Data Mining and Machine Learning
Coursework 1

Emanuele De Pellegrin
Fraser Garrow
Odhran McHugh
Lucía Parga

November 2019

Contents

1 Introduction

1.1 Contribution

The work was split evenly among all group members.

1.2 Aim

The aim of this coursework is to understand and gain knowledge in data mining and machine learning methods. For that, we are asked to work with given data for further analysis, preparing it beforehand using tools as: confusion matrices, correlation and feature selection for machine learning tasks. In the analysis part, as explained in the lectures, two methods are used: data clustering and probabilistic data analysis.

1.3 Data Set

The data set for the coursework is a sample from *Stallkamp et al's German Street Sign Recognition Benchmark*. Originally the data set consisted of 39,209 RGB-coloured train and 12,630 RGB-coloured test images of different sizes displaying 43 different types of German traffic signs.

For this coursework, 11 training data sets are provided: one entire sample [**train_gr_smpl.csv**] contains train features and labels from the entire sample represented as row vectors and [**train_smpl_<label>.csv**]

The data given consists of 10 classes and 12660 images. The images have been given converted to grey-scale with pixel values ranging from 0 to 255, and were re-scaled to a common size of 48*48 pixels. Hence, each row (= feature vector) in the data set has 2305 features, and represents a single image in row-vector format (2304 features) plus its associated label.

1.4 Tools

Python has been used for this Coursework. For the Beyond Naive Bayes part, Weka has been used in the Beyond Naive' Bayes section. The link to our Github repository is: https://github.com/Cryoscopic-E/Data-mining-coursework1

2 Data preparation

Since Python has been used, the given data in *.csv* format was used: there was no need to convert the data into *.arff* format.

In order to maximise the value of the data, the following pre-processing steps were taken:

1. The rows of image data which were originally ordered according to class, were randomised in order (but their related attributes were still kept together). We use *seed*(100) to produce a consistent randomization of the rows of picture data.

2. Data is normalised to improve the contrast in the darker and blurrier images. The minimum and maximum pixel values for each image are found, and all pixels in the class are normalised form 0 - 255 from that. The core part of that function is:

```
for pixel in pixels:
    pixel = int((pixel-minVal)*(255)/(maxVal-minVal))
```

3. Images were cropped to attempt to remove the background. This was done by:

```
re = np.reshape(image, (48, 48))
sub_matrix = re[9:37, 9:37]
```

2.1 Data Randomisation

The data was randomised to ensure that the train test split functions were sampling an equal amount of each class.

2.2 Data Normalization

The difference normalization makes to an image from the data-set is shown below.

(a) Original image (b) Normalised image

Figure 1: Improvement in contrast using normalisation methods

Normalization modifies the pixels value of each image in the data set. We transformed a gray scale image with pixel value between [minValue, ..., maxValue] to [0, ..., 255]. All the data set now is consistent, and this lead us to better classification accuracy.

2.3 Data Slicing

The difference image cropping makes can be seen below.

(a) Normalised image (b) Normalised image with cropping

Figure 2: Output of cropping every picture equally

The rationale for cropping the images was a rudimentary form of attribute selection (prior to step 5 of the assignment, of choosing the best 10 features). It was noted in the lecture slides (Lecture 9) [1] that fewer attributes could lead to better accuracy. The reasoning being that the background is nothing but noise and is interfering with the accuracy. Although it was noted in the coursework specification that the signs were not centered, it was found experimentally that selecting a 28*28 slice of the 48*48 image from the centre captured to sign and removed the background very well. Had the data come with a bounding box, other more accurate methods could have been employed however this worked in place of such accurate measures.

3 Classification: Naive Bayes

3.1 Aim

Since exploiting all conditional probabilities is computationally expensive, in this part of the coursework we work with Naive Bayes. Naive Bayes is a classification method based on the Bayes' theorem, and hence provides the probability that the a feature (in this case, a pixel) belongs to a class (sign). The way that Naive Bayes' work is based on the assumption that, given the class, the attributes are independent of each other, Thus, when calculating the probabilities for each attribute, this make the calculations tractable. Naive Bayes algorithm's task is thus of training a model based on labeled training data which then can be used to assign a pre-defined class label to new objects. [9] [1]

3.2 Methodology

The data set prepared as explained in the section before is now split in two: 70% of the data is used as a training set and 30% of the data is used as test set. The following code shows how the data is split:

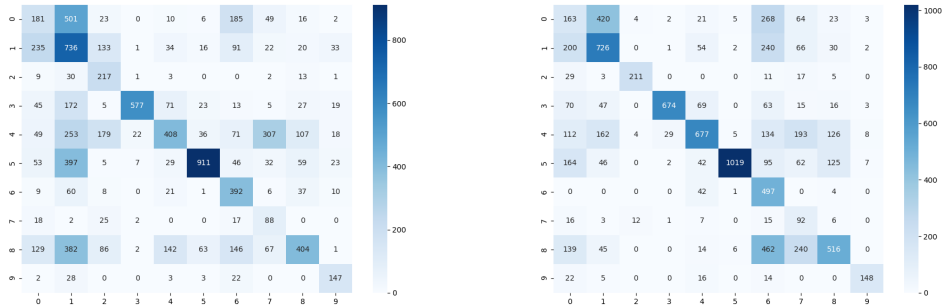
```
def split_sets(data_set, class_set, train_set_perc):
train_set_perc = max(min(train_set_perc, 100), 0)
n_tr_set = int(train_set_perc/100 * len(data_set))
n_ts_set = len(data_set) - n_tr_set

return (data_set[:n_tr_set], class_set[:n_tr_set],
data_set[n_ts_set:], class_set[n_ts_set:])
```

We obtain a python set that contains: training set, training set classes, test set and test set classes.

The method created for the Naive Bayes Classification uses the GaussianNB imported from sklearn. Training data set is used to train the model (`.fit()`) and then this one is tested with the test set(`.predict()`). Results as a classification report, confusion matrix and accuracy score are obtained (as an output file).

3.3 Results



(a) Normalized data set (b) Sliced and normalized data set

Figure 3: Confusion matrix for the data set

Results obtained show that the accuracy is *higher* when we use the **sliced data** (*accuracy=0.53*) than with the **full size data** (*accuracy=0.46*)

3.4 Conclusions

It was assumed that cropping the image would increase our accuracy, as the background is essentially acting as noise which affects our results in a negative manner. An interesting example of this was a machine learning experiment in the University of Washington, which sought to differentiate between huskies and wolves, which was initially thought to be very good classifier but in the end it's main distinguishing factor was not the dog itself, but rather was there snow in the background of the picture (as the pictures of huskies were typically in the snow). [8]

This results confirm our prior hypothesis and hence, by using the sliced images (with the border pixels eliminated) we remove the unnecessary objects such as the background from the images for the classification [6], giving us a better accuracy in our results (almost 10% better using our sliced images).

From the information given in the confusion matrices we can say that the the signs with label 2, 6 and 9 are the most easily to be correctly classified but signs with labels 0,1,4,5 and 8 are the hardest to classify (they have confusing features that enhances their probability to be incorrectly classified). Ex: label 0 sign is easily classified in class 0 and 6, also class 8 is easily incorrectly classified in class 6. The information also shows that signs with *label 0, 1 and 2* are the easily confused between each others (ie: 200 features were predicted as label 0, when they belonged to label 1).

This makes us think that some features affect more than other for a correct classification. We can conclude that to obtain a better accuracy in our classification we need to get rid of some of the "noisy" data [1] and hence, we move on with the next task.

It is thought that visual similarities between classes led to decreased accuracy in identifying that class. For example, the confusion matrix shows it has high accuracy in guessing what was a *right of way at crossing* sign, which is the only upward facing triangular sign out of all the classes. However, it had very poor accuracy in guessing the *speed limit 60* sign, which could be due to the fact that there are other classes of circular signs and other classes of signs with numbers on them.

4 Deeper analysis of the data

In order to obtain a better accuracy in our Naive Bayes Classification (as it was seen and explained in the lectures), we are asked to create and use a method that selects the "10 best attributes" of each class which correspond in our case with the "10 most representative pixels" of each of our classes.

4.1 Methodology

In order to do this, we need to state a relation between our pixels (features) and the correspondent classes. With this, they can be 'scored' by the strength of their relation, meaning that the most representative pixels of the sign class will be highly scored and weak pixels (ie: background) will be low scored. The method chosen for uses the the Pearson Chi Square. This test is commonly used for categorical features, the algorithm works by calculating how the expected observation of a class deviates from the observed class.

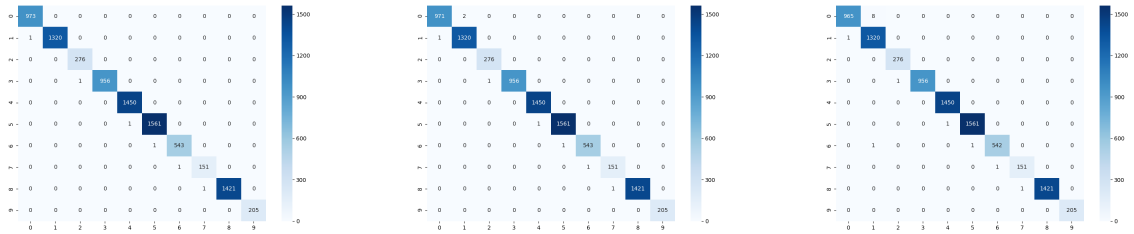
$$\tilde{\chi}^2 = \sum_{k=1}^{n} \frac{(O_k - E_k)^2}{E_k}$$

If for example two features are independent then we will get a lower chi2 value. Higher values indicates a features is more dependent, thus it can be used as representation of a class and used for training [4]

4.2 Results

The accuracy obtained for the different best features is:

- 10 best features: accuracy = 99.93%



(a) 10 Best features (b) 5 Best features (c) 2 Best features

Figure 4: Confusion matrix for the data set

- 5 best features: accuracy = 99.91%
- 2 best features: accuracy = 99.83%

4.3 Conclusions

Based on the Lecture 9 slides [1] of this Course information and papers found about this topic (from the Lecture notes and Vision), we know beforehand that reducing the number of attributes will give us a better accuracy. The reason for this is: 1. The data can have irrelevant instances for the desirable classification that only adds 'noise' to it. 2. Machine learning will pick up on spurious correlations, that might be true in the training set, but not in the test set.

By using a more representative set of our data we expect an increase in the accuracy. incl then that the accuracy increases a lot. To enhance this accuracy, we have also selected the best 10 features in our reduced data set, and we highlight here that the accuracy is even better.

In this part we are also asked to select the best 5 and 2 features. As taught in the Lecture 9, the accuracy and the data is linearly related until a peak: accuracy increases with the amount of data for the experiment, until a peak where more data gives 'noise' and errors and reduces the accuracy.

With the information of the previous experiments, we know that with the 10 best features, the accuracy has increased by till 99% which is a huge increase.

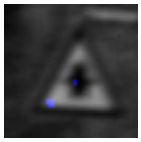
The results obtained show that the accuracy decreases when we only use 2 and 5 features, hence, we are still located in the "growing" regime, and we can state that the peak of maximum features we can use will be around 10.

To obtain a better visual image of where those best features are allocated we combined these in the images in grey scale. Using opencv we have randomly selected an image for each class, and combined with a new image created by reading the file containing the most important features and highlight them in blue. The first 10 features extracted using this method are displayed in the images below using the normalized images not sliced.

(a) Class 0

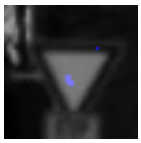
(b) Class 1

(c) Class 2



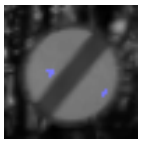
(d) Class 3

(e) Class 4



(f) Class 5

(g) Class 6



(h) Class 7

(i) Class 8

(j) Class 9

5 Beyond Naive Bayes

5.1 Methodology

We weren't be able to find a python package that fit our needs. We decided to use python and create a smaller data set with the 2 top most pixel feature for each class. We then convert to '.arff' to use in Weka and build the Bayesian networks models with the following search algorithms:

1. K2 (1 and 3 parents)
2. TAN
3. Hill Climbing

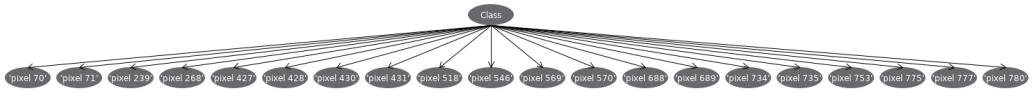
We used the first 70% of data as a training set and the remaining 30% as a test set

The *hill climber* algorithm is the most basic search algorithm, where most of weka's other search algorithms stem from (i.e.: *LAGD Hill Climbing* & *K2* are modified versions of the hill climbing algorithm). The hill climber algorithm traverses the search space and tends to maxima (be it local or global maxima). The algorithm uses heuristic values to guide its search. 7b

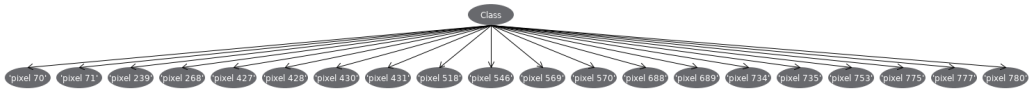
The *K2* algorithm also is a heuristic search model, which looks at the most likely belief structure from it's network, with all assumptions taken into account.[3]

The *TAN* algorithm develops on the Chow-Liu algorithm. The structure of this representation is the exact same structure, with connection from the classifier to each attribute. This algorithm calculates the parameters that would lead to the best weighting tree structure of the naive bayes network. [2]

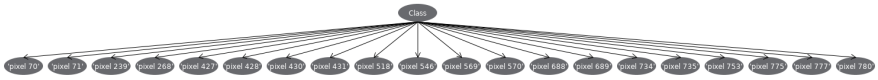
5.2 Results



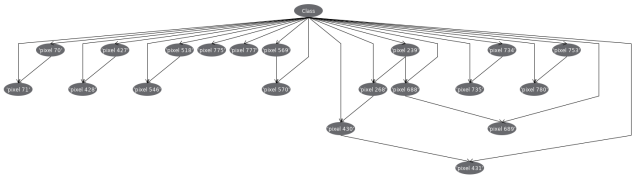
(a) Hill climber matrix confusion matrix



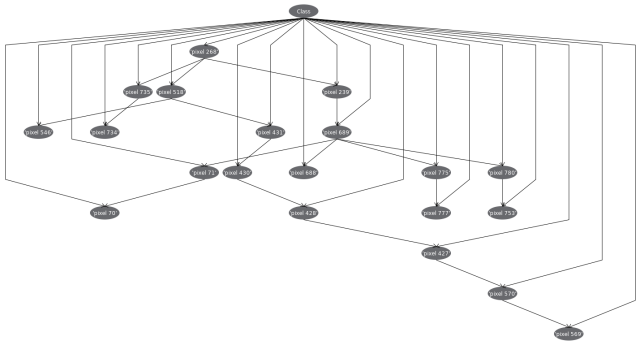
(b) Hill climber



(c) K2 - 1 parent



(d) K2 - 3 parents



(e) TAN

Figure 6: Beyond Naive Bayes graphs

a	b	c	d	e	f	g	h	i	j	<-- classified as
142	164	5	5	26	11	31	6	58	0	a = 0
103	272	10	3	55	26	30	11	60	0	b = 1
2	8	81	0	14	0	0	0	19	0	c = 2
1	3	6	283	41	9	26	2	32	4	d = 3
2	24	8	6	334	15	15	44	134	18	e = 4
17	21	2	20	24	431	33	5	79	4	f = 5
2	3	1	1	15	7	165	0	46	2	g = 6
0	0	5	0	13	1	4	36	5	0	h = 7
20	19	6	1	130	16	65	34	296	26	i = 8
1	3	0	0	17	4	1	0	21	47	j = 9

(a) Hill climber. Confusion matrix

=== Summary ===		
Correctly Classified Instances	2087	54.95 %
Incorrectly Classified Instances	1711	45.05 %
Kappa statistic	0.4824	
Mean absolute error	0.0928	
Root mean squared error	0.2708	
Relative absolute error	53.5739 %	
Root relative squared error	91.9646 %	
Total Number of Instances	3798	

(b) Hill climber. Summary

a	b	c	d	e	f	g	h	i	j	<-- classified as
142	164	5	5	26	11	31	6	58	0	a = 0
103	272	10	3	55	26	30	11	60	0	b = 1
2	8	81	0	14	0	0	0	19	0	c = 2
1	3	6	283	41	9	26	2	32	4	d = 3
2	24	8	6	334	15	15	44	134	18	e = 4
17	21	2	20	24	431	33	5	79	4	f = 5
2	3	1	1	15	7	165	0	46	2	g = 6
0	0	5	0	13	1	4	36	5	0	h = 7
20	19	6	1	130	16	65	34	296	26	i = 8
1	3	0	0	17	4	1	0	21	47	j = 9

(c) K2 - 1 parent. Confusion matrix

=== Summary ===		
Correctly Classified Instances	2087	54.95 %
Incorrectly Classified Instances	1711	45.05 %
Kappa statistic	0.4824	
Mean absolute error	0.0928	
Root mean squared error	0.2708	
Relative absolute error	53.5739 %	
Root relative squared error	91.9646 %	
Total Number of Instances	3798	

(d) K2 - 1 parent. Summary

=== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	<-- classified as
129	170	7	8	35	20	27	5	47	0	a = 0
103	254	8	6	69	30	27	8	64	1	b = 1
2	8	77	0	12	2	0	2	21	0	c = 2
9	7	9	284	46	5	10	2	33	2	d = 3
8	27	6	11	397	19	3	14	103	12	e = 4
23	33	2	20	38	408	15	1	95	1	f = 5
6	11	0	0	17	4	131	0	72	1	g = 6
1	2	7	0	13	0	1	30	8	2	h = 7
18	12	4	12	108	23	22	14	391	9	i = 8
1	6	0	0	15	7	1	0	27	37	j = 9

(e) K2 - 3 parents. Confusion matrix

=== Summary ===		
Correctly Classified Instances	2138	56.2928 %
Incorrectly Classified Instances	1660	43.7072 %
Kappa statistic	0.494	
Mean absolute error	0.092	
Root mean squared error	0.2578	
Relative absolute error	53.0764 %	
Root relative squared error	87.5356 %	
Total Number of Instances	3798	

(f) K2 - 3 parents. Summary

=== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	<-- classified as
145	125	3	4	39	49	7	0	76	0	a = 0
88	266	0	8	65	66	4	1	72	0	b = 1
10	25	27	2	20	8	0	0	32	0	c = 2
19	50	1	191	37	43	5	0	61	0	d = 3
14	38	2	10	351	71	0	0	113	1	e = 4
14	46	1	10	74	416	2	0	73	0	f = 5
5	7	0	2	9	9	134	0	76	0	g = 6
3	15	4	2	19	5	0	3	13	0	h = 7
11	17	0	8	61	45	5	0	465	1	i = 8
4	3	0	4	13	5	2	0	53	10	j = 9

(g) TAN. Confussion matrix

=== Summary ===		
Correctly Classified Instances	2008	52.8699 %
Incorrectly Classified Instances	1790	47.1301 %
Kappa statistic	0.4474	
Mean absolute error	0.1036	
Root mean squared error	0.2578	
Relative absolute error	59.8073 %	
Root relative squared error	87.531 %	
Total Number of Instances	3798	

(h) TAN. Summary

Figure 7: Beyond Naive Bayes information

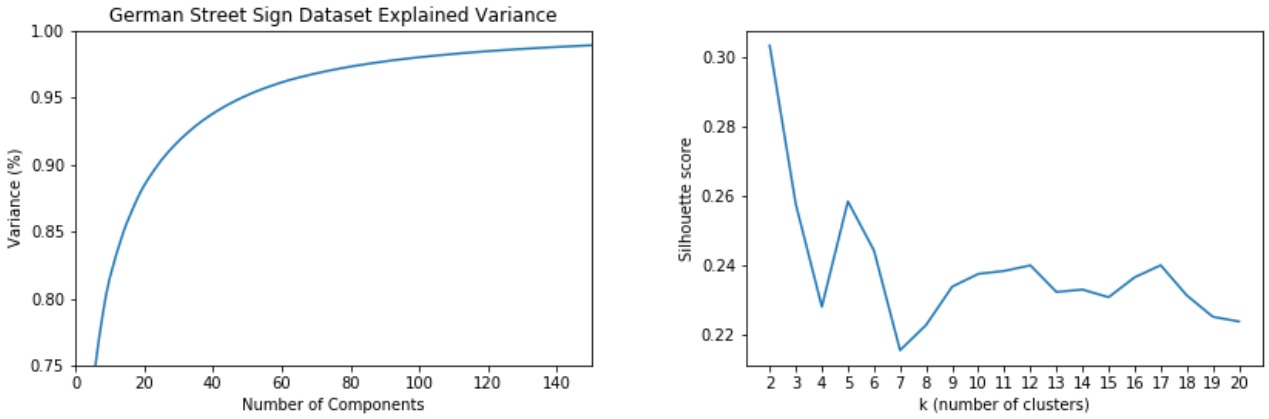
5.3 Conclusions

The hill climber algorithm yielded a Naive Bayes network, where the attributes were all children of one parent node, which was the class. The K2 algorithm was run twice, first with one parent, which gave the same structure as the hill climber algorithm. However, the K2 algorithm was also executed with three parents, which takes much longer to run, but it addition to the class being a direct parent to all attribute nodes, there were chains from class to attribute to another attribute and so on, in the structure. The TAN algorithm had a similar structure to the K2 algorithm with 3 parents.

The difference in runtime was significant, with the hill climber algorithm only taking 13.91 seconds to build, compared to the K2 with three parents for example, which took 830.01 seconds to build. However, the difference in correctly identified was only an increase from 54.95% to 56.2928%. This small increase in performance for a factor of 10 increase in building time was deemed negligible.

6 Clustering

Before K-means clustering was performed on the dataset, the dimensionality of the data-set was reduced by principal component analysis. The explained variance is plotted (Figure 8a) and showed that a reduction to 100 components (from 784, i.e. the reduced 28 x 28 image) preserves about 98% of the total variance of the data.



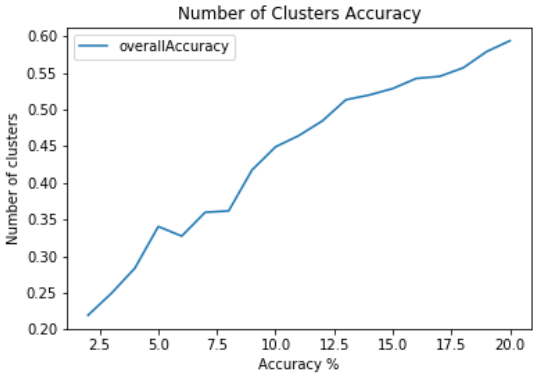
(a) Explained Variance

(b) Silhouette Score

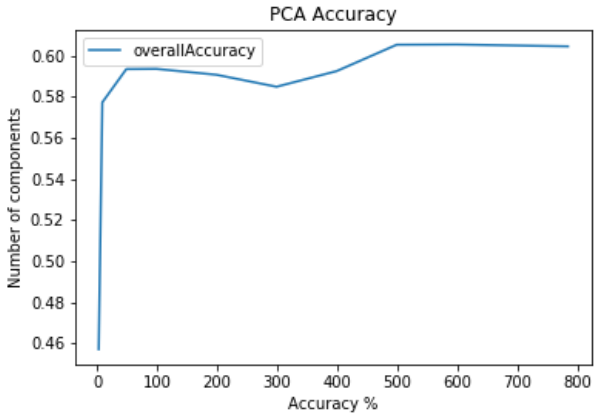
For K-means clustering the number of clusters is specified upfront and the number of clusters can drastically affect the quality of the results. There are a number of methods to determine the optimal number of clusters. For low-dimension data it may be possible to visualise, however, that is not possible in this case. Here, we already know that we have 10 classes, although counter-intuitive, it is unlikely that this means that the clustering algorithm will determine 10 distinct clusters with a one to one relationship. Signs in one class may be photographed differently and assigned a different cluster even though they are of the same class. The silhouette score is normally a suitable metric for predicting the a good choice for the number of clusters. The silhouette score is the mean silhouette coefficient of all the instances, where the coefficient describes how similar an instance is to its own cluster compared to the other clusters. The silhouette score for clusters of 2 through 20 is presented in Figure 8b.

As the silhouette score did not give a clear indication of the optimal number of clusters to use, K-means was performed for multiple initialisation of different numbers of clusters. To measure the 'performance' of the clustering algorithm, the clusters were evaluated against the actual corresponding labels. The cluster labels can be thought of as aliases for the actual labels, determined by the most commonly occurring actual label in each cluster. How well defined a cluster is, is defined by how closely grouped the instances are within it, i.e. whether there are many 'incorrect' instances in the assigned aliases. The overall accuracy is

defined by the sum of the most common occurring labels in all of the clusters divided by the total number of instances. The overall accuracy for k-means performed on the data set for clusters of 2 through 20 has been presented in Figure 9a. This overall accuracy for K-means has also been checked, varying the number of principal components, this shows that for most values the accuracy is about 60%, 3 components was checked as this can be helpful for visualisation, however it had a lower accuracy of only 46%. The PCA accuracy has been presented in 9b.



(a) Accuracy per number of clusters



(b) Accuracy per number of Primary Components

K-means is finally run using 19 clusters and a PCA reduced dataset of 100 components and achieves an overall accuracy of 60%. Analysis of the results shows that classes 0 and 1 have often been classified in the same cluster. These classes correspond to 60 and 80 kph signs, respectively - so poor separation between classification for these classes might have been expected. Classes 2, 7 and 9 were generally not well classified - this is in part down to the fact there were many fewer examples of classes 2 and 7 in the original data set.

Another clustering algorithm, Gaussian Mixture Model (GMM), was applied to the data set. GMM can cluster the data by hard clustering, like K-means, or by soft clustering. Hard clustering definitively ascribes instances to clusters whereas soft clustering gives instance a probably density of their likelihood of belonging to a certain class.

The hard clustering results for the GMM algorithm are similar to those found in K-means with better accuracy - GMM achieved roughly 75% . Classes 0 and 1 are, again, often clustered together. Class 9 was never given its own distinct cluster.

A number of texts were referenced to develop the clustering code [5], [7].

7 Research Question

The research question chosen relates with the Naive Bayes' classification and the feature selection. The question arisen was how to calculate the maximum and optimal number of features necessary for this classification. As it is explained in the lecture notes [1] and we already have mentioned in the previous tasks, using the 'correct' amount of data will improve the accuracy of the classification, whereas if too much data is used, the accuracy drops and the same as using not as sufficient data. There is a peak of "optimal" data which should be used. How to select this data and how much of it is what we are trying to answer here.

For this study we use the best feature selection method and the Naive Bayes' classification and the sliced data for our calculations.

7.1 Methodology

With the following code:

```
if __name__ == '__main__':
sliced = data_operations.load_dataframe(constants.NORMALIZED_SLICED_SMPL)
classes = data_operations.load_dataframe(constants.ORIGINAL_CLASSES)
classes = data_operations.randomize_data(classes, constants.SEED)
max_accuracy = 0.0
current_accuracy = 0.0
accuracy_list = []
feat = 1
for n in range(len(sliced.columns)):
    features_dataframe = select_features(sliced,feat)
    features_dataframe = data_operations.randomize_data(features_dataframe,
        constants.SEED)
    current_accuracy = naive_bayes(features_dataframe, classes)
    accuracy_list.append(current_accuracy)
    if max_accuracy < current_accuracy:
        max_accuracy = current_accuracy
    feat +=1

pyplot.plot(range(len(accuracy_list)),accuracy_list)
pyplot.title('Accuracy vs n Features')
pyplot.xlabel('No of features')
pyplot.ylabel('Accuracy')
pyplot.show()
```

We create a simple algorithm which finds how many necessary "best features" are needed to reach the peak of accuracy. The algorithm first selects the best feature of the data and carries out the Naive Bayes classification. After this, it loops for a search of the 2 best features and continues with the NB classification again, comparing with the previous accuracy obtained and if this latter one is bigger, it will update our result. The search will stop when the new accuracy obtained is lower than the previous one. This way, we can easily obtain the answer for how many features given the data and classification method will be necessary with the best accuracy, and hence, improve the optimization of our classification.

We used the sliced data (785 features) for our experiments so the processing time would be faster.

7.2 Results

As a result of our experiment we obtain that the maximum accuracy is obtained when we select the best **6 features** and the value of the accuracy is 0.9993. The accuracy drops when we reach the 700 features. Our experiment show that there is stable accuracy from the feature best 6 till the 700. Using only the 6 best features will then optimize our experiments and reduce the processing time in our calculations.

We show the plot for the drop in accuracy. The y-axis is the accuracy % and the x-axis shows the feature. In this graph, as it is only the drop in accuracy we have made a cut and it is showing the last features (hence, when it shows 35 it is referencing to 785 pixel/feature)

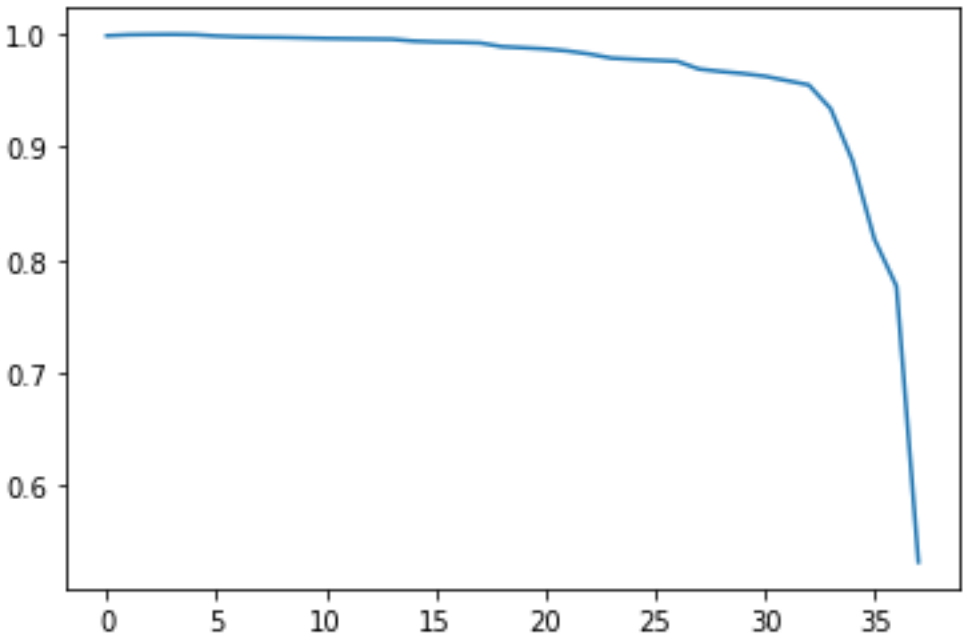


Figure 10: Drop in accuracy after 700 features

8 References

[1] Diana Bental, Ekaterina Komendantskaya, and David Corne. *Lecture Notes F21DL*. Nov. 2019.

[2] Jie Cheng and Russell Greiner. "Comparing Bayesian network classifiers". In: *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 1999, pp. 101–108.

[3] *A Bayesian method for constructing Bayesian belief networks from databases*. 1990, pp. 86–94.

[4] Sampath Kumar Gajawada. *Chi-Square Test for Feature Selection in Machine learning*. Oct. 2019. URL: `https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223`.

[5] A. Geron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017. ISBN: 9781491962244. URL: `https://books.google.co.uk/books?id=bRpYDgAAQBAJ`.

[6] Sruthi Nair and Aneesh R P. "Recognition of Speed Limit from Traffic Signs Using Naive Bayes Classifier". In: Dec. 2018, pp. 1–6. DOI: `10.1109/ICCSDET.2018.8821142`.

[7] A.A. Patel. *Hands-On Unsupervised Learning Using Python: How to Build Applied Machine Learning Solutions from Unlabeled Data*. O'Reilly Media, 2019. ISBN: 9781492035619. URL: `https://books.google.co.uk/books?id=-SKJDwAAQBAJ`.

[8] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016, pp. 1135–1144.

[9] Ian H Witten et al. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.