Data Mining and Machine Learning Coursework Jamie McCulloch H00189648

https://drive.google.com/open?id=16WJi8FYXqDDTzcU_z9D7xKt16RarfAo4 (Link to files)

1. The csv files downloaded from Vision can be loaded into weka, where there is a tool called "arff viewer" found in the 'tools' menu of the weka explorer. This converts csv files into arff format to be saved. There is a folder of all the arff files attached on Google Drive.
To split the pixels into individual columns I used Microsoft Excel, and went into data→ text to columns. Then to choose the delimited option. Then I delimited by space, which split up the pixels into individual columns, and thus different attributes. Once loaded into weka this was 2305 attributes, with 35888 instances.
2. To randomise the instances within fer2017, I applied the Randomise filter within weka, to shuffle the instances around.
Settings used:
weka.filters.unsupervised.instances.randomize
3. To use the Bayes Net algorithm on this dataset, it first had to be filtered. To begin with, I used the NumericalToNominal filter found in weka.filters.unsupervised.attribute.NumericToNominal to change the class attribute 'emotion' from numeric to nominal. This then gave the class attribute a noticeable distribution and it helped understand the data.
I then moved the emotion attribute from the first column of the csv file to the last column, as weka only recognises class attributes if they are the last attribute in the file.
4. By changing both of these, I was able to run the naive bayes algorithm. Below is the output from this with the settings for the algorithm being :
With the accuracy being 21.6318% correctly classified, 7763 out of 35887 instances. Individual accuracies are given in the table below:

Class	0-Angry	1-Disgust	2-Fear	3-Happy	4-Sad	5-Surprise	6-Neutral
Accuracy (%)	4.86	20.8	5.48	15.44	39.82	59.82	15.06

Accuracy found by dividing the amount of correctly classified instances divided by the amount of the specific class * 100.
From this we can see that class 5 has the highest accuracy, with ~60% and class 0 has the lowest ~5%. These values are found by taking the amount of correctly classified instances in the row, and dividing by the amount of that class in the full dataset. So for class 0 was calculated by doing 232/4953*100.
5.1. Using the RemovePercentage filter within weka, I removed 50% of the values in the dataset, resulting in 17943 instances instead of 35888. I then ran the NumberToNominal filter to ensure that emotion was nominal. I then loaded each dataset into weka and ran the naive bayes net algorithm on them and recorded the accuracy.
All of these were run on the training set. Below are the accuracies of each set:

Set	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Accuracy	60.6142	51.0784	56.7185	57.1365	59.8785	65.5409	56.2838

It is clear by comparing these two tables that the files that are strictly binary - 0 for no, 1 for yes - are much more accurately classified using the bayes net algorithm. This is partly due to the fact

https://drive.google.com/open?id=16WJi8FYXqDDTzcU_z9D7xKt16RarfAo4 (Link to files)

that there is only two choices for the *fer2017emotionX* files, rather than 7 in the original fer2017 file.

5.2. By using the CfsSubsetVal evaluator and BestFirst search method, I found the top 10 attributes in each emotion dataset. This was run with only half the instances, using the removepercentage filter, as otherwise the computation would have taken too long on my laptop. The results are given below:

	1	2	3	4	5	6	7	8	9	10
Happy	Pixel48	Pixel146	Pixel433	Pixel455	Pixel595	Pixel734	Pixel756	Pixel769	Pixel799	Pixel847
Sad	Pixel1	Pixel5	Pixel148	Pixel52	Pixel94	Pixel96	Pixel98	Pixel145	Pixel192	Pixel240
Neutral	Pixel1	Pixel5	Pixel9	Pixel10	Pixel11	Pixel23	Pixel24	Pixel25	Pixel33	Pixel35
Disgust	Pixel17	Pixel80	Pixel105	Pixel117	Pixel124	Pixel185	Pixel213	Pixel253	Pixel286	Pixel367
Surprise	Pixel1	Pixel47	Pixel48	Pixel49	Pixel96	Pixel99	Pixel192	Pixel261	Pixel429	Pixel476
Fear	Pixel1	Pixel40	Pixel115	Pixel118	Pixel224	Pixel278	Pixel343	Pixel480	Pixel511	Pixel540
Angry	Pixel57	Pixel59	Pixel151	Pixel536	Pixel674	Pixel712	Pixel749	Pixel785	Pixel789	Pixel798

These results, screenshots, and text files are all included in the appendix files of this report.

6. Taking the top 2 attributes in each emotion from the table above, I removed all other attributes (apart from the class attribute) resulting in 10 attributes (as there are 4 duplicates) and then ran the naive bayes algorithm on this dataset. I then repeated this for the top 5 attributes and the top 10 attributes respectively, the results are in a table below.There is also the standard fer2017 file there for comparison (this had 2305 attributes)

Original (2305)	**Top 2 Attributes (10)**	**Top 5 Attributes (30)**	**Top 10 Attributes (63)**
21.6318	22.7331	20.8605	19.0102

7. Using all of the test ran above, it is safe to assume that anger and fear were the hardest to classify when using the original fer2017 file. Both had ~5% correct classification rate. The easiest to categorise is surprise, as this had around a 60% correct classification rate.

As Pixel1 appeared in 4 of the 7 datafiles as the top correlated attribute, this can possibly be the most important attribute in defining what emotion an image belongs to.

The purpose of question 5/6 was to find out if there was a difference in accuracy of correctly classified instances if the top 2, 5, and 10 attributes were found for each emotion file, and then running the same naive bayes algorithm. This meant that the algorithm was not running over 2305 attributes, and was a lot quicker to run. As seen in figure 6, the accuracy stays consistent, ~20%, and actually increases when only 10 attributes are used. This shows just how many attributes are redundant in terms of the fer dataset, instead of 2305, we only need 10.

8. Using the K2 algorithm, and repeating task 4 described earlier using the bayes net algorithm instead.

To repeat task 5 with bayes net instead, and to get a smaller dataset, I used the RandomSubset filter on weka to remove half the attributes and then used the RemovePercentage filter to remove half the instances, so there was 1153 attributes and 17943 instances.

Settings used:

https://drive.google.com/open?id=16WJi8FYXqDDTzcU_z9D7xKt16RarfAo4 (Link to files)

weka.classifiers.bayes.BayesNet -D -Q weka.classifiers.bayes.net.search.local.K2 -- -P 1 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5

I then ran with 2 parents.

Settings used:

weka.classifiers.bayes.BayesNet -D -Q weka.classifiers.bayes.net.search.local.K2 -- -P 2-S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5

When running with 2 parents maximum, it took upwards of 2 hours until something was computed, whether this is due to my dataset or due to my laptop's processing power I am unsure but I could not run it for any more than 2. Due to this being on the original fer2017 file, there was no use in comparing this to earlier results as these were done on the individual emotion files.

For repeating task 6, I removed all but the top attributes (2, 5, and 10) same as before but ran the bayes net algorithm instead of the naive bayes.

All results for this are below in figure 7.

TAN Algorithm

Settings used: weka.classifiers.bayes.BayesNet -D -Q weka.classifiers.bayes.net.search.local.TAN -- -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5

Tree augmented naive bayes algorithm works by using a tree structure so that each attribute depends on class attribute and one other attribute. A maximum weighted spanning tree is used to classify the training data.Unfortunately as this is such a large, and heavily computational it proved very time consuming to get results for the below columns. This is very computationally heavy so I have decided to have < 100 attributes for each test.

Hillwalking

This is an iterative mathematical algorithm which finds a random solution then attempts to improve the solution by incrementally changing one thing at a time. Again, this is very computationally heavy so I have decided to remove all but 100 attributes for each test.

Settings used: weka.classifiers.bayes.BayesNet -D -Q weka.classifiers.bayes.net.search.local.HillClimber -- -P 1 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5

Dataset	**Naive Bayes (%)**	**K2 (%)**	**TAN (%)**	**Hill Climbing (%)**
Full Dataset	21.63	24.69	N/A	N/A
K2 Random subset (parent=1)	N/A	25.40	N/A	N/A
K2 Random subset (parent=2)	N/A	42.74	N/A	N/A

Random subset (94 attributes)	29.4	N/A	35.66	25.55
Top 2 attributes	22.73	24.54	29.64	24.54
Top 5 attributes	20.86	23.14	28.79	23.14
Top 10 attributes	19.01	22.10	29.70	22.10

Figure 7

As seen from this table, it can be deduced that for the most part, naive bayes test is less accurate in terms of classifying the fer2017 dataset. However, hill climbing is identical to the K2 algorithm in terms of the top 2/5/10 attribute datasets. The TAN algorithm is the best on average, however with multiple parents, the K2 algorithm is the best, but due to computational constraints I was unable to carry out more tests on this.

9. When running the k-means clustering algorithm on the training set it always returned two clusters, one of 51% and the other of 49% regardless of the dataset used.

Figure 8 belows shows the accuracy of the other datasets when k-means clustering is used.

Settings used:

weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Dataset	Class to Clustering
fer2017	21.96
ferangry	50.35
ferdisgust	51.03
ferfear	53.36
ferhappy	50.88
ferneutral	51.67
fersad	52.58
fersurprise	55.4

Figure 8