# **READ ME – Homework 3**

The program will first build the look up table. For doing so, it will prompt user to enter the path of the folder where “lookup.csv” is stored. The file is in the package. User can either use the preloaded file or give the path of a folder where they want to create the file. Creating the file takes the program 10-12 minutes. Reading from existing pickle takes less than 30 seconds

**Enter path in the following format:**

**‘/Users/rashmivarma/Documents/’**

As there are 60 images each in the folders, a total of 120 images; the look up table pixelates every image and stores it’s RGB value. This RGB value has further been divided into R1, R2, R3, R4, B1, B2, B3, B4, G1, G2, G3 and G4, depending on their value. If R lies between 0 and 64, it falls under R1, if it is between 65- 127, it falls under R2 and so on. Doing so for each pixel, makes classification easier.

Once the table has been computed/read, a User Driven Menu appears. This Menu runs continuously and can be quit by entering ‘0’.

Pressing 1, enables KNN classification algorithm.

Here the user is asked to input the path of the image the user wishes to classify. Input the file path in the following format:

**‘/Users/rashmivarma/Documents/sample.jpeg’**

The training set has been entirely built on jpeg images but the algorithm classifies both jpg and jpeg files.

The user is then prompted to enter a value of k for KNN. Please enter k less than 120.

On entering so, the algorithm computes and classifies the image into landscape or headshot.

Pressing 2, enables 3-fold cross validation.

Here the algorithm extracts headshots and landscapes from our look up table. It then shuffles them both and partitions them randomly without repetition into 3 folds. We once again shuffle the partitions amongst themselves to ensure proper shuffling

We combine first two folds to make our training set and third fold to create the validation set. We then compare distances to classify our validation set. And then find accuracy. The accuracy of our algorithm currently is around 50%.

Pressing 3, enables k-means clustering

Here the algorithm considers one row of the look up table as a point. Each point is considered. We randomly pick 2 centers and compute distances of every point with the centers and accordingly divide the clusters. For the sake of this assignments, we have stopped segregating clusters when we have achieved an accuracy of more than 50 i.e. when k-means has correctly classified half the data

Pressing 4, enables hierarchical clustering

Here, the program clubs points in a cluster together. Consider our output of the dataset:

(((((((((((((((((landscape01.jpeg,((((((((((((((((((landscape05.jpeg,landscape30.jpeg),((((((((((landscape06.jpeg,landscape45.jpeg),(((((landscape13.jpeg,((landscape28.jpeg,landscape41.jpeg),((((landscape50.jpeg,headshot22.jpeg),headshot41.jpeg),headshot17.jpeg),headshot46.jpeg))),((landscape18.jpeg,headshot03.jpeg),((landscape19.jpeg,((headshot04.jpeg,headshot29.jpeg),headshot33.jpeg)),(((((landscape21.jpeg,headshot23.jpeg),landscape22.jpeg),landscape26.jpeg),landscape25.jpeg),headshot49.jpeg)))),(((((landscape34.jpeg,headshot25.jpeg),headshot16.jpeg),headshot53.jpeg),(landscape56.jpeg,headshot12.jpeg)),(headshot27.jpeg,headshot45.jpeg))),landscape35.jpeg),(landscape60.jpeg,headshot52.jpeg))),(landscape47.jpeg,headshot32.jpeg)),((((headshot02.jpeg,headshot37.jpeg),(((((headshot06.jpeg,headshot38.jpeg),(headshot30.jpeg,headshot58.jpeg)),(headshot31.jpeg,headshot39.jpeg)),headshot50.jpeg),headshot44.jpeg)),headshot15.jpeg),headshot11.jpeg)),headshot09.jpeg),((landscape08.jpeg,(landscape09.jpeg,landscape33.jpeg)),landscape48.jpeg)),(((((landscape10.jpeg,((landscape16.jpeg,(headshot20.jpeg,headshot48.jpeg)),landscape32.jpeg)),(headshot18.jpeg,headshot59.jpeg)),((headshot19.jpeg,headshot34.jpeg),headshot42.jpeg)),landscape11.jpeg),landscape14.jpeg)),landscape15.jpeg),landscape27.jpeg),(landscape44.jpeg,landscape52.jpeg))),(landscape23.jpeg,landscape58.jpeg)),headshot01.jpeg),landscape12.jpeg),landscape29.jpeg),landscape39.jpeg),(landscape54.jpeg,headshot47.jpeg)),headshot26.jpeg),landscape24.jpeg),landscape43.jpeg),(((((((headshot05.jpeg,((headshot08.jpeg,headshot55.jpeg),headshot14.jpeg)),headshot24.jpeg),(headshot13.jpeg,headshot35.jpeg)),headshot21.jpeg),headshot28.jpeg),headshot56.jpeg),(((headshot10.jpeg,headshot60.jpeg),headshot43.jpeg),headshot51.jpeg))),landscape53.jpeg),landscape59.jpeg),headshot57.jpeg),landscape31.jpeg),landscape36.jpeg),headshot36.jpeg)),landscape20.jpeg),(landscape38.jpeg,landscape57.jpeg)),landscape02.jpeg),landscape49.jpeg),(landscape03.jpeg,landscape55.jpeg)),headshot40.jpeg),landscape46.jpeg),headshot54.jpeg),landscape40.jpeg),landscape17.jpeg),headshot07.jpeg),landscape37.jpeg),landscape07.jpeg),landscape51.jpeg),landscape42.jpeg),landscape04.jpeg)

On cutting the resulting dendogram in 2, we get the above 2 clusters. Cluster 1 (red) and cluster 2 (green).

The image in the second cluster is a landscape with low intensities of RGB. That is the most probable reason that the image got classified separately.

Now observing the misclassifications, one such misclassification is shown below (highlighted in output)

… 

Above are headshot47 (left) and landscape54 (right). We notice that both images have a lot of yellow in it. Hence while computing RGB, the algorithm must have gotten the distance as less between these images and therefore misclassified it.

On pressing 5, single linkage hierarchical clustering is performed on the Flags dataset. Our output is as follows:

(((((((((((((((((((((((((((((((((((((Afghanistan.jpg,Malawi.jpg),(((((((((((((((((((((((((((((((((((((((Albania.jpg,(((China.jpg,Morocco.jpg),(Kyrgyzstan.jpg,Vietnam.jpg)),((Hong\_Kong.jpg,Tunisia.jpg),Turkey.jpg))),(Denmark.jpg,(Switzerland.jpg,Tonga.jpg))),(East\_Timor.jpg,Trinidad\_and\_Tobago.jpg)),(((Austria.jpg,Peru.jpg),Bahrain.jpg),Canada.jpg)),Bermuda.jpg),((England.jpg,Georgia.jpg),((((Greenland.jpg,((Indonesia.jpg,Monaco.jpg),Poland.jpg)),Malta.jpg),Singapore.jpg),Lebanon.jpg))),Norway.jpg),Nepal.jpg),Montenegro.jpg),Taiwan.jpg),(Andorra.jpg,(((Chile.jpg,(((Croatia.jpg,Paraguay.jpg),(France.jpg,Russia.jpg)),Malaysia.jpg)),Philippines.jpg),Kiribati.jpg))),Cuba.jpg),Samoa.jpg),(Japan.jpg,South\_Korea.jpg)),((((Antigua\_and\_Barbuda.jpg,(((Egypt.jpg,(Iraq.jpg,Syria.jpg)),Yemen.jpg),(((Jordan.jpg,Palestine.jpg),Kuwait.jpg),Sudan.jpg))),((Madagascar.jpg,Tajikistan.jpg),Oman.jpg)),Puerto\_Rico.jpg),theGambia.jpg)),(Cambodia.jpg,(Haiti.jpg,Liechtenstein.jpg))),(Macedonia.jpg,Spain.jpg)),(Panama.jpg,(Slovakia.jpg,Slovenia.jpg))),(((((Burundi.jpg,Iran.jpg),Wales.jpg),Equatorial\_Guinea.jpg),Comoros.jpg),Central\_African\_Republic.jpg)),((((Costa\_Rica.jpg,(Dominican\_Republic.jpg,Thailand.jpg)),United\_Kingdom.jpg),Serbia.jpg),(Czech\_Republic.jpg,Mexico.jpg))),Cyprus.jpg),Seychelles.jpg),(Colombia.jpg,Ecuador.jpg)),(Ethiopia.jpg,((Mali.jpg,Zimbabwe.jpg),Myanmar.jpg))),(Moldova.jpg,(Mongolia.jpg,North\_Korea.jpg))),(Angola.jpg,Papua\_New\_Guinea.jpg)),(((Belgium.jpg,Chad.jpg),Romania.jpg),Venezuela.jpg)),((Bolivia.jpg,Ghana.jpg),(Maldives.jpg,Portugal.jpg))),Algeria.jpg),((((Australia.jpg,Cape\_Verde.jpg),Iceland.jpg),Nauru.jpg),Bosnia\_and\_Herzegovina.jpg)),(((Cameroon.jpg,Grenada.jpg),Mozambique.jpg),Togo.jpg)),(Germany.jpg,Uganda.jpg)),Niger.jpg),((((((((Benin.jpg,((Guinea-Bissau.jpg,Guinea.jpg),Senegal.jpg)),Mauritius.jpg),Burkina\_Faso.jpg),Namibia.jpg),Guyana.jpg),(Libya.jpg,(Saint\_Kitts\_and\_Nevis.jpg,Vanuatu.jpg))),United\_Arab\_Emirates.jpg),((Bulgaria.jpg,Italy.jpg),Hungary.jpg))),Republic\_of\_the\_Congo.jpg),Armenia.jpg),(Jamaica.jpg,Sao\_Tome\_and\_Principe.jpg)),Luxembourg.jpg),South\_Sudan.jpg)),Laos.jpg),(Barbados.jpg,(((Cayman\_Islands.jpg,Cook\_Islands.jpg),New\_Zealand.jpg),Guam.jpg))),(Israel.jpg,Uruguay.jpg)),India.jpg),((Gabon.jpg,Saint\_Vincent\_and\_the\_Grenadines.jpg),(Ireland.jpg,Ivorycoast.jpg))),(New\_Caledonia.jpg,South\_Africa.jpg)),(Rwanda.jpg,Sierra\_Leone.jpg)),((Brazil.jpg,Zambia.jpg),Tanzania.jpg)),Brunei.jpg),Eritrea.jpg),(Kenya.jpg,Suriname.jpg)),Lithuania.jpg),(Netherlands.jpg,Swaziland.jpg)),(Azerbaijan.jpg,(Bahamas.jpg,Democratic\_Republic\_of\_the\_Congo.jpg))),Estonia.jpg),((Dominica.jpg,Pakistan.jpg),(Mauritania.jpg,Saudi\_Arabia.jpg))),Uzbekistan.jpg),Bhutan.jpg),Finland.jpg),(Liberia.jpg,United\_States.jpg)),Djibouti.jpg),((((Belize.jpg,(El\_Salvador.jpg,(Greece.jpg,Scotland.jpg))),Sweden.jpg),Ukraine.jpg),Solomon\_Islands.jpg)),Sri\_Lanka.jpg),(((Argentina.jpg,Guatemala.jpg),((Botswana.jpg,Fiji.jpg),Palau.jpg)),Somalia.jpg)),Belarus.jpg),Lesotho.jpg),Kazakhstan.jpg),Nigeria.jpg),Turkmenistan.jpg),(Honduras.jpg,Nicaragua.jpg)),(Saint\_Lucia.jpg,San\_Marino.jpg)),Latvia.jpg),Qatar.jpg),Bangladesh.jpg),Tuvalu.jpg)

Here again, our one cluster is shown in green whereas the other is red. On observing the dendogram, we notice the following.

Consider one pair of the dendogram. Let’s consider Mauritania(left) and Saudi Arabia. Their flags are as follows:

…. 

Both flags are dominated by their green value and hence justified as to why the distance between them was low for them to be clustered together. We notice that these two flags are further clustered with the flags of Dominica and Pakistan which are also green dominated.

Pressing 6, shows the graph of accuracy for k from 1 to 10. The graph is constant and around 50%

Pressing 7 performs GA between the RGB values of two flags