K-Means Clustering
K-Means Clustering is an unsupervised learning method. If data set samples have labels, we prefer to use supervised method, but in real world mostly we don't have labels and that's why we prefer clustering methods which are known as unsupervised methods. The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are:
The centroids of the K clusters, which can be used to label new data
Labels for the training data (each data point is assigned to a single cluster)
Rather than defining groups before looking at the data, clustering allows you to find and analyze the groups that have been formed organically. The "Choosing K" section below describes how the number of groups can be determined.
Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents[1].
Project Details
As, I have mentioned previously I used OpenCV library for image processing with Python3 for this project. The other libraries that I have used NumPy for numerical arrays, Matplotlib for visualizing my result, Sklearn for machine learning.
First we will read image data using cv2.imread() function from OpenCV as cv2. Once the image is read using cv2, our image color channel comes to us as Blue-Green-Red. But we want Red-Green-Blue as our image color channel, so we convert it to the required channel using cv2.cvtcolor() function. Now we have 3-D parameters in our image data: row number X column number X colour channel number. But, we won't need row and column information separately. Besides, it is hard to deal with 3-D Matrix that's why we reshape() image and make it 2-D Matrix data. This was the preparation part for images and now we are ready to go on with clustering. Since we will import K-Means from top of our code, we can easily use it by only giving n_clusters, which represents cluster number originally. After that we will use fit() function to apply K-Means Clustering algorithm on our pre-processed image data and result will come back to clt objects. We use find_histogram() function to limit the number of histograms to the desired number of clusters. You can find the full details about find_histogram() and plot_colors2() function in this article below. As we don't want to find histogram for all pixel, and all the color palette, so we would like to limit it to the desired number of clusters. The plot_colors2() function will prepare a bar, and put colors on this bar as shown in the image below:
Figure 1. dominant colour bar
A histogram is an accurate graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable. When we come to find_histogram() function it takes one parameter which is trained clt object and returns histogram information. np. arange(0, len(np.unique(clt.labels_))+1) function will return evenly spaced values within a given interval. For our example, because we have 3 clusters numLabels' value will be [0 1 2 3]. After that, we will calculate the histogram according to numberLabels and cluster's label values. clt.labels_ is an array which includes cluster labels for each pixel. After calculating the histogram value, we will calculate the ratio of the histogram values we obtained because it is more difficult to work with numbers than to work with ratios. After calculating the histogram ratios, we will jump in to plot_colors2() function. This function will plot a bar which shows colours according to clusters and histogram. First we prepare a 3-D matrix, all values are zero. Then we plot the relative percentage of each cluster. cv2.rectangle() function draws a simple, thick, or filled up-right rectangle. We are filling a rectangle with colors in terms of histogram ratios. Below you can find all the code, I have mentioned above or Github version can be found here.

In the end, we prepare an image processing application which aims to find the dominant color for an image. We have used K-Means Clustering and calculate histogram to find the following result: