# Combining Bayes and Density Tree Classifier

Constantin Pape
Marcus Theisen

March 10, 2015

## 1 Introduction

In this project we combine two different classifiers to address the issue of generating new instances for data with interdependent feature dimensions. As basis we take the Naive Bayes Classifier, that treats feature dimensions as independent and the Density Tree Classifier, that captures the dependency between the feature dimensions.
Both Classifiers share the concept of learning the data by learning the posterior $p(B|A)$ in Bayes Theorem

$$p(A|B) = \frac{p(B|A)\ P(A)}{P(B)}. \tag{1}$$

As a reference dataset we use a downsampled version of the MNIST-dataset and filter it for the numerals 3 and 8. We test the classifiers on the full dataset and a version that is reduced to the two most relevant dimensions. For further applications we use more complex datasets: Words
The Classifiers are implemented in C++, the source code is available on GitHub:
`https://github.com/consti123/machine_learning`

## 2 Basis Classifiers

First we give a brief summary of the two underlying classifiers and present their results on the MNIST-dataset.

### 2.1 Bayes Classifier

The Naive Bayes Classifier treats all feature dimensions as independent. It learns the posterior by creating a histogram for each dimension and class. The bin width (in each dimension) is chosen according to the formula

$$\Delta X = \frac{2\ \mathrm{IQR}(X)}{N^{\frac{1}{3}}} \tag{2}$$

where $X$ is the dataset, IQR the inter-quartile-range and $N = |X|$. However this rule might lead to too small bins for data with a small spread. To counter this effect, we enforce a maximal bin number of $\sqrt{N}$, adjusting the bin width if necessary. After determining the bin width the histograms are constructed from the training data and normalised.
The Bayes Classifier predicts the labels for new data by calculating the likelihood according to 1 for each class. The posterior $p(B|A)$ is given by the probability of the corresponding bin. The prior $p(A)$ is given by the fraction of class instances to total instances in the training data. Then the class with the biggest value for $p(B|A)\ p(A)$ is predicted.
For generating the Bayes Classifier iterates over the dimensions. For each dimension one of the 5 most probable bins is chosen with its probability and then it is sampled uniformly from this bin.

#### 2.1.1 Results

The Bayes Classifier performs well on the full and reduced dataset, see table 1 for details.

For the generation method chosen about 10 % of the results can be identified as 3s. The results are quite sparse, because we only sample from the 5 most probable bins. See figure 1 for examples.

| Data | Classification Rate |
|---|---|
| Full | 0.8654 |
| Reduced | 0.8376 |

Table 1: Results for the Bayes Classifier

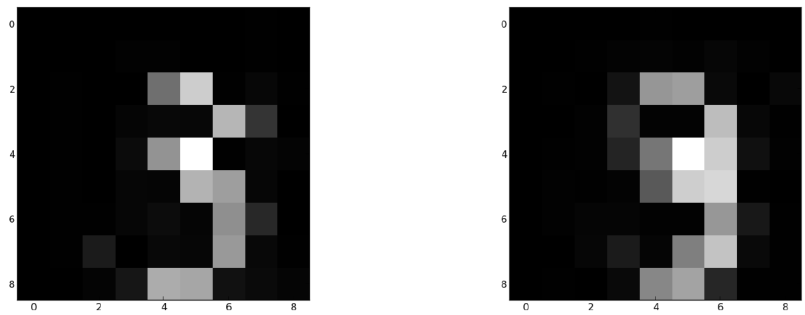


Figure 1: Instances generated with the BayesClassifier.

### 2.2 Density Tree Classifier

The Density Tree Classifier captures the interdependence of the feature dimensions. This is done by building a binary tree from the data for each class. First all data is put in the root node of the tree. This node is splitted in two children nodes according to a split criterion. This is repeated for all nodes until a termnination criterion is met.
We started with a simple splitting criterion in the beginning: We iterate over the dimensions. For each dimension we create a set of thresholds. The instances to the left of this threshold (in current dimension) belong to the left node, the ones to the right to the right node. For each instance (except the two outer ones) two thresholds are created, one to the left of the instance and one to the right. Then a gain-function is calculated for each threshold. Finally the split that maximizes this gain is chosen. We implement the criterion with gain-function

$$\text{gain} \;=\; (\frac{N_l}{N})^2 \, \frac{1}{V_l} \;+\; (\frac{N_r}{N})^2 \, \frac{1}{V_r} \tag{3}$$

where $N$ is the total number of instances in the node, $N_l$, $N_r$ the number of instances to the left / right and $V_l$, $V_r$ the volume to the left / right. From now on we will refer to this criterion as the standard criterion. We will introduce different split criteria later.
To each node the probability $p_i = \frac{N_i}{N\ V_i}$ is assigned. Here $N$ is the total number of instances and $N_i$, $V_i$ the values for the corresponding node.
A node is terminated if it has either reached a maximal depth in the tree or has fallen below the minmal number of instances $N_min = N^{\frac{1}{3}}$.
To predict the labels for new data the posterior is calculated for each class by finding the leave node in which this data point belongs. Then the posterior is identified with the probability of this node. The prior is calculated like in the Bayes Classifier. The class with the biggest product of posterior and prior is predicted.
New instances are generated by walking the tree: Starting at the root node the children nodes are selected with their probability until a leaf node is reached. Then the new data point is sampled uniformly from this node.

#### 2.2.1 Results

This classifier performs considerably worse than the Bayes Classifier. For the reduced data it achieves a classification rate of approximately 68 %, which is 15 % worse than the Bayes Classifier. For the full dataset the classification rate is below 50 %!

To test the generation ability we generate 50 new instances of data. Only in some of these 3s are barely discenible. In general all are noisy. See 2 for examples.

| Data | Classification Rate |
|---|---|
| Full | 0.4928 |
| Reduced | 0.6860 |

Table 2: Results for the Density Tree Classifier

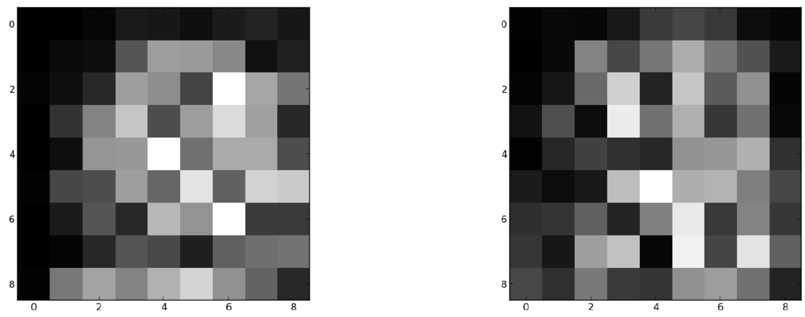


Figure 2: Instances generated with the Density Tree.

# 3 Copula Classifier

The main task of this project is to construct a classifier that exploits the ability of the Density Tree Classifier to capture the interdependency of the feature dimensions while keeping the prediction performance of the Bayes Classifier. This way we hope to increase the generation power compared to Bayes Classifier and Density Tree Classifier.
This is achieved by combining both classifiers: We use the fact that the cumulative density function (CDF) can be calculated from the (normalised) histograms of the Bayes Classifier. This can be combined with a Density Tree Classifier that is trained of the rank order transformation of the training data (result is called copula).
The training of the new classifier can be summarized as follows:

1. Train the Bayes Classifier on the training data.
2. Compute the copula of the training data.
3. Train the Density Tree on the copula.
4. Compute the CDF of the histograms of the Bayes Classifier.

Then labels can be predicted for new data via the following steps:

1. The posterior is calculated with the Bayes Classifier.
2. The CDF is applied to the data to transform it to copula space.
3. The posterior of the Density Tree is calculated on the transformed data.
4. The product of both posteriors and the prior is taken.

This is done for each class and the class with the biggest resulting value is predicted.
New data is generated via:

1. Copula data is generated with the Density Tree.
2. It is mapped back with the inverse CDF from the Bayes Classifier.

In the following we call the resulting classifier the Copula Classifier.

### 3.1 Splits

To improve on the underlying Density Tree Classifier, we implement new split criteria. First we use the split criterion already discussed in 2.2. Then we implement a similar criterion, which uses a different gain function:

$$\text{gain} \;=\; |\; N_l \; V_r \;-\; N_r \; V_l \;| \tag{4}$$

This criterion is supposed to yield more balanced splits than the original, because it balances the number of instances and the volumes of the splits. We will refer to it as alternative criterion.
Finally we implement a criterion based on the gradient of each datapoint. For this we need an estimation of the gradient. First we find the k nearest neighbors of the datapoint. Then we estimate the gradient as the displacement to the center of these neighbors. Additionally the density is estimated by the average distance of the datapoint to the neighbors. We choose to split at the datapoint, which has the biggest value for the gradient divided by density. Then we split along the dimension, where the gradient is biggest.

For a first comparison of the criteria we plot the distribution of the values for the gains / gradient over the thresholds / instances. See figures 3, 4 and 5.

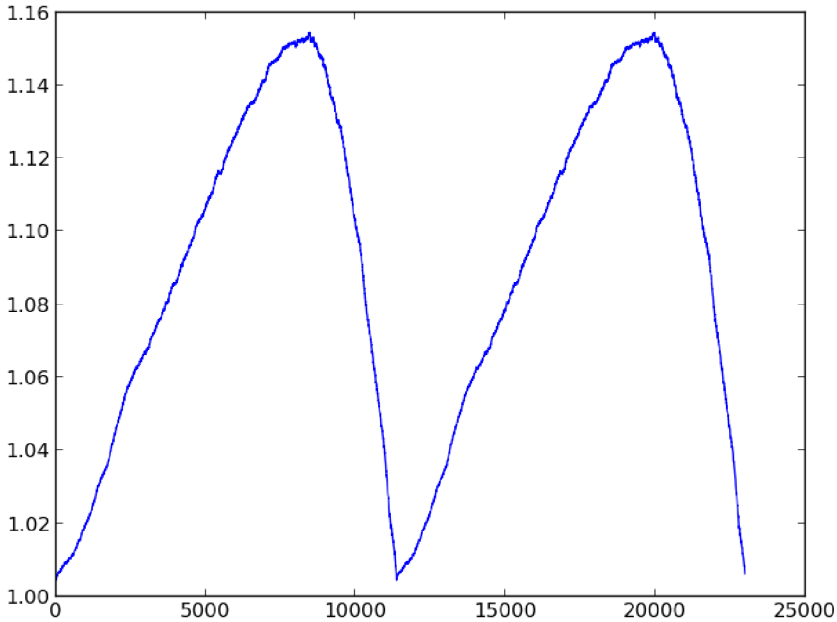



Figure 3: Distribution of the gain over the thresholds for the standard split. Split for the 2nd most important dimension, top tree level.

One can see that the standard and alternative criterion yield a distribution of gains with 2 peaks (this is also true for other dimensions). The distribution of the gradients is quite noisy and there is no clear structure discernible.

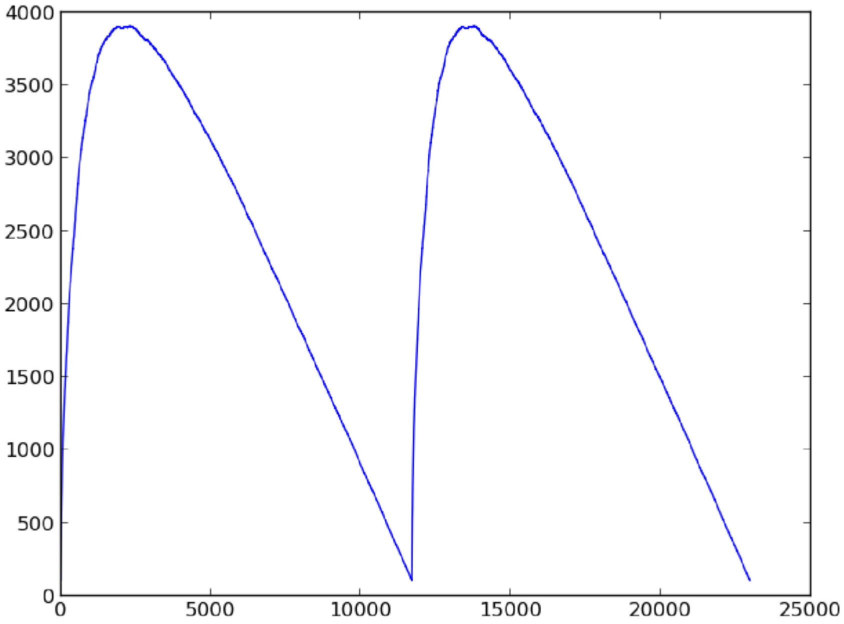



Figure 4: Distribution of the gain over the thresholds for the alternative split. Split for the most important dimension, top tree level.

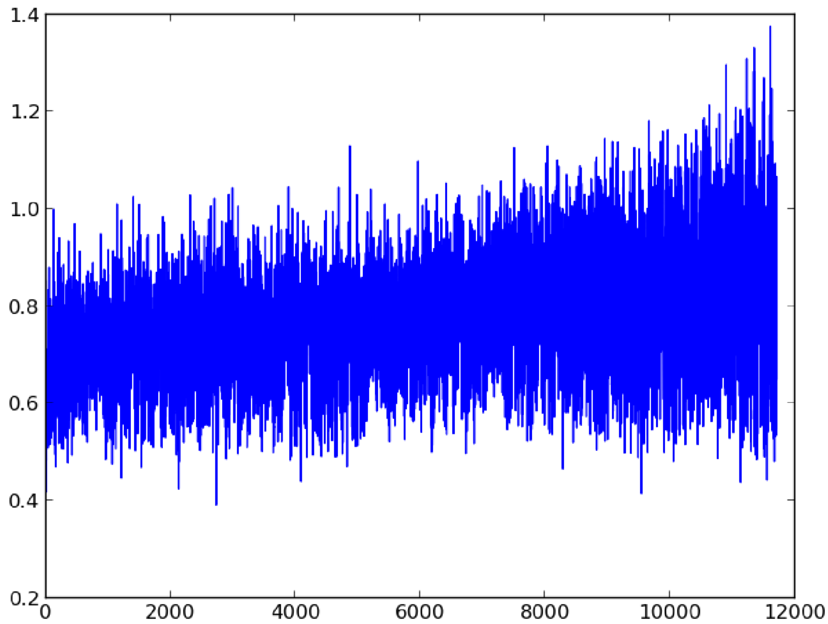



Figure 5: Distribution of the gradient over the instances for the gradient split. Split for the full data, top tree level.

To further examine the splits we also plot the binary tree produced for the reduced dataset. These are shown in figures 6, 7 and 8. Here the label of the nodes corresponds to the number of instances assigned to them.

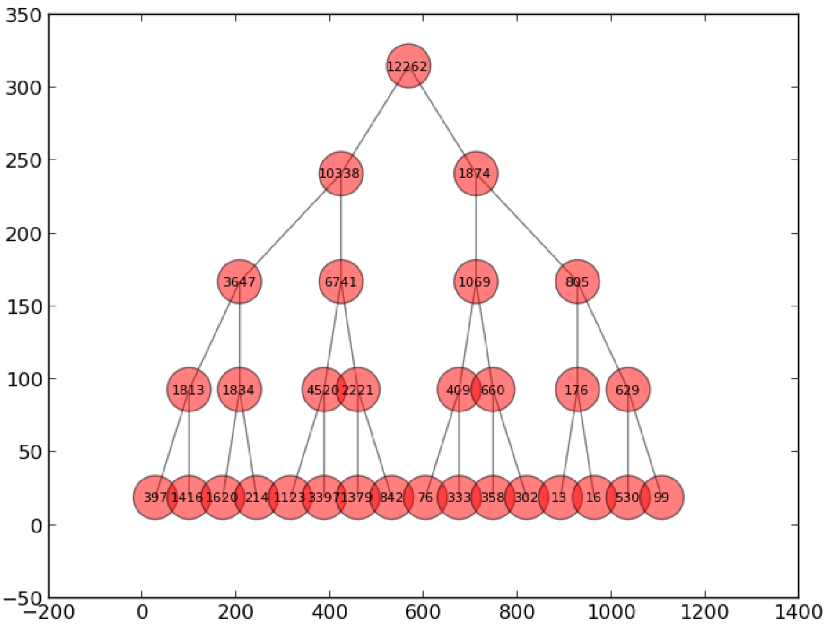



Figure 6: Tree for the standard criterion on reduced dataset.

These trees suggest, that the alternative criterion produces the most balanced splits, whereas the standard criterion produces the trees with the big leaves, but not too many very small leaves. The gradient criterion seems to produce splits with many big leaves and many very small leaves.

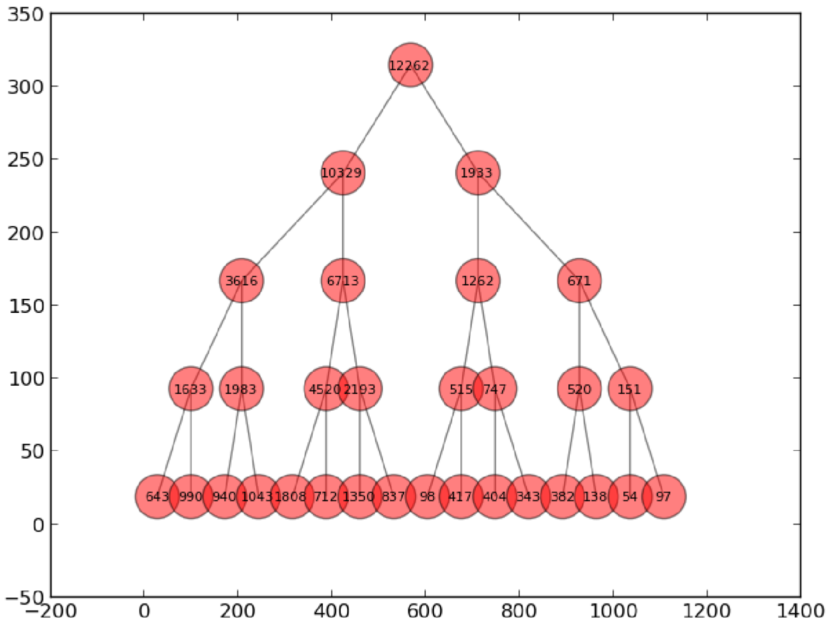


Figure 7: Tree for the alternative criterion on reduced dataset.

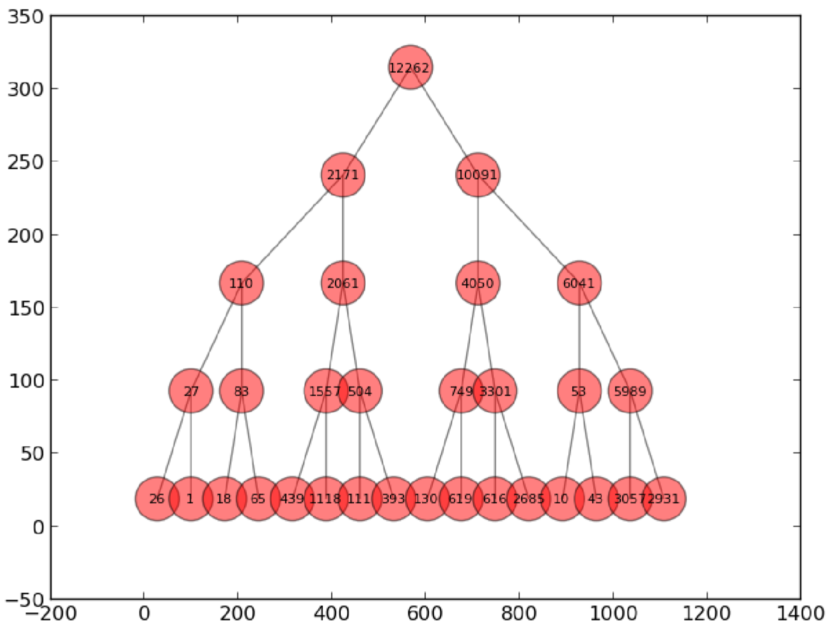


Figure 8: Tree for the gradient criterion on reduced dataset.

### 3.2 Results

To evaluate the split criteria thouroughly, we sweep the parameters they depend on.
First we sweep different tree depth values for the standard and alternative criterion. See table 3 for the results. The performance of the standard criterion on the full data is better by approximately 5 % for the larger depth values. However the performance for the reduced data slightly with increasing depth. For the alternative criterion the performance decreases for full and reduced data with increasing depth. In addition we fear that overfitting effects might be stronger for too large depth values. Hence we continue with a maximal tree depth of 4 for the following tests.

| | Standard | | Alternative | |
|---|---|---|---|---|
| Depth | Full Data | Reduced Data | Full Data | Reduced Data |
| 4 | 0.5285 | 0.8376 | 0.6372 | 0.8364 |
| 8 | 0.5285 | 0.8363 | 0.5286 | 0.8355 |
| 10 | 0.5285 | 0.8343 | 0.5285 | 0.8308 |
| 15 | 0.5806 | 0.8307 | 0.5285 | 0.8299 |
| 20 | 0.5823 | 0.8298 | 0.5285 | 0.8299 |
| 40 | 0.5847 | 0.8299 | 0.5285 | 0.8299 |

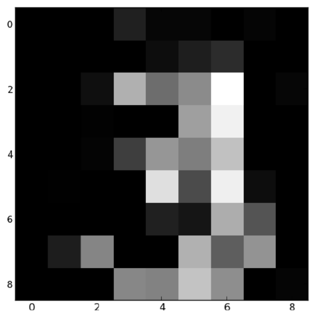
Table 3: Results for different tree depths for the standard and alternative criterion.

Then we sweep the number of neareset neighbors k, that are taken into account for the calculation of gradient and density in the gradient split criterion.
The results are shown in 4. In contrast to the standard criterion the gradient split performs better on the full data. There it performs considerably better than the standard criterion. We further see, that the quality of the splits decrease drastically with too big k-value. We choose to continue with k = 10 for the following tests.
It should be noted that calculating this split criterion takes considerably longer than calculating the standard or alternative criterion.

| k | Full Data | Reduced Data |
|---|---|---|
| 5 | 0.8406 | 0.8188 |
| 10 | 0.8447 | 0.8245 |
| 15 | 0.8428 | 0.8348 |
| 30 | 0.5357 | 0.8159 |

Table 4: Results for different numbers of nearest neighbors for the gradient criterion.

Next we look at the generation of new instances with the Copula Classifier. Figures 9, 10 and 11 show instances generated with the three split criteria.



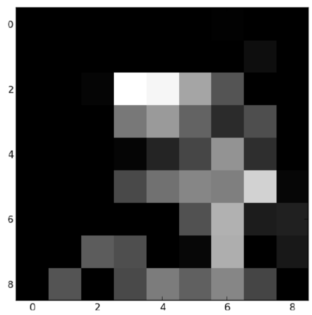




Figure 9: Instances generated with the Copula Classifier (default criterion).

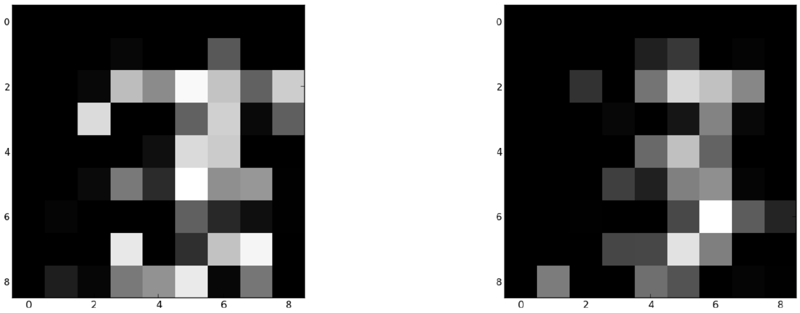


Figure 10: Instances generated with the Copula Classifier (alternative criterion).

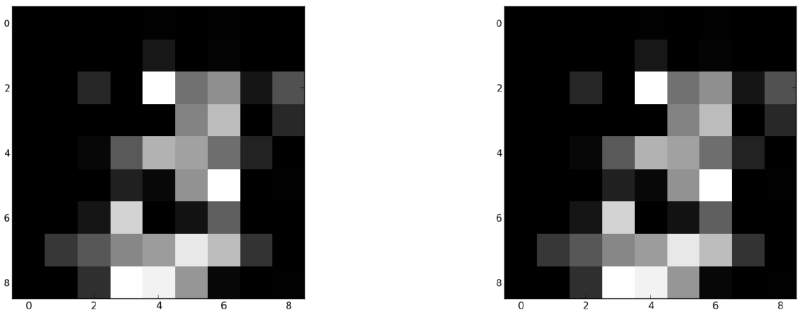


Figure 11: Instances generated with the Copula Classifier (gradient criterion).

About 10 to 15 % of the instances generated with all three criteria are discernible as 3s. In comparison the results with the Bayes Classifier they look more noisy, but the shape of the 3 is in general more distinct. Qualitatevely there is no difference between the generated data of the three criteria.

Next we compare all three criteria and also compare them with the Bayes Classifier. Table 5 shows all results. We see that the Bayes Classifier has the best classification results on full and reduced data. The only split criterion that yields good results on the full dataset is the gradient criterion. However it takes about 140 times longer for training than the other criteria.

| Split / Class. | Bayes | Standard | Alternative | Gradient |
|---|---|---|---|---|
| Full | 0.8654 | 0.5285 | 0.6372 | 0.8245 |
| t_train | 0.7 s | 61.3 s | 62.7 s | 140 min |
| Reduced | 0.8376 | 0.8375 | 0.8364 | 0.8245 |
| t_train | 18.7 ms | 1.6 s | 1.6 s | 573 s |
| Generating | $0.9391 \pm 0.0265$ | $0.8886 \pm 0.0095$ | $0.8158 \pm 0.0123$ | $0.7998 \pm 0.0126$ |

Table 5: Results for different numbers of nearest neighbors for the gradient criterion.

We further evaluate the generating performances with the Random Forest Classifier from sklearn. For this we train it on the complete MNIST dataset (i.e. with all numerals not only 3 and 8). Then we predict the labels for 1000 instances generated with the different split criteria and the Bayes Classifier. This is repeated 50 times to exclude randomness effects from the Random Forest Classifier. The forest is set up to consist of 30 trees. It achieves a classification rate of $0.9223 \pm 0.0007$. The classification for the generated data is reported in the last row of table 5. The errors were estimated with the standard deviation of the 50 trials.

According to this classification the Bayes Classifier generates the best data. However this might be explained by the fact that it generates sparse images with the generation method we have chosen. This leads to less noise in the images and might increase the correct prediction of the Random Forest Classifier also for generated data that is not discernible (by human standards) as 3. To investigate this hypothesis one could try to weaken the noise in the generated instances of the Copula Classifier, e.g. by introducing a threshold and test whether this way one obtains better generation results.

The split criterion that performs best in this validation is the standard criterion. It is better than the other two criteria, which perform at level, by nearly 8 %. This is quite surprising, because this criterion performed worst with regard to prediction.
To further evaluate the generating performance of the Copula Classifier, we will turn to more complex data, finalizing with a dataset of words. We hope that the Copula Classifier will profit from its ability to take into account feature dimension interdependency here.

# 4 Complex Data

Firstly, we test our classifiers against the classification problem generator from sklearn `sklearn.datasets.make_classification` which produces a Madelon-like dataset. Secondly, we set up a scheme in which we train our algorithm to generate valid words. The lists of British words used are themselves generated invoking scowl `http://wordlist.aspell.net/`.

## 4.1 Madelon (sklearn)

We consider datasets composed of 10000 instances (8000 train, 2000 test) to check the validity of our algorithm for a non-linear hypercube problem. Here, the instances are clustered at the vertices and split into two classes with 20 feature dimensions each. By incrementing the number of informative features, we can add covariance in order to increase complexity. Those randomly generated classification problems are studied by taking the mean values for 100 problems generated. Like this we quantify the behaviour of a classifier in response to an arbitrary chosen classification task of given difficulty.

| # informative features | Bayes | Standard | Alternative | Gradient |
|---|---|---|---|---|
| 2 | 89.7± 3.7 | 90.5± 3.7 | 83.8± 7.3 | 84.7± 8.7 |
| 3 | 84.6± 6.0 | 83.9± 6.6 | 78.8± 7.9 | 77.7± 9.7 |
| 5 | 81.1± 6.4 | 80.1± 5.9 | 77.5± 6.4 | 76.2± 6.5 |
| 10 | 79.7± 4.5 | 79.6± 4.8 | 80.0± 5.3 | 77.3± 4.0 |

Table 6: Correct classification rate (accuracy) in % for different types of classifiers. The mean over 100 (20 for the Gradient) samples is calculated with the standard derivation taken as the error.

As seen in Table 6 the Bayes and Standard Copula perform equally well in terms of overall response. While the mean values are close, the same holds for each problem individually: In case Bayes performs better for a given problem the Copula classifier performs better as well, even at high complexity. This behaviour is truly unexpected, since the Copula should take the covariant interaction into account.
Regarding different splits exposes that the choice of the split criterion strongly effects the accuracy of the Copula classifier. The malperformance of Alternative and Gradient suggests that the Standard split is more robust with respect to clustering of the data. This is in contrast with the results for the MNIST data where the Gradient and Alternate splits perform at least as well as the Standard approach.

Somehow the results of the Standard Copula seem to be restricted by the Bayes results. Therefore it might be worth reconsidering the probability assigned to each node in the Density Tree as defined in Section 2.2. A change would reweight the Densety Tree against the Bayes part in the Copula classifier. Alternatively, one may aim to outrank the performance of the Bayes by developing even more sophisticated split criteria. Such splits should be sensitive to clustering of the data.

## 4.2 Wordlists

In this section the Bayes, Density Tree and Copula Classifiers are trained with datasets consisting of 'valid words' and 'non-words' of a given length. Our lists of valid words incorporates all British-spelled words excluding abbreviations, special signs and proper names (e.g. abbrv., we're, Africa, Einstein, ...). Non-words are random character strings not included in those wordlists. The goal is to subsequently generate valid words of this length.
The number of features is determined by the word length and the feature space has the dimension 26 according to the low-letter alphabet. We study the behaviour of the classifiers for words of length 5,7 and 10 and discuss the results. Again the train set makes up 80% and the test set 20% of the data. As the standard split performs best with the clustered data, we use it in the following.

Note that the setup is formally equal to a binary classification, where 'true' stands for valid word and 'false' for non-word. Hence we can analyse the errors of the classifiers accordingly.

| wordlength | Bayes | | Copula | | Density Tree | |
|---|---|---|---|---|---|---|
| | ACC | GAIN | ACC | GAIN | ACC | GAIN |
| 5 | 0.846 | 1.691 | 0.843 | 1.686 | 0.638 | 1.368 |
| 7 | 0.926 | 1.852 | 0.926 | 1.853 | 0.708 | 1.416 |
| 10 | 0.967 | 1.935 | 0.968 | 1.935 | 0.719 | 1.438 |

Table 7: Correct classification rate (accuracy) and gain in accuracy.

Table 7 shows the classification rate and gain of each classifier tested. As expected from earlier tests the Bayes and Copula perform equally well and the Density Tree (depth 10) lacks in accuracy. The gain is jet another measure for the performance and is calculated by

$$\text{gain} = \frac{\text{accuracy}}{\text{random accuracy}} \, ,$$

where

$$\text{random accuracy} = \left(\frac{\text{positive targets}}{\#\text{instances}}\right)^2 + \left(\frac{\text{negative targets}}{\#\text{instances}}\right)^2 \, .$$

In this setup a gain equal to one would indicate no better classification than random and it is true that the higher the gain, the better the classifier. This quantity is useful for comparison of different classifiers and different splits. Here it confirms that Bayes and Copula performance is approximately equal while the pure Density Tree is less informative due to the imbalance in splits mentioned in Section 2.2.1.

There are evidently two types of classification errors that can occur – a word that gets classified as a nonword (type I error) and the opposite, a nonword classified as a valid word (type II error). Analysis on this error behaviour is shown in Table 8.

| wordlength | Bayes | Copula | Density Tree |
|---|---|---|---|
| 5 | 1.200 | 0.946 | 4.933 |
| 7 | 1.116 | 1.143 | 8.249 |
| 10 | 1.127 | 1.192 | 38.55 |

Table 8: Error analysis (type I error/type II error) for different classifiers.

What is striking is the high occurrence of type I errors for the Density Tree. It explains the low classification rate and states that a lot of valid words are classified as nonwords. For the Bayes and Copula the errors are balanced. This indicates that i) Bayes suppresses the type I errors coming from the Density Tree part of the Copula or ii) that the Density Tree behaves differently when trained on the cumulative density. Either way the Bayes classifier seems to govern the overall behaviour of the Copula classifier. This of course prevents Copula from neither performing better nor significantly worse than Bayes.
Finally we will check which classifier generates the best results.

### 4.2.1 Generation Bayes

We generated 5000 words of length 5, 7 and 10 each and found 62, 9 and 0 words to be valid words respectively. They are:

length 5: ['cuber', 'benny', 'saker', 'snits', 'peaky', 'giber', 'gooks', 'babes', 'sober', 'seine', 'bilks', 'certs', 'filer', 'raked', 'finer', 'bulks', 'rooks', 'pukes', 'poker', 'shine', 'conte', 'coons', 'bakes', 'garbs', 'raker', 'conks', 'panne', 'bikes', 'boons', 'biker', 'barer', 'sales', 'foams', 'bobby', 'goner', 'pales', 'robed', 'beats', 'bares', 'pones', 'faker', 'coles', 'cured', 'booby', 'abase', 'seams', 'gorse', 'fakes', 'cools', 'foals', 'booms', 'forby', 'soaks', 'beans', 'cabby', 'poles', 'shims', 'cooks', 'pains', 'sties', 'piker', 'ruler']

length 7: ['carrier', 'carried', 'darkies', 'ceasing', 'parties', 'pansies', 'rabbles', 'sensing', 'coolies']

The Bayes generation process samples from the 5 most probable bins uniformly at random. Therefore the letters for each feature dimension are restricted to the 10 most probable ones. Each produced word then consists of letters which were sampled according to the probability of its feature only. This restriction can be seen as most of the words feature vowels like 'a' and 'o' in the middle and often end on 's', 'r' or 'y'.
Regarding words of length 7 'carrier' and 'carried' as well as 'parties' and 'pansies' are quit alike. Bayes' restriction to independence of features and to 10 generated features is not a good one since only 0.2% of the words generated are valid. Moreover since no interactions between the features are taken into account we do not observe dominant occurrences of probable phrases as 'th' or 'ly'.
No words of length 10 were generated. This may be understood as with growing word length the number of independent features rises. Longer words usually include a larger number of different letters.

#### 4.2.2 Generation Copula

We generated 5000 words of length 5, 7 and 10 each and found 65, 2 and 0 words to be valid words respectively. They are:

length 5: ['lated', 'tamer', 'farad', 'paper', 'cager', 'raper', 'facer', 'haler', 'taper', 'rares', 'baler', 'laser', 'paced', 'tater', 'famed', 'rarer', 'tares', 'rared', 'basic', 'baker', 'faker', 'emmer', 'fatly', 'baser', 'raced', 'fader', 'faded', 'caber', 'rater', 'rates', 'tabes', 'rated', 'faqir', 'radio', 'radar', 'fared', 'raker', 'laced', 'gapes', 'maser', 'rainy', 'barer', 'baked', 'cameo', 'koala', 'based', 'paged', 'takes', 'lames', 'taler', 'later', 'taser', 'laker', 'gamed', 'gamer', 'farer', 'parer', 'ratel', 'cutey', 'baaed', 'raked', 'lamer', 'baled', 'pared', 'tapas']

length 7: ['statant', 'hatcher']

As the Copula generator is based on the Bayes we only observe ten different letters per feature dimension. The overall generating performance of the Copula is equal to Bayes. Still a different type of valid words seems to be produced. Except for 'emmer', 'koala' and 'cutey' the second letter is always an 'a'. Thus the probability to produce those words is enlarged. The fact that the Copula classifier fails to generate valid words of length 10 entirely and only two for a wordlength of 7 is disappointing. We conclude that the Copula does not take the interaction into account properly.

## 5 Summary

In this evaluation we studied the behaviour of the Copula classifier with regard to the naive Bayes and Density Tree classifiers. Focus lay on the classification and generation performance for the MNIST and wordlists datasets. The idea was to create a Copula classifier that achieves better performances by splitting the classification problem in a non-interacting and interacting part. Thus it combines a naive Bayes which treats features independently and a Density Tree classifier by using its capability to include feature interactions.
Unfortunately, our results do not indicate such a gain in performance neither in classification, generating nor time efficiency. The Bayes classifier achieves equal or even better results than the Copula in 1/60 the time. We conclude that the Copula does not take the interaction into account properly.

Different split criteria have been tested for the Density Tree and Copula in terms of classification rate. The Standard and Gradient splits are found to behave equally in most regimes while the Alternate split performs badly for clustered data. More sophisticated splits may be developed to include the interaction between the features.