Primitive annotations

Master Thesis

L.D. Stooker, 0819041

Supervisor: Joaquin Vanschoren
May 25, 2018

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 [33], [29]. Python is a trending machine learning language, as it holds premade machine learning algorithms in the scikit-learn (sklearn) library.[30]. This library is made to easily choose between different algorithm for a machine learning problem. Figure 1 depicts a flowchart with the different machine learning algorithms that can be chosen in the sklearn library. It can be seen that different algorithms are recommended depending on the early results, size and problem.
It can be seen that different algorithms are recommend depending on the problem, size of the dataset and early results. It can also be seen that four different groups are present: classification, regression, clustering and dimensionality reduction. The selection of a group depends on the type of the problem that needs to be solved. Each group contains a different selection of algorithms. A specific algorithm is recommended depending on the size of the data and early results. However, with time new algorithms are added to the library. This allows for the recommendation of more specialized algorithms for a particular problem. Each different problem cannot be solved by the same algorithms. This concept is based on the 'no-free-lunch' theorem, which describes that one particular algorithm cannot be used to solve all optimization problems [11].

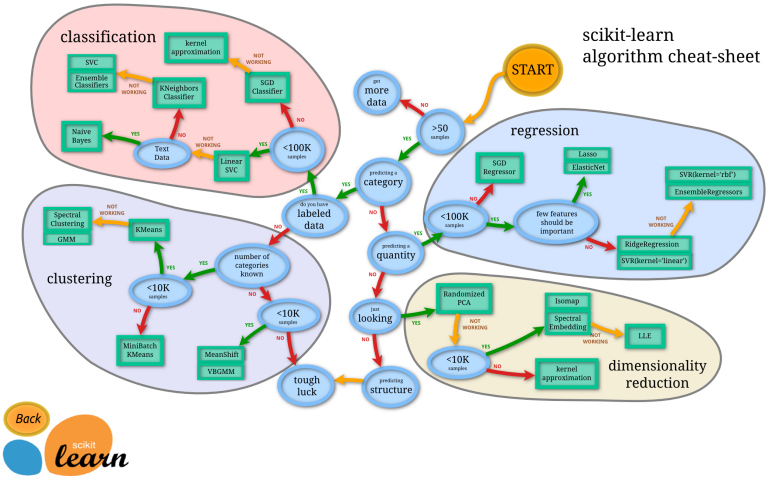


Figure 1: FlowChart with the different machine learning algorithms in the sklearn library [34]

1.2 Research question

In this research we focus on the classification group to give room to new algorithms This thesis focuses on the selection of algorithms in the classification group. We base our research question on the work of Joaquin to give properties to classifiers[24]. 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[14]
The work of Joaquin describes that properties can be given to the classification algorithms. This allow for earlier research to improve the algorithm choice of new research. Out of the quantitative properties given by Joaquin the resilience properties and the bias-variance are investigated. The resilience properties describe how an algorithm can learn from noisy, irrelevant or redundant data, while the bias-variance

characterizes the errors that are made by learning algorithms. Some features of these properties have been investigated over a decade ago. However new algorithms have become popular, which makes this thesis more relevant in today's worlds.
Research questions have been formulated. These questions have been made to determine the properties of the research and to reflect on earlier work.

- What is the impact of the number of features for classifiers on runtime?
- What is the impact for the number of instances for classifiers on runtime?
- What is the impact of irrelevant or redundant features for classifiers on predictive accuracy?
- What is the impact of noisy features for classifiers on predictive accuracy?
- What is the percentage of Bias error or variance error for classifiers?

To summarize we rank the classifiers on their performance(predictive accuracy), robustness and scalability to find a ranking like done by Quan Sun and Bernhard Phahringer[28].

1.3 Outline

In the first chapter the research question is introduced, followed by the second chapter, this chapter discusses the relevant background information. The third chapter gives the setup of the experiments for the classifiers. The fourth chapter shows the results for each classifier. The fifth chapter presents the conclusion and answers the research questions and discusses future extensions.

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 [6]. The random forest classifier constructs directive trees on subsamples of the input data. A directive tree is a decision tree classifier which splits the features on certain thresholds to decide on the type of class. The highest probabilistic prediction of all the directive trees is the prediction. The subsamples for the directive trees are chosen randomly so the results can vary between runs on the same input. This splitting of the data is either randomly or choosing the best split. The criterion to judge the best split is either Gini or entropy based. The number of splits, features and samples are also considered during fitting. RandomForestClassifier is made up of decision trees. These trees split the features on certain thresholds or values in each node. The top most node of the tree is the most deciding part as it is defines the first step in the tree. 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 top most nodes in the trees of a random forest. The value of feature importance can be calculated as the percentage of occurrence in all the top most nodes of trees in the RandomForestClassifier. This feature importance value tells us what the RandomForestClassifier finds to be important features in a dataset.

2.1.2 KNeighborsClassifier

In the scikit-learn library KNeighborsClassifier is an implementation of the k-nearest neighbors classification algorithm(kNN)[1]. The kNN uses as it names tells the k nearest neighbors for calculation. This is done by instance-based learning and results in non-generalizing learning. Instance-based learning and non-generalizing learning means that during fitting no complete model is made but only the given feature set is stored in order of appearing. The target predicting is done by going through the inputted feature set to find the nearest k points for each instance, depending on the majority class of the k neighbors the classification is vote is decided. The default metric for distance measuring is Euclidean distance(absolute distance) another option is the Manhattan distance. The Manhattan distance is the sum of all differences in features between two instances, this is quicker than calculating the Euclidean distance but less accurate. A ball tree, a kd-tree or a brute search is used find the k nearest neighbor. These can heavily influence the search time, depending on the number of instances and number of features. These methods can influence the prediction time heavily but should not influence predictive accuracy. The previously mentioned k parameter is an influencer for prediction quality.[22]

2.1.3 SGDClassifier

SGDClassifier is an incremental function to stochastically approximate the gradient descent of a loss or cost function [20]. The default classifier to optimize is a linear SVM on its loss function. The classifier expects continuous features with a mean of zero and 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. The fitted functions are linear, if the input has multiple target classes a classifier predicts one class versus all other classes.

2.1.4 AdaBoostingClassifier

AdaBoostingClassifier (AdaBoost) is an ensemble classifier that fits other classifiers and outputs the weighted results of those classifiers [13]. 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 [23]. The solution of J. Zhu 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. The standard support vector machine splits for each feature pair, doing this for a large feature space computation grows outside a feasible duration for classification [7]. The feasible duration doing a run of fitting and predicting in a day or two. This fit time is quadratic with the number of samples based on the implementation of libsvm[26]. The fitting of a SVC will assign each example to one of two categories and will map them in a new dimension space. This mapping is to make a clear separation between the two categories. However, with the radial basis function this is done with the distant from the points indicated by a separation area in a multi feature space. Classifying a point is finding in which class area this point falls. The multi class problem is solved by all combinations of classes pairs and the class with the most votes is picked [19]. To optimize the decision surface 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 [16]. Fitting the data, a partial fit function is used based on the work of Chan, Golub and LeVeque [2]. 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. This means that no adjustment to the given input is made on importances for classification. 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 binarization is done by binning numerical values in ranges of their value or splitting categorical values into separate classes for each class their own. This binarization can be disabled or reduced by a threshold on 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

GradientBoostingClassifier builds from other classifiers, which makes it an ensemble classifier, like AdaBoost [17]. GradientBoostingClassifier builds an additive model in a forward stage-wise fashion. This means that the default loss function is optimized in a stage-wise fashion. A logistic regression is created to classify a probabilistic output [15].

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.

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 is mostly used for the relative change of a feature. The amount of noise is mostly a ratio to relate different level of noise disruption.
annotations	Adjectives of something like an algorithm, examples can be robust or biased
bootstrapping	is a test to calculate accuracy on a dataset. The techniques uses random sampling with replacements.
categorical	as property of a feature. This means that the feature has distinct categories or classes. The numerical relation between two categorical values has no meaning.
classifier	machine learning algorithm to specifically predict classes
class	categorical features consist of at least 2 classes
cross validation	is a method to measure accuracy. This is done by splitting the data in k folds. In each fold 1/k of the dataset is tested and the rest is used for classification. The value 10 is chosen for k as this is on average a well working number[5].
datapoint	a datapoint is a single value of a feature. For example a single instance has for all features a single datapoint.
dataset	a dataset are values in a matrix format, where each row represents a single Instance and each column represent all the values of a feature.
dataset manipulation	like the word suggested is the manipulation of a dataset by adding or removing; of features or instances. This influences the internal structure of a dataset.
dense matrix	most values in a matrix are different and fluctuate with each row or column.(non-zero)
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, number of leaves and age. Features describe someone or something in this context. Features are part of a dataset and the input to a machine learner.
feature importance	is a value for all features in a dataset calculated with either RandomForest-Classifier, AdaBoost or GradientBoostingClassifier. The value explains the perceived importance of a feature in a dataset, 100 percent meaning most important and 0 no importance.
fit	preparing the classifier for prediction, synonym with inputting the training data
Github	an online platfrom to host data. It uses git commands and is mostly used with programming project to organize a common project wich each member can locally alter and centralize share updates or modifications.
machine learning	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. Multiple flavors of machine learners exists, for example for regression, classification, clustering.
numerical	exact values, more uniquely than a category. For example a temperature value or time value. Such a value can be subtracted or divided

robustness	The ability of an classifier to cope with changes in the dataset. A classifier is more robust if it deteriorates less than another in a value like predictive accuracy.
sample	A part of a larger something.
scalability	is the capability of a system to handle growing amount of work. Work can be something like more input in or more output out.
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 mostly zero values, the counterpart of dense were all values fluctuate a lot.
target	In a classification problem the value or classes that needs to be predicted. This is value is most likely also a feature of the origin of the data. Each feature in a dataset can be chosen to be the target feature in a supervised machine learning setting. The supervised machine learning has the precondition of knowing what needs to be predicted.
weight	The influence or power of a value, function or object. It can be expressed as a fraction of 1 to indicate 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
sklearn	scikit-learn
std	standard deviation
TU/e	Eindhoven University of Technology
SVC-rbf	Support vector classification with a radial based function kernel

3 Experimental setup

The different experiments use different datasets. The datasets used for some experiments need some prerequiste on the dataset to be used in the experiment. The chosen datasets can have a large impact on shown results. The results are mostly shown as an average result of the datasets involved. This can average out outliers but can also give a false sense of the reality as other patterns can exist in the dataset.

3.1 Datasets

Depending on the property that an experiment focuses on different datasets are chosen or datasets are split on a property. Categorical 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.

3.1.1 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.

3.1.2 Categorical datasets

Categorical datasets hold only features with categorical values. These features are useful for a RandomForestClassifier to make decisions 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.

3.1.3 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 uniqueness of numerical values for predictions.

3.2 Collected data

Experiment data is saved to in 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.

3.2.1 Duration

The duration for each classification instance and for each prediction the time is added to indicate how long the total cross validation took.

3.2.2 Predictions

The predictions of the outputted target value are saved. Individual files are chosen for each case of predictions. The true value for the prediction is also saved as reference and option for recalculation.

3.2.3 scores

The scores give 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

3.2.4 SummaryGuesses

SummaryGuesses give a quick overview of the obtained results. It stores in python dictionaries the total number 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 3.2.2.

3.2.5 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.

3.2.6 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.

3.2.7 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.

3.3 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.

3.3.1 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. Training a RandomForestClassifier to find the feature importance of all features. The next step is to remove the least important features. In the other case of 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 value 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. Most classifiers consider each instance 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.

3.3.2 Duplicate features

Duplicate features experiments have multiple goals in mind. By adding existing feature we can measure scalability of datasets with some number 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 predictive accuracy is measured over the whole dataset by doing a 10 fold cross validation.

3.3.3 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. The accuracy that is measured can explain what the impact is for different classifiers. Two sorts of random features are added; Numerical and categorical features. We add either the categorical or numerical depending on the the dataset. The odds of either a categorical or numerical feature being added is depended on the amount of numerical and categorical features of the initial dataset. The predictive accuracy is measured over the whole dataset by doing a 10 fold cross validation. These experiment can redo the results of Wettschereck et al., Hilario and Domingos and Pazzani [10] [14] [9]. Those works indicated the poor robustness of kNN and the great robustness of Naive bayes solutions.

3.3.3.1 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.

3.3.3.2 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.

3.3.4 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

3.3.5 Noisy data

Datasets can get more noisy over time. During training of a 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. This injection of noise is a replication of an older study of Aha et al [3].

3.3.5.1 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)

3.3.5.2 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)

3.3.6 Bias Variance

To measure the bias and variance error factor we use the calculation of Kohavi and Wolpert[8]. This is the same measure used in the work of Joaquin et al.[27]. 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.

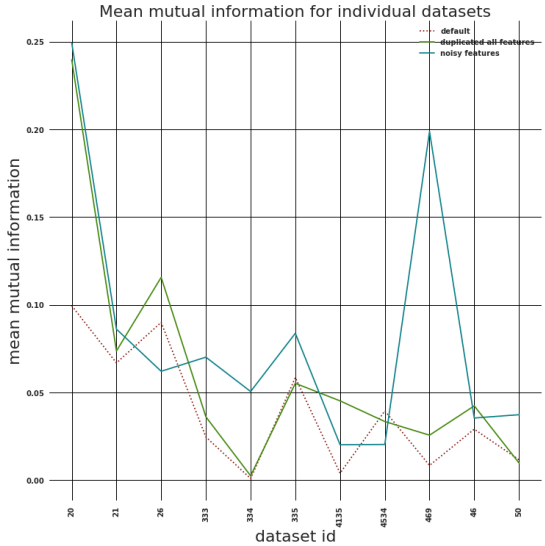
3.3.7 PreProcessing

Preprocessing is a necessary job for most 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 [4][18]. The proposed preprocessing steps are OneHotEncoder and Standardscaler in that order. OneHotEncoder translates the categorical features in features corresponding to the number 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.

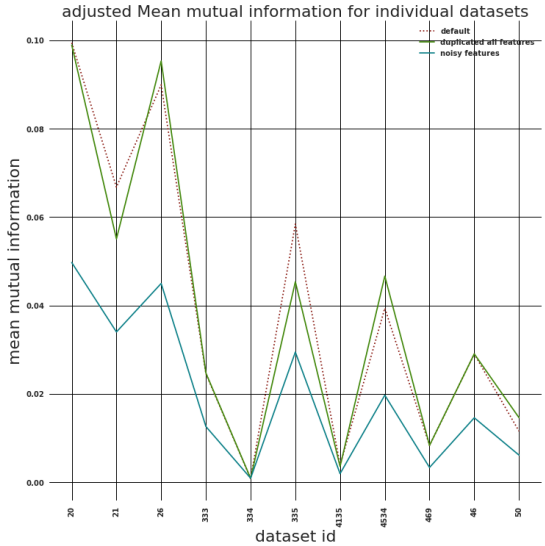
3.4 MetaFeatures

3.4.1 Mean Mutual Information

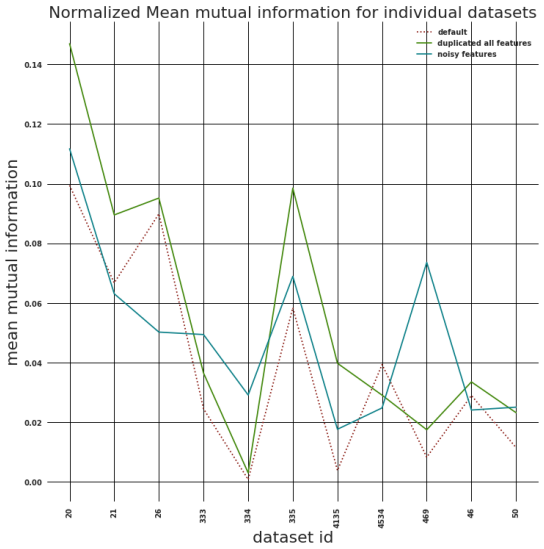
Meta features like mean mutual information or entropy for categorical features are calculated for our enhanced dataset with duplicate feature and random features. The result is that they do little to change these values and so do not indicate reduced information even tough commonly the results deteriorate 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.



(a) Mean Mutual Information for manipulation of datasets



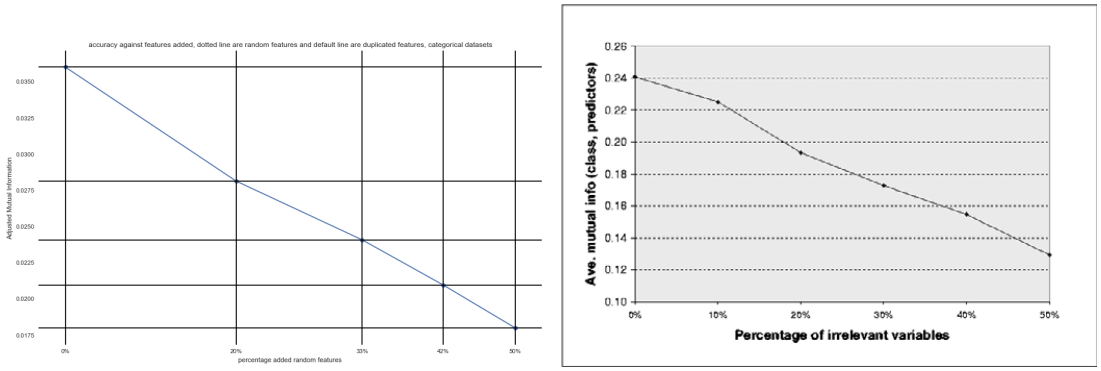
(b) adjusted Mean Mutual Information for manipulation of datasets



(c) normalized Mean Mutual Information for manipulation of datasets

Figure 2: Mutual information difference between measure

In figure 3 the experiment setup of Hilario and the experiment setup of this thesis are shown[14]. Hilario shows 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. Comparing any results is harder as the versions of the classifiers have been changed over time.



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

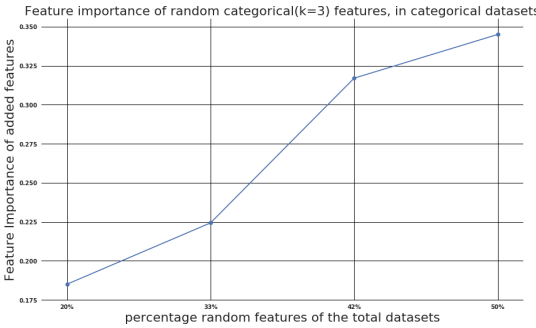
(b) average Mutual Information as a measure of relevance

Figure 3: Mutual information decline between previous work and this work

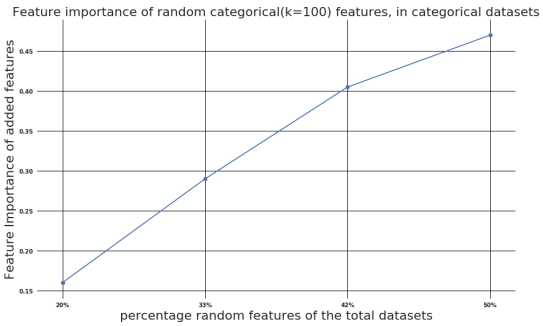
3.4.2 Feature Importance

Feature importance calculated with a RandomForestClassifer gives an indication what features are important for the decision trees. In figures 4 and 5, the feature importance of the added random features are shown. This makes it clear that RandomForestClassifier does not recognize these features as unimportant. 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 number of features. Random Forest Classifier has a limit on the number 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 4,6 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 duplicate features having importance is because they hold the same value of information, the order of the dataset is the factor that makes those features not equally important. The end results is 47% feature importance for duplicate features when the dataset is half duplicates. The feature importance of the GradientBoostingClassifier seen in figure 7 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.

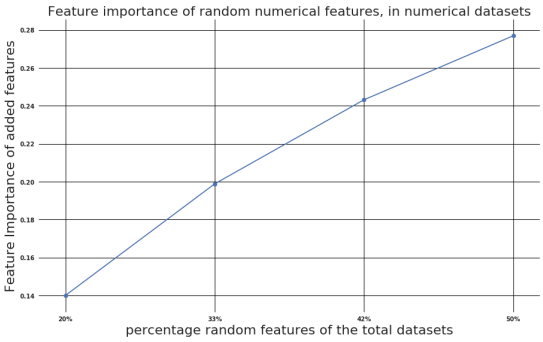


(a) Feature importance for random categorical features with k = 3.

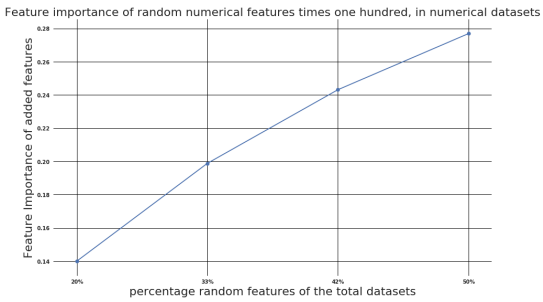


(b) Feature importance for random categorical features with k = 100.

Figure 4: Feature importance of random categorical features. The k value has an impact on the perceived importance of the features.

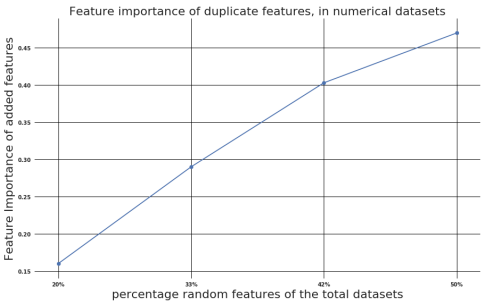


(a) Feature importance for random numerical features with the default uniform spread between 0 and 1.

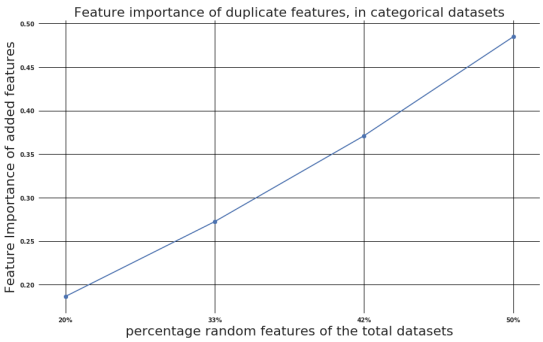


(b) Feature importance for random numerical features which are multiplied by a 100.

Figure 5: 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 6: Feature importance of duplicate features and their effect on categorical or numerical datasets

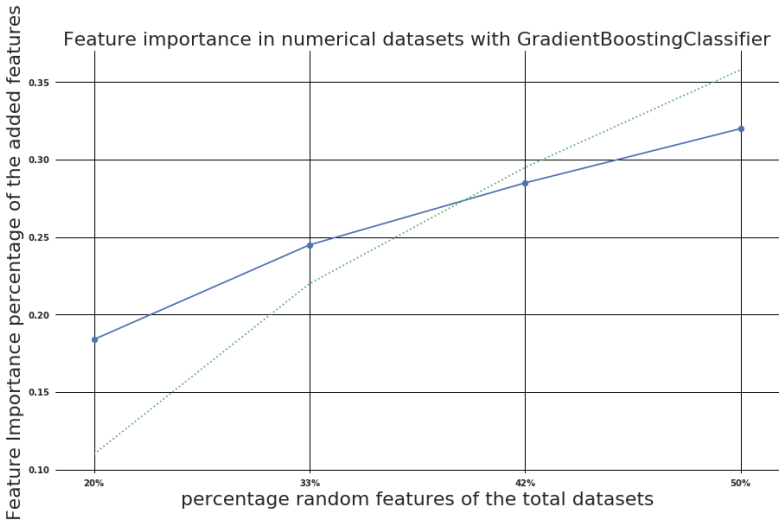


Figure 7: Feature importance for duplicate features and random features calculated with a GradientBoostingClassifier instead of the RandomForestClassifier on a numerical dataset

4 Experimental Results

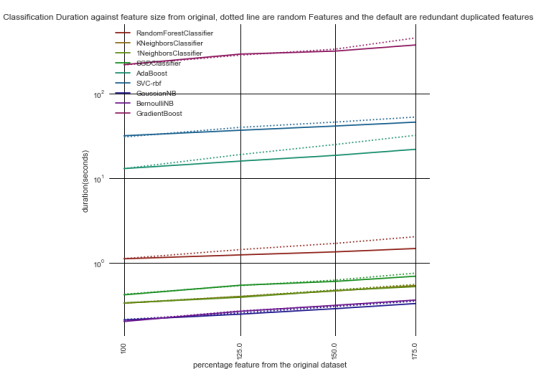
4.1 Scalability

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.

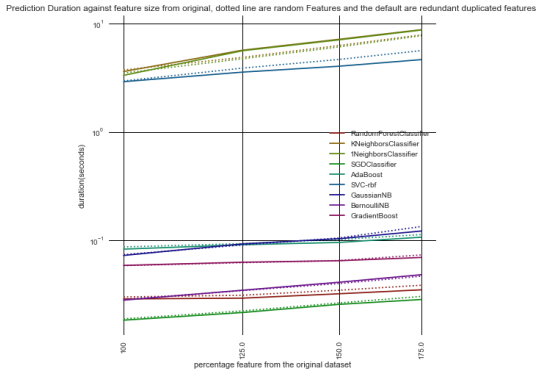
4.1.1 features

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.

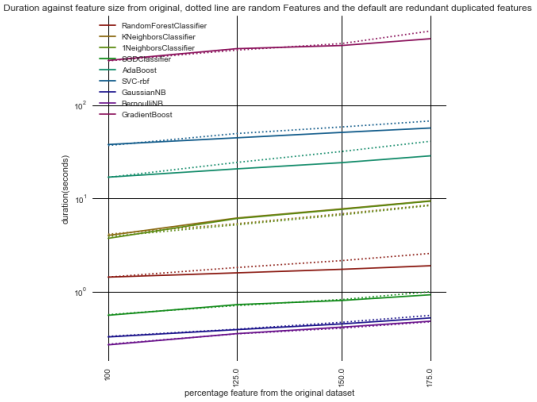
4.1.1.1 Added features In figure 8 the durations of adding redundant duplicate or random features are shown. In figures 9 the durations of adding duplicate or random features are shown In figures 8a, ,9a the classification times are shown for a growing feature size, these durations are a great part of most classifiers. In figures 8b, 9b the prediction time is shown, here most classifier take only a short while. The combined time can be seen in figures 8c, 9c which shows the total difference. The averaged size of this dataset is 3797 instances and 200 features. he slope of these added figures is seen in figure 10.



(a) only the time needed to fit

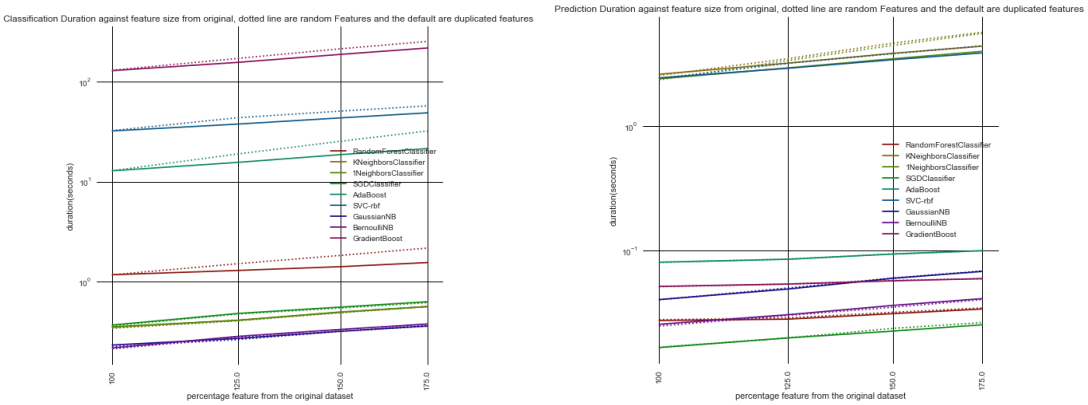


(b) only the time needed to make predictions



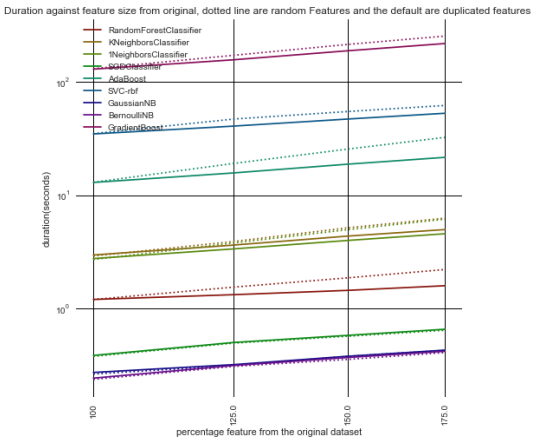
(c) The combined time of predicting and classifying on a dataset with added random and redundant duplicated features

Figure 8: 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 9: 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 section3.4.2 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.

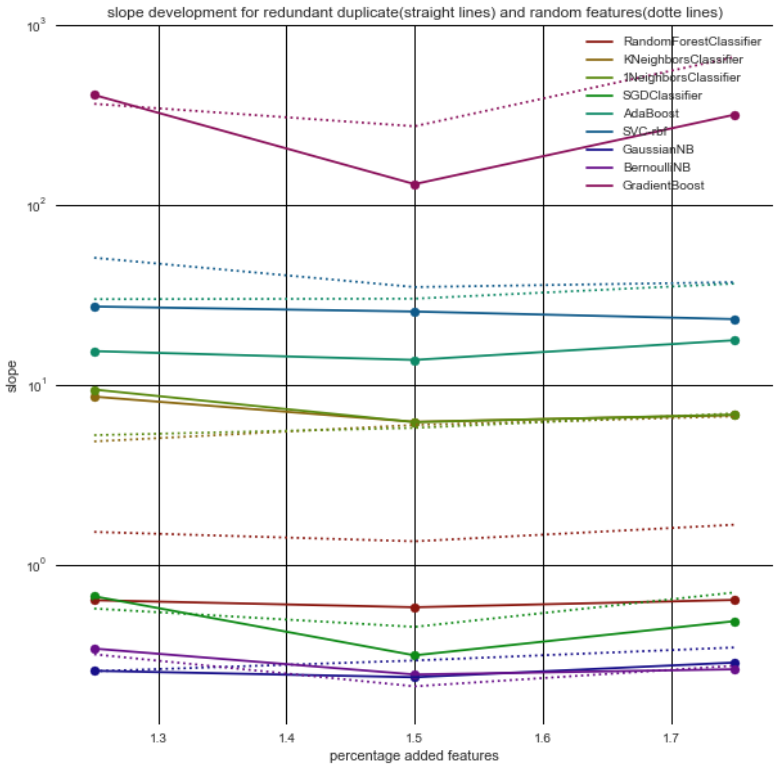


Figure 10: Slope for 8c with the dotted line for random added features and the straight line for redundant duplicate features

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.

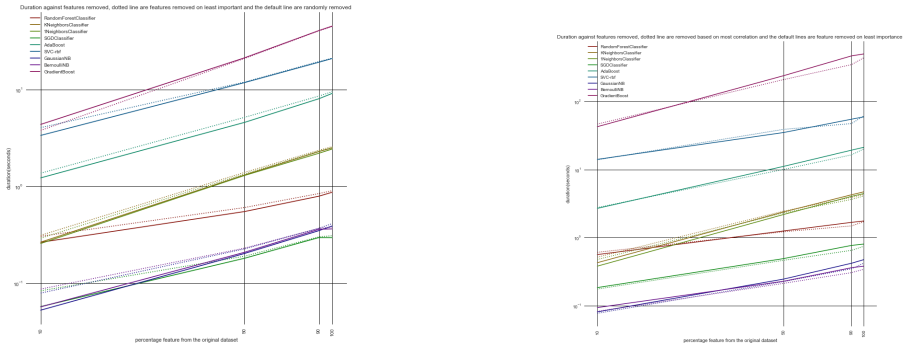
SVC-rbf SVC-rbf has one of the slowest classification and prediction times. With the calculation of the decision surface taking the most time. The duration of prediction is also slow in comparison to the other classifiers. The impact of the random features is noticeable with the variation present in 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 needs the most time of all classifiers for classification and prediction. 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

4.1.1.2 Removed features In figure 11 the duration against features removed from a dataset are shown. Three different techniques of feature removing are shown. In figure 11a features are randomly remove or a RandomForestClassifier is fitted on the training set and the least important feature is removed. In figure 11b the features are removed by their importance or by having the most correlation with another feature in the training set. In figure 11a the average dataset is 2230 instances with 307 features and in figure 11b 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 11: 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 KNeighborsClassifier need more for least 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. This is also the case for important features 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 11a but a bit more time consuming in figure 11b. 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 GradientBoostingClassifier 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 12 adding and removing features is shown. in figure 13 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 13b has the most balanced approach as basing the calculation on the clean dataset reduces the influence of outliers. The slopes in figure 10 are also considered in these results.
Most classifiers paint a clear picture in the results. 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 13b 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. In the case of SGDClassifier nearly no difference in duration between 90% and 100% features of the dataset is found and can be seen as the slope dropping outside of figure 13a. Most classifiers have a similar but less drastic drop. The drops can be seen in figure 13b as being low. This is because the difference between instances is small and the duration calculation is inaccurate. With the short duration needed for training and prediction in the SGDClassifier the difference can be larger between the results.

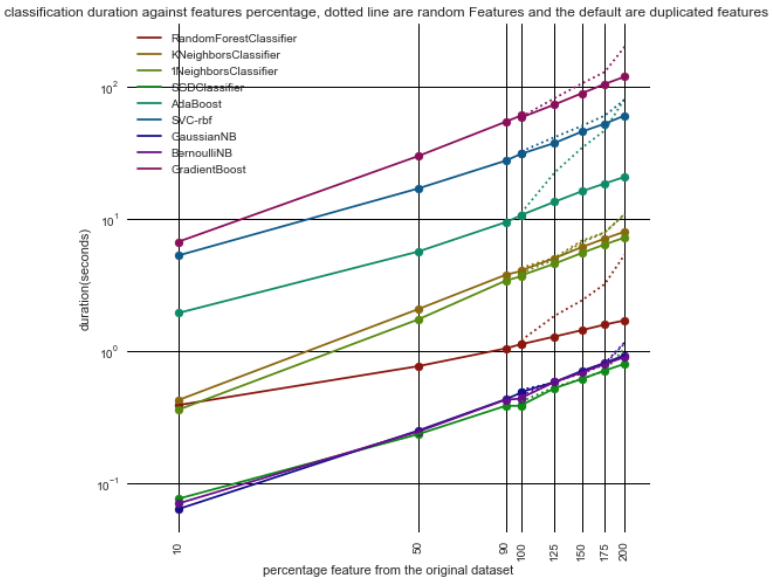
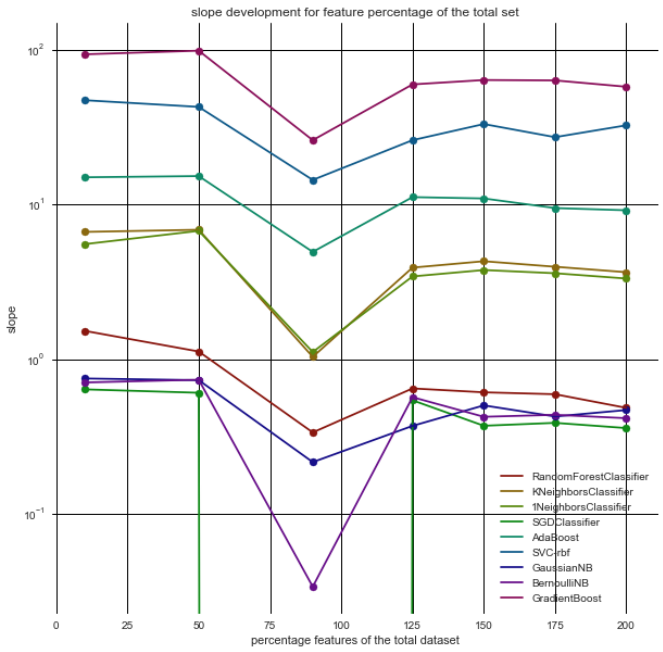
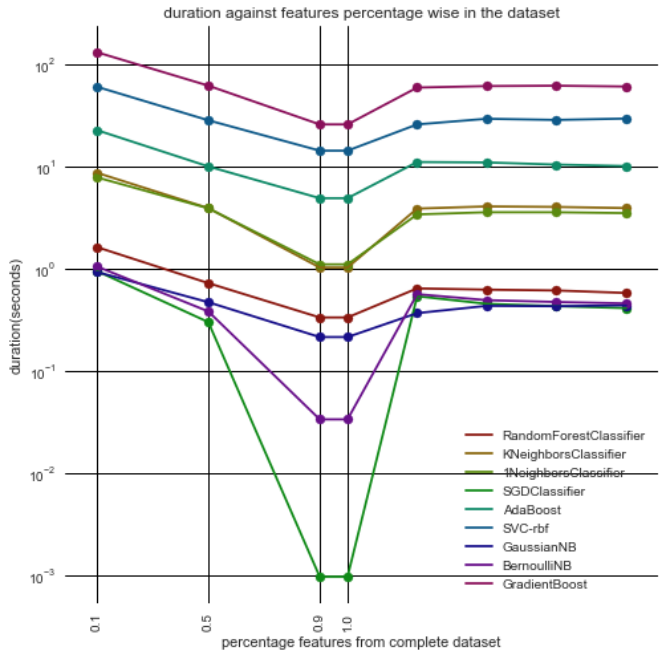


Figure 12: features removed randomly and features redundantly duplicated(straight lines), randomly added (dotted line)



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

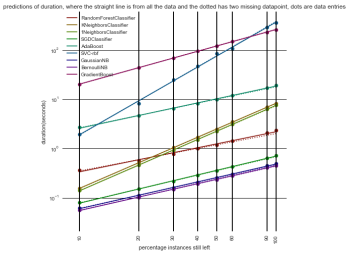


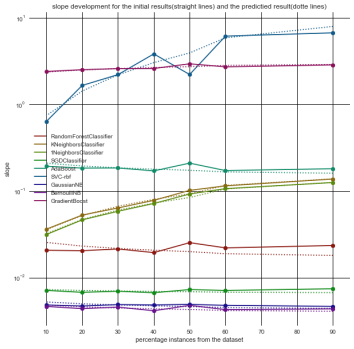
(b) features removed randomly and features duplicated slope based on the average point of a clean dataset, with a log scale

Figure 13: Slopes of Feature percentage total duration of the original dataset calculated from figure 12.

4.1.2 Instances

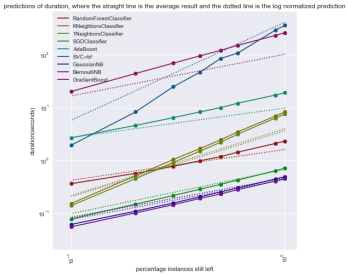
In this sections datasets are split reduced in the instance dimension. First the set is shuffled and then a percentage of the dataset is cross validated. In figure 14 the results can be seen. In these results the scalability in the instance dimension of the classifiers is shown. The averaged dataset is 7786 instances with 154 features for figure 14
Comparing the lines visible in figure 14a with the actually slope in figure 14b the classifiers GradientBoostingClassifier, RandomForestClassifier, AdaBoost, 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 number of instances. 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 14a the predictions lines for most classifiers, line up with the actually results. Together with the slope seen in figure 14b 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 14c as the dotted line, that line clearly does not line up with the averaged result. Visible 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 are fitted on these dots. The scale is log based as input.

(b) The default line is the slope of the experiments and the dotted line is a prediction line with figure 14a lines are fitted on these dots.



(c) The default line are the result of experiments done, the same as the ones in figure 14a. The dotted line is prediciting by a KernelRidge for each inputted dataset the trend line of the duration.

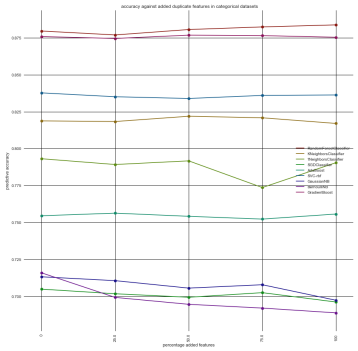
Figure 14: Instances scalability, the slope and the prediction attempts.

4.2 Redundant features

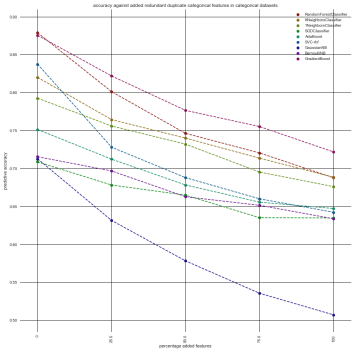
This section focusses on feature manipulation of clean datasets and the accuracy impact. By either removing or adding features the dataset changes and maybe also the workings of classifiers. By comparing the results of a clean dataset to a manipulation in the feature dimension we see how classifiers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

4.2.1 Adding features

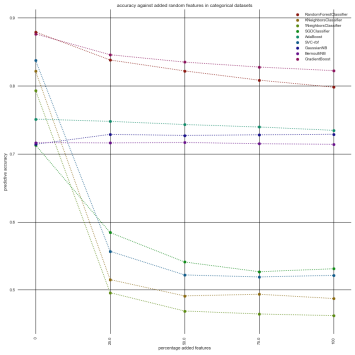
We consider here random, duplicate and redundant duplicate features as they are all redundant. In section 3.4.2 the importance and in section 3.4.1 the mutual information is shown which changes with the addition of certain features.



(a) Adding increasingly more duplicate features to categorical datasets

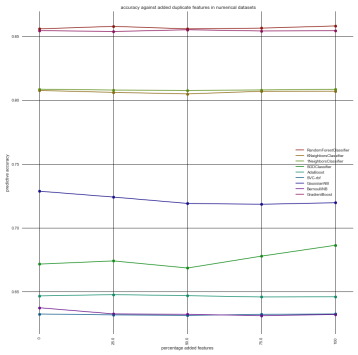


(b) Adding increasingly more redundant duplicate features to categorical datasets

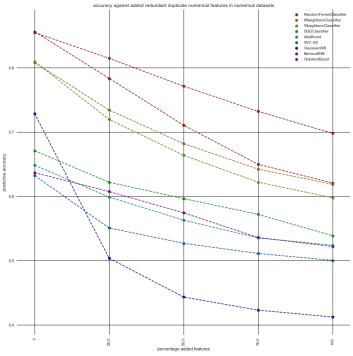


(c) Adding increasingly more random features(k = 100) to categorical datasets

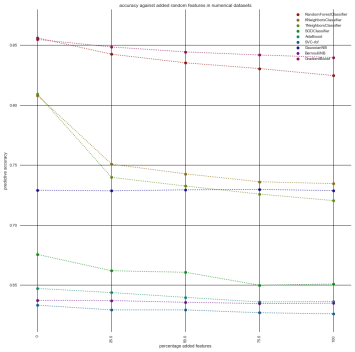
Figure 15: Predictive accuracy of categorical datasets injected with irrelevant or redundant features



(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 16: 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 3.4.2 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 duplicate 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 dependence 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 Similar 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 performance 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.

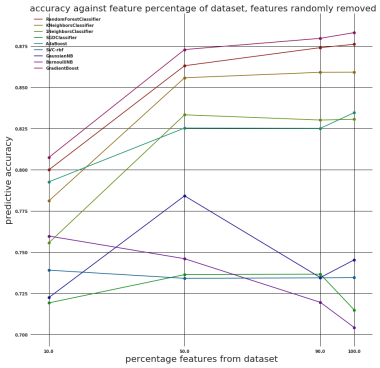
GradientBoostingClassifier The effect of duplicate features has a similar story to both AdaBoost and RandomForestClassifier in other words no effect. The decision trees handle the duplicate features as a bit less important than the original dataset(figure 7). This explains a less decrease in accuracy 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

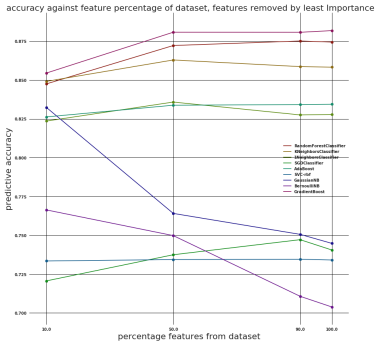
Table 1: all changes in predictive accuracy in adding both kind of features(results from figures 16 and 15)

4.2.2 Removing Features

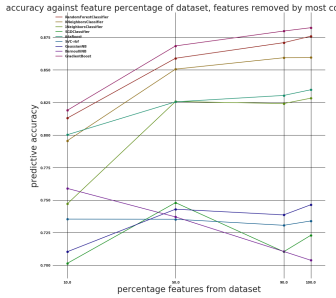
Randomly removing features, based on least importance or based on most correlation. The accuracy against the feature left is shown.



(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 17: 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 similar 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. Similar 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 noticeable 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 interesting behavior of fluctuating between improving or deteriorating for randomly removed features. This means that similar for SVC-rbf that most features have an similar 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 features 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 similar 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 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.

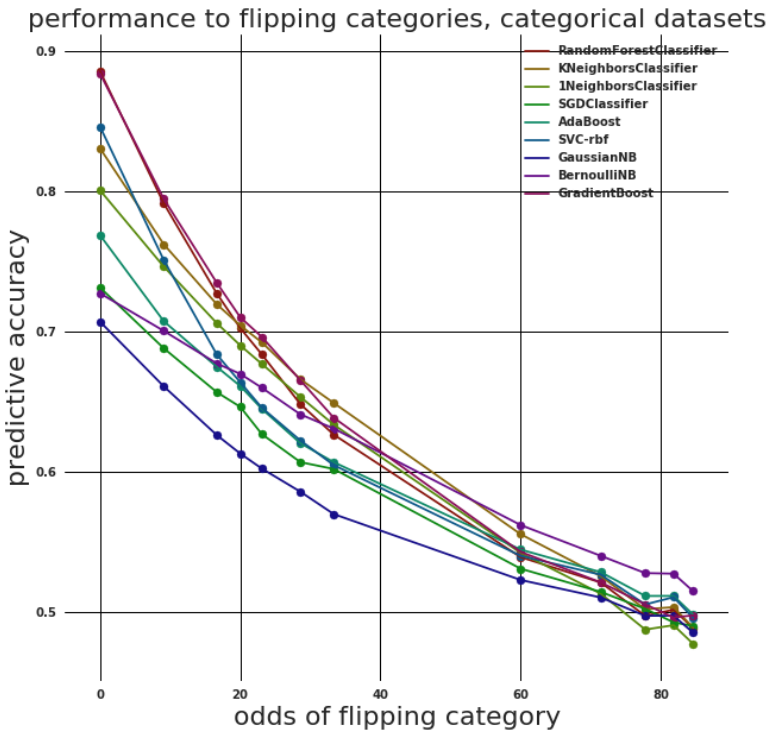


Figure 18: flipping categories for categorical features

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 SGDClassifier. Considering that AdaBoost is not optimized and that the datasets are not all particularly

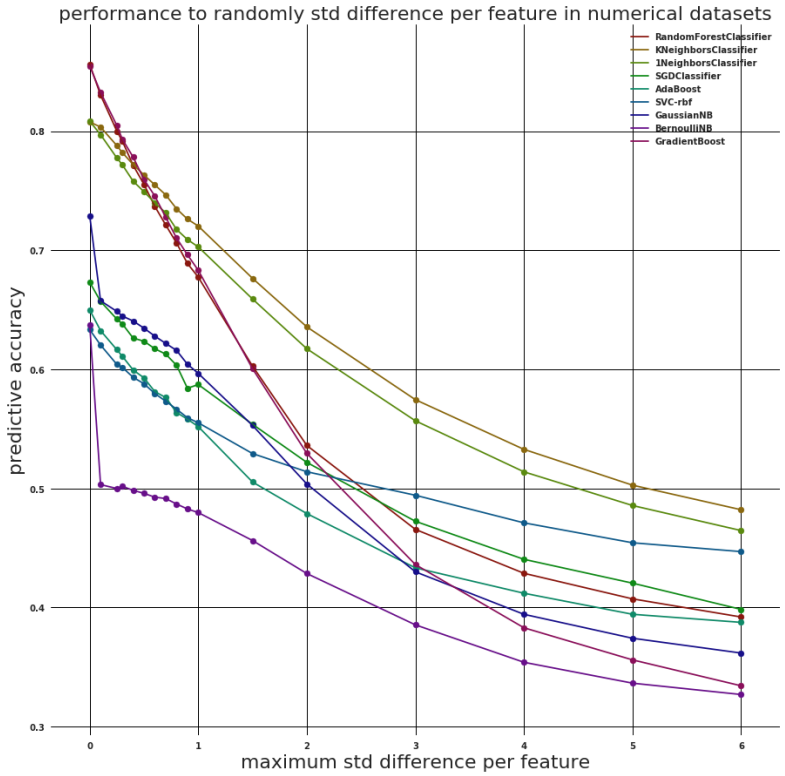


Figure 19: adding or removing uniformly random std to numerical features

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.

GradientBoostingClassifier GradientBoostingClassifier has one of the best predictive accuracy on

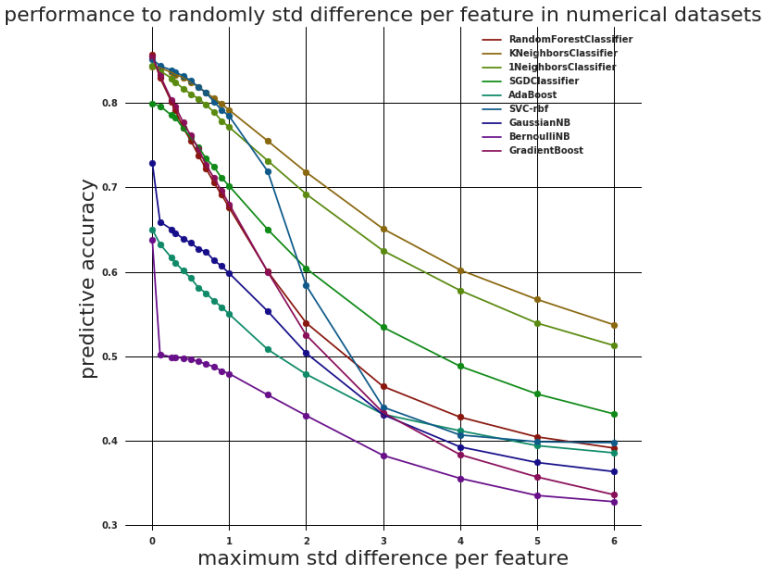


Figure 20: adding or removing uniformly random std to numerical features and preprocessing for some classifiers.

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 21 the bias percentage is given against the instance percentage of a dataset. The averaged dataset has 3186 instances and 160 features. In figure 22 the accuracy of the same experiments as figure 21 can be seen. The result of these bias variance experiments indicate little of a repeat of previous work. The most prominent reason is the less number of instances considered. This means the results are the comparable to the varying bias percentages in the below 10000 instances case. The only clear results are the high bias percentages of SVC-rbf, GaussianNB and BernoulliNB, the low bias of the SGDClassifier and the increase in bias of AdaBoost and GradientBoostingClassifier. The high bias percentage overlap with previous results and the other results are not yet researched.

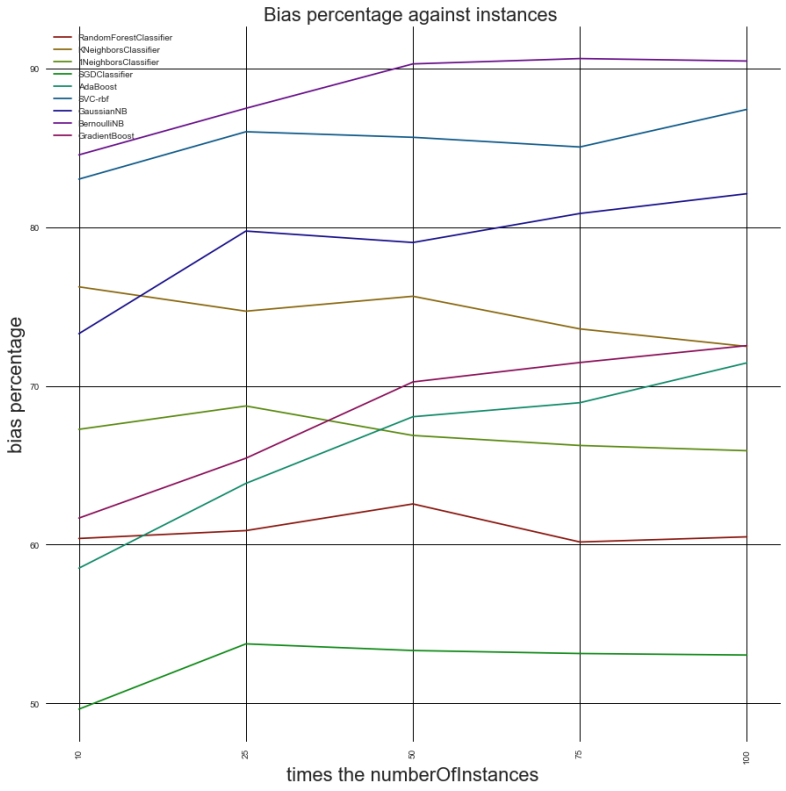


Figure 21: Bias percentage against percentage of instances from a dataset.

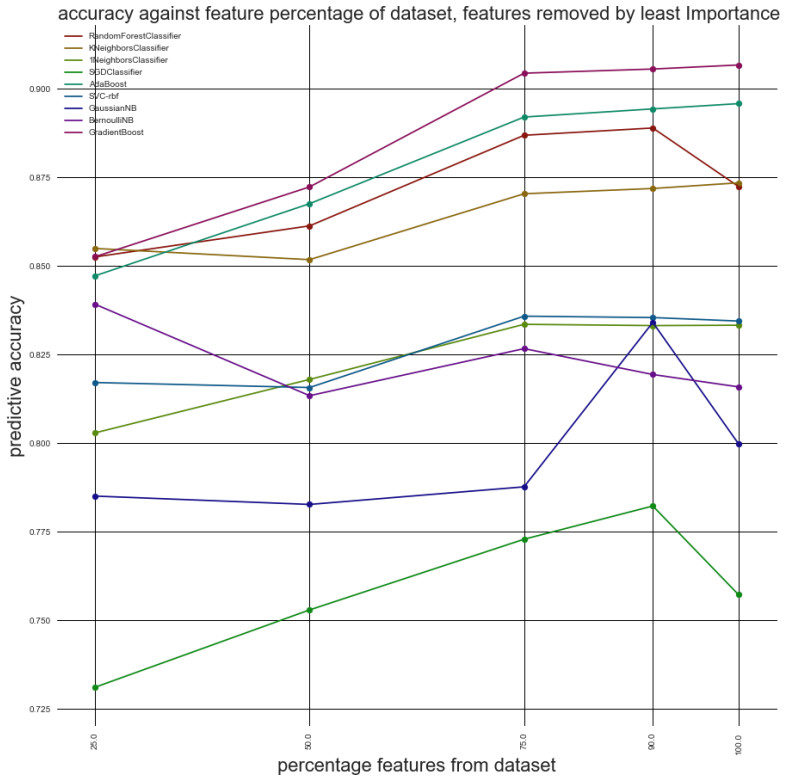


Figure 22: Bias accuracy against percentage of instances from a dataset.

5 Conclusion and discussion

5.1 Discussion

This section discusses mistakes, opportunities and pitfalls.

5.1.1 Missed opportunities

More options exist to save more data of the executed experiments. The first missing data is probability predictions. These predictions come from some of the classifiers which work with predicting based on the highest probability. This means that for each instance the probability it is a certain target class is calculated. The second missed opportunity of 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 1NN 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. One of the reason for these differences in duration is because of multiple used processing systems and other processes running during computation. These other process or the experiments themselves maxed out the cpu or memory of the systems and so interrupting the experiments.

5.1.3 Bias Variance

The datasets used by Joaquin in [27] were UCI datasets, some of these are also published on OpenML under the dataset ids (1369-1377). These are artificially created to have datasets with a large number of instances.

5.1.4 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. Changing all features for some datasets meant a drop in accuracy but the continuation of upping the number meant little to change accuracy further. In figure 26 the predictive accuracy of adding some flat amount to the features is shown. The resulting accuracy of most classifiers drops significantly in case of the addition the only classifiers with a steady accuracy is AdaBoost. The steady line of accuracy for AdaBoost is because of the focus AdaBoost has on misclassification. The sample size of this experiment is too small to further assess the failing results of the other classifiers.

5.2 Conclusion

To answer the research from the results of the experiments, a ranking is chosen. The different classifiers are compared to each other in this way to give some indication of the impact. These rankings can be compared to results from a review from 2007 of classification techniques [21] The result will be a ranking of classifiers on different scales shown in figure 23

	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 23: rankings of all experiment results compared to each other. A plus signs means the degree that it needs something and minus indicate the lack of needing it.

The impact of features on runtime for most classifiers is near linear with for all but kNN an increase with random features. The impact of features for kNN is depended on the values used for fitting and the values for predicting. In the case these values differ largely the duration is longer in comparison. The other classifiers have more problems with the variation in random features gives an increase in classification duration as that is the learning way of those classifiers.
The impact of instances on runtime for most classifiers is near linear except for kNN and SVC-rbf. With a calculation based on multiple runs over the input set do kNN and SVC-rbf need more than linear time for prediction and classification. The other classifiers might have linear increase in duration but the differences are exceedingly large between classifiers.
The end results of durations indicate that the Naive Bayes classifiers and SGDClassifier are the quickest in classification with a runner up of the RandomForestClassifier. The KNeighborsClassifier and AdaBoost are behind RandomForestclassifier in average lowest duration. However the factor between RandomForestClassifier and AdaBoost is already around 10 times slower. The remaining two classifiers GradientBoostingClassifier and SVC-rbf are the slowest of them all with GradientBoostingClassifier taking around 200 times longer than RandomForestClassifier for the duration experiments.
The impact of irrelevant features or redundant features for predictive accuracy is highly depended on the type of features. Straight duplicate features are not really a concern for most classifiers except SGDClassifier. This classifiers takes all features into account and can so gain or lose accuracy depending on the duplicate feature. Redundant duplicate features has a bad impact on all classifiers as they are not

identified as redundant internally. The classifier most depended on all features like RandomForestClassifier or GaussianNB are most impact by these features. Lastly random features has the most flavors of impact of all of them. The Naive bayes classifiers, GaussianNB and BernoulliNB gain little or lose little accuracy. kNN classifiers is the worst off with a deep impact with both types of random features. Classifiers with a bit better results than kNN are SVC-rbf and SGDClassifier who lose a lot of accuracy from categorical random features but lose little from numerical random features. The remaining classifiers RandomForestClassifier, AdaBoost and GradientBoostingClassifier lose only some accuracy based on the total number.
The impact of noisy data for predictive accuracy is similar for most classifier as a reduction in accuracy. Classifiers of the Naive Bayes variant GaussianNB and BernoulliNB are not resilient against the numerical noise, they depend on close to categorical translation. Classifiers kNN and SVC-rbf are most resilient, with SVC-rbf being only resilient for a single std. The close runner up in robustness against noisy data are classifiers AdaBoost and SGDClassifier with some more resilience than most others. The classifiers kNN, SVC-rbf and SGDClassifier do benefit from some preprocessing on the data to better reflect their impact. Lastly the classifiers RandomForestClassifier and GradientBoostingClassifier are left as being not very robust to the noise but more than the Naive bayes's classifiers.
The percentage of bias error for each classifiers can be read from figure 21 however, these results are not on a large enough scale to give conclusive evidence this is ranking between the algorithms.
In total these results illustrate the 'no-free-lunch' theorem as no classifier has the best quantitative properties of all the classifiers. The easy algorithms of Naive bayes, BernoulliNB and GaussianNB are the most scalable together with the SGDClassifier. kNN is most robust against noisy numerical features and the Naive bayes are also most robust against added random features. In perspective this robustness of the Naive bayes classifiers may not be worth that much as their predictive accuracy is nearly always out shined by classifiers like RandomForestClassifier or GradientBoostingClassifier.

5.3 Future work

Adding more experimental runs to most of the experiments can give way to more broader conclusions of the results. As benchmark datasets have a hard time to be a universal standard for testing. The results of a universal standard benchmark may also limit results because of the no free lunch theorem. This may result in an average result of shared robustness.
A source of attention can be the use of only datasets that are available on OpenML are being used for this research. 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 the research nature of OpenML.
A partly solution for the previous mentioned problem of datasets is the splitting of datasets in certain categories. In this thesis there was for some cases 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] Nearest neighbor pattern classification, T. Coverm P. Hart, IEEE Transaction on Information Theory 13:21-27,(1967)

[2] 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)

[3] Instance-based learning algorithms, Aha, David W., D. Kibler, Marc K., Machine learning 6:1 pages 37-66, (1991)

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

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

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

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

[8] 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)

[9] On the optimality of the simple Bayesian classifier under zero-one loss,P. Domingos, M. Pazzani, Machine learning 29 2-3 pages 103-130, (1997)

[10] A review and empirical evolutions of feature weighting methods for a class of lazy learning algorithms, Wettschereck, D. Aha, T. Mohri, Artificial Inteligence Review 11(1-5) pages 273-314, (1997)

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

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

[14] 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)

[15] Greedy function approximation: A gradient boosting machine, J. H. Friedman, The annals of statistics volume 29 issue 5, pages 1189-1232,(2001)

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

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

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

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

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

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

[22] 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)

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

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

[25] 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)

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

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

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

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

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

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

[32] An empirical study of hyperparameter importance across datasets, Jan van Rijn, Frank Hutter, Proc. of AutoML : 97-104, (2017)

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

[34] Choosing the right estimator, l,Scikit-learn, Retrieved from http://scikit-learn.org/stable/tutorial/machine_learning_map/index.htm on 28th March (2018)

7 Appendix

7.1 Datasets

7.1.1 Datasets per figure

OpenML dataset identifiers per figure

figures 2, 3a , 4, 6b, 15, 18, 25 - 20, 21, 26, 333, 334, 335,40668, 4135, 4534, 469, 46, 50
figures 5,6a, 7, 16, 19, 20 - 1038, 1043, 1046, 1049, 1050, 1063, 1067, 1068, 1176, 11, 12, 1459, 1462, 1464, 1466, 1467, 1468, 1475, 1476, 1478, 1479, 1485, 1487, 1489, 1491, 1492, 1493, 1494, 1497, 14, 1501, 1504, 1510, 1515, 1570, 16, 18, 22, 28, 300, 30, 32, 36, 375, 37, 39, 40499, 40509, 40, 4134, 41, 44, 4538, 458, 53, 54
figures 8, 9, 10, 24 - 11, 12, 1038, 14, 16, 18, 1043, 20, 21, 1046, 22, 1176, 1049, 1050, 26, 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, 333, 1487, 334, 1489, 335, 1491, 1492, 1493, 1494,

469, 1497, 1501, 1504, 1510, 1515, 375
figure 12 and 13 - 22, 1176, 4134, 1063, 1067, 1068, 44, 53, 4538, 458
figure 14 - 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
figure 15 - 20, 21, 26, 333, 334, 335, 40668, 4135, 4534, 469, 46, 50
figure 16 - 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
figures 17, 11 - 1038, 1043, 22, 1176, 1049, 1050, 1063, 1067, 1068
figures 22,21 - 1025, 1067, 1036, 1068, 1038, 1043, 1014, 1049, 1050, 1022
figure 26 - 10,12,18
figure 27 - 11, 12, 37, 54, 458, 1038, 1043
figure 29 - 40706, 6, 1036, 1038, 1043, 1046, 1176, 1049, 1050, 1570, 4134, 1063, 40, 1067, 1068, 44, 46, 53, 4534, 54, 953, 4538, 26, 60, 1489, 458, 40668, 40670, 478, 1120, 40677, 40680, 40686, 40687, 40702, 40695, 40696, 40697, 1022 figure 30 - 1038, 1043, 1049, 1050, 1176, 12, 1466, 1468, 1475, 1476, 1478, 1479, 1485, 1487, 1491, 1492, 1493, 1494, 14, 1501, 1504, 1510, 1515, 16, 22, 28, 300, 40499, 40, 4134, 44, 4538, 458

7.1.2 Values

In figure 24 the number of features and instances for datasets used in figure 8 is shown. This shows the large amount of datasets in the range before 2000 instances and the outliers in the feature dimension. The outliers in the feature dimension seem to have hundred times more features than most dataset.

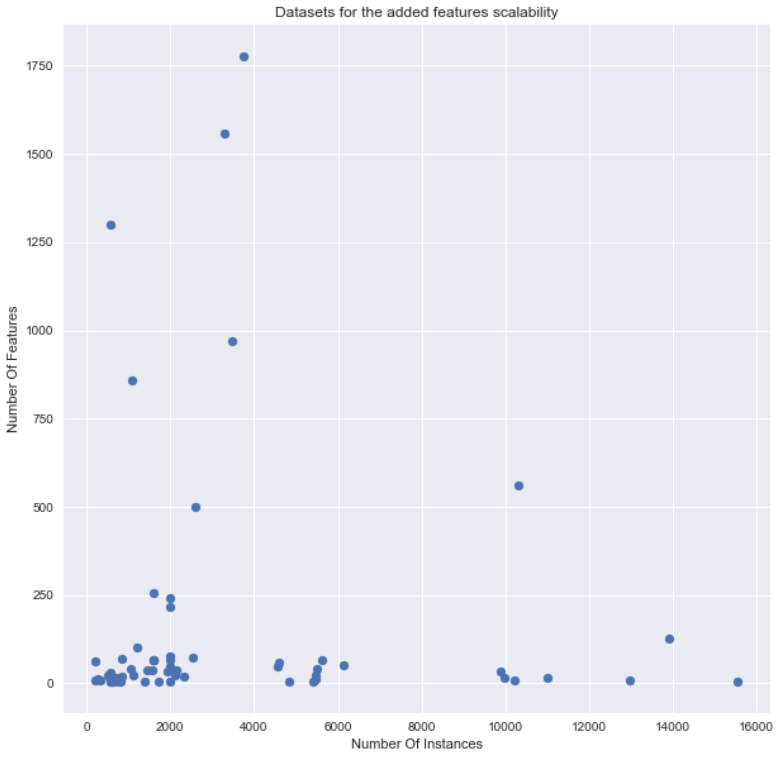


Figure 24: Dataset metafeatures for figure 8

7.2 Different approach

In figure 27 classifiers are optimized on their hyperparameters, based on the work of van Rijn and Hutter [32]. The classifiers SVC-rbf, AdaBoost and RandomForestClassifier were analyzed to find the most important hyperparameters to optimize on. In the work of Islam the importance k parameter in a kNN classifier is mentioned and so this value is tuned to increase accuracy[22]. In figure 25 the same experiment as in figures 15c, 15b is done but with instead of categorical random features we add numerical random features. The difference in result is mostly for the large losers in accuracy. The kNN, SVC-rbf and SGDClassifier lose nearly no accuracy, kNN even gains a little predictive accuracy with the added numerical features.

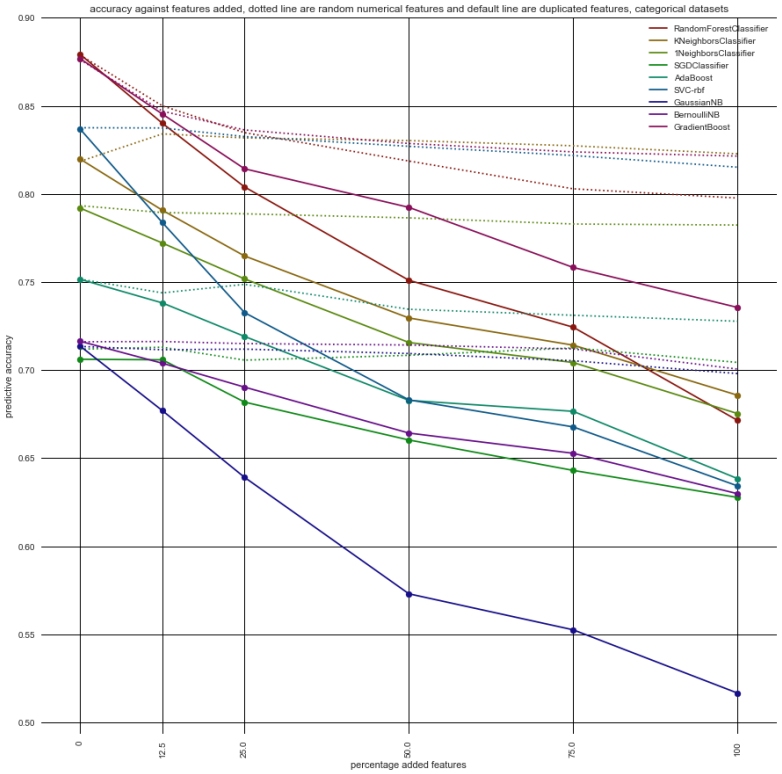


Figure 25: adding numerical random features or redundnat duplicate features for a categorical dataset.

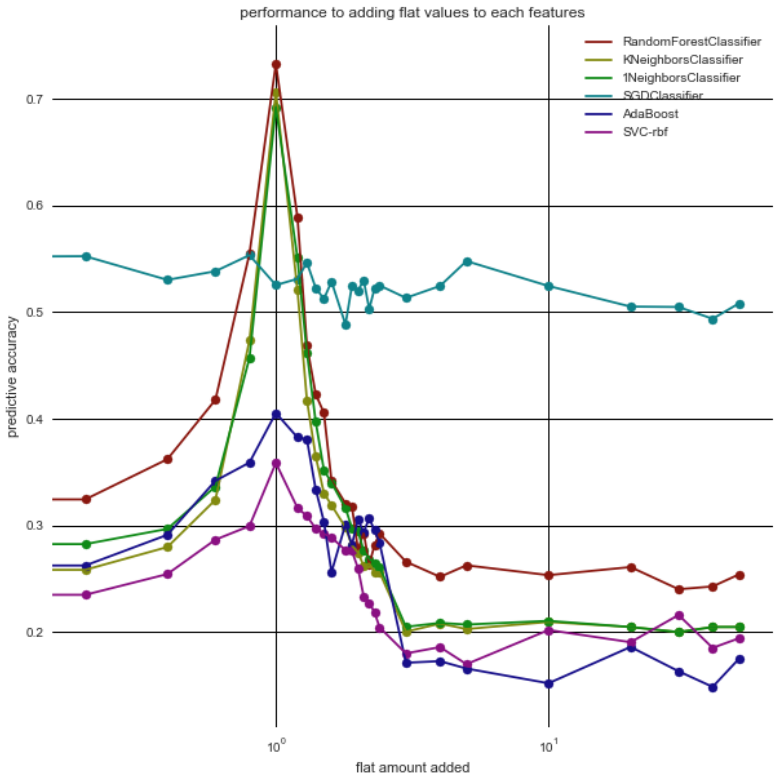


Figure 26: adding flat values to each feature

7.3 Duration variance

In this section duration examples are shown which show behavior that should not happen. In figure 28 the duration against the percentage of features of the total dataset is shown. The problem in this figure is the lack of overlap between the zero points of 100 % percentage features. The classifiers AdaBoost, kNN and RandomForestClassifier have no overlap on that point. This means that the average duration does not match. The duration of added features is done in the same time frame as a run on the clean dataset. This means that all the results of these classifiers have a different duration multiplier. The reason for this can be a busy computer or false information. In figure 29 the duration against adding duplicate features or random features is shown. The results are not normal as no line is having similar behavior in previous work. Figure 9c shows the expected behavior. The spike in duration for kNN cannot be weird coincidence of k features lining up to be zero distance from a point over multiple datasets. These odd duration values are because of a form of outside influence. In figure 31 an even more extreme case of this is shown. In this case not even the average align giving an easier explanation of a difference in duration registration. Lastly in figure 30 a mismatch between two different sort of feature manipulation against duration is shown. The lines of feature removed from the dataset and feature added to the features do not algin for any example. This is a clear indication that there is a mismatch between the duration calculation of feature removing and feature addition. In figure 12 the more expected behavior of the average duration is shown. The main difference between the two is that figure 30 uses more datasets in the calculation. This may mean that the duration of a clean dataset is influenced by the duration of the manipulated dataset.

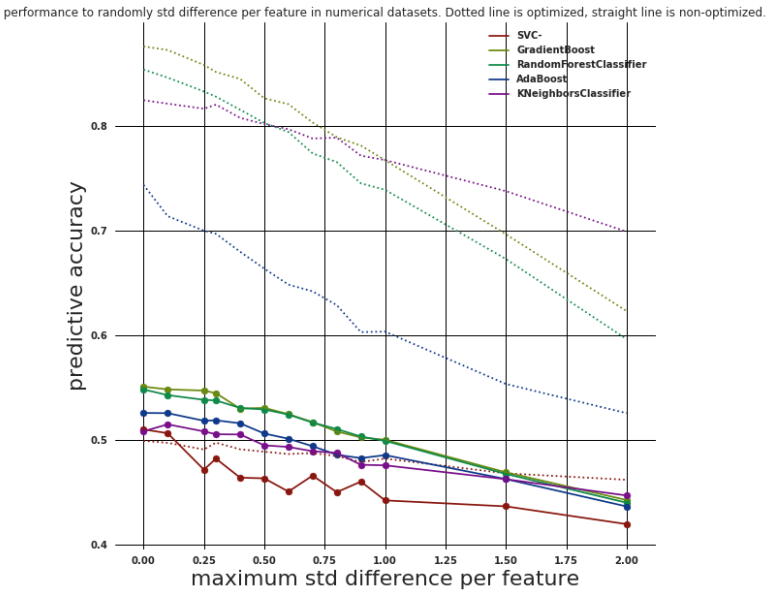


Figure 27: predictive accuracy against numerical datasets with std difference, hyperparameter optimization(dotted line), default settings(straight line)

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 done in figure 19. In figures 32, 33, 35, 36, 37, 38, 39, 40 the individual results of each classifier is shown.

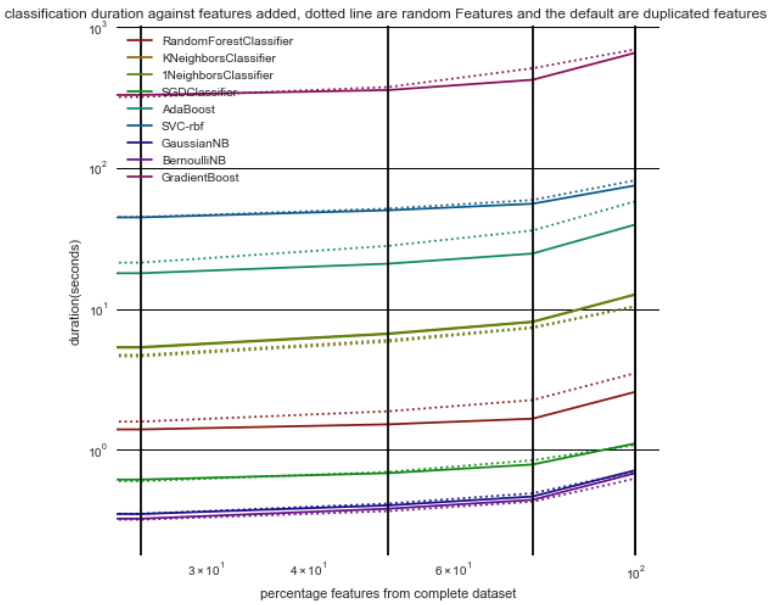


Figure 28: Duration of the oldest version of adding random features

7.4.2 Instance duration

Percentage of instances remaining of a dataset per individual dataset with the average result in figure 14. In figures 41, 42, 44, 45, 46, 47, 48, 49 the duration of cross validation in each individual dataset for each classifier.

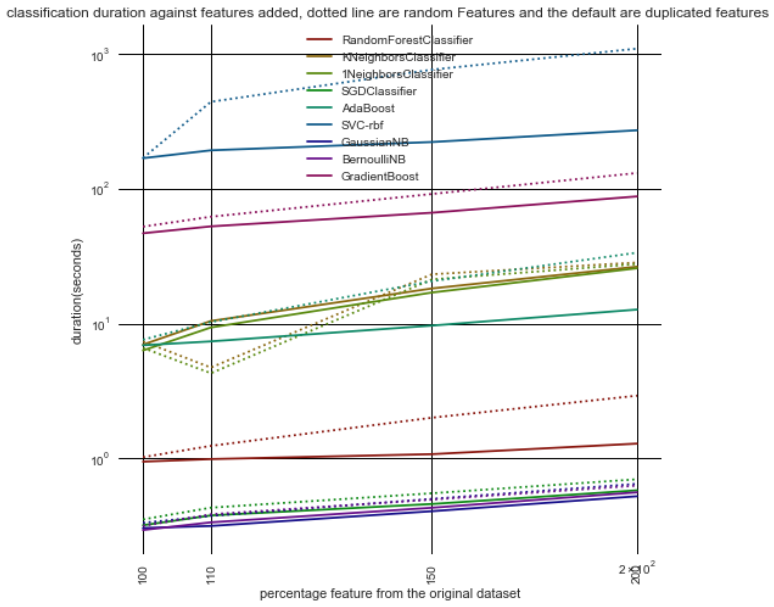


Figure 29: Duration of the newest version of adding random or redundant features

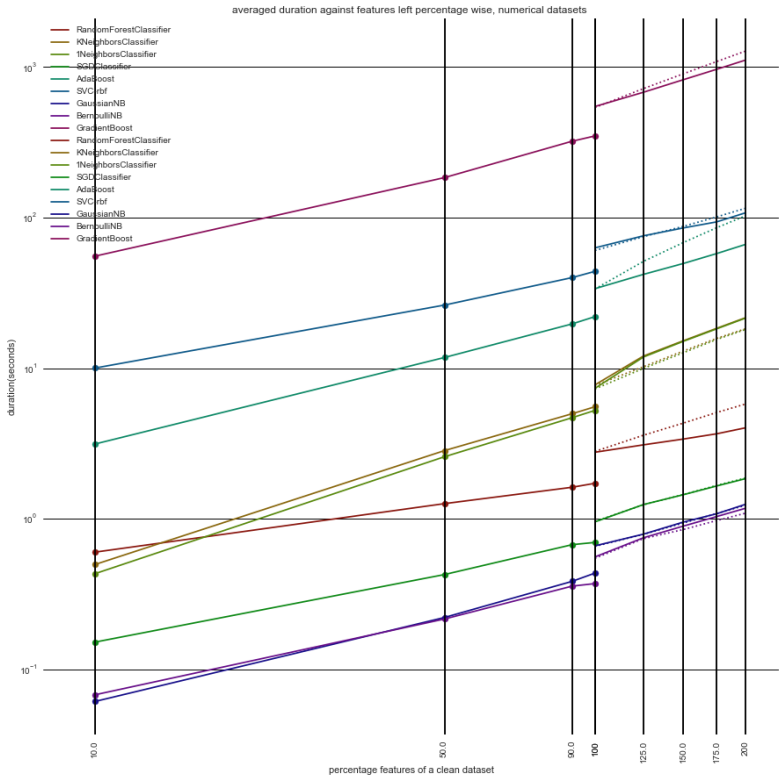


Figure 30: Duration of adding and removing features combined result

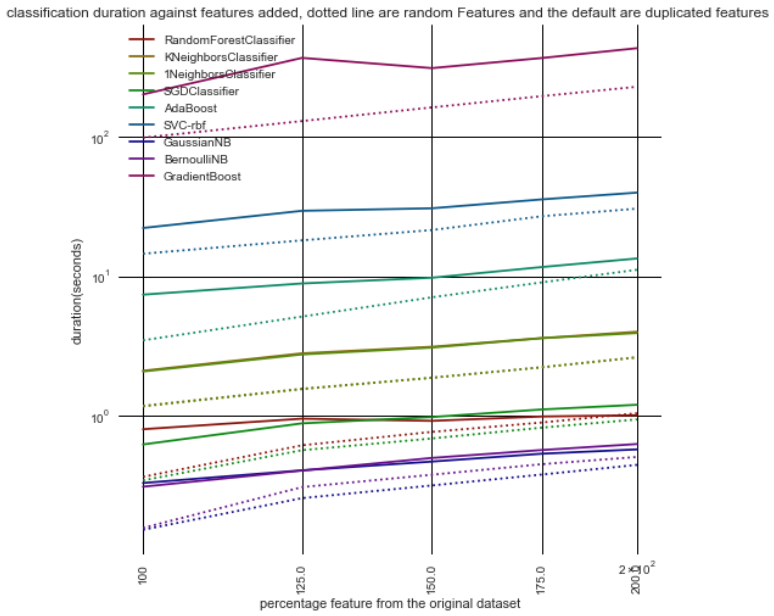


Figure 31: Duration of duplicated features against random features with clear spikes in duration registering

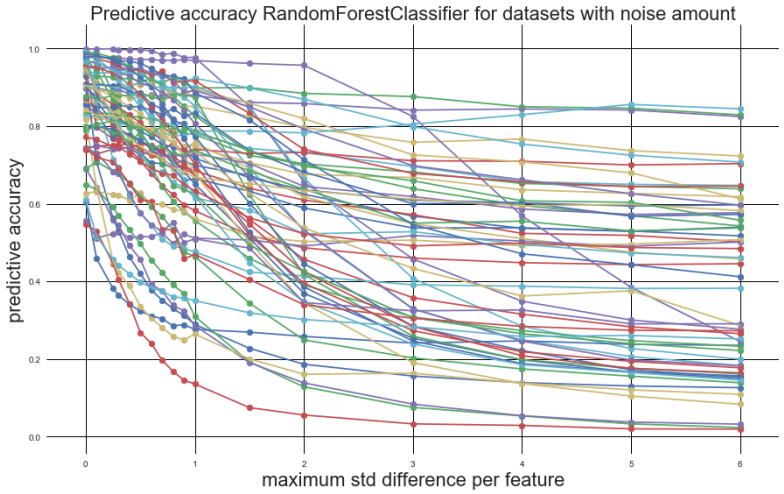


Figure 32: adding or removing uniformly random std to numerical features for RandomForestClassifier.

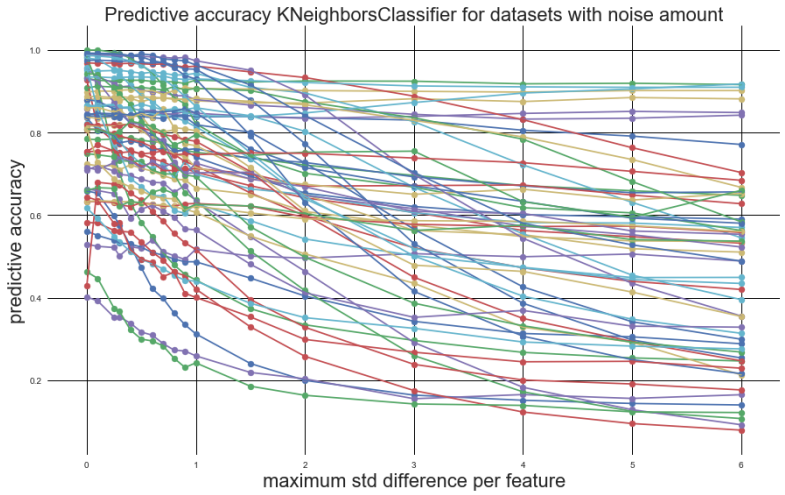


Figure 33: adding or removing uniformly random std to numerical features for KNeighborsClassifier.

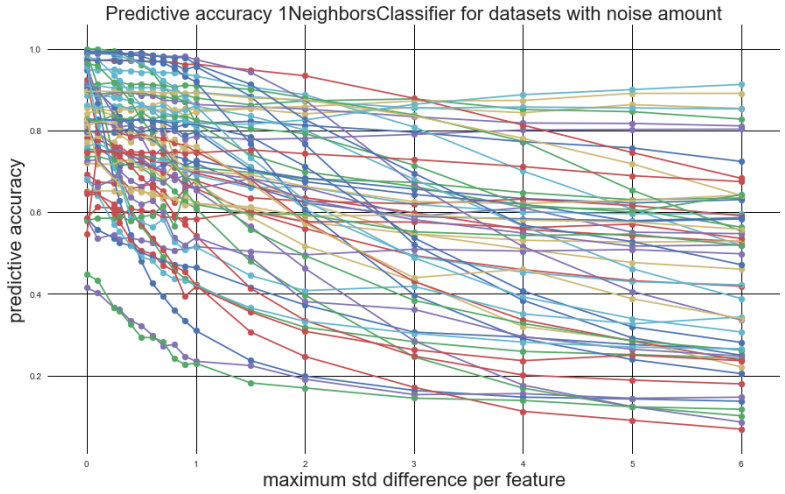


Figure 34: adding or removing uniformly random std to numerical features for 1NeighborClassifier.

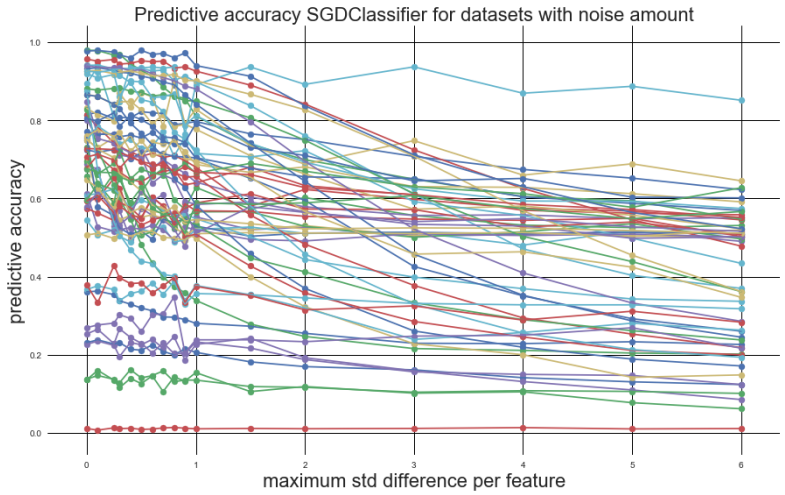


Figure 35: adding or removing uniformly random std to numerical features for SGDClassifier.

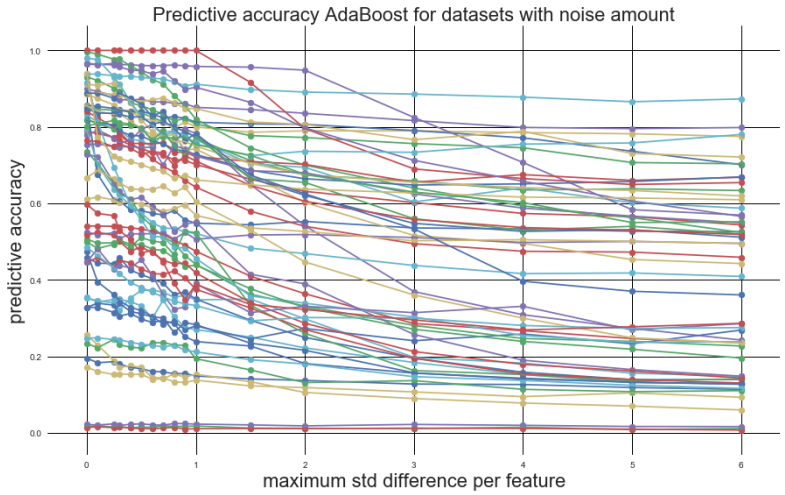


Figure 36: adding or removing uniformly random std to numerical features for AdaBoost.

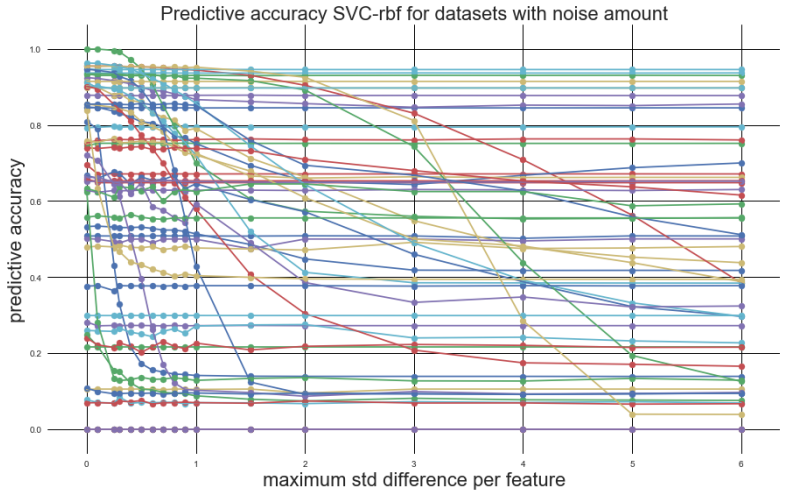


Figure 37: adding or removing uniformly random std to numerical features for SVC-rbf.

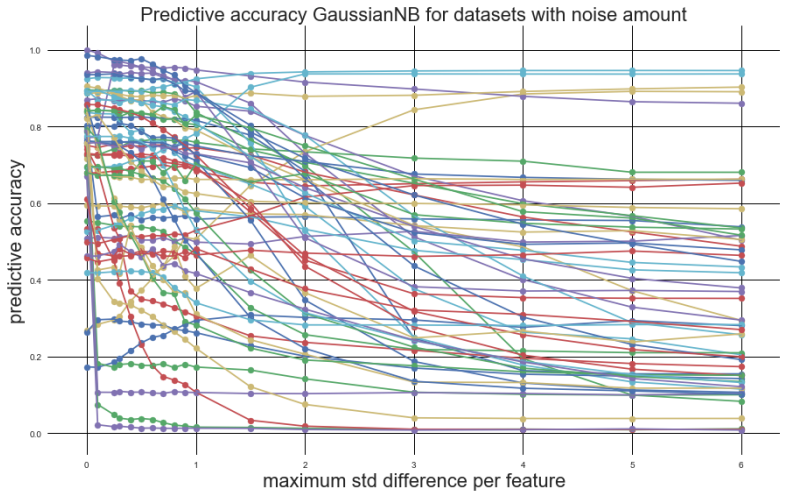


Figure 38: adding or removing uniformly random std to numerical features for GaussianNB.

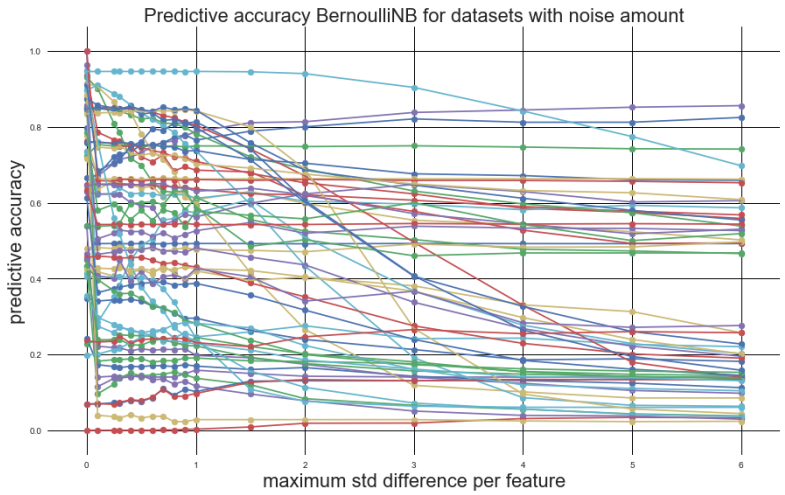


Figure 39: adding or removing uniformly random std to numerical features for BernoulliNB.

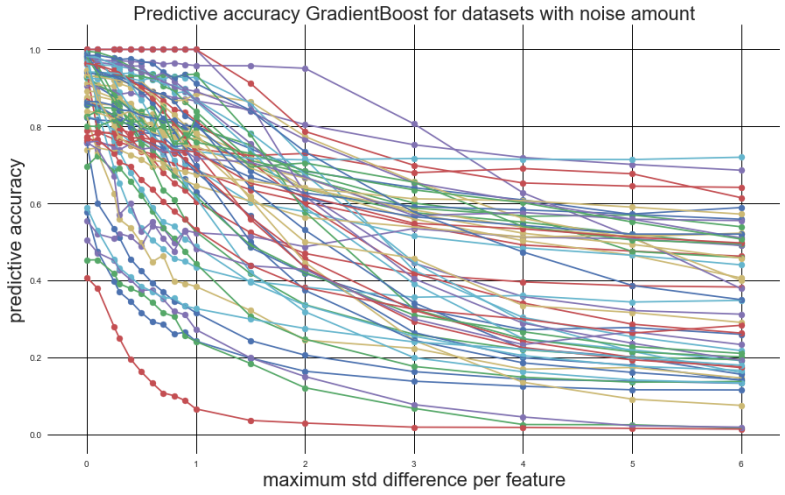


Figure 40: adding or removing uniformly random std to numerical features for GradientBoostingAlgorithm.

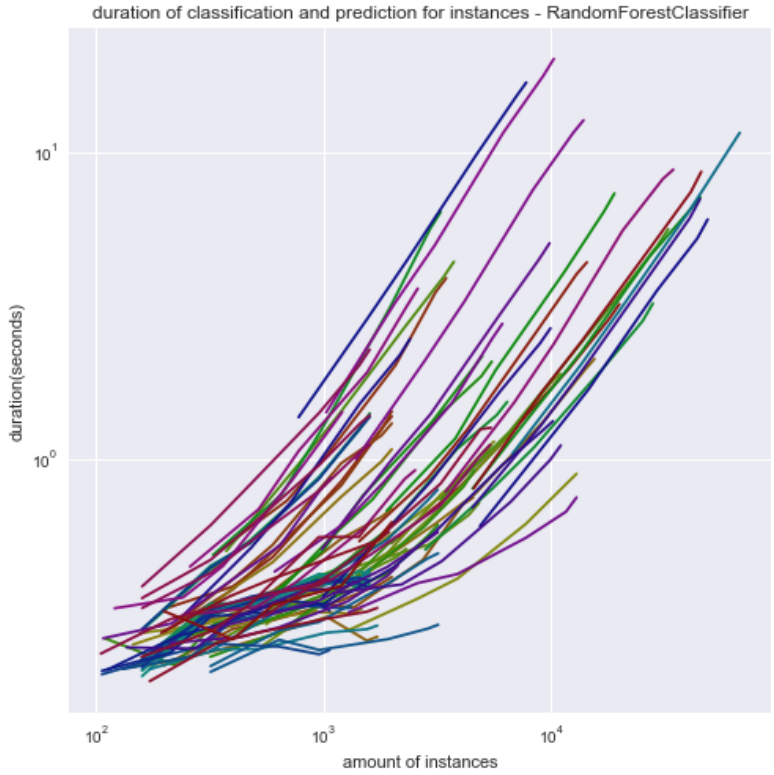


Figure 41: number of instances against duration for RandomForestClassifier

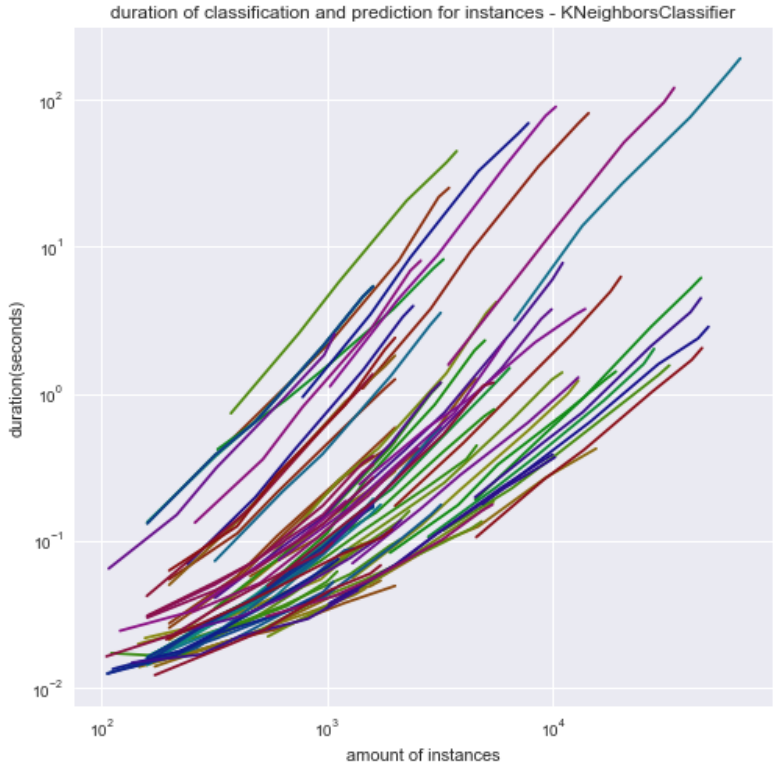


Figure 42: number of instances against duration for KNeighborsClassifier

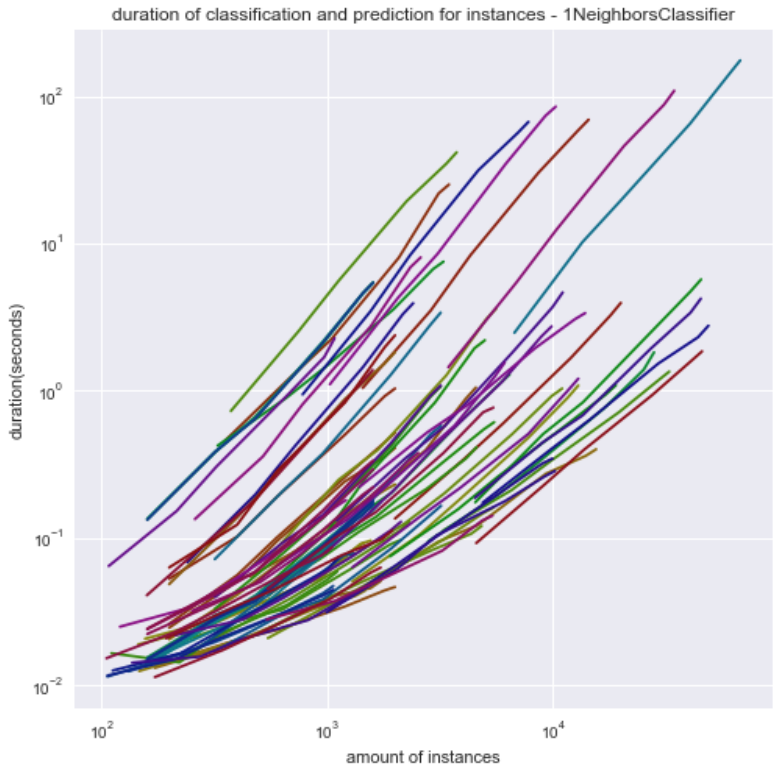


Figure 43: number of instances against duration for 1NeighborsClassifier

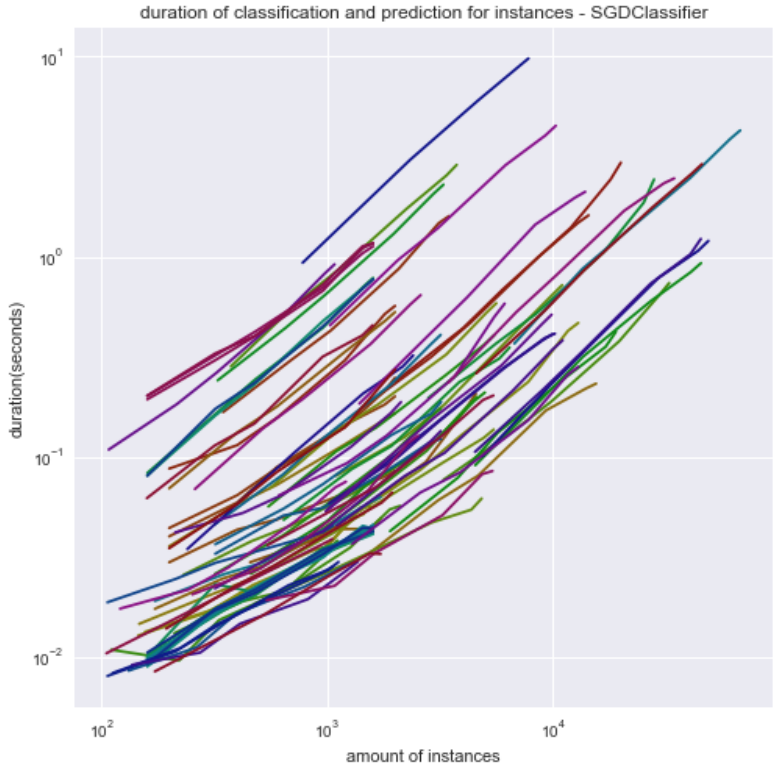


Figure 44: number of instances against duration for SGDClassifier

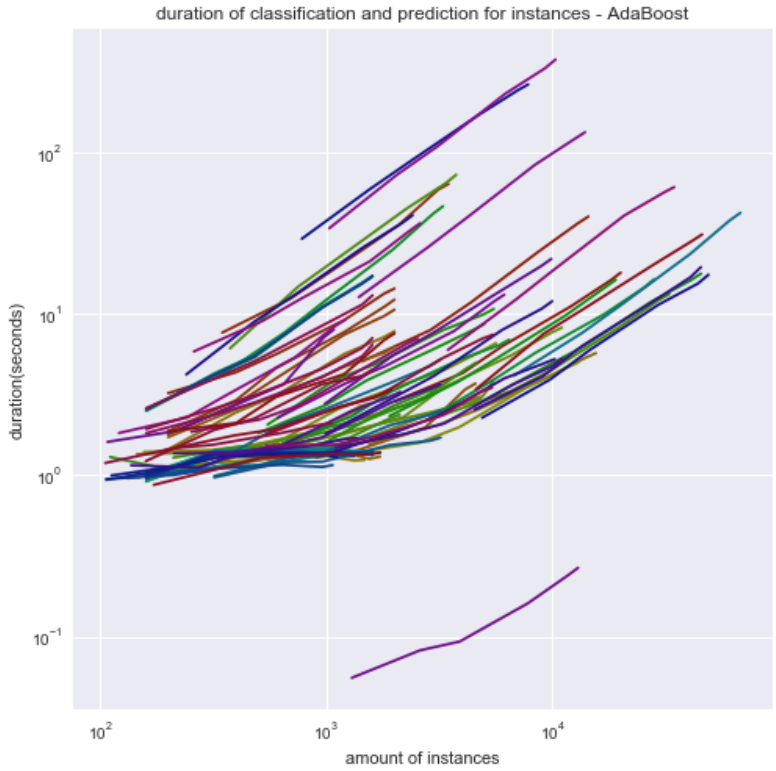


Figure 45: number of instances against duration for AdaBoostingClassifier

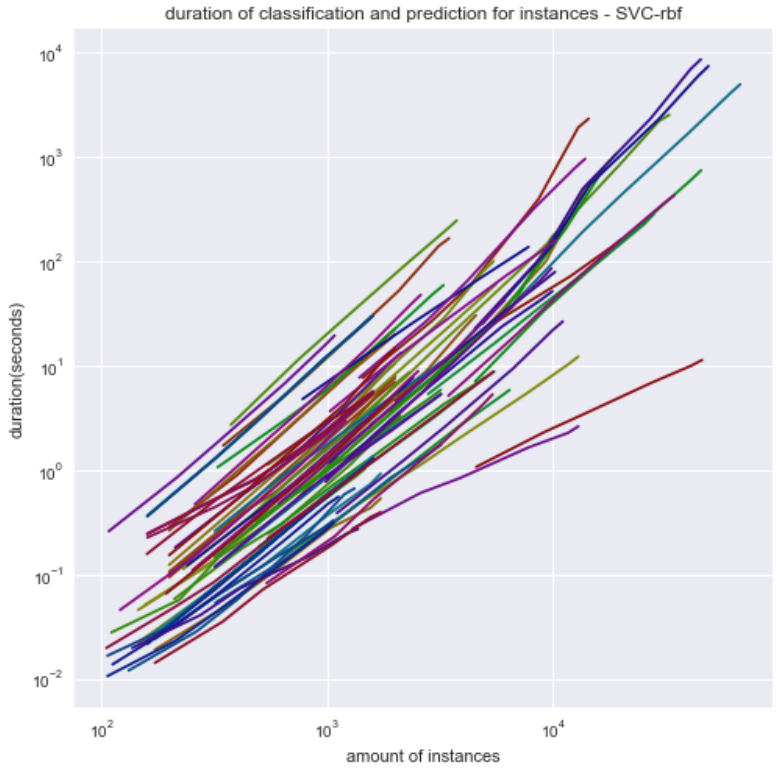


Figure 46: number of instances against duration for SVC-rbf

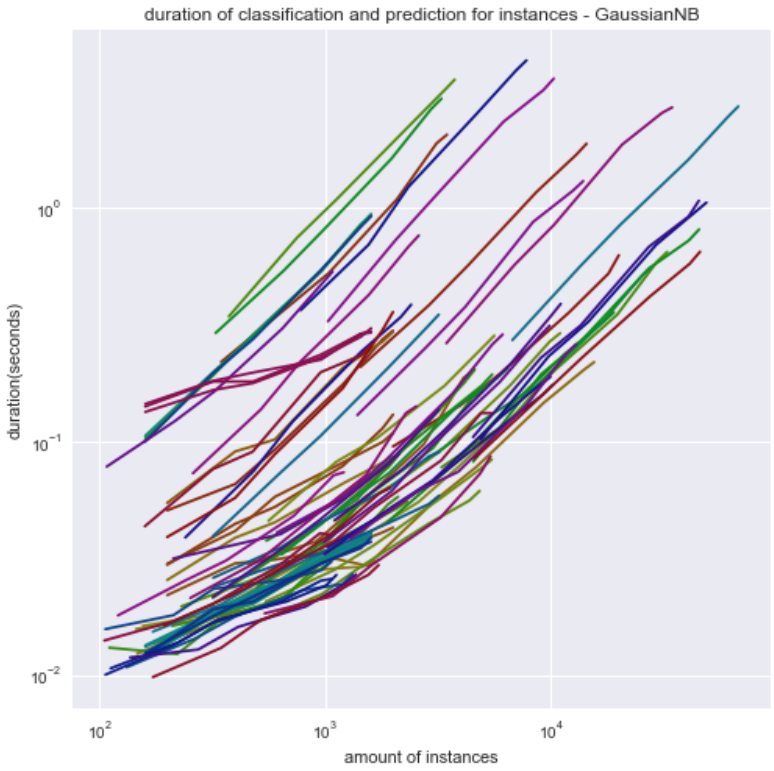


Figure 47: number of instances against duration for GaussianNB

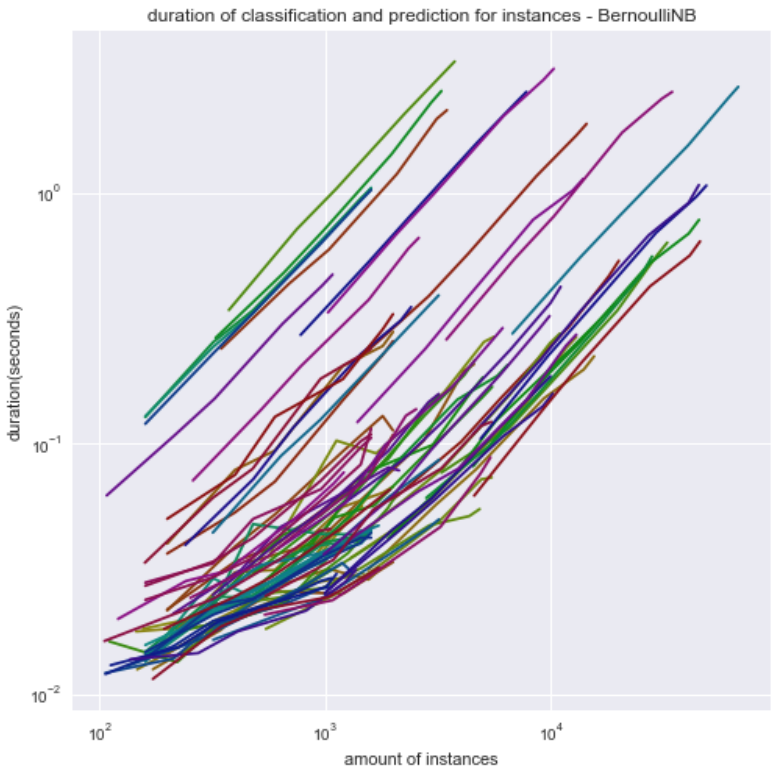


Figure 48: number of instances against duration for BernoulliNB

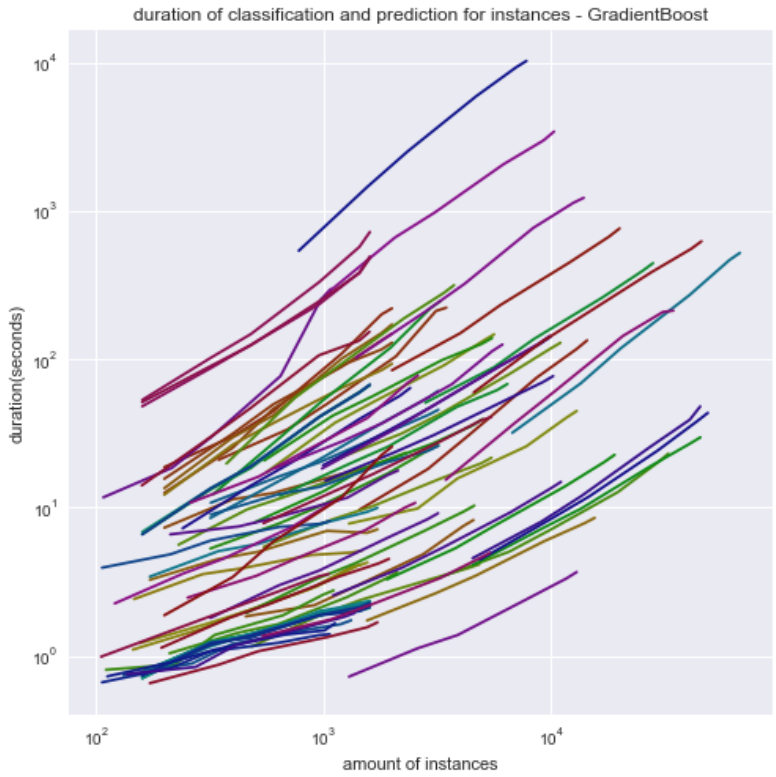


Figure 49: number of instances against duration for GradientBoostingClassifier