CIS 6930 - Introduction to Data Science

-- Project 4 --

Done by: Sethuraman Sundararaman

UF #: 11321142

Introduction:

Data preprocessing:

Approach 1:

1. For each csv file read R array, G array and B array. To determine the bin in which each it falls is found as follows:

Do the following separately for R, G, B arrays separately: new_val=(Max(Array)- curr_val)/ (Max(Array)- Min(Array))*16

2. Once done generate a line in the train csv file for each image. The line is created as new R array concatenated with G array followed by B array. Finally the class label is appended.

3. The above preprocessing is applied for test data as well.

4. This process can be described as the normalization of the data set and used to set the data set on a same scale (here 0 to 16).

Approach 2:

1. Read train file from the previous approach. For each data set in the file create a 48 length integer array (feature vector) with all the elements initialized to 0.

2. For each line update its corresponding feature vector as:

```
W=line.split(‘,’)
Fv=[[0]*48]  # 48 length histogram representation
D=[ int(i) for i in W[:-1] ]  #integer array of the dataset
count=0
while count<len(D):
    Fv[(count/100)*16+D[count]]+=1
    count+=1
```

Now the Fv consists of histogram feature vector consistent with the approach 2 representation.

3. Repeat the above process for all the datasets and write its corresponding feature vector to a new file with class label appended.

4. Do the same processing for test file as well.

Approach 3:

1. In order to cluster all the pixels in the train file the approach is as follows: From the training file used in approach 1:

Each data set generates 100 pixels of RGB triplet. Write down the pixel data to the train file without the label. Do this process for all the data set. Perform the same data preprocessing for test data and generate the test file.

Rapid miner operators used:

Operator	Description
Read csv	This operator is used to parse the csv file and generate the dataset object for classification or clustering.
Write csv	This operator is to write the data set to the csv
Decision tree	This operator takes in a dataset and builds the decision tree model
Apply model	Used to apply a model on test data
performance	Given a labelled sequence of test data and a model this operator returns the performance measure
K means	Given a data set this model returns both the clustering model and labels for the training set

Process description:

Approach 1 and 2:

Operators used:

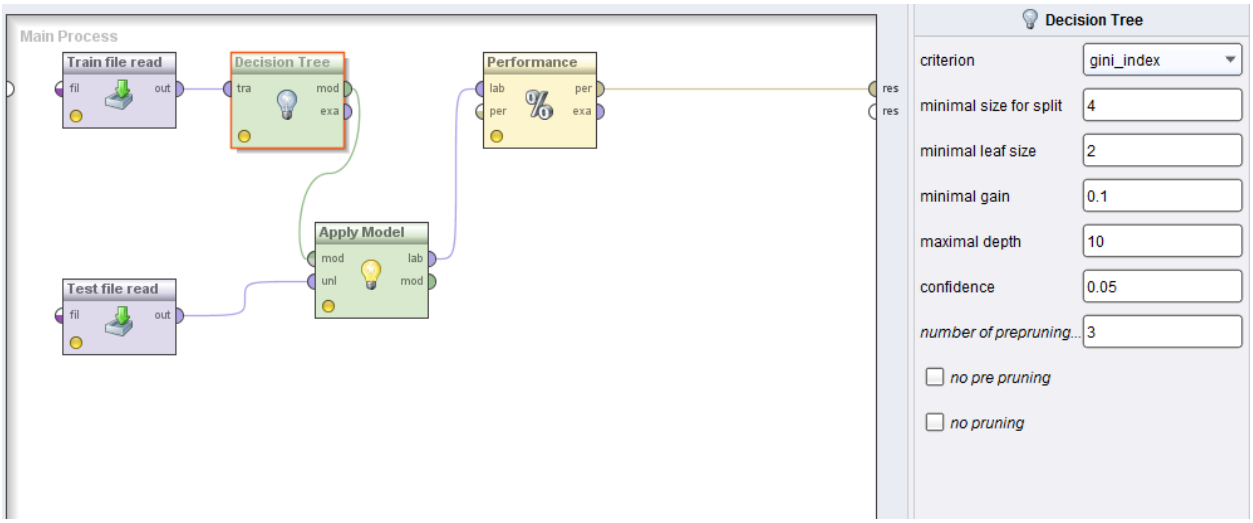
Read csv, Decision tree, Apply model and performance

Process:

1. Read train and test files.
2. Construct a decision tree on training data using gini index criterion.
3. The above model is applied on test data.
4. The performance is read using the performance operator.

Differences in the approaches:

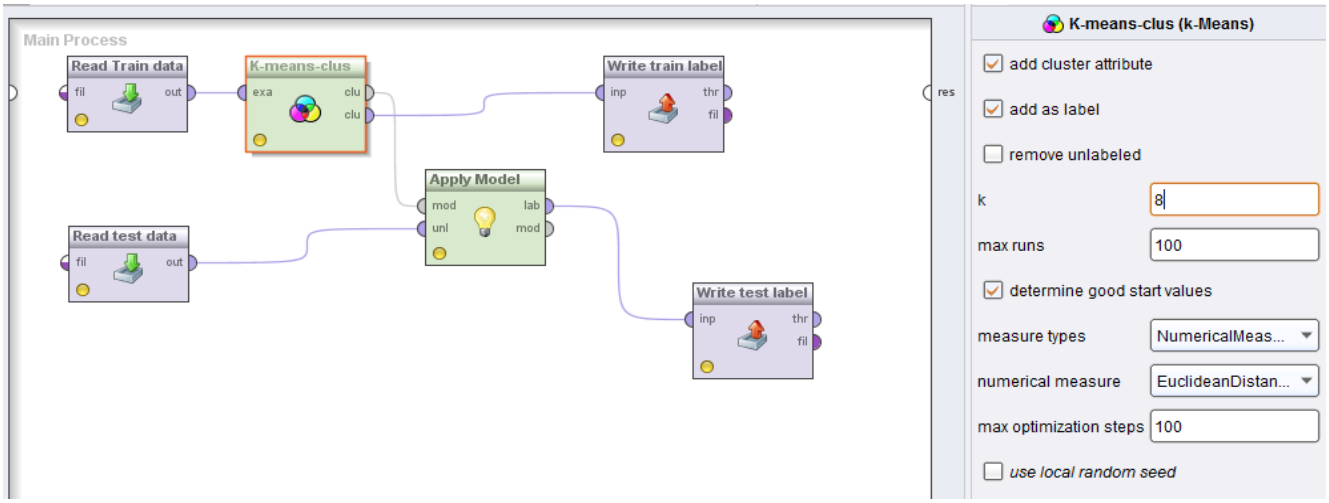
Only the train and test data files differ. The entire process is same for both the approaches.



Approach 3:

Intermediate processing:

1. Apply K means clustering on the training file for third approach.
2. Label all the pixels based on its cluster.
3. Apply the model on the testing data and label all the pixels in the test file as well



Now this generates the intermediate train and test files which needs to be further processed to generate the feature vectors.

The feature vector is represented as follows:

The feature vector belongs to R^k vector space where the k is the k factor supplied in the clustering phase. The i^{th} component represents the frequency of i^{th} cluster in the labelled data.

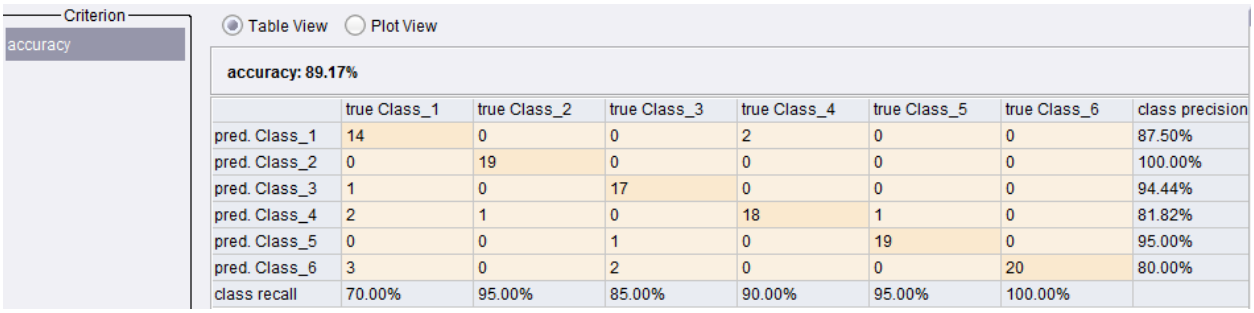
Generate feature vectors for all the training samples and append it with its corresponding label.

Generate feature vectors for the testing sample as well. Here the model constructed using training data is directly applied on the test data to label the pixels.

Now use the same classification process used in the previous two approaches with the training and test files generated above.

Results:

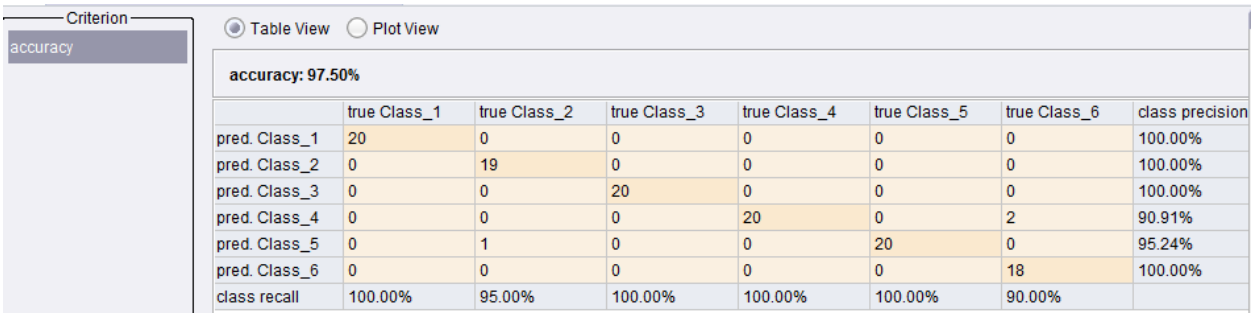
Approach 1:


Criterion
accuracy
Table View
Plot View
accuracy: 89.17%

	true Class_1	true Class_2	true Class_3	true Class_4	true Class_5	true Class_6	class precision
pred. Class_1	14	0	0	2	0	0	87.50%
pred. Class_2	0	19	0	0	0	0	100.00%
pred. Class_3	1	0	17	0	0	0	94.44%
pred. Class_4	2	1	0	18	1	0	81.82%
pred. Class_5	0	0	1	0	19	0	95.00%
pred. Class_6	3	0	2	0	0	20	80.00%
class recall	70.00%	95.00%	85.00%	90.00%	95.00%	100.00%	

The relatively poor performance of Approach 1 is due to the dimensionality of the feature vector. The decision tree's performance gets degraded because of the abundance of irrelevant information.

Approach 2:

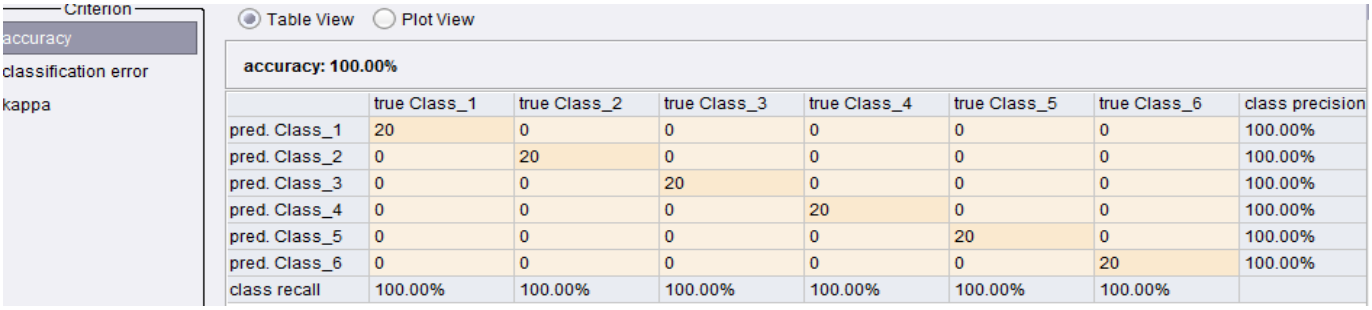

Criterion
accuracy
Table View
Plot View
accuracy: 97.50%

	true Class_1	true Class_2	true Class_3	true Class_4	true Class_5	true Class_6	class precision
pred. Class_1	20	0	0	0	0	0	100.00%
pred. Class_2	0	19	0	0	0	0	100.00%
pred. Class_3	0	0	20	0	0	0	100.00%
pred. Class_4	0	0	0	20	0	2	90.91%
pred. Class_5	0	1	0	0	20	0	95.24%
pred. Class_6	0	0	0	0	0	18	100.00%
class recall	100.00%	95.00%	100.00%	100.00%	100.00%	90.00%	

The performance has improved because of the dimensionality reduction.

Approach 3:

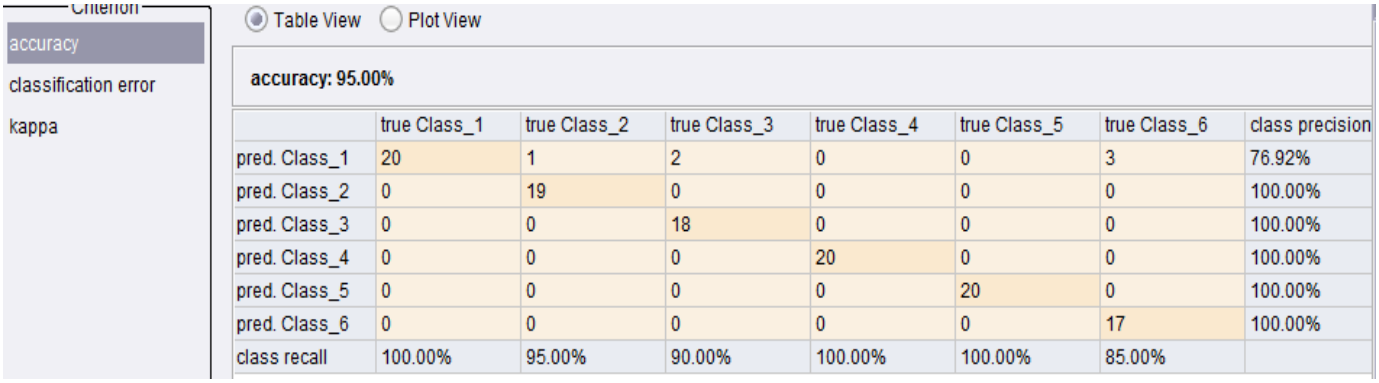
For k=8



	true Class_1	true Class_2	true Class_3	true Class_4	true Class_5	true Class_6	class precision
pred. Class_1	20	0	0	0	0	0	100.00%
pred. Class_2	0	20	0	0	0	0	100.00%
pred. Class_3	0	0	20	0	0	0	100.00%
pred. Class_4	0	0	0	20	0	0	100.00%
pred. Class_5	0	0	0	0	20	0	100.00%
pred. Class_6	0	0	0	0	0	20	100.00%
class recall	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	

The amazing performance is due to considering the statistics of the pixel color as such by clustering it, rather than individual RGB intensity. The fact that k=8 produced 100% accuracy reflects the fact that the pixels were indeed drawn from 8 clusters and the constructed cluster model is the true model from which the pixels are drawn to construct the image.

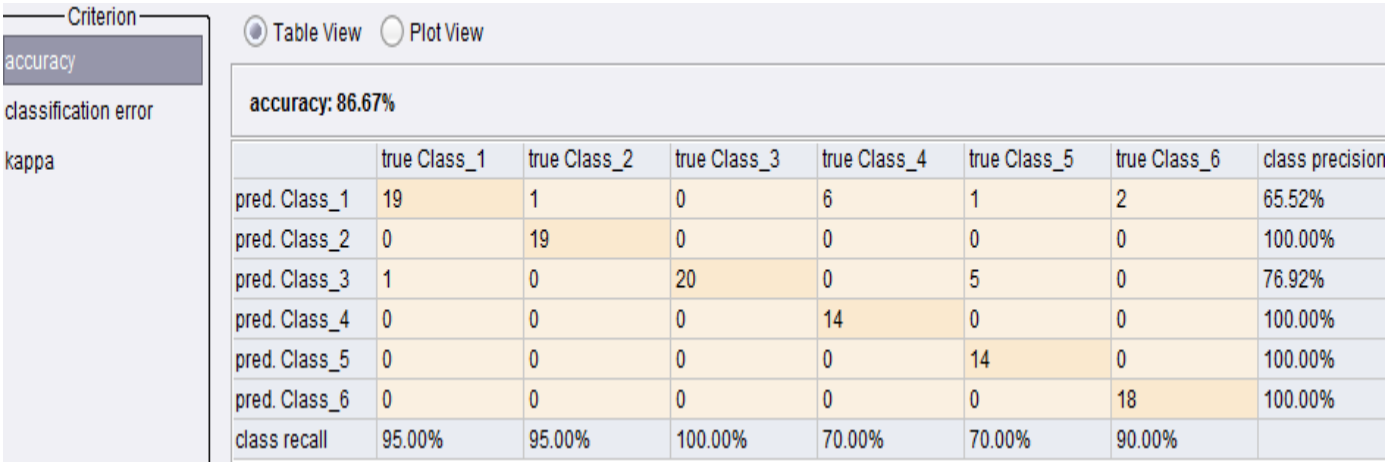
For k=16



	true Class_1	true Class_2	true Class_3	true Class_4	true Class_5	true Class_6	class precision
pred. Class_1	20	1	2	0	0	3	76.92%
pred. Class_2	0	19	0	0	0	0	100.00%
pred. Class_3	0	0	18	0	0	0	100.00%
pred. Class_4	0	0	0	20	0	0	100.00%
pred. Class_5	0	0	0	0	20	0	100.00%
pred. Class_6	0	0	0	0	0	17	100.00%
class recall	100.00%	95.00%	90.00%	100.00%	100.00%	85.00%	

Lowered performance of k=16 and k=32 represents deviation from the true model and verifies that k=8 is the perfect model.

For k=32



	true Class_1	true Class_2	true Class_3	true Class_4	true Class_5	true Class_6	class precision
pred. Class_1	19	1	0	6	1	2	65.52%
pred. Class_2	0	19	0	0	0	0	100.00%
pred. Class_3	1	0	20	0	5	0	76.92%
pred. Class_4	0	0	0	14	0	0	100.00%
pred. Class_5	0	0	0	0	14	0	100.00%
pred. Class_6	0	0	0	0	0	18	100.00%
class recall	95.00%	95.00%	100.00%	70.00%	70.00%	90.00%	

Conclusion:

Thus the impact of feature representation on the classification is studied and the results are tabulated. The more relevant information we provide the better the results are. The approach 1 uses no information extraction and provides lot of irrelevant information. The approach 2 decreases the dimensionality but the statistics it reported was based on individual RGB components. Approach 3 reports the statistics of pixel as a whole by clustering like pixels. The results proves the fact that the images were indeed constructed from 8 clusters and the approach 3 with k=8 method of feature representation is the optimal representation.