W BOOTH SCHOOL OF ENGINEERING
PRACTICE AND TECHNOLOGY

Artificial Intelligence – SEP 786

Final Project

Harsh.D. Shah
Shahh34@mcmaster.ca

Introduction: The scope of this project was to implement either a regressor or a classifier and apply it to a dataset. This project focuses on the PCA and backward feature selection processes that were applied to a dataset from the UCI repository of a binary classification of authenticating banknotes. The program used was Python and the IDE was a jupyter notebook. The accuracy provides how well the model I created performed, and the possibilities of improving it. The confusion matrix was used to provide how many points of data were classified wrongly.

Dataset: The dataset had 1372 rows and 5 columns. The data was initially pictures taken of banknotes (although it is unclear for what country), and then transformed with a wavelet transformation tool. It is all continuous data. This dataset would later be split at a 75:25 ratio of Training data and Testing data respectively.

Let's delve deeper into the 5 columns, since it is the most integral part of the project:

Column 1: The variance of the wavelet transformed image
Column 2: The Skewness of the wavelet transformed image
Column 3: The curtosis of the wavelet transformed image
Column 4: The entropy of the image
Column 5: The class which has two classes – 0 and 1

Column 5 was made to be the y or target variable and omitted from the training data. This helped me confirm how my model classified the data and put the sum of all the appropriate values in the right box of the confusion matrix.

This dataset was then mean centered and scaled. This is a very integral requirement for implementing PCA. It also makes the data unitless, which can provide a lot of information that is valuable to the data analysis process.

Choosing and training the classifier(s): The objective was to choose two classifiers and compare the differences of the accuracy when the testing data is introduced. Since the focus was on the behavior, and not on the classifiers themselves, I chose to classify the data using Decision Tree and KNearestNeighbour algorithms and fit the PCA and a backward feature selection model. Choosing the classifiers was easier since the data had fairly small number of components, and from what I learned in class, this seemed to be the best fit for my solution in mind.

Training the data had its own challenges, but the implementation was made easier using the libraries. The timing and implementation of model fitting was all fit into a data frame. This is shown in table 1.

Table 1 – Results with Training and Testing Time

Accuracy	Algorithm	Hue	False_Negative	False_Positive	Method	Test_T	Total_Attributes	Train_T	True_Negative	True_Positive
0.679300292	Decision Tree	1	55	55	PCA	0.012299	1	0.009923	136	97
0.865889213	Decision Tree	1	20	26	PCA	0.01248	2	0.010495	165	132
0.962099125	Decision Tree	1	6	7	PCA	0.013413	3	0.010758	184	146
0.688046647	KNN	2	58	49	PCA	0.080967	1	0.006533	142	94
0.880466472	KNN	2	25	16	PCA	0.062501	2	0.003536	175	127
0.970845481	KNN	2	3	7	PCA	0.061454	3	0.003628	184	149
0.976676385	Decision Tree	3	5	3	BS	0.006512	4	0.005404	188	147
0.988338192	Decision Tree	3	4	0	BS	0.003752	3	0.002654	191	148
0.935860058	Decision Tree	3	11	11	BS	0.004363	2	0.003148	180	141
0.787172012	Decision Tree	3	39	34	BS	0.003058	1	0.002048	157	113
0.997084548	KNN	4	0	1	BS	0.019506	4	0.002681	190	152
0.991253644	KNN	4	0	3	BS	0.01846	3	0.001491	188	152
0.935860058	KNN	4	11	11	BS	0.015239	2	0.00118	180	141
0.825072886	KNN	4	27	33	BS	0.015653	1	0.001605	158	125
					PCA - Principal Component Analysis					
					BS - Backward feature selection					

Results: The results shown in table 1 provide very good information about my model. Training times are very, very small, and hence it just proves that not a lot of processing power is needed to train the model. This can be because there are only 4 columns of data, which I had previously considered to be large, since there are 1372 instances, but it was shocking how fast the model was trained across all iterations of the model. In the PCA model, as the number of attributes increased, it was expected that the training time would increase, but for the KNN, the timing remained somewhat consistent with the increasing accuracy of the model.

In terms of best accuracy, the KNN – BS combination worked best, especially since it was seen that even with fewer attributes, the accuracy was quite high for the model. In Figure 1, it shows that the accuracy was really high with a reduction of attributes (in the pink), this was also the most consistent, even at the lowest attribute.

Conclusion: Looking at the behavior of the data, through several graphs and iterations, that a high accuracy is achieved in all of the classifiers, but there are some that work better than the others (which was obviously expected), I would like to add that I would choose to go with the KNN-BS model because it fit the model the best through all the variations. The training time is fast, and the results very thorough. This was a great learning process, and the results are very conclusive. The model works and the classification was done successfully by reducing error to the minimum.

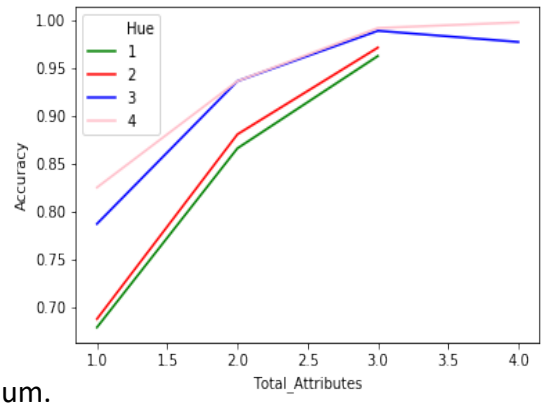

Image 1 – Total Attributes vs Accuracy

Citations:

1.) Owner of database: Volker Lohweg (University of Applied Sciences, Ostwestfalen-Lippe, volker.lohweg '@' hs-owl.de)
Donor of database: Helene DÃ¶rksen (University of Applied Sciences, Ostwestfalen-Lippe, helene.doerksen '@' hs-owl.de)
Date received: August, 2012