**184.702 Machine Learning 2019SS**

**Assignment 1: Classification Experiments**

Aleksandra Radecka (e11847505)

Niklas Speckle (e1515258)

Nazika Moeininia (e1535524)

# Introduction

The following report summarizes the results of machine learning experiments on classification. Three different types of algorithms were tested on four different datasets (4x3) to get to know the nature of the classifiers. The aim was to analyse algorithms with different underlying mathematical fundamentals. Therefore Naive Byes, K-Nearest-Neighbours and Random Forest were chosen. The experiments delivered insights on how the algorithms work and perform on different datasets. Also the impact of different parameter settings and preprocessing steps was investigated.

# Data Description

## Drugs Dataset

This dataset contains information about the frequency of drug use depending on different demographic and psychological attributes. It contains 1885 observations on 32 variables, whereas one is the ID and 19 can be considered as target variables as they contain information on the frequency of drug use on different kind of drugs. We decided to only work on one specific drug (cannabis). Therefore the dataset consists of 12 features. The features are numerical and categorical. The target variable has seven different values (never used the drug - used it yesterday), therefore it can be considered as a multiple class problem. The target variable is unequally distributed as most people either never consumed cannabis or used it within the last day (both > 20%). In between the values vary about 10%. Most people are and come from UK or USA and are rather young (< 25 years). Almost all participants have a degree, indicating that the study was conducted at one university. The psychological attributes are approximately normally distributed. There are only few outliers and does not contain any missing values. Beside that there are no peculiarities. A detailed description of the dataset can be found in Assignment 0.

## Cancer Dataset

This Dataset contains information about different properties of breast cancer and if a recurrence event occurred or not. It contains 285 observations on 32 variables, whereas one is the ID and one is the target variable "class". Therefore the dataset consists of 30 features. All features are floating point values and the target variable has two different values, therefore it can be considered as binary classification. The target variable is slightly unequally distributed as the proportion for B is about 66%. Generally all features have similar distribution patterns with several outliers to the top. There are only very few values which are extreme outliers beside that there are no peculiarities.

## Kick Dataset

This dataset is about kicked car which can be very costly for dealer. The biggest challenge of this dataset is to anticipate whether the purchased car at the auction is Kick (bad buy) or not. This dataset consists of 72.938 samples with missing values on 33 attributes. The main challenge encountered is how to deal with missing data which is 149271 distributed in various attributes. Hence, it is crucial to determine whether they follow any specific pattern or even completely at random manner since missing values have considerable impact on final results. Moreover, mainly two feature data types can be found in this dataset – categorical which is principally nominal (e.g. car model, color and IsBadBuy) and numerical e.g. current retail average price. In addition, due to large amount of observations, it seems to be necessary to select a representative sample by using sampling method.

As it is mentioned earlier, there is a very large population in this dataset and it is vital to determine an appropriate data selection in order to adequately answer the stated research question (IsBadBuy). Therefore, random sampling methodology was approached to obtain a representative sample. This sampling method tries to ensure the representativeness from the entire population by incorporating an element of ‘randomness’ to the selection procedure, and thus a greater ability to generalize findings to the targeted class.  To handle missing values regarding Kick’s variables, the most frequent imputation methodology was applied which can properly work with categorical features. Moreover, another preprocessing seems to be crucial is categorical encoding which refers to transforming a categorical feature into one or multiple numeric features. One hot encoding is one of the best methods to encode categorical feature, as they are substituted by one or zero.

## Amazon Dataset

This dataset is about predicting who wrote an Amazon review based on 10000 different attributes of the review. All those attributes are numeric. The training dataset contains 750 different instances, each representing an Amazon review. The target variable “class” is nominal and is simply the name of the person who wrote the review. There are no missing values and every instance seems to make sense, therefore, it is not required to deal with either missing values or deleting instances.

As 10000 attributes are clearly a lot to classify data and the majority of them do not have any special meaning for instance “and”, “a”, hence it is logical to select a subset of the most relevant and important attributes in order to reduce the complexity of a model and training time. It is noteworthy to mention that irrelevant or partially relevant features can negatively impact on this model performance and feature selection should be employed in the first step of model designing. To create an accurate predictive model in Amazon dataset, univariate feature selection by selecting K best was implemented to eliminate redundant data and improve model accuracy. As the amazon attributes are sparse, one of the appropriate methods which can deal with such data is chi\_square (chi2). Chi- square used to test whether the relationship of a dependent variable is significant to an independent variable or not. Finally, 500 prominent attributes were chosen to use in classification method.

## Dataset characteristics Summary

The decision to work on these datasets (Drugs Dataset and Kick Dataset) was based on their variety of characteristics. They differ in the number of samples, features, target classes and in presence of missing values. Therefore they fit perfectly to analyse the performance of different algorithms. The different characteristics of the datasets are summarized in table 1.

**Table 1: Dataset characteristics**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Number of samples | Number of features | Data types of features | Number of target classes | Presence of missing values |
| Drugs Dataset | 1.885 | 12 | Categorical and numerical | 7 | None |
| Kick Dataset | 72.983 | 33 | Categorical, numerical and date | 2 | 149.271 present in 20 features |
| Cancer Dataset | 285 | 30 | Numerical | 2 | None |
| Amazon Dataset | 750 | 10.000 | Numerical | 50 | None |

# Classifiers and Performance Measurements

The three classifiers were primarily assessed by accuracy and its standard deviation. On top of that precision and recall were considered. These metrics were calculated after performing a 10-fold cross validation. Another important performance measurement was the time it took to fit the model and to perform predictions. Why these classifiers and performance measurements were chosen is stated below.

## Classifiers

As mentioned above Naive Byes, K-Nearest-Neighbours and Random Forest were investigated. The decision to analyse these algorithms is based on different aspects. One of the main reasons is their difference in their underlying algorithmic method. Naive Bayes is a so called probabilistic model and is based on Bayes’ theorem. Conditional probabilities are calculated behind the scenes on which predictions are based. K-Nearest-Neighbours on the other hand does not even fit a model but looks for similar observations in the dataset and predicts according to the k most similar observations. Once the neighbours are found majority voting decides about the prediction. Random Forest is based on decision trees where a specific set of decision rules decide on the prediction. A certain number of decision trees is calculated where they do not include all features but a random subsample. Prediction is again determined by majority voting of all single decision tree predictions.

Another reason why these classifiers were chosen is the fact that these are very well known algorithms and their fundamental ideas are also influencing other techniques. Also the question appears if very simple algorithms like Naive Bayes and K-Nearest-Neighbours can compete with more modern and sophisticated approaches like Random Forest. On top of that these algorithms can be adapted by a number of parameters.

## Parameter Settings

In the experiments we varied parameters for each algorithm. For Naive Bayes we decided to vary the assumptions made regarding the distribution of the conditional probabilities. Gaussian, Bernoulli and Multinomial distribution assumptions were made.

For k-Nearest-Neighbours we obviously decided to vary the number of neighbours. In addition to that we tested two different kinds of algorithms to compute the nearest neighbours. Firstly, we used brute-force search, where all data points are considered as neighbours. Secondly, KDTree was tested, where the dataset gets split into several subspaces. For predictions only neighbours within the subspace where the unseen data is located is considered. This speeds up the prediction time what might be important for K-Nearest-Neighbours as it does not build a model beforehand.

Regarding Random Forest we mainly varied the number of trees but also the functions to measure the quality of a split. Precisely gini index and entropy was used. In addition to that we varied the minimum number of samples required to split a node. This causes the trees to not perfectly fit every single data point and therefore avoids overfitting. For each classifier we tried different preprocessing steps. For example Z-score standardisations and one hot encoding was made and sometimes the dataset was discretized.

## Performance measurements

The most important metric obviously is accuracy. It measures the ratio of correct predictions to the total number of predictions made, hence the higher the better. This can be considered as the overall performance of the model and is therefore obligatory to consider. Still accuracy can be misleading if the frequency for each class in the target variable is unequally distributed. Therefore accuracy needs to be treated with caution.

To see if accuracy is stable standard deviation is needed especially to compare results. This measure gives an insight on how likely it is to achieve such accuracy again. Here a low standard deviation is better as this means that the result is reliable.

To get a deeper insight into the models’ prediction behaviour recall and precision is considered as well. Recall indicates how many percentage of one specific class were predicted correctly. This can be interpreted as conditional probability of the prediction being true if the class is actually true. Recall gives an insight on how well the model performs on one specific class and helps to counter the missleading aspect of the accuracy described above. Precision measures how many percents of predictions of one specific class were actually correct. Again this can be interpreted as conditional probability of the class actually beeing true if the prediction for the class was true. In combination these performance measurements can be very powerful if interpreted correctly. All metrics were calculated by averaging them after a 10-fold corss validation. This is one of the most common and sufficient ways to validate predictive machine learning models. Last runtimes are considered as well to assess the different classifiers. Regarding runtime one have to keep in mind that this is depending on the machine used, but growth trends can be considered as consistent.

# Classifications

All three algorithms were used to predict the same scenarios. For the drugs dataset predictions were done on the frequency of cannabis use. The target variable has seven different values like “Never Used”, “Used in Last Year” and “Used Last Day”. Features are either psychological metrics or demographical attributes. Regarding cancer dataset predictions were done on whether a recurrence event occurred after a breast cancer disease. Different cancer attributes are taken into account. The kick dataset is used to figure out if a used car bought at an action can be sold to customers or not. Features are different attributes of the car like age, model and producer. Four attributes which are year of purchase, vehicle year model, VNST (in which state), wheel type ID were dropped to proceed in data classification. In Amazon dataset, the target variable is who wrote the commerce review and 50 classes are recognised. The ID attribute which refer to each review is clearly not useful for data classification, therefore, it cannot be considered for further process. As it is mentioned in preprocessing section of this dataset, feature selection method was approached due to numerous variables to reduce dimensionality. 500 features were eventually chosen which can aid in our mission to create a better accuracy in predictive model.

## Naive Bayes

### Drugs Dataset

If Gaussian Naive Bayes is applied one has to keep in mind that categorical data types are treated as continuous variables. If Multinomial or Bernoulli Naive Bayes is used continuous variables are treated as categorical. Obviously this leads to a bad performance of the classifier when used on the original data with an accuracy of 36.69% (5.79) for Gaussian, 36.27% (4.38) for Multinomial and 34.35% (3.57) for Bernoulli. But this is also due to the nature of the dataset as described in K-Nearest-Neighbours below. One solution for this problem might be to discretize the numerical features in five equally distributed classes (20% quantiles). This increases the accuracy of both Multinomial and Bernoulli algorithms whereas the performance for Gaussian decreased minimal. When all variables are discretized and one hot encoded Gaussian falls far back with an accuracy of 19.09% (1.81) whereas the others perform close to 40%. Runtimes do not differ between the classifiers but with the input data.

### Cancer Dataset

As this dataset only contains numerical data it perfectly fits Gaussian Naive Bayes. For both Multinomial and Bernoulli the features need to be discretized again. This is also shown in the experiments. Gaussian performed best on the original numerical and the others on the one-hot encoded dataset. Discretization was again done by cutting each feature in five equally distributed classes (20% quantiles). This dataset was then used to create a one hot encoded dataset. The following matrix shows the accuracies with standard deviation for all algorithms.

***Table 2: Summary of all experiments based on Naïve Bayes in Cancer datset***

|  |  |  |  |
| --- | --- | --- | --- |
|  | Gaussian NB | Multinomial NB | Bernoulli NB |
| Source Data | 96.54% (3.08) | 90.86% (3.25) | 66.33% (1.05) |
| Discretized Data | 94.09% (5.33) | 75.76% (7.87) | 75.73% (5.49) |
| One hot encoded Data | 93.05% (6.14) | 94.80% (4.42) | 94.80% (4.42) |

### Kick Dataset

This dataset contains categorical and numerical attributes. As it is mentioned in data description section, categorical attributes in preprocessing step were converted into a form of one and zero by using one hot encoding method that could be benefical to perform different machine learning algorithms. As it can be seen in table 3, Multinomial methodology is a weak predictor regarding lowest accuracy and highest standard deviation. Whilst Bernoulli and Gaussian methods demonstrate almost same standard deviations, their accuracy estimations are 84.6 % and 75.8 % respectively. In this algorithm Bernoulli Naïve Bayes represents a better performance as a predictor.

***Table 3: Summary of all experiments based on Naïve Bayes in Kick dataset***

|  |  |  |  |
| --- | --- | --- | --- |
|  | Gaussian NB | Multinomial NB | Bernoulli NB |
| One hot encoding | 75.8(1.3) | 50.8% (3.6) | 84.6% (1.2) |

### Amazon Dataset

This dataset only consists of numerical data it perfectly fits Gaussian Naive Bayes. Therefore, there is no need to one hot encoding and all algorithms are investigated according to source data. The following table indicates the accuracies of three algorithms with standard deviation (Table 4). As it can be seen from table Multinomial has a best accuracy, although standard deviation is measured in the higher value which can be a representative of widely spread data around mean and less reliable. On the other hand, Gaussian algorithm shows a lowest standard deviation which are relatively better clustered around the mean and more reliable, whereas its accuracy is downgrading compared to another algorithms. The result of Bernoulli algorithm is more close to Multinomial. Other experiments are also investigated by using min\_max scale data, as it is shown in following table. The Gaussian and Bernoulli experiments indicate the exact outcomes, whereas Multinomial experiment with scaling method shows surprisingly reduction of accuracy and standard deviation. All of implemented experiments in this algorithm had the fastest fitting and score time, as every feature being classified is independent of the value of any other feature. This would be advantageous because Naïve Bayes classifier converge quicker than other algorithms.

***Table 4: Summary of all experiments based on Naïve Bayes in Amazon dataset***

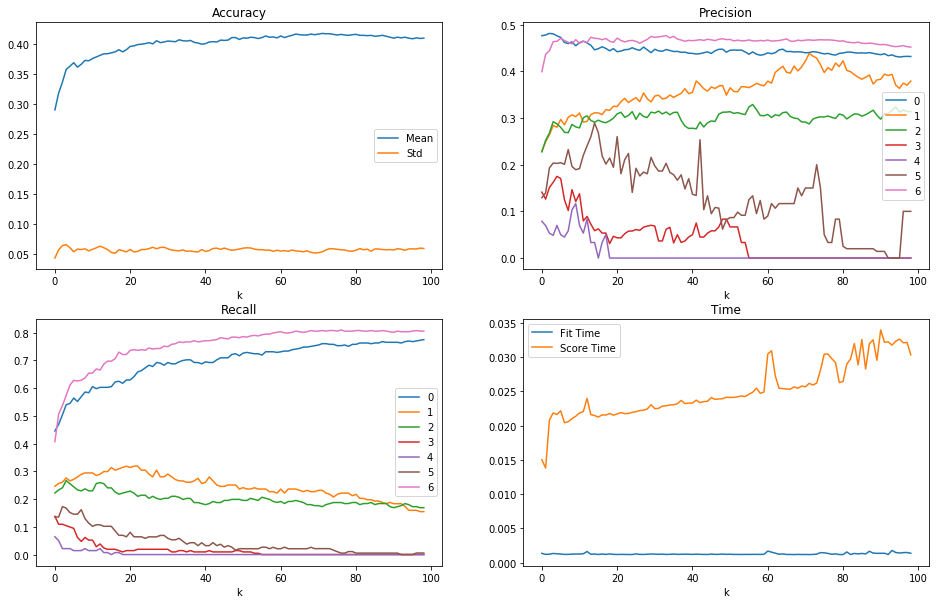
|  |  |  |  |
| --- | --- | --- | --- |
|  | Gaussian NB | Multinomial NB | Bernoulli NB |
| Source Data | 39.7% (4.3) | 62.5% (5.8) | 55.4% (5.6) |
| Min\_max scale | 50.8(4.3) | 49.9% (2.3) | 55.4% (5.6) |

## K-Nearest-Neighbours

### Drugs Dataset

Overall the Classifier did not perform very well on this dataset with an accuracy varying from 25-40%. One reason might be the unequal distribution of target classes as mentioned in the dataset description. When taking a closer look to accuracy and recall one realises that especially recall is high (up to 80%) for “Never Used” and “Used Last Day” but very low for the other values. This is also true for precision, but the value varies only about 40%. KNN just decides on two of the seven classes. This is why accuracy increases with k.

In general recall increases with a higher number of neighbours and precision slightly decreases. Both fitting time and score time were very fast. The best results were achieved when using min-max scaling on numerical and one hot encoding for categorical data and applying distance weights (Figure 1). The classifier performed very similar when KDTree was used to determine the neighbours. Interestingly it took more time to fit the model using KDTree but also to conduct a prediction. The following table shows some representative experiments.



***Figure 1: kNN on one hot encoded (categorical) and min-max scaled data with distance weights in Drugs dataset***

***Table 5: Summary of all experiments based on KNN in Drug dataset***

|  |  |  |  |
| --- | --- | --- | --- |
| Setting | Accuracy | Standard Deviation | Runtime |
| Data: source  Weights: uniform  Algorithm: brute | 24.94% (k = 2)  30.48% (k = 10)  35.88% (k = 50)  36.52% (k = 100)  logarithmic growth | 3.33 (k = 2)  3.27  3.92  3.66  constant | Fit Time:  < 0.0025  constant  Score Time:  0.015 – 0.025  linear growth |
| Data: min-max  Weights: distance  Algorithm: brute | 29.66% (k = 2)  37.12% (k = 10)  39.19% (k = 50)  38.28% (k = 100)  logarithmic growth  slightly decreases for k > 40 | 3.63  5.03  4.96  4.46 | Fit Time:  < 0.0025  constant  Score Time:  0.015 – 0.03  linear growth |
| Data: dummy (for categorical variables) and min-max  Weights: distance  Algorithm: brute | 28.95% (k = 2)  37.19% (k = 10)  41.00% (k = 50)  40.99% (k = 100)  logarithmic growth | 4.25  5.79  5.63  5.84 | Fit Time:  < 0.0025  constant  Score Time:  0.015 – 0.035  linear growth |

### Cancer Dataset

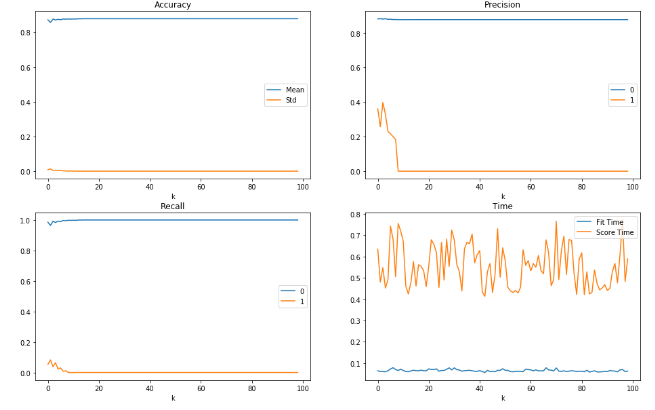
K-Nearest-Neighbours perfectly fit this dataset as all variables are numeric. In all experiments the classifier achieved accuracy greater than 90%. Anyway one can see a decent impact of the preprocessing here. Opposed to the Drugs Dataset the accuracy decreased with a larger number of neighbours but increases for up to 13 neighbours. When distance weights were used accuracy was greater and did not decrease as rapidly for a higher number of neighbours. Surprisingly using KDTree to determine the neighbours did not have an impact on accuracy and a minor impact on runtime. For all experiments recall for class B was greater than for class A. This might be again due to an unequal distribution of both classes (66% for B). Precision is higher for A. The gap between recall and precision accuracy increases with the number of neighbours. The best results were achieved using min-max scaled data and distance weights with 10-15 neighbours. The following table again shows some representative experiments.

***Table 6: Summary of all experiments based on KNN in cancer dataset***

|  |  |  |  |
| --- | --- | --- | --- |
| Setting | Accuracy | Standard Deviation | Runtime |
| Data: source  Weights: uniform  Algorithm: brute | 92.25% (k = 2)  92.28% (k = 10)  92.26% (k = 50)  89.44% (k = 100) | 4.14  3.44  3.78  3.59 | Fit Time:  0.0015  constant  Score Time:  0.003 – 0.005  linear growth |
| Data: source  Weights: distance  Algorithm: brute | 93.67% (k = 2)  92.98% (k = 10)  92.25% (k = 50)  92.26% (k = 100) | 4.07  3.48  4.94  3.78 | Fit Time:  0.0015  constant  Score Time:  0.003 – 0.004  linear growth |
| Data: min-max  Weights: uniform  Algorithm: brute | 96.87% (k = 2)  98.28% (k = 10)  94.79% (k = 50)  89.80% (k = 100) | 3.26  2.31  2.89  4.27 | Fit Time:  0.001  constant  Score Time:  0.003 – 0.005  linear growth |
| Data: min-max  Weights: distance  Algorithm: brute | 97.56% (k = 2)  98.62% (k = 10)  95.43% (k = 50)  91.55% (k = 100) | 3.50  2.29  2.25  3.62 | Fit Time:  0.001  constant  Score Time:  0.003 – 0.004  linear growth |

### Kick Dataset

In this dataset, both experiments based on brute and KdTree illustrate similar outcomes if more than 15 neighbours are applied (Table 7). Standard deviation also has a slight difference which can be neglectable. However, the significant discrepancies are related to time when fitting the defined train part and predicting for test part. In this case, score computation time in KdTree algorithm is considerably higher than brute force search which shows also linear growth. Different method of weighting also could not contribute on the prediction. Moreover, min\_max scalar was examined to investigate the performance of accounted algorithms and enhance the efficiency of predictor. Nevertheless no improvement has not been observed in this implemented experiment. For all experiments recall for class 0 (is not bad buy) which is approximately 100% is a far greater than precision in this class (90%). For class 1 in precision accuracy if less than 8 number of neighbours is considered, class 1 in recall and precision accuracy shows big gap; however, more than 8 number of neighbours is similar constant value (close to zero). To summarize, despite similar accuracy performances in all experiments, the best achievement of result is regarded to brute algorithm by utilising either uniform or distance weight due to less computation time (Figure 2).



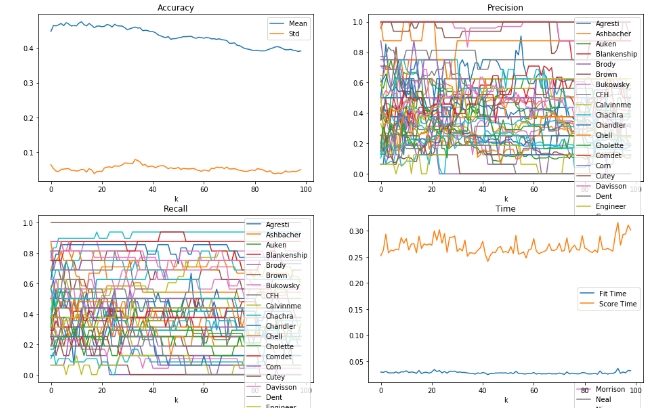
***Figure 2: kNN on one hot encoded (categorical) and min-max scaled data with distance weights in Kick dataset***

***Table 7: Summary of all experiments based on KNN in Kick dataset***

|  |  |  |  |
| --- | --- | --- | --- |
| Setting | Accuracy | Standard Deviation | Runtime |
| Data: source  Weights: uniform  Algorithm: brute | 87% (k = 2)  89% (k = 10)  90% (k = 50)  90% (k = 100) | 1.5  1.1  1.1  1.1  constant | Fit Time:  0.05-0.07  Fluctuated  Score Time:  0.42 – 0.76  Fluctuated |
| Data: source  Weights: distance  Algorithm: KdTree | 81% (k = 2)  88% (k = 10)  90% (k = 50)  90% (k = 100) | 2  1.5  1.4  1.4  constant | Fit Time:  0.14-0.18  Fluctuated  Score Time:  1 – 2.45  Linear growth |
| Data: min-max  Weights: distance  Algorithm: brute | 81% (k = 2)  88% (k = 10)  90% (k = 50)  90% (k = 100) | 2  1.5  1.4  1.4  constant | Fit Time:  0.14-0.18  Fluctuated  Score Time:  0.47 – 0.76  Fluctuated |

### Amazon Dataset

K-Nearest-Neighbours has not shown any perfect match in this dataset, even though all variable s are numeric. Both considered approaches (brute and KdTree) illustrate similar behaviour in terms of accuracy accounting same criterion. However, the computation time in KdTree algorithm is higher than brute force search. A first investigated criterion is weight that can be determined based on unique or distance. Distance weight can enhance the accuracy, although this improvement is not significant compared to the implemented experiment based on uniform weighting parameter (approximately 5%). For all Knn experiments, higher number of neighbours (more than 15 neighbours) not only reduces accuracy result but also boosts considerably run time. Furthermore, for all experiments recall and precision accuracy were plotted according to our diverse classes; however, overall view of precision and recall show the discrepancy is not clearly distinguishable due to high number of classes. In addition to this, using min\_max scaled data can surprisingly achieve a better accuracy result and less computation time. The best obtained result is belonging to KdTree by using distance weight and min\_max scale data with 10-15 neighbours (Figure 3). As it is mentioned earlier, mean and standard deviation accuracy show almost same trend result, hence the major differences are related to fitting and scoring time which has main impact on choosing as a best predictor. The following table again shows some representative experiments.



***Figure 3: kNN on min-max scaled data with distance weights in Amazon dataset***

***Table 8: Summary of all experiments based on KNN in Amazon dataset***

|  |  |  |  |
| --- | --- | --- | --- |
| Setting | Accuracy | Standard Deviation | Runtime |
| Data: source  Weights: uniform  Algorithm: brute | 15% (k = 2)  22.5% (k = 10)  20% (k = 20)  17% (k = 50)  14% (k = 100) | 2.3  3.25  3.3  2.3  2.9 | Fit Time:  0.01-0.03  Fluctuated trend  Score Time:  0.09 – 0.48  Fluctuated trend |
| Data: source  Weights: distance  Algorithm: brute | 20% (k = 2)  24% (k = 10)  23% (k = 20)  19.5% (k = 50)  17% (k = 100) | 2.5  4.1  4.8  2.7  4.6 | Fit Time:  0.01-0.03  Fluctuated trend  Score Time:  0.085 – 0.48  Fluctuated trend |
| Data: min-max  Weights: distance  Algorithm: brute | 47% (k = 2)  47.5% (k = 10)  46% (k = 20)  42% (k = 50)  38% (k = 100) | 3.50  2.29  2.25  3.62  5.5 | Fit Time:  0.02-0.05  Fluctuated trend  Score Time:  0.08 – 0.45  Fluctuated trend |
| Best result:  Data: min-max  Weights: distance  Algorithm: KdTree | 47% (k = 2)  47.5% (k = 10)  46% (k = 20)  42% (k = 50)  38% (k = 100) | 3.50  2.29  2.25  3.62  5.5 | Fit Time:  0.04-0.06  Fluctuated  Score Time:  0.24 – 0.32  Fluctuated |

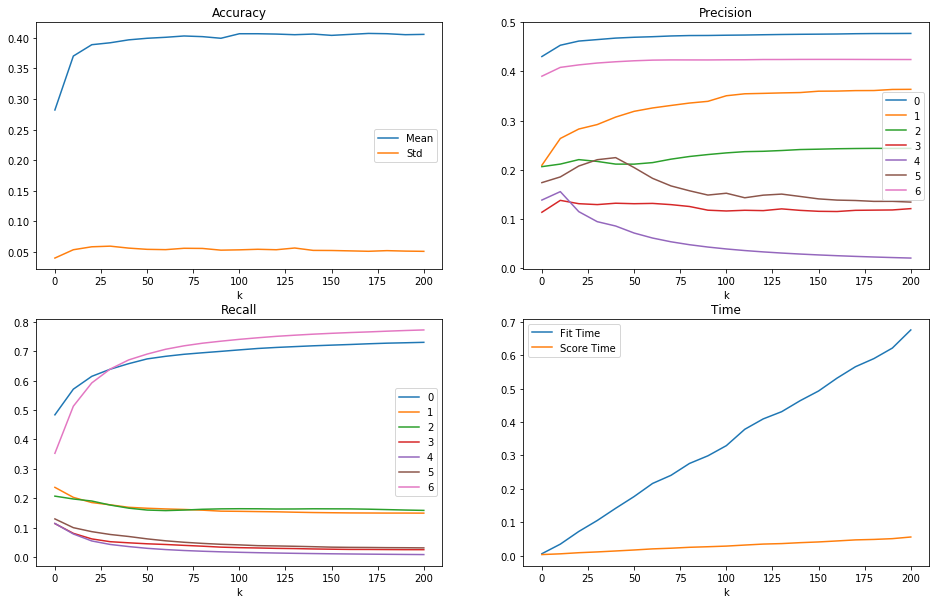
## Random Forest

### Drugs Dataset

Random Forest performed similarly as the other classifiers. Again, precision and recall show the same pattern as mentioned above (Figure 1 and Figure 4). All experiments run on Random Forest delivered similar results. Only the number of estimators had a bigger impact as accuracy increased logarithmically with a higher number of trees. Surprisingly standard deviation did not decrease. As expected, the minimum sample size to split a node only had an impact on models with a small number of trees as it avoids overfitting. Different scaling methods did not affect the results. Accuracy increased with a higher number of estimators so did runtimes. Especially the time to fit the model increased linearly. The following table again shows some representative experiments.

***Table 9: Summary of all experiments based on Random Forest in Drugs dataset***

|  |  |  |  |
| --- | --- | --- | --- |
| Setting | Accuracy | Standard Deviation | Runtime |
| Data: source  Criterion: Gini  Minimum Sample Split: 2 | 25.10% (n = 1)  34.53% (n = 10)  36.39% (n = 20)  38.02% (n = 50)  38.40% (n = 100)  38.93% (n = 200) | 2.81  4.49  4.91  5.62  5.42  5.04 | Fit Time:  0.01 – 0.6  linear growth  Score Time:  0.01 – 0.06  linear growth |
| Data: source  Criterion: Gini  Minimum Sample Split: 10 | 28.63% (n = 1)  37.23% (n = 10)  38.29% (n = 20)  39.50% (n = 50)  39.61% (n = 100)  40.25% (n = 200) | 4.16  5.65  4.57  4.84  5.04  5.30 | Fit Time:  0.4 – 0.5  linear growth  Score Time:  0.01 – 0.06  linear growth |
| Data: source  Criterion: Entropy  Minimum Sample Split: 2 | 27.79% (n = 1)  34.13% (n = 10)  36.64% (n = 20)  38.06% (n = 50)  39.08% (n = 100)  39.51% (n = 200) | 3.50  4.28  5.56  4.00  4.59  5.19 | Fit Time:  0.01 – 0.85  linear growth  Score Time:  0.01 – 0.08  linear growth |
| Data: source  Criterion: Entropy  Minimums Sample Split: 10 | 28.20% (n = 1)  37.02% (n = 10)  38.89% (n = 20)  39.93% (n = 50)  40.68% (n = 100)  40.57% (n = 200) | 3.96  5.32  5.81  5.39  5.31  5.06 | Fit Time:  0.01 – 0.7  linear growth  Score Time:  0.01 – 0.07  linear growth |



**Figure 4: Random Forest on source data with entropy as criterion and 10 as minimal sample split in Drugs dataset**

### Cancer Dataset

Again Random Forest performs very well with all parameter settings (Table 10). Scaling Methods again did not have a significant impact. Overall the accuracy varied from 94% up to almost 99%. Results became better with a higher number of estimators but only 20 were needed to achieve the best results possible. Again there is the same pattern for recall and precision as described above for the cancer dataset. Runtime increases linearly, especially the time to fit the model increases and takes up to 0.3 seconds.

***Table 10: Summary of all experiments based on Random Forest in Cancer dataset***

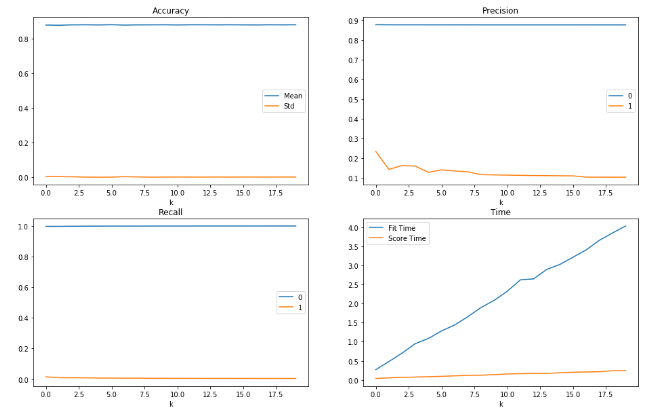
|  |  |  |  |
| --- | --- | --- | --- |
| Setting | Accuracy | Standard Deviation | Runtime |
| Data: source  Criterion: gini  Minimum Sample Split: 2 | 96.85% (n = 1)  96.17% (n = 10)  97.91% (n = 20)  97.19% (n = 50)  97.56% (n = 100)  97.91% (n = 200) | 2.42  2.89  2.80  2.63  2.73  2.34 | Fit Time:  0.01 – 0.30  linear growth  Score Time:  0.01 – 0.025  linear growth |
| Data: source  Criterion: gini  Minimum Sample Split: 10 | 96.46% (n = 1)  97.56% (n = 10)  97.56% (n = 20)  97.53% (n = 50)  97.56% (n = 100)  97.56% (n = 200) | 2.78  2.73  3.14  2.75  2.73  2.73 | Fit Time:  0.01 – 0.27  linear growth  Score Time:  0.01 – 0.04  linear growth |
| Data: source  Criterion: entropy  Minimum Sample Split: 2 | 96.85% (n = 1)  96.50% (n = 10)  97.54% (n = 20)  97.88% (n = 50)  97.91% (n = 100)  98.25% (n = 200) | 3.19  2.70  2.80  2.37  2.37  2.37 | Fit Time:  0.01 – 0.25  linear growth  Score Time:  0.01 – 0.04  linear growth |
| Data: source  Criterion: entropy  Minimum Sample Split: 10 | 95.12% (n = 1)  96.84% (n = 10)  97.19% (n = 20)  97.19% (n = 50)  97.56% (n = 100)  97.56% (n = 200) | 4.68  2.90  2.63  2.63  2.73  2.73 | Fit Time:  0.01 – 0.25  linear growth  Score Time:  0.01 – 0.04  linear growth |

### Kick Dataset

The performance of all experiments in Random forest based on the considered criteria which are gini or entropy with minimum split sample 2, 5 and 10 demonstarte the exact performance results (Table 11). After 10 estimators, the optimum result which is 90 % accuracy with low standard deviation can be achieved (Figure 5). Thus, it seems to be advantegous to implement this algorithm regarding low fit and score run time. To compare the prescion and recall accuracy, it should be noted that the results are similar to K Nearest neighboring algorithm.

***Table 11: Summary of all experiments based on Random Forest in Kick dataset***

|  |  |  |  |
| --- | --- | --- | --- |
| Setting | Accuracy | Standard Deviation | Runtime |
| Data: source  Criterion: gini  Minimum Sample Split: 2 | 89% (n = 1)  90% (n = 10)  90% (n = 50)  90% (n = 100)  90% (n = 200) | 0.5  0.5  0.5  0.5  0.5 | Fit Time:  0.5 – 4.5  linear growth  Score Time:  0.01 – 0.02  linear growth |
| Data: source  Criterion: gini  Minimum Sample Split: 10 | 89% (n = 1)  90% (n = 10)  90% (n = 50)  90% (n = 100)  90% (n = 200) | 0.5  0.5  0.5  0.5  0.5 | Fit Time:  0.3 – 4  linear growth  Score Time:  0.01 – 0.025  linear growth |
| Data: source  Criterion: entropy  Minimum Sample Split: 2 | 89% (n = 1)  90% (n = 10)  90% (n = 50)  90% (n = 100)  90% (n = 200) | 0.5  0.5  0.5  0.5  0.5 | Fit Time:  0.3 – 4.7  linear growth  Score Time:  0.01 – 0.02  linear growth |
| Data: source  Criterion: entropy  Minimum Sample Split: 10 | 89% (n = 1)  90% (n = 10)  90% (n = 50)  90% (n = 100)  90% (n = 200) | 0.5  0.5  0.5  0.5  0.5 | Fit Time:  0.3 – 4.2  linear growth  Score Time:  0.01 – 0.02  linear growth |



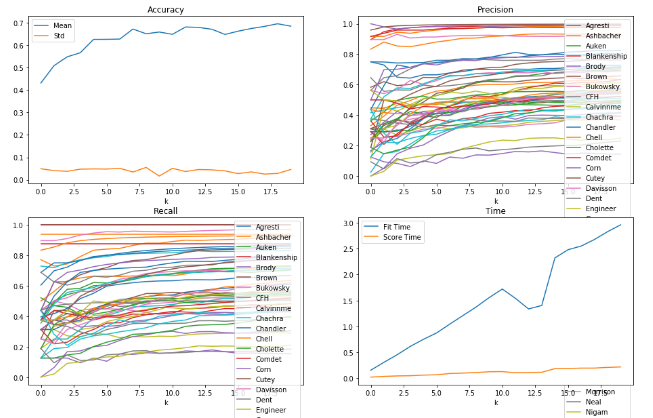
**Figure 5: Random Forest on source data with gini as criterion and 2 as minimal sample split in Kick dataset**

### Amazon Dataset

The outcome which is derived from Random forest algorithm compared to other examined algorithms shows that this predictor algorithm can anticipate test data up to 70.5% correctly (Table 12). Min\_max scalar have no positive impact on accuracy of forecasting. Thus, it is not included in the table result. Though minimum sample split 2 can negatively contribute on score time and fitting time in the gini method, from accuracy point of view it can elaborate the most efficient performance among all of the experiments in employed methodology. On top of that, high number of estimators can improve the efficiency of this algorithm. Moreover, fitting time in best experiment after 100 estimators was surprisingly dropped and in 120 estimators again depicts an upward growth. Overall precision and recall in different classes can not be distinguished again, thus we can not simply interpret them (Figure 6). To conclude, the best result has obtained based on gini method with minimum sample 2 among all of implemented experiments in different classifiers.

***Table 12: Summary of all experiments based on Random Forest in Amazon dataset***

|  |  |  |  |
| --- | --- | --- | --- |
| Setting | Accuracy | Standard Deviation | Runtime |
| Best experimnet:  Data: source  Criterion: gini  Minimum Sample Split: 2 | 42% (n = 1)  58% (n = 25)  64% (n = 50)  68% (n = 100)  70.5% (n = 200) | 4.8  4.8  5.5  5.6  5.2 | Fit Time:  0.4 – 2.8  linear growth, dropping trend and again increasing  Score Time:  0.02 – 0.25  linear growth |
| Data: source  Criterion: gini  Minimum Sample Split: 10 | 45% (n = 1)  60% (n = 25)  63% (n = 50)  65% (n = 100)  69.5% (n = 200) | 4.8  4.6  5.4  3.8  2.5 | Fit Time:  0.12 – 1.80  linear growth  Score Time:  0.05 – 0.13  linear growth |
| Data: source  Criterion: entropy  Minimum Sample Split: 2 | 28% (n = 1)  40% (n = 25)  47% (n = 50)  52% (n = 100)  56% (n = 200) | 3.1  2.7  4.5  5.4  4.1 | Fit Time:  0.3 – 3.7  linear growth  Score Time:  0.01 – 0.04  linear growth |
| Data: source  Criterion: entropy  Minimum Sample Split: 10 | 28% (n = 1)  44% (n = 25)  53% (n = 50)  55% (n = 100)  56% (n = 200) | 3.2  3.1  4.5  5.6  4.1 | Fit Time:  0. 2 – 4.8  linear growth  Score Time:  0.01 – 0.04  linear growth |



**Figure 6: Random Forest on source data with gini as criterion and 10 as minimal sample split in Amazon dataset**

## Summary

In general, the information content of the dataset seems to make the biggest difference as all algorithms performed similarly on the datasets.

### Naive Bayes

The experiments approved what was expected from theory but still some unexpected results occurred. Gaussian Naive Bayes works best for numerical data and also surprisingly well on multileveled data but performs badly on one-hot encoded data. Multinomial and Bernoulli Naive Bayes performed better on categorical data. Both achieved higher accuracies and lower standard deviations when the features where discretized or one-hot encoded. Surprisingly Multinomial Naive Bayes sometimes works better for binary or numerical variables as for multiclass variables. Runtimes increase with the size of the dataset but still stay small.

### K-Nearest-Neighbours

For this algorithm the accuracy improved with more neighbours considered up to a certain number. As expected, this ceiling is higher for datasets with a bigger sample size. With additional neighbours the performance might decrease again. Surprisingly the standard deviation of accuracy seemed to be independent from the number of neighbours. Scaling methods are highly recommended as it equalizes the impact of features. According to the experiments run it is best to apply min-max scaling to numerical data and one-hot encoding to categorical data. Weighting the considered neighbours by distance seem to slightly improve the performance. Surprisingly KDTree did not have an impact on performance. In all experiments the score time increased linearly whereas fitting time was constant. This is due to the reason that this algorithm actually does not build a model beforehand.

### Random Forest

This algorithm achieved similar accuracies as the other classifiers but has advantages regarding standard deviation. Similar to K-Nearest-Neighbours the model becomes better with a higher number of estimators up to a certain ceiling. Opposed to KNN the model does not become worse by adding additional trees. Surprisingly drugs dataset took more estimators to ceil as cancer dataset. This is contrary to the fact that drugs dataset has a lower dimension. Criterion does not significantly influence the results. It takes some time to fit the models especially for a higher number of trees. Surprisingly in general the standard deviation did not decrease with a higher number of estimators. Minimum sample split only had an impact on drugs dataset for small number of trees. Scaling had no impact.

Overall the conclusion can be drawn that most classification performance is mainly dependent on the information content of the dataset. Although naive, Naive Bayes can perform pretty well if applied correctly by assuming the correct distribution of conditional probabilities. This is also true for K-Nearest-Neighbours where one has to find the appropriated number of neighbours. Equalizing the impact of the features by scaling is essential here. Even when those simple algorithms are able to compete with the random forest this algorithm might be the best as it is very stable and easy to apply. Scaling the data is not required and to be sure one can use rather use more estimators. There are also tiny advantages for random forest regarding standard deviation.

# Conclusion