F20DL Data Mining & Machine Learning

Coursework 1

Cameron McBride (H00160548)

Jordan Walker (H00222194)

Christopher Walsh (H00122841)

Date: October 18, 2018

Link to all documents including scripts, data Files etc. can be found at:
https://github.com/cdw2/data_mining_cw1
*will be public after Submission Date.

Contributions

All group members involved contributed equally to this assignment.

1 Tasks 1, 2 & 3

1.1 Task 1

The first task was to convert the CSV files into the WEKA specific Attribute-Relation File Format (ARFF) [3]. To do this, we used Python to write a script "**csv_arff.py**" which accepts as input a CSV file.
The script skips the first line of the input file to ignore the header information. Then we use Python's built-in CSV library to parse one row at a time and split the values up into a list. We take the first value of the list and convert it to a string representing the emotion. We take the second value string and split it into spaces to create an array of pixel values. From this, we create a string representing each instance, formatted to suit the .arff file and add it to a list for output. We then write out the header information to the .arff file and for each item in the output list write it to a new line in the .arff file.

1.2 Task 2

For this part of the coursework, we decided that it would be better to create two different scripts for the randomisation of the data we are working with throughout the course these include;

1. **shuffle.py**
2. **makeR.py**

The **shuffle.py** program works on any .arff and is used to swap entries in the data of that particular file. By doing this, it results in no loss of any data; however, the position of each data entry changes which removes any bias. I.e. if you trained the model one after the other in the original file and the file which was created in such a manner that the properties of the predecessor data entry affected the following entry then this is the bias which will be removed by shuffle.py.

makeR.py works by making one new file from all the data entries from each fer2018EmotionX arff file combining them and creating a reliable dataset where there are equal amounts of entries from each emotion file. This is achieved by allowing the user to select the data size they want and checking that is a modulo of 7. This produces a robust dataset where there could perhaps be issues with more entries of one emotion than another which may affect the learning of the model. Using this with the shuffle in our opinion was the best practice for making good data samples that are randomised.

By shuffling and taking random samples of the data files, we can say that we serve the idea of reducing variance and making sure that the data model stays general and we reduce the risk of overfitting. By taking a fair amount of data entries from each EmotionX file, we can say that our training, testing and validation are more representative of the data-set as a whole.

1.3 Task 3

For our implementation we increased the "heap" size of *WEKA* to 4GB, allowing us to use the whole data set rather than just a smaller collection of the data entries. However, due to a large number of attributes for each image (2305), analysis of Nearest neighbour Algorithm performed on the dataset showed that it yielded an accuracy of around 62% correctly classified instances. To achieve higher accuracy, we decided to reduce the number of attributes for each image by a factor of 4. This reduction aimed to filter the data and reduce the noise of images.

To reduce the number of attributes, we took the average value of a quadrant of four pixels and stored this as a single value. This process was repeated for every quadrant in each image. This technique is synonymous with reducing the resolution of the images.

We implemented this by writing the Python script "**part3_attributeReduction.py**". The output file contained 577 attributes per instance instead of 2305. The script iterated over each instance in the input file, calculating the average value of quadrants of four pixels, before writing the new values to the output file.

2 Tasks 4, 5, 6 and 7

2.1 Task 4 - Classification: Performance of the Nearest Neighbour Algorithm on the given data set

After we had reduced our number of attributes we also normalised the pixel values using the *WEKA* filter. The motivation for this is that the pixel values can vary over a wide range (0-255) and the K-nearest neighbour algorithm works by looking at the nearness of other data points typically based on Euclidean distance. Therefore normalisation allows for more accurate classification by bringing the values into a smaller range (0-1).

During Task 3 our solution to the task allowed us to reduce the "noise" of the images. Thus because of this we did not need to add another *WEKA* filter as we essentially created our own implementation of a "noise reduction" filter.

angry	disgust	fear	happy	neutral	sad	surprise
3880	21	145	309	337	156	105
52	430	9	16	23	6	11
574	67	3591	257	259	195	178
882	164	767	6083	652	281	160
681	100	616	914	3573	183	131
811	116	739	996	549	2773	93
249	51	326	382	208	53	2733

Figure 1: Confusion Matrix

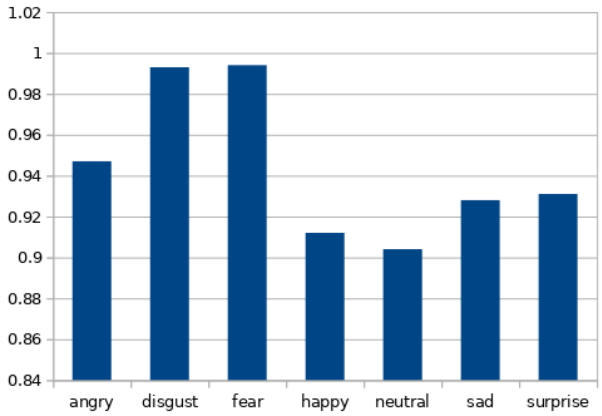

Figure 2: ROC Bargraph

Above you can see the results of our first run of the nearest neighbour with k =3.

2.2 Task 5 - Deeper analysis of the data

For this task we amended our "**csv_arff.py**" script to allow the conversion of the individual emotion files to only have the correct emotion name and "other". This allowed them to be loaded into *WEKA* and have correlation tests performed on them.

Using the *WEKA* select attribute tab with the Correlation Attribute Evaluator selected and the Ranker search method. Here are the values for the 10 most relevant attributes for each emotion file.

File	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
angry	344	351	345	368	320	396	327	321	275	350
disgust	14	12	13	11	15	38	63	39	62	40
fear	192	216	239	241	193	217	215	191	167	240
happy	468	467	492	491	469	493	516	490	515	466
neutral	5	50	44	26	2	20	6	4	27	31
sad	155	156	131	132	179	154	130	180	157	107
surprise	367	343	344	203	179	204	180	366	368	391

Figure 3: Task 5 : Top 10 pixel correlation values

File	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
angry	223	36	125	55	45	114	172	407	310	285
disgust	351	466	151	325	396	198	166	526	74	464
fear	540	471	517	512	539	538	518	464	442	177
happy	20	31	276	21	66	18	275	341	498	282
neutral	345	342	443	319	448	369	444	442	513	471
sad	117	141	68	51	93	27	74	4	98	44
surprise	444	443	494	470	445	514	489	540	517	442

Figure 4: Task 5 : Bottom 10 pixel correlation values

Note: Our filter reduced the pixel range from 48 x 48(2304) to 24 x 24 (576).

2.3 Task 6 - Improve the classification

For part 6 we wrote a script "**extract_pixels.py**", to extract specific pixel values from each file and assemble them into a whole dataset.

We used the data gathered in part5 to extract the pixels for the top 2, 5 and 10 class attributes. This resulted in datasets with 14, 35, and 70 pixel attributes.

We imported these into *WEKA*, normalised the pixel values and ran the IBK Nearest Neighbour classifier again with K set to 3.

2.4 Task 7 - Make Conclusions

Which emotions are harder to recognise?

The happy and neutral emotions were hardest to recognise.

Which emotions are most easily confused?

Angry happy and neutral were the most easily confused.

Which attributes (fields) are more reliable and which are less reliable in classification of emotions?

See figures 3 and 4 for the top 10 best and worst attributes for the classification of emotions.

What was the purpose of Tasks 5 and 6?

The purpose of tasks 5 and 6 was to perform attribute selection on our data. The reason behind attribute selection is to reduce the dimensionality of the data. This reduces the training time of the models and can increase accuracy by reducing noise in the data, reducing overfitting and avoiding spurious correlations.[2]

What would happen if the data sets you used in Tasks 4, 5 and 6 were not randomised?

The resulting classifier would be poorly trained as the data would be grouped by class, so you would not get a good representative sample of the data between the training and test data.

What would happen if there is cross-correlation between the non-class attributes?

If there was a cross correlation between each image there would be no way to classify which emotion is which. If we imagine that the "strongest" pixels were always the same or close then the classifier would experience a high error rate as each entry regardless of emotion would share the same values that make them distinguishable.

Other methods we could have tried?

Two possible methods appear when thinking about image classification:

1. Using another model
2. Using different data

When working with any data mining set it is apparent that there are many different models that could be used to extract the data you are looking for. For example in this project we could have implemented a Neural Network where we feed the network line by

line training data and then have that line "classed" as a certain emotion. Afterwards we could insert our fer2018 datafiles and achieve a different type of classifier for the images.

The second aspect we could have tested would be to use different data i.e. instead of pixel data we could have used ratio data where we feed the classifier attributes such as the width between eyebrows, the height between the left eye and the left lip etc.

In general practice we could assume these attributes may give similar or even greater results for classifying emotions from images.

Would there be more accuracy by using a Neural Network for this type of project?

In 2015, an experiment was performed to determine if combining a Neural Network and K- Nearest Neighbour models would produce better classification results. The experiment was run on ratios as attributes rather than pixel values as our own. However, during this experiment the Neural Network showed a better classification average of 0.95%. [1].

We can assume that had we implemented a Neural Network rather than K-Nearest Neighbour there would be an increase of accuracy in our classification.

3 Masters Question (For our own understanding)

3.1 The effects on Nearest Neighbour of varying numbers of neighbours

To test the classification on changing the K value of the Nearest Neighbour algorithm we ran tests on K set to one, two and the original testing requirement of three. As seen in figures 5 and 6, we observe that when the K value is changed we see a very dramatic decrease in the correctly classified instances and a sharp increase in the incorrectly classified images.

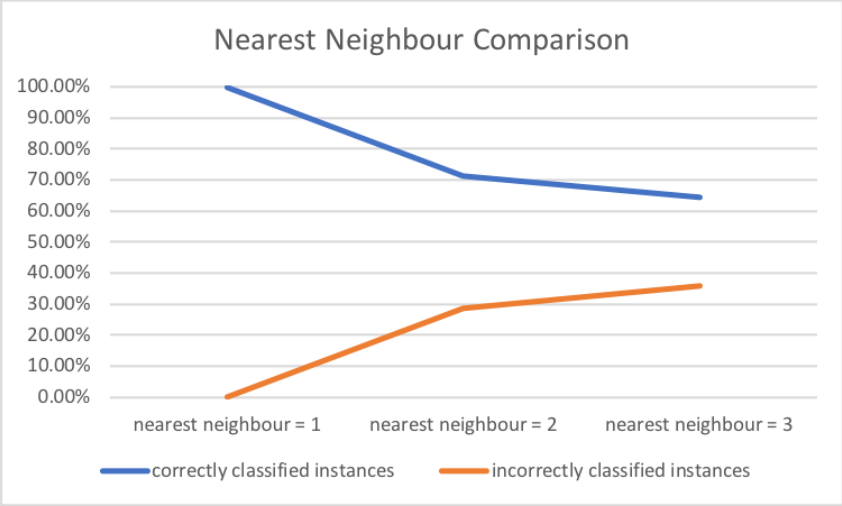

Figure 5: Masters: Nearest Neighbour Comparison

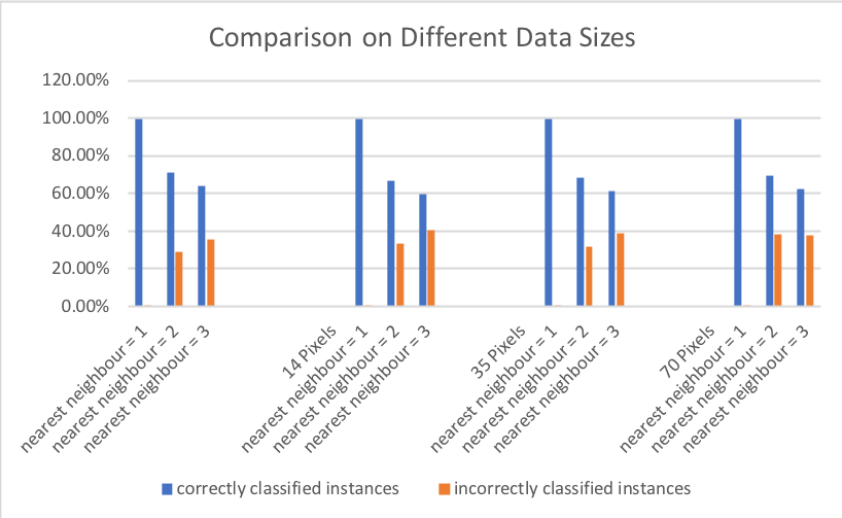

Figure 6: Comparison running IBK on different numbers of attributes.

We can establish that the data size doesn't matter. Because of the type of data (numeric) grouping nearest neighbour becomes a lot more difficult. Because of that the noise of the image itself is increased.

Hence the vast reduction in correctly classified instances and the increase in incorrectly classified instances.

References

[1] T S Hai. "Facial Expression Classification Using Artificial Neural Network and K - Nearest Neighbor". In: *MECS (* (2015). DOI: DOI:10.5815/ijitcs.2015.03.04. URL: http://www.mecs-press.org/ijitcs/ijitcs-v7-n3/IJITCS-V7-N3-4.pdf.

[2] R. Saian and K. R. Ku-Mahamud. "Comparison of Attribute Selection Methods for Web Texts Categorization". In: *2010 Second International Conference on Computer and Network Technology*. 2010, pp. 115–118. DOI: 10.1109/ICCNT.2010.13.

[3] University of Wakito. *Attribute-Relation File Format (ARFF)*. 2008. URL: https://www.cs.waikato.ac.nz/ml/weka/arff.html.

4 Appendices

4.1 Task 4 Screen-shot

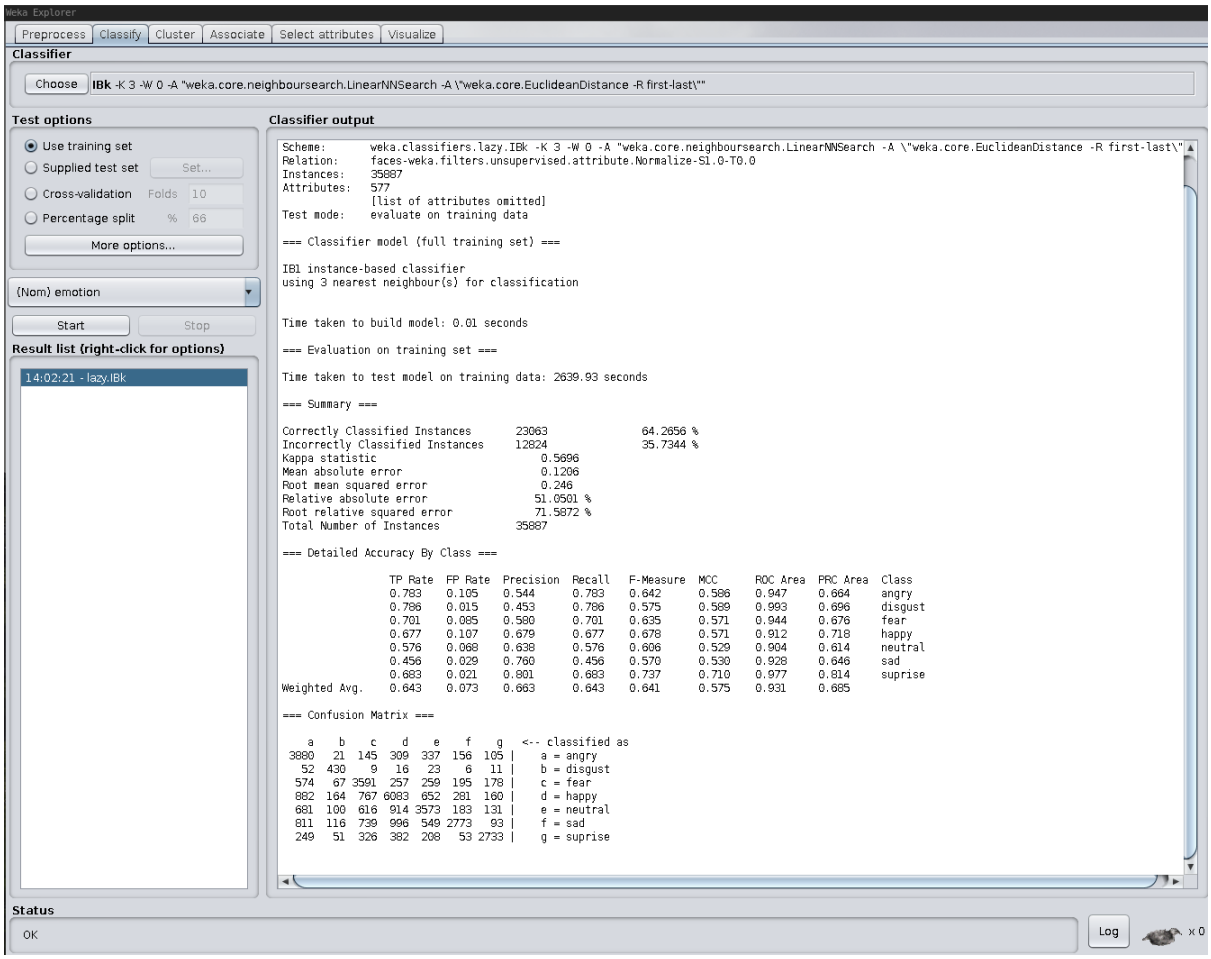

Figure 7: Task 4: Reduced Dataset, Normalised NN3

4.2 Task 6 Screen-shots

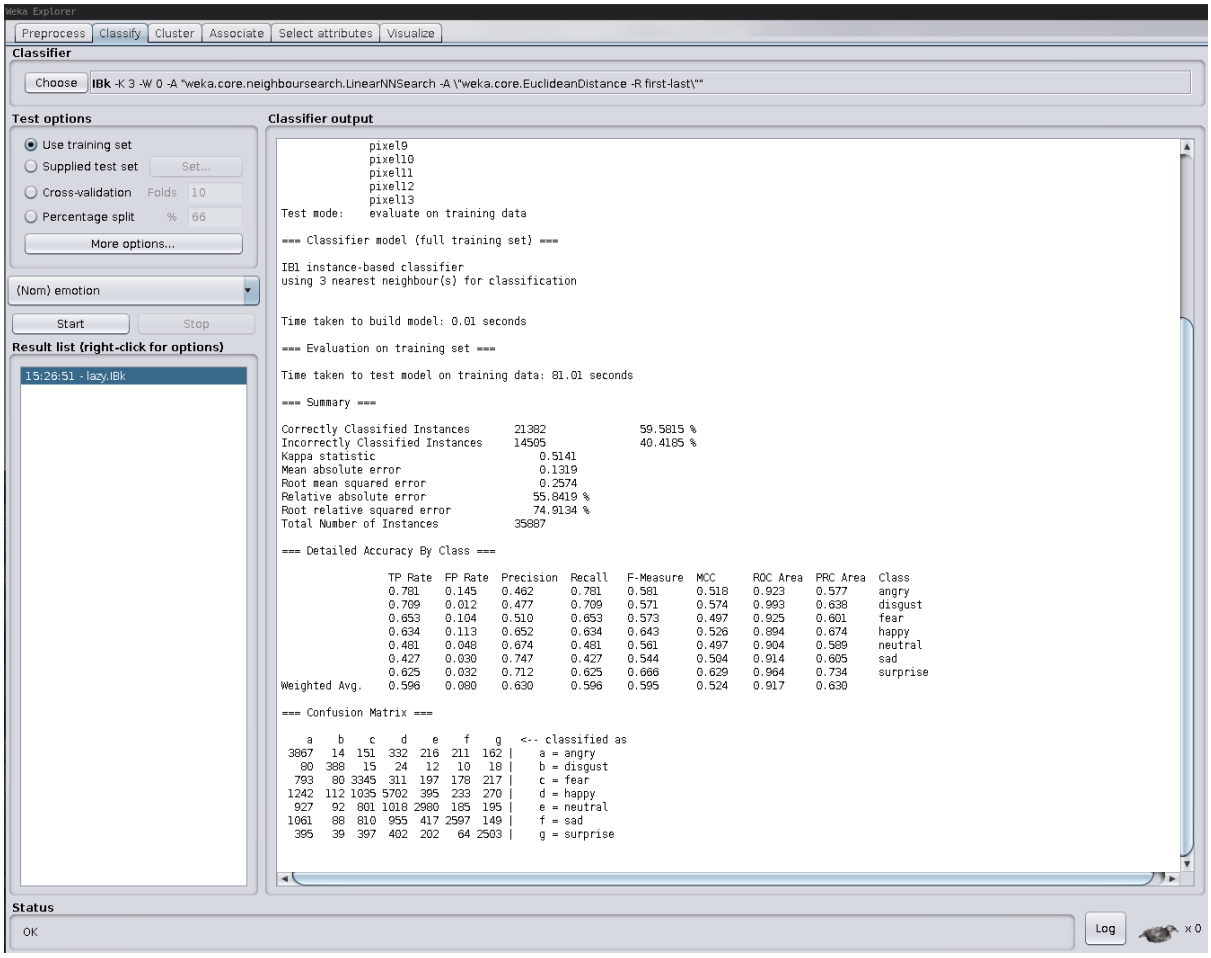

Figure 8: Task 6: 14 Pixels

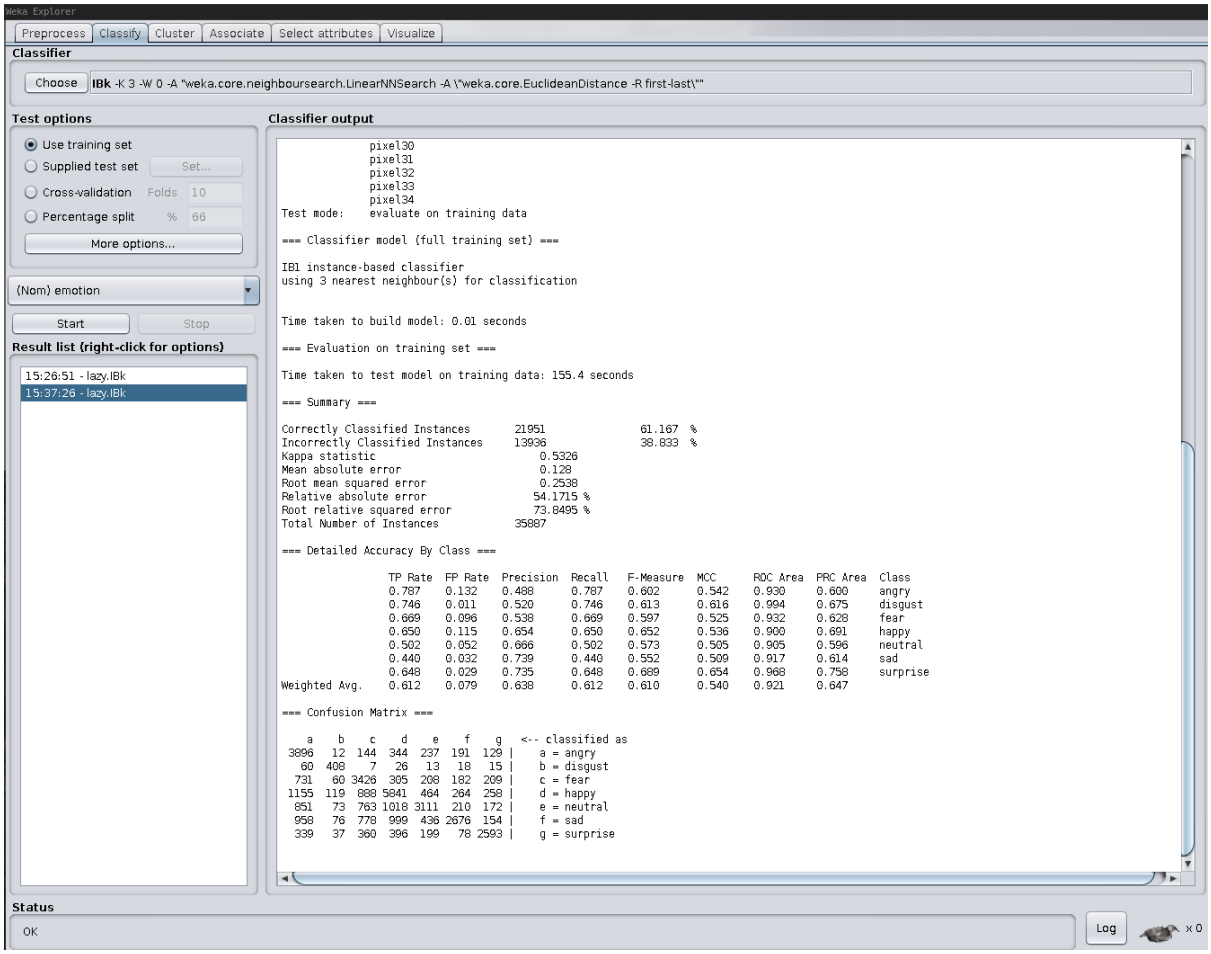

Figure 9: Task 6: 35 Pixels

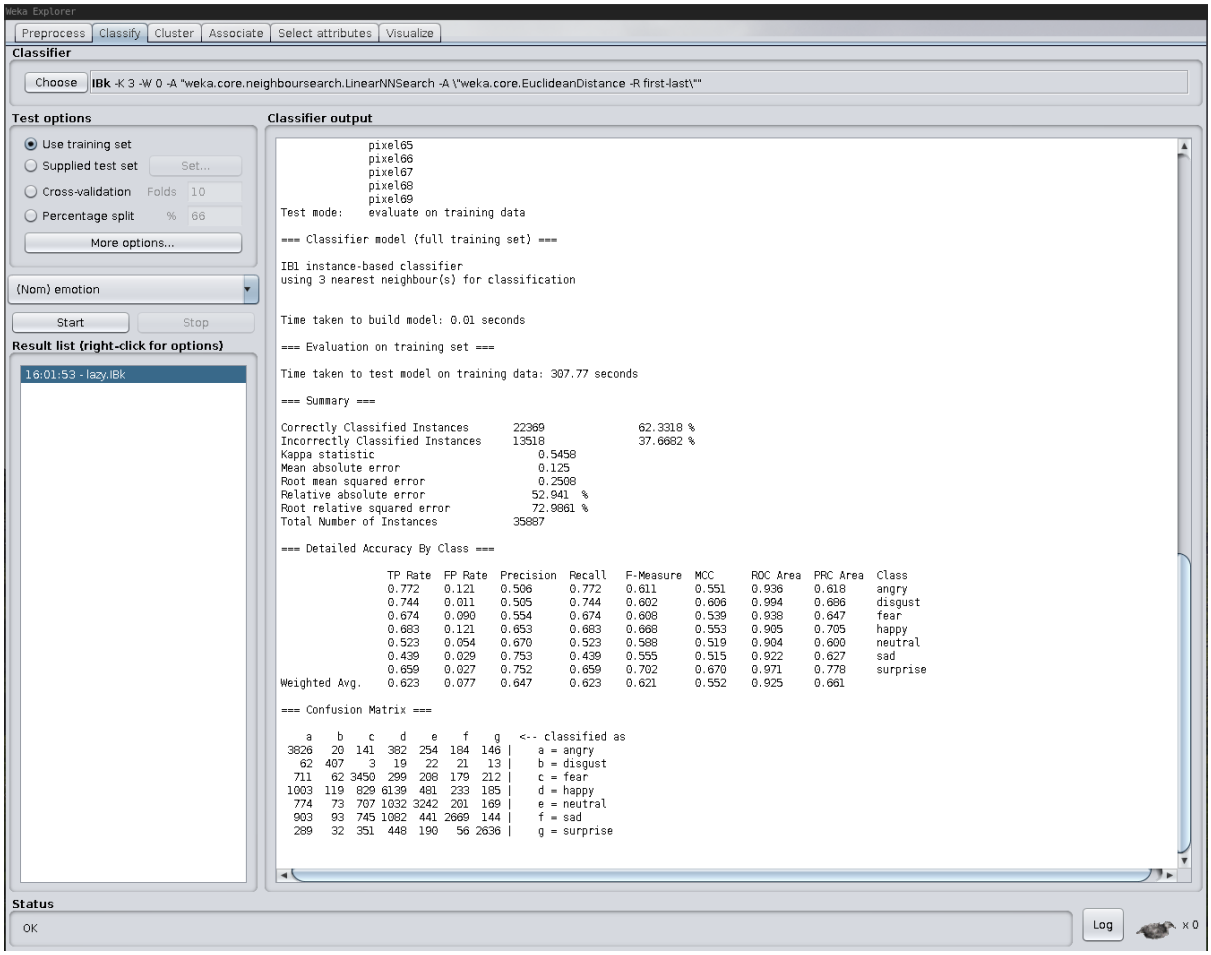

Figure 10: Task 6: 70 Pixels