Mention that I took the inspiration from the four methods from opencv. So I decided to manually implement each of them and compare the results to find the best performing model

Test functions using RBG, then move to using HSV because its better

* Reference paper about why HSV is better than RGB
* Reference paper about why 80/20 split is a good ratio for tests

431 total images

20% test images (86.2)(86) – test folder

80% image gallery (344.8)(345) - gallery folder

4 image categories. Split test images accordingly.

86/4 = 21.5

22 crowds, 21 F1-cars, 22 horses, 21 landscapes

-Xmx9216m

Correlation and Bhattacharyya have the best results in terms of accuracy to the label associated to the search image and image results (Crowds, F1 Cars, Horses, and Landscapes).

Averages of 65% and 70% respectively show a strong baseline of accuracy because these functions have no object detection so I would have initially guessed the results may have been lower because these methods are only dealing with colour. This means that from the test images above you can see that even the images that do not match the query image category they are still very similar in terms of colour so it would make sense for even my final search model to return images like those shown.

From here I should try to find a way to implement both of these methods with an object detection function.

The results from Chi-square and Intersection were very unimpressive in terms of label accuracy and returning images with similar colours. They will not be used in future models.

In this example, the compareHistograms method takes two histograms as input, normalizes them, and returns the correlation coefficient as the similarity score. The correlation coefficient is computed by first computing the mean and standard deviation of each histogram, and then computing the covariance between them. The resulting score is a value between -1 and 1, where a score of 1 indicates a perfect similarity and a score of -1 indicates a perfect dissimilarity.