Heriot Watt University

Data Mining and Machine Learning

Coursework One

Group 2: Report

Authors:

Gordon Rennie (H00155853), Samuel Haley (H00103552), Alexandros Mitrousis (H00307077), Andrew Nowlan (H00104104)

November 11, 2019

Contents

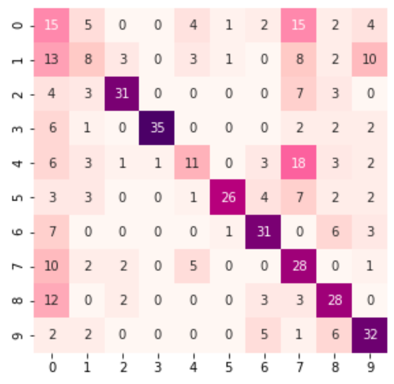
1 Introduction

For this coursework we chose to do our analysis using Python, Bayesian networks were done in Jython with the Weka API. We recognised that each class had different numbers of instances with some classes significantly less represented than others. This would have meant after training, models would be biased towards over represented classes (1). To fix this, we found the least represented class (class7 - 240 images) and we took a random sample of the same number of instances for each class to create a balanced dataset of 2400 instances. We also used random.seed() function to save the state of random function, so that we can keep the same values after multiple executions. Finally, we split the sample to train and test with an 80-20 balance. All code can be found in GitHub following: `https://github.com/anowlan123/DMMLGroup2`

2 Naïve Bayes

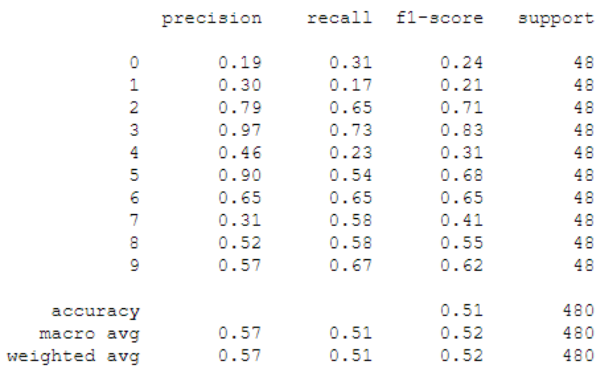
2.1 Full Data Set Q4

We selected a Multinomial Naïve Bayes model. This selection was due to the non-normally distributed data. Multinomial models uses discrete counts which suits non-normally distributed data the best(2).
After the training and testing of the model the following results of successful and mistaken classification in each class were found (X-axis=predicted class, Y-axis=Actual class):



	0	1	2	3	4	5	6	7	8	9
0	15	5	0	0	4	1	2	15	2	4
1	13	8	3	0	3	1	0	8	2	10
2	4	3	31	0	0	0	0	7	3	0
3	6	1	0	35	0	0	0	2	2	2
4	6	3	1	1	11	0	3	18	3	2
5	3	3	0	0	1	26	4	7	2	2
6	7	0	0	0	0	1	31	0	6	3
7	10	2	2	0	5	0	0	28	0	1
8	12	0	2	0	0	0	3	3	28	0
9	2	2	0	0	0	0	5	1	6	32

Figure 1: Heat Map



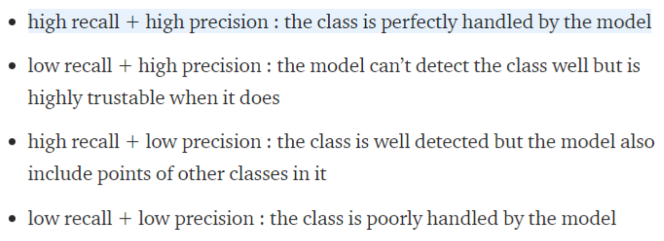
```
              precision    recall  f1-score   support

           0       0.19      0.31      0.24        48
           1       0.30      0.17      0.21        48
           2       0.79      0.65      0.71        48
           3       0.97      0.73      0.83        48
           4       0.46      0.23      0.31        48
           5       0.90      0.54      0.68        48
           6       0.65      0.65      0.65        48
           7       0.31      0.58      0.41        48
           8       0.52      0.58      0.55        48
           9       0.57      0.67      0.62        48

    accuracy                           0.51       480
   macro avg       0.57      0.51      0.52       480
weighted avg       0.57      0.51      0.52       480
```

Figure 2: Confusion Matrix

As can be seen from the heat map the results were mixed. The average accuracy of the model taken from 10-fold cross-validation was 51%. Figure:3 below shows how the model handles the classes depending on recall and precision.



- high recall + high precision : the class is perfectly handled by the model
- low recall + high precision : the model can't detect the class well but is highly trustable when it does
- high recall + low precision : the class is well detected but the model also include points of other classes in it
- low recall + low precision : the class is poorly handled by the model

Figure 3: (3)

Where: Recall is defined as the number of true positives divided by the total number of elements that actually belong to the positive class, Precision is the number of true positives divided by the total number of elements labeled as belonging to the positive class and the F1 score of a class is given by the harmonic mean of precision and recall.

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{1}$$

The best results were by far in class 3 as we expected because this sign is unique. The classes with the worst results were 0, 1, and 4. We expected 0 and 1 to be confused because they are similar. The bad results from sign 4 being mistaken for sign 7 were quite unexpected, however, upon further examination of these two signs, despite the difference in shape the colour contrasts are quite similar, with large swathes of white, outlined by gray with gray through the center. An interesting fact is that in classes 8 and 9 even though they are the same sign mirrored, only images from class 9 were confused with class 8 but never the opposite. Furthermore, we can see that the triangle sign images from classes 3 and 5 hardly ever predicted in other classes, probably because of their unique shape. Finally, we can see that class 0 and 1 have by far the worst precision and that means that most of the signs which were confused were classified as being of this class. In conclusion, we can see that this classifier did not work very well, probably due to the noise in the images.

2.2 Top Features Q6

After running the Naïve Bayes on the whole data set, we identified the top 2, 5, and 10 correlating features for each class. This was done using the SKLearn function SelectKBest (4). A F_regression method was used to select these top features. This involves computing the correlation between each attribute and the class.
After finding the top features for each class, new data frames were created with 20, 50 and 95 attributes respectively. The last data frame was expected to have 100 attributes. The difference was caused by the duplication of top attributes across classes. It was decided not to add additional attributes as technically the top 10 attributes for all classes were still represented in the final data frame and to add more attributes would favour certain classes without justification. Figure: 4 displays the top 2, 5, and 10 pixels plotted on images representing each class. Top 2 are coloured yellow, top 5 blue, and top 10 red.

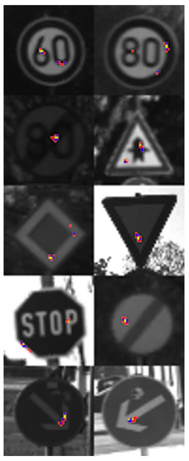


Figure 4: Class images with top attributes

Naïve Bayes classification was then run on these data frames. The heat maps below show the successful and mistaken classification in each case.
As can be seen from the heat maps all three conditions resulted in far better classifications than when the full data set was used. A 10-fold cross validation was used to discover the average accuracy of the models using the top features. Top 2 attributes resulted in an average of 68% where Top 5 and Top 10 averaged 71%.
The first conclusion that can be drawn from this is that the full data set contains noise, the removal of this noise leads to better classification. This makes sense from a conceptual level. All the images in the data set had the sign with a background. As the background is random it creates noise in the data set.

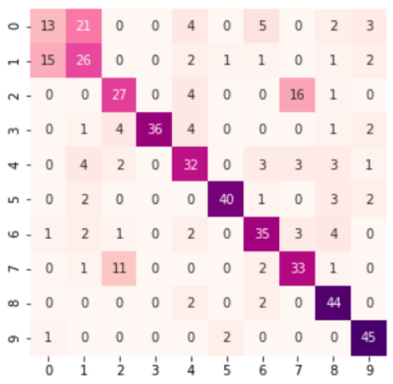


Figure 5: Top 2 Attributes

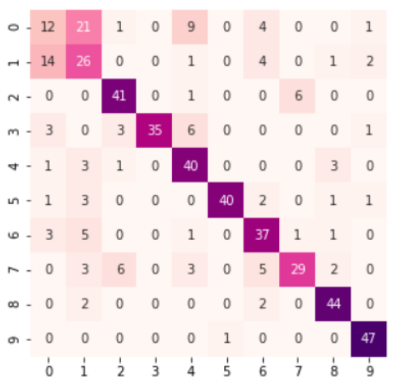


Figure 6: Top 5 Attributes

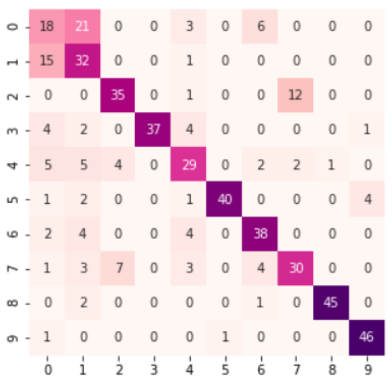


Figure 7: Top 10 Attributes

Examining where the top attributes are it's clear that for the more unique classes (3, 4, 5, 6, 8, 9) the top correlating attributes exist in areas of the image where they are unique, for example the edge of a Octagon (class 7).
An especially interesting point is the difference in accuracy in signs which share a 'pair'. Classes 0 and 1, 2 and 7, and 8 and 9 could all be paired due to their similar content and silhouette. The first two pairs are both commonly confused for each other as can be seen in the heat maps. Whereas classes 8 and 9 are never confused with each other.
The reason for this is obvious when Figure: 4 is considered. Both 8 and 9 have highly unique attributes at the tip of their arrows. The diagonals are uniquely positioned when considered against other signs which leads to the high accuracy. This accuracy stands in contrast to the accuracy when the whole data set is used because when the whole data set is used, the common features across the pairings are included (such as the circular silhouette) resulting in confusion in the model.

3 Complex Bayes Q8

Naïve Bayes assumes all attributes are independent. In Bayesian networks, the attributes are no longer considered independent, and are dependent on a subset of attributes (the parents). A set of conditional probability tables are produced for each attribute given its parents. Due to the limitations of the current Python libraries, the Weka API in Jython was used to carry out the Bayesian network analysis.

To assess the usefulness of a Bayesian network, the data sets generated in the previous section (top 2, 5 and 10 attributes for each class) were re-analysed, the datasets were exported from Python and converted to "arrf" file format for use with the Weka API in Jython. For each of these datasets various search algorithms were used to model the data: K2, TAN and HillClimber (5).

K2: Uses a hill climbing algorithm, using the given order of the attributes (6).

TAN: Determines the maximum weight spanning tree and returns a Naïve Bayes network augmented with a tree (7).

HillClimber: Uses a hill climbing algorithm. The search is not restricted by the order of the attributes (8) (unlike K2).

For both K2 and HillClimber the number of parents was adjusted and the effect recorded. To assess the Bayes model, and ensure comparability with the Naïve Bayes results, the data sets were split into the same 80/20 split train/test as in previous section.

Data set	**Sorting algorithm**	**F score – Number of parents:**			
		1	2	3	4
Top 10	K2	0.523	0.728	0.758	0.758
	TAN	0.701			
	HillClimber	0.523	0.681	0.667	0.667
	Naïve Bayes	0.74			
Top 5	K2	0.536	0.715	0.752	0.752
	TAN	0.706			
	HillClimber	0.536	0.719	0.716	0.720
	Naïve Bayes	0.73			
Top 2	K2	0.490	0.700	0.696	0.696
	TAN	0.660			
	HillClimber	0.490	0.643	0.641	0.641
	Naïve Bayes	0.68			

Table 1: F scores for Bayesian Networks

The K2 and HillClimber algorithms produce similar results, with K2 consistently scoring higher. It can be seen that for both K2 and HillClimber increasing the number of parents beyond 3 has little effect. Inspecting the networks that these algorithms produce shows that no attribute is ever found to have more than three parents, hence increasing the number of parents to four has no effect. One difference between the results of K2 and HillClimber is that hill climber tends to produce "deeper" networks with only a few attributes having the Class as a parent, for K2 all attributes always have Class as a parent.
The TAN algorithm effectively uses two parents and the F scores are in line with the scores obtained from the other algorithms with two parents.
Comparing the complex Bayes to Naïve Bayes results, shows that there is little, if any, increase in the F score using Bayesian networks. The Bayesian networks and the Naïve Bayes models struggled to classify the same classes, for example class 0 and 1 are often confused for each other. The results for a one parent Bayesian network is substantially worse than the results obtained by a Naïve Bayes, this is most likely due to the use of the Multinomial Naïve Bayes model suiting the distribution of the data better.

4 Clustering

4.1 kMeans Clustering Q10 & 11

We began exploring clustering by executing the KMeans(9) algorithm on the full unaltered dataset, first excluding the class attribute. With a boolean mask, we compared the cluster assignments to the true class labels and obtained a poor accuracy score with most instances being incorrectly classified. Over half of the instances were assigned to cluster one with the remaining instances being distributed across the other clusters, with some clusters remaining empty.

The KMeans algorithm is only guaranteed to converge to a local and not global minimum, so we ran an experiment to alter the random number generation for the centroid initialization and checked for significant changes in the results. Altering the random seed between 1 and 10, we observed a relatively small differences between the max and minimum accuracy - approximately 2% . While this experiment could have been extended with many more random seed values, it wasn't practicable to do so, due to the size of the data set and the time taken to execute the algorithm for each seed.

Repeating the above experiment, this time including the class attribute yielded indistinguishable results. The lack of change can be attributed to the fact that the 2304 pixel attribute take values an order of magnitude larger than the class attribute. This behavior emerges from the E step in the KMeans algorithm. The value of X_n in Equation:2 is used to assign instances to clusters. That is, the instance is assigned to a particular cluster, say A, if X_A is minimal for all X_n. This means the contributions of the class attribute term would be negligible in comparison to 2304 pixel attributes.

$$X_n = \sum\nolimits_{j=1}^{n} (pval(i, X_j) - val(e, X_j))^2 \tag{2}$$

To show this by means of example, let C_{nj} denote the centriod value of cluster n for attribute j and suppose we are calculating X_n for cluster A, where the class attribute has index j. Then we have

$$X_A = \underbrace{(C_{A1} - X_1)^2 + \cdots + (C_{A(j-1)} - X_{j-1})^2}_{\text{Very Large}} + \underbrace{(C_{Aj} - X_j)^2}_{\text{Trivialy Small}}$$

Here, it is clear that even one pixel's error term could be significantly larger than the class attribute error term. E.g if $C_{A1} = 100$ and $X_1 = 247$ then this attribute would contribute 21609 to the X_A term while the maximum error that can be produced by the class attribute is $(0 - 9)^2 = 81$.

To verify our mathematical intuition, we normalized the pixel values to range between 1 and 0 across the whole dataset and increased the class attribute values by a factor of 100. Repeating the the experiment on the modified data set yielded a 100% accuracy. This demonstrates the converse scenario where the class attribute contribution to the X_n is significantly larger than that of the pixel attributes combined.

4.2 More Clustering Q12 & 13

4.2.1 Pre-Processing

With the aim of improving the clustering results, we decided to eliminate attributes that contribute least to the classification. We first used the OpenCV's resize(10) method to reduce the number of attribute without loosing a significant amount of information. Through trial and error, we observed that reducing the images size to 24 × 24 did not significantly impact the results produces in previous experiments. With an already significant reduction in attributes, we used SelectKBest as done previously in Section(2.2) to extract the best 100 attributes. As shown in (11), applying the t-distributed stochastic neighbor embedding (t-SNE) algorithm(12) to a dataset of images can both improve classification results and provide a means of visualising the data. This is possible since t-SNE can preserve points within clusters whilst reducing the dimension of the data (11). This reduction in number of attributes will inevitably lead to improved algorithm execution time when exploring the dataset, particularly for KMeans.

4.2.2 Optimal K

The Elbow method is widely used to calculate the optimal number of clusters in situations when the number of classes are unknown. "*It calculates the Within-Cluster-Sum of Squared Errors for different values of k, and chooses k for which this error term first starts to diminish*"(13). This can be achieved by plotting a the WWS vs K, inspecting the graph and recording the value of k for which "elbow" or turning point is observed. Applying this method to our dataset, we obtained an approximate value of 8 for k - later experiments verified that instances were only assigned to 8 classes when running KMeans.

4.2.3 Comparison of Clustering Algorithms

Running KMeans on the pre-processed data returned a significant improvement in accuracy when compared to runs on the initial data. The f-score for the unaltered dataset was 0.27 while three-attribute dataset returned as much higher score at 0.63. Since the data can now be visualised in three dimensions, we plotted the ground truth, shown in Figure: 8, against the KMeans clustering assignments, shown in Figure: 9. By comparing these figures, we observe that KMeans returns a relatively good model of the ground truth, but struggles to represent smaller and sparsely distributed classes. This can be attributed to the fact that kMeans is hard clustering algorithm that performs best when the data is hyper spherical(14), and as we can see fro Figure: 8, this not true of our data.

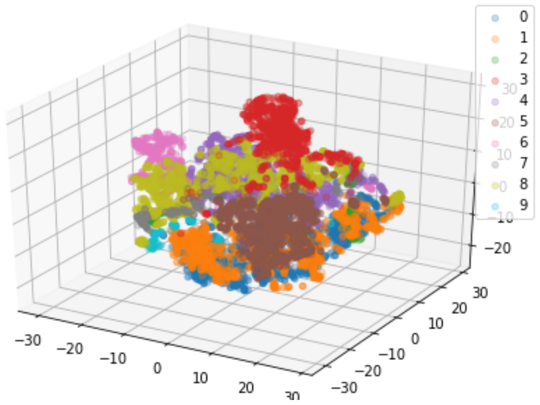


Figure 8: Ground Truth

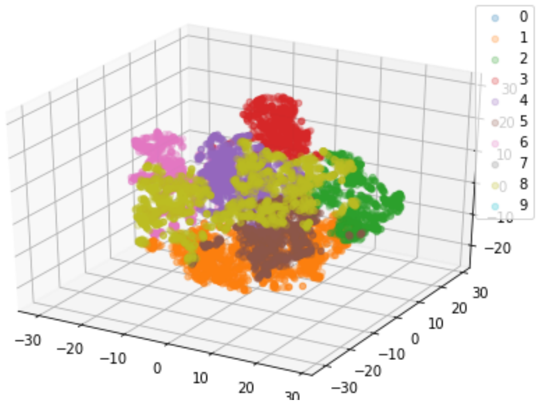


Figure 9: KMeans cluster assignment

To determine if a soft clustering algorithm would perform better, we ran expectation-maximization using a GaussinaMixtureModel(15). The result returned an f-score of 0.64, marginally higher than KMeans. Since the algorithm combines aspects of KMeans with Bayesian learning, it assigns probabilities to instances belonging to a certain cluster. As such. this model is better equipped to deal with data that is not hyper spherical, though in our case, the improvements were modest.

Finally we ran hierarchical clustering using Agglomerative Clustering(16) which achieved f-score of 0.62, only marginally worse than the previous algorithms. This type of clustering uses a bottom up approach where each instance begins a cluster, then through out execution, individual clusters are merged to form larger clusters. This approach presents some benefits over kMeans. Firstly, results are reproducible in Hierarchical clustering where KMeans starts with random choice of clusters with different results being produced each time.KMeans often requires prior knowledge of k requires prior knowledge of k, where hierarchical clustering can stop an appropriate number of clusters by interpreting its dendrogram(14).

With the Bayesian learning we, we also saw a significant increase from f-score of 0.37 to 0.66 and on Naïve Bayes and achieved an f-score of 0.91 utilising complex Bayes Nets, both using the reduced dataset. Comparing these to our clustering algorithms, Naïve Bayes provided results not too dissimilar from our EM analyses, however complex Bayes Nets provided a far superior result overall.

5 Research Question

5.1 Research Direction

Due to the nature of the problem posed the goal of our research was to improve the accuracy of our models. Based on the better classification accuracy seen when top attributes were selected, we decided that discovering the 'perfect' number of attributes would be a worthy research direction. We discussed attempting to increase the number of instances in each class as we currently cut out around 75% of our data pool. We discovered a method of oversampling by rotating other classes to give images of the target class (17), such as flipping class 9 about the y axis to give more instances of class 8. We decided to not use this method as when the data set was examined the classes with the lowest number of instances - class 7 - could be doubled in size to 480 but that was then the limit of this oversampling method.

5.2 Testing

We modified our code used in Question 6 to select x number of top correlating features, find the accuracy of a model built using those x attributes and 10-fold cross validation, and then plot the accuracy against the number of features. This was initially done from 1 to 1000 top attributes resulting in Figure: 10.

5.3 Results

As can be seen using only 1 feature results in an accuracy of around 65%. This peaks at around 80% using 100 features and then gradually tails off until it reaches 55%. This process was repeated from 1-200 features (Figure:11) and 1-100 features (Figure:12) which shows in higher definition the same trend.

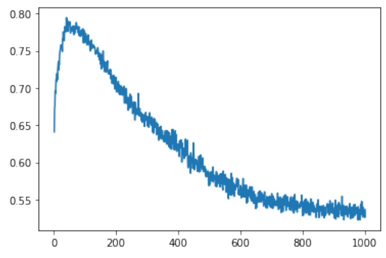


Figure 10: Top 1-1000 Attributes

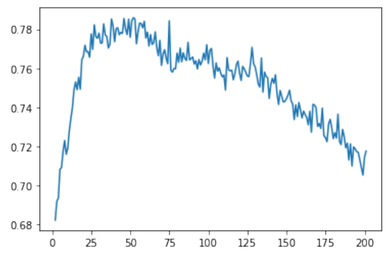


Figure 11: Top 1-200 Attributes

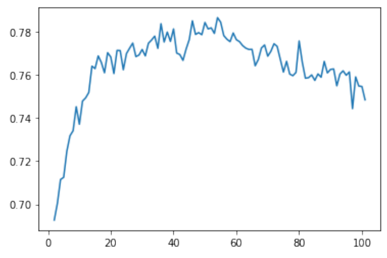


Figure 12: Top 1-100 Attributes

The highest accuracy varied from 78-81% with the optimum number of attributes ranging from 40-60.

5.4 Discussion

It is obvious from the above results that there is an optimum number of attributes for the given data set and that number is around 50 attributes. Finding the turn optimum number depends on the training set used. As done in section: 2.2 we then plotted the top attributes over their classes to discover what features were most important. As expected the areas that are unique between classes are by far the most important attributes.

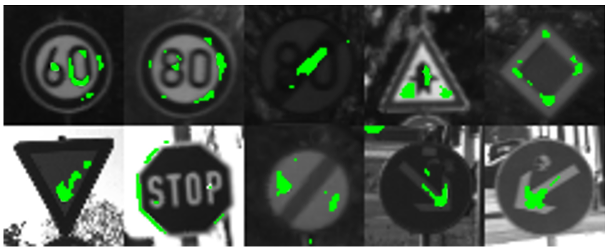


Figure 13: Class images with top 60 attributes

6 Conclusion

Classification of street signs is a difficult machine learning issue. This is due to the commonality of features in road signs such as their base shape. This point is especially true when considering which signs are most commonly confused. Obeying the correct speed limit is vital but all of our models often mistook the 60kph and 80kph classes. This leads to perhaps the most important conclusion which is that automated cars should not base their understanding of vital information purely off of road sign recognition. One option to overcome this is the creation of unique road signs with the specific purpose of being read by computers. At present we are attempting to read signs fit for human recognition. It seems creating signs tailored to machine recognition would be a safe solution. This approach would mimic placing AR Markers around a nation's roads for automated cars to read.

7 GitHub Link

https://github.com/anowlan123/DMMLGroup2

8 Contributions

We feel that everyone in the group contributed equally.

References

[1] D. S. Longadge, R., "Class imbalance problem in data mining review," *CoRR*, vol. 12, p. abs/1305.1707, 2013.

[2] S. Ray. (2017) 6 easy steps to learn naive bayes algorithm with codes in python and r. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

[3] B. Rocca. (2019) Handling imbalanced datasets in machine learning. [Online]. Available: https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28

[4] Scikit-learn. (2011) Selectkbest. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html,

[5] R. Bouckaert, "Bayesian network classifiers in weka," 2007.

[6] ——. (2019) Weka classifiers k2. [Online]. Available: http://weka.sourceforge.net/doc.stable/weka/classifiers/bayes/net/search/local/K2.html

[7] ——. (2019) Weka classifiers tan. [Online]. Available: http://weka.sourceforge.net/doc.stable/weka/classifiers/bayes/net/search/local/TAN.html

[8] ——. (2019) Weka classifiers hillclimber. [Online]. Available: http://weka.sourceforge.net/doc.stable/weka/classifiers/bayes/net/search/local/HillClimber.html

[9] Scikit-Learn. (2011) Kmeans. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

[10] OpenCV. (2019) Resize method. https://docs.opencv.org/trunk/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d

[11] PythonDataScienceHandbook. (2019) In Depth: KMeans. https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html

[12] Sklearn.manifold.TSNE. (2011) t-sne. https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.

[13] Medium. (2019) How to Determine the Optimal K for K-Means? https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb

[14] A. Vidhya. (2019) An introduction to clustering and different methods of clustering. https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

[15] Scikit-Learn. (2011) Gaussian mixture models. https://scikit-learn.org/stable/modules/mixture.html 2019-04-11.

[16] SK-Learn. (2011) Agglomerativeclustering. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.

[17] M. Pechyonkin. (2017) Traffic sign classification. [Online]. Available: https://pechyonkin.me/portfolio/traffic-signs-classification/

Acknowledgements

Equations and notations were borrowed from E.Komendantskaya's Data Mining and Machine Learning Course delivered at Heriot Watt University in 2019. We would also like to thank Daniel Kienitz for his informative python guest lectures which assisted the programming aspect of our project greatly.