Machine Learning Assignment - LLLL76

1 DISCUSSION/DETAILS OF APPROACHES CHOSEN AND EXPERIMENTAL PROCEDURE

1.1 kNN AND VARIATIONS

K nearest neighbours was implemented and tested first. The benefits of kNN is that is is very fast to train only needing the time taken to plot all train data. The disadvantages of kNN is the slow query time, this is due to slow look up in high dimensional space. There is only a single value that can be changed in basic kNN to alter its behaviour, the number of neighbours that are used, k. There are a few more advancements on kNN based around the weighting of the neighbours. The weighting cane be done based on the inverse square of the distance between the query and its neighbour or the similarity between the query and its neighbour, both of which have been implemented.

1.2 SVM

Support Vector Machines are based around the idea of a linear separator, that any two classes can be split by a line, and to find the best line there should be a maximum separating margin. The implementation of this approach has a significant number of variables, kernel, degree, C, gamma and the class weighting.

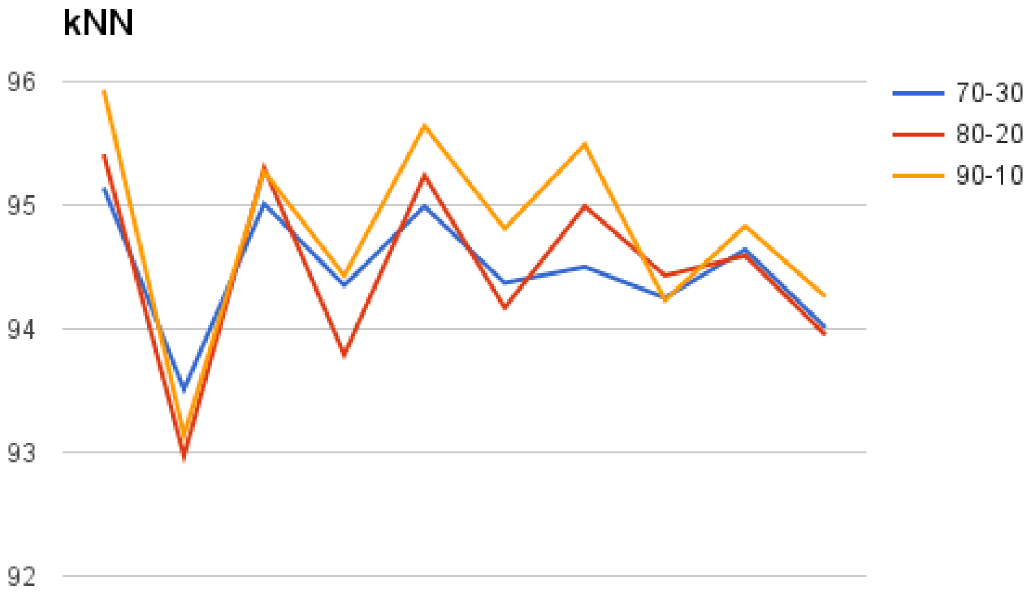
1.3 EXPERIMENTAL PROCEDURE

Different data splits were implemented, each was used to test the three different kNN implementations. A grid search was performed on all three implementations of kNN a k value of one to one hundred was used with the data from one to ten being shown in the tables bellow. An external library was used to perform a grid search on SVM, this performed SVM with different kernals, gamma, degree and C values, then returned the percentage of correct tests. Any results taken are averages of ten runs of those values, this includes accuracy, precision, recall and F1 score.

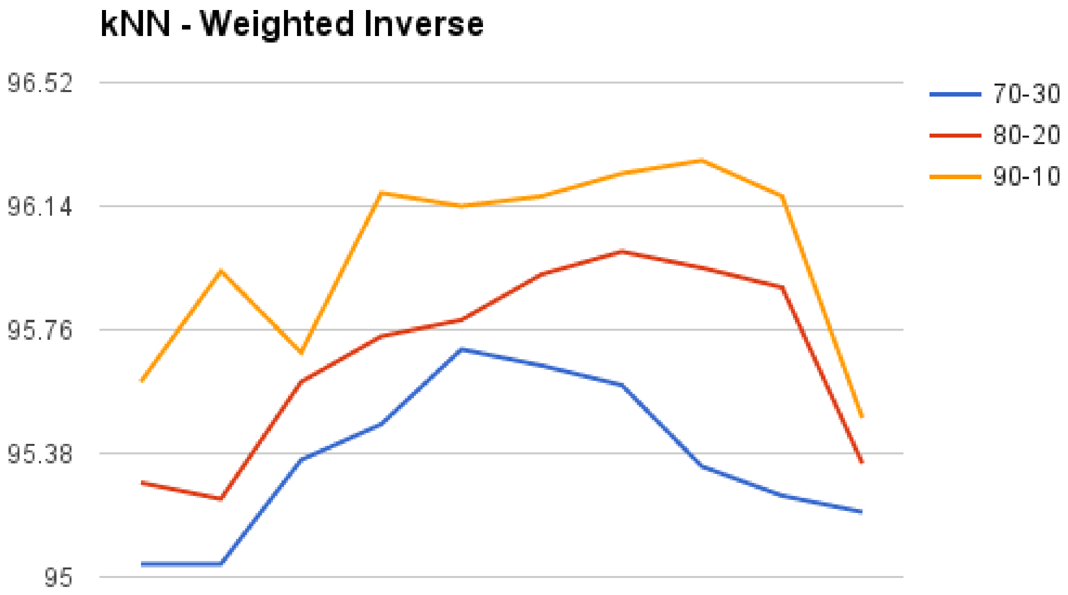
2 Evidence of the performance of your chosen approaches on the data

2.1 Percentages

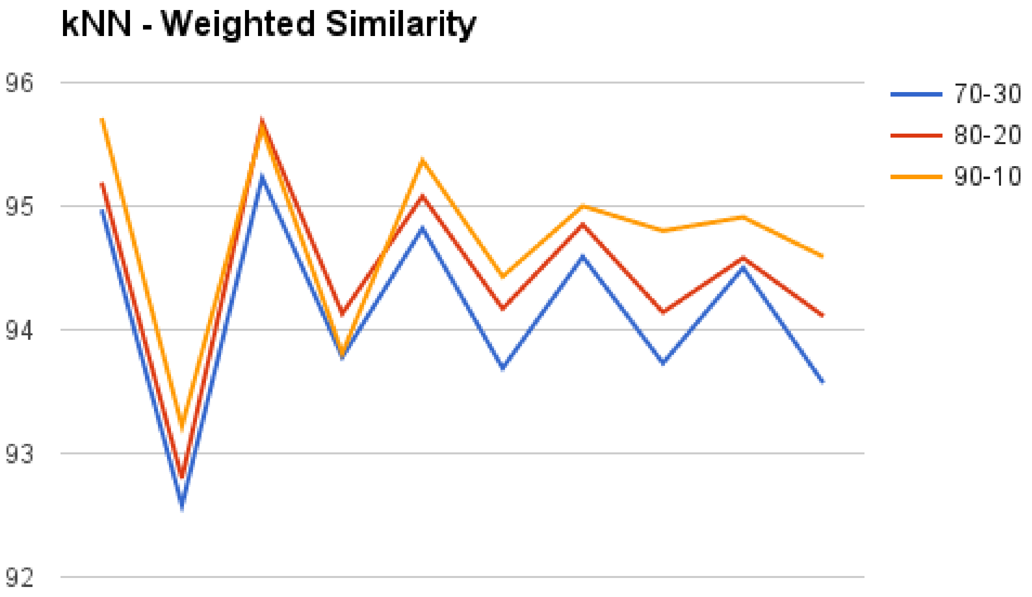
kNN Results			
K Value	kNN - 70:30	kNN - 80:20	kNN - 90:10
1	95.14	95.41	95.93
2	93.51	92.97	93.14
3	95.01	95.30	95.27
4	94.35	93.79	94.43
5	94.99	95.24	95.64
6	94.37	94.17	94.81
7	94.50	94.99	95.49
8	94.25	94.43	94.23
9	94.64	94.59	94.83
10	94.01	93.95	94.26
Best	1 - 95.14	1 - 95.41	1 - 95.93



kNN Weighted Inverse Results			
K Value	kNN Weighted Inverse - 70:30	kNN Weighted Inverse - 80:20	kNN Weighted Inverse - 90:10
1	95.04	95.29	95.60
2	95.04	95.24	95.94
3	95.36	95.60	95.69
4	95.47	95.74	96.18
5	95.70	95.79	96.14
6	95.65	95.93	96.17
7	95.59	96.00	96.24
8	95.34	95.95	96.28
9	95.25	95.89	96.17
10	95.20	95.35	95.49
Best	5 - 95.70	7 - 96.00	8 - 96.28



kNN Weighted Similarity Results			
K Value	kNN Weighted Similarity - 70:30	kNN Weighted Similarity - 80:20	kNN Weighted Similarity - 90:10
1	94.97	95.19	95.71
2	92.58	92.80	93.22
3	95.23	95.68	95.63
4	93.78	94.13	93.81
5	94.82	95.08	95.37
6	93.69	94.17	94.43
7	94.59	94.85	95.00
8	93.73	94.14	94.80
9	94.50	94.58	94.91
10	93.57	94.11	94.59
Best	3 - 95.23	3 - 95.68	1 - 95.71



SVM				
Kernal	C Value	Gamma	Degree	Result
RBF	100	0.01	-	98.08
LINEAR	1	-	-	96.31
POLY	1	0.1	5	97.63
SIGMOID	1000	0.001	-	96.8

The best from each table are:

- kNN - 90:10 train:test, where K = 1 with 95.93% accuracy
- kNN Weighted Inverse - 90:10 train:test, where K = 8 with 96.28% accuracy
- kNN Weighted Similarity - 90:10 train:test, where K = 1 with 95.71% accuracy
- SVM using the RBF kernal with C = 100 and Gamma = 0.01 with 98.08% accuracy

2.2 Accuracy, Precision, Recall and F-Measure

- kNN - 90:10 train:test, where K = 1
Accuracy: 0.959706959707
Precision: 0.959706959707
Recall: 0.959706959707
F1: 0.959706959707

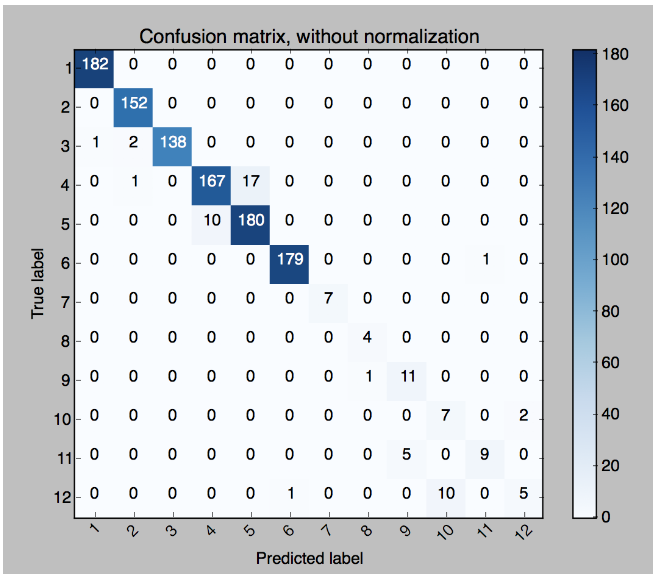
- kNN Weighted Inverse - 90:10 train:test, where K = 8
Accuracy: 0.951465201465
Precision: 0.951465201465
Recall: 0.951465201465
F1: 0.951465201465

- kNN Weighted Similarity - 90:10 train:test, where K = 1
Accuracy: 0.957875457875
Precision: 0.957875457875
Recall: 0.957875457875
F1: 0.957875457875

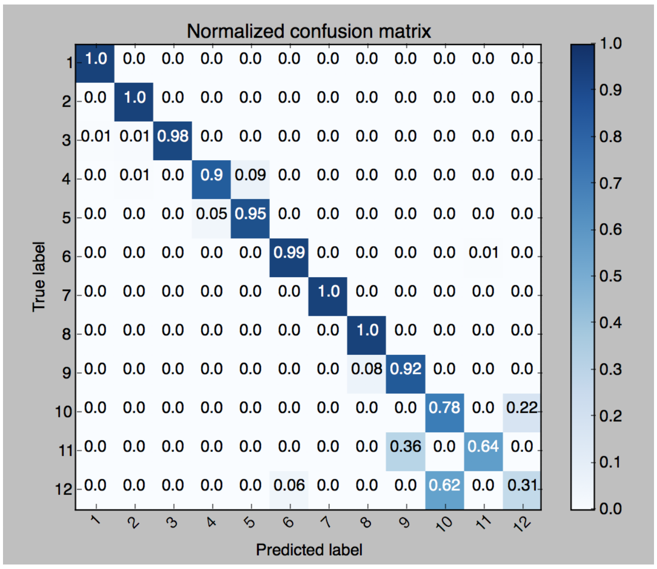
- SVM using the RBF kernal with C = 100 and Gamma = 0.01
Accuracy: 0.981391092129
Precision: 0.981391092129
Recall: 0.981391092129
F1: 0.981391092129

2.3 Confusion Matrix

2.3.1 kNN - 90:10 train:test, where K = 1

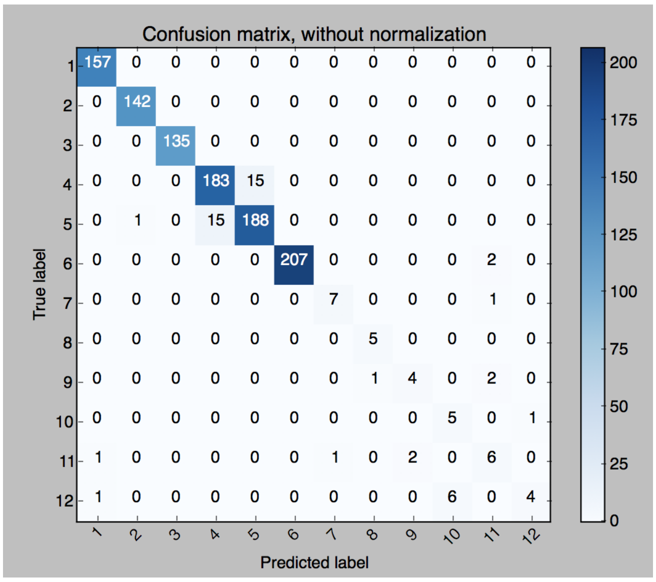


	1	2	3	4	5	6	7	8	9	10	11	12
1	182	0	0	0	0	0	0	0	0	0	0	0
2	0	152	0	0	0	0	0	0	0	0	0	0
3	1	2	138	0	0	0	0	0	0	0	0	0
4	0	1	0	167	17	0	0	0	0	0	0	0
5	0	0	0	10	180	0	0	0	0	0	0	0
6	0	0	0	0	0	179	0	0	0	0	1	0
7	0	0	0	0	0	0	7	0	0	0	0	0
8	0	0	0	0	0	0	0	4	0	0	0	0
9	0	0	0	0	0	0	0	1	11	0	0	0
10	0	0	0	0	0	0	0	0	0	7	0	2
11	0	0	0	0	0	0	0	0	5	0	9	0
12	0	0	0	0	0	1	0	0	0	10	0	5

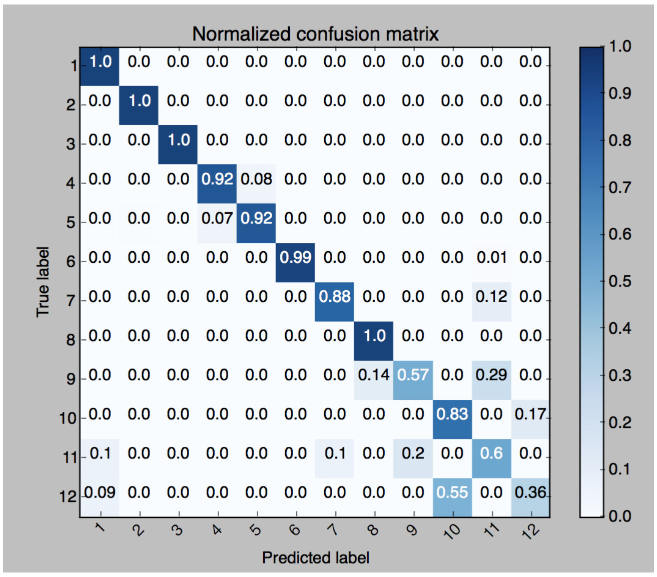


	1	2	3	4	5	6	7	8	9	10	11	12
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.01	0.01	0.98	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.01	0.0	0.9	0.09	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.05	0.95	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.99	0.0	0.0	0.0	0.0	0.01	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.08	0.92	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.78	0.0	0.22
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.36	0.0	0.64	0.0
12	0.0	0.0	0.0	0.0	0.0	0.06	0.0	0.0	0.0	0.62	0.0	0.31

2.3.2 kNN Weighted Inverse - 90:10 train:test, where K = 8

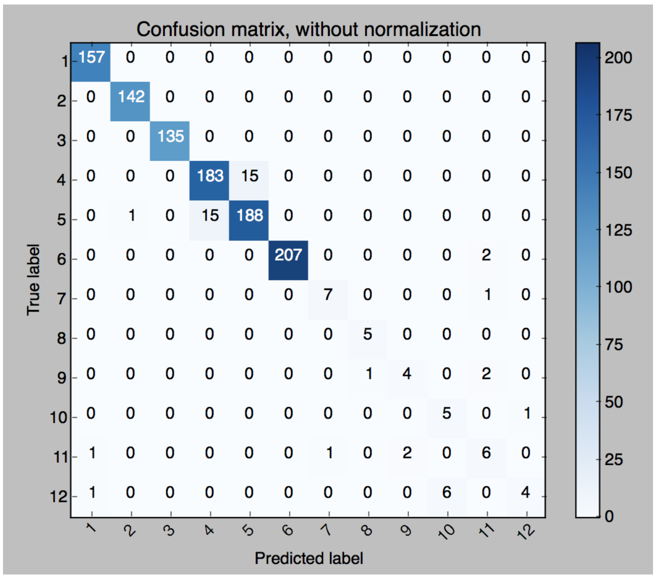


	1	2	3	4	5	6	7	8	9	10	11	12
1	157	0	0	0	0	0	0	0	0	0	0	0
2	0	142	0	0	0	0	0	0	0	0	0	0
3	0	0	135	0	0	0	0	0	0	0	0	0
4	0	0	0	183	15	0	0	0	0	0	0	0
5	0	1	0	15	188	0	0	0	0	0	0	0
6	0	0	0	0	0	207	0	0	0	0	2	0
7	0	0	0	0	0	0	7	0	0	0	1	0
8	0	0	0	0	0	0	0	5	0	0	0	0
9	0	0	0	0	0	0	0	1	4	0	2	0
10	0	0	0	0	0	0	0	0	0	5	0	1
11	1	0	0	0	0	0	1	0	2	0	6	0
12	1	0	0	0	0	0	0	0	0	6	0	4

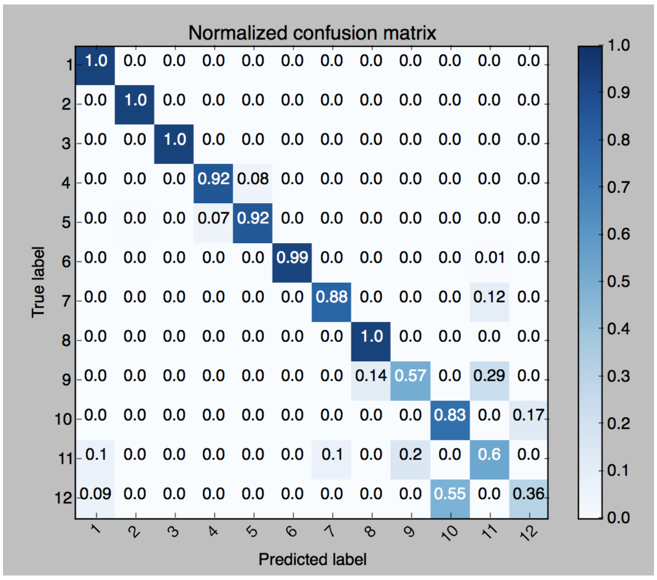


	1	2	3	4	5	6	7	8	9	10	11	12
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.92	0.08	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.07	0.92	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.99	0.0	0.0	0.0	0.0	0.01	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.88	0.0	0.0	0.0	0.12	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.14	0.57	0.0	0.29	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.83	0.0	0.17
11	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.2	0.0	0.6	0.0
12	0.09	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.55	0.0	0.36

2.3.3 kNN Weighted Similarity - 90:10 train:test, where K = 1

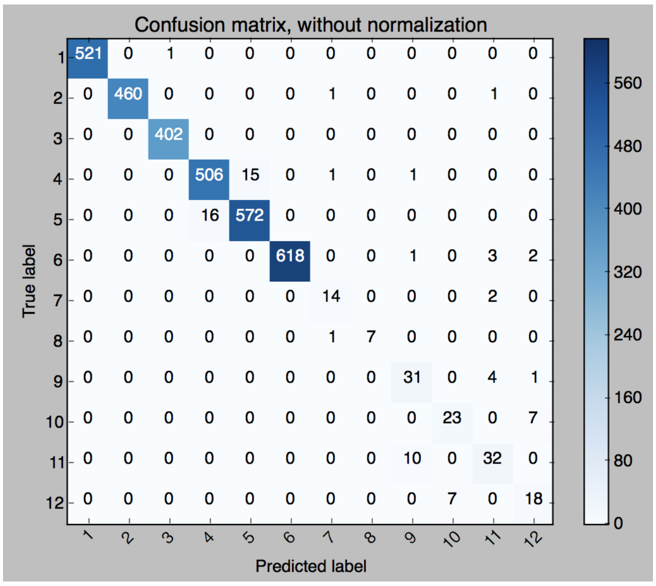


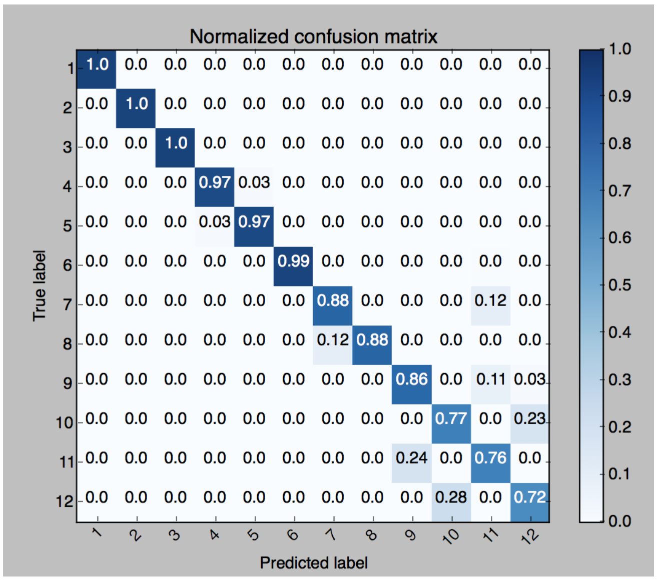
	1	2	3	4	5	6	7	8	9	10	11	12
1	157	0	0	0	0	0	0	0	0	0	0	0
2	0	142	0	0	0	0	0	0	0	0	0	0
3	0	0	135	0	0	0	0	0	0	0	0	0
4	0	0	0	183	15	0	0	0	0	0	0	0
5	0	1	0	15	188	0	0	0	0	0	0	0
6	0	0	0	0	0	207	0	0	0	0	2	0
7	0	0	0	0	0	0	7	0	0	0	1	0
8	0	0	0	0	0	0	0	5	0	0	0	0
9	0	0	0	0	0	0	0	1	4	0	2	0
10	0	0	0	0	0	0	0	0	0	5	0	1
11	1	0	0	0	0	0	1	0	2	0	6	0
12	1	0	0	0	0	0	0	0	0	6	0	4



	1	2	3	4	5	6	7	8	9	10	11	12
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.92	0.08	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.07	0.92	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.99	0.0	0.0	0.0	0.0	0.01	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.88	0.0	0.0	0.0	0.12	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.14	0.57	0.0	0.29	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.83	0.0	0.17
11	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.2	0.0	0.6	0.0
12	0.09	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.55	0.0	0.36

2.3.4 SVM using the RBF kernal with C = 100 and Gamma = 0.01





3 Conclusions from the experimentation

3.1 Results

As the results show SVM with specific parameters was the most correct algorithm, with an average of 98.08% correct. There was an issue with lack of data on the last six classes, leading to very high correctness for the first six then a fairly obvious drop off on the last six.

3.2 Ethics

The ethics of having this data with the participants full knowledge and permission is perfectly acceptable, this can change however if consent is not given. If this data was used by any third parties to use in any activities such as marketing or tracking, this is unacceptable. This boundary can however be moved in the case of emergency fire and rescue needing the data for a search and rescue in the case of a fire or earthquake.

4 References

Toby Breckon - www.github.com/tobybreckon/python-example-ml