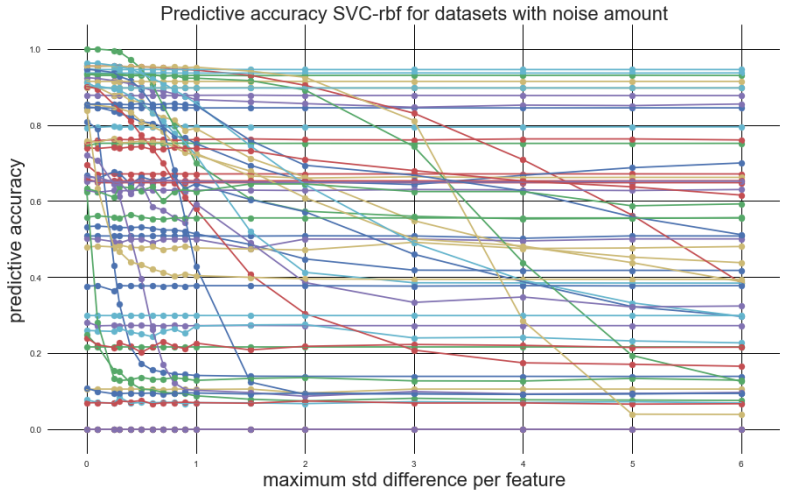


Figure 35: adding or removing uniformly random std to numerical features for SVC-rbf.

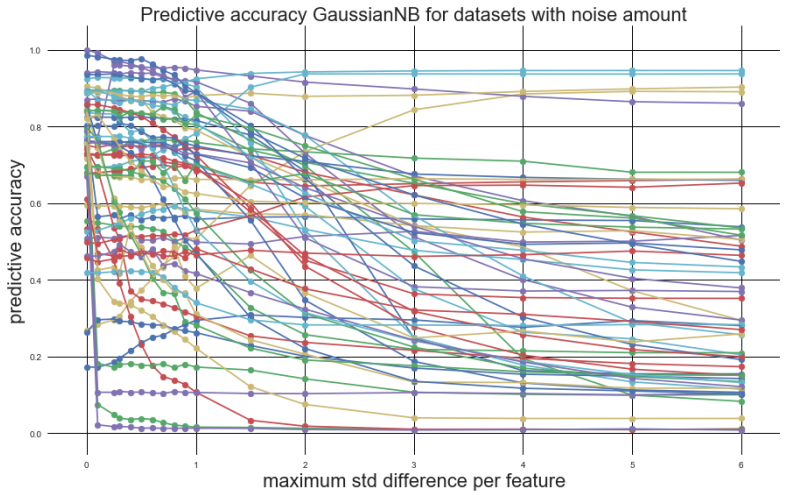


Figure 36: adding or removing uniformly random std to numerical features for GaussianNB.

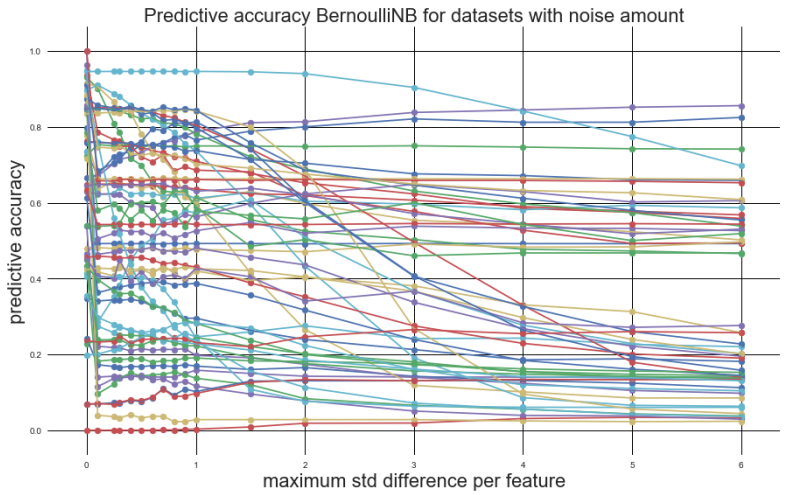


Figure 37: adding or removing uniformly random std to numerical features for BernoulliNB.

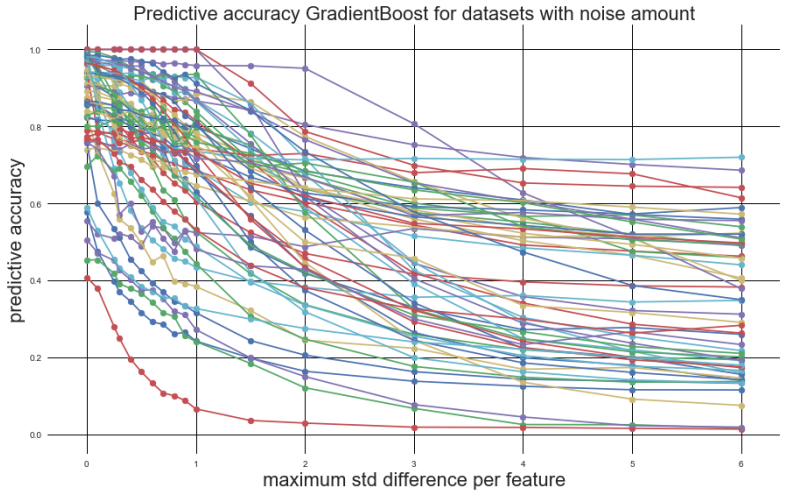


Figure 38: adding or removing uniformly random std to numerical features for GradientBoostingAlgorithm.