Edward Mina ML Exam 2

Problem 1:

For the case of image segmentation, an input image is brought in and is converted into its pixel values normalized by the 255 possible pixel color representations (intensities) and also thus their proximity to each other in a image. Note that colored images are combinations of pixel intensities for Red, Green and Blue pixels which can roughly produce most of the visible light spectrum. This however requires 3 bytes of data storage per pixel- one byte per pixel color intensity. In this exam, there is an equal number of bytes to the number of pixels ranging from grayscale intensities from 0 (black) to 255 (white). This was normalized so that the pixel intensities range from 0 to 1. Images are imported by python module imread where the image is converted into a 2-Dimensional numpy array illustration of pixel values. Python's openCV library can also convert images into a 3 Dimensional array factoring colors (Pixel Height, Pixel Length, Pixel Colors (3)). Since colors are only in greyscale, the third component doesn't play much of a role and the data can be compressed 2-D set of points that produces and image.

Explanation of K-Means and Implementation

K means is a unique and yet simple clustering algorithm used in unsupervised learning. Effectively what the model is that it take in some input, whether it be cluster of several gaussians, pixel groupings in an image, or some other cluster segmentable data set, and assigns labels and clusters. The first component of the algorithm, k represents the number of centroids which is a user-selected parameter. These centroids are initially arbitrary centers which represent the point that has the shortest distance from all points of its own cluster. For the purposes of this current implementation, euclidean distance is the basis for all distance calculations. After setting the random k centroids, the model begins to solve for the best representation of those k clusters. What this means is the first set of clusters are formed such that near by data points belong to that centroid class. The centroid is then recalculated after the data points near the initial random centroid are labelled. This process then continues until convergence such that the centroids no longer change after each iteration or until the model approaches some set max iteration count. Essentially what happens from a higher level perspective is that k random centroids are initially set, and within each iteration of the algorithm, points near the centroid are constantly being rellabelled and the centroid is being recalculated. K-means can be thought of as an extension of (unsupervised) k-nearest neighbors where each iteration finds the nearest neighbor centroid for a point. This process is done so that all the pixel points are classified and grouped

Steps in the Algorithm:

1. Set/Solve for Centroid
2. Assign and re/label points
3. Repeat

Code Process:

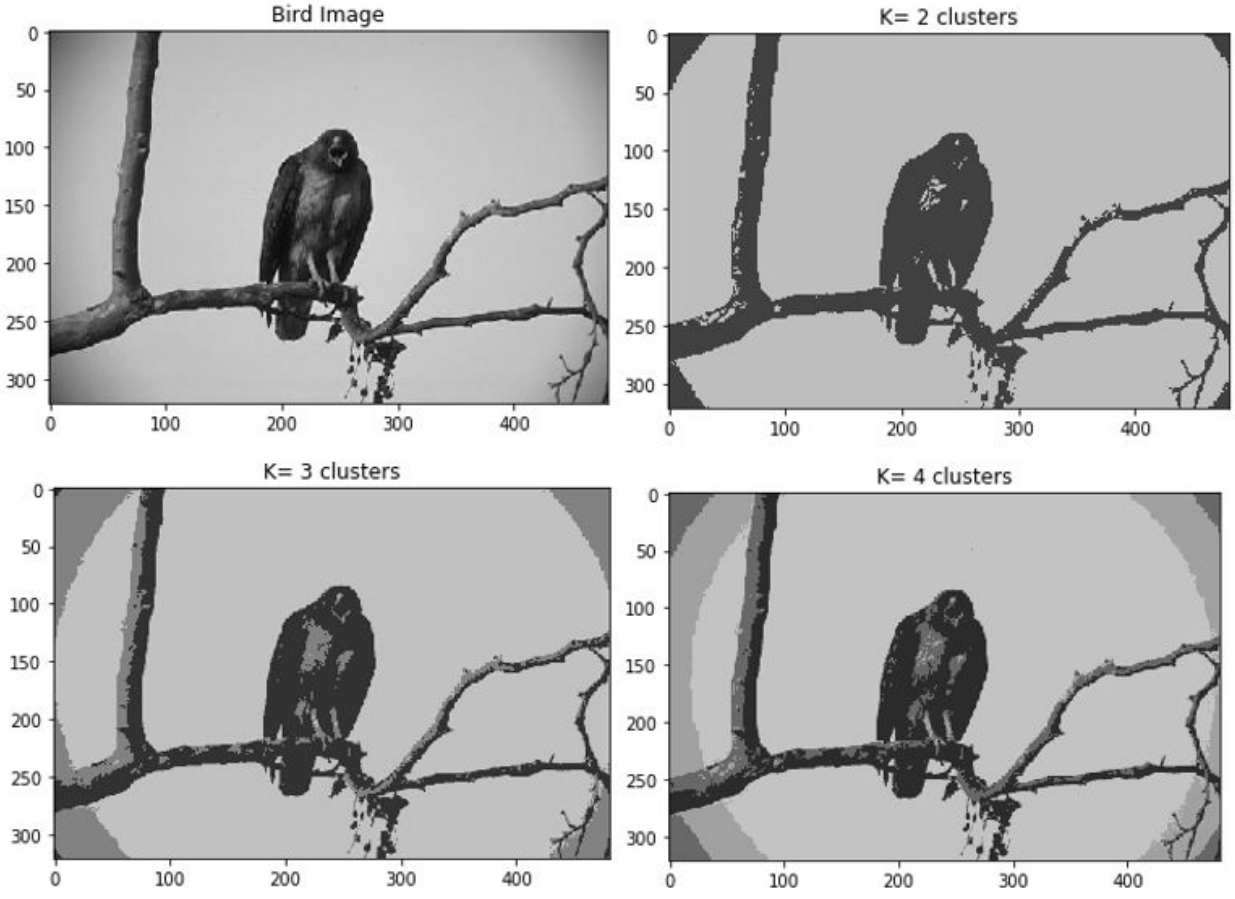
1. Display data scatter or in the case of data segmentation take image as a set of pixel values based on location (and color)
2. Create a k random (numpy.random.shuffle) centroids of 2 dimensions
3. For n = 0:k centroids, Solve for $d = \sqrt{(x_i - centroid_x(n))^2 + (y_i - centroid_y(n))^2}$
4. Using ex of 4 points and 3 centroids:
    a. Compute distance of all the points from all the centroids
        i. $||Point\ 1\ -\ 3\ centroids||_2^{\ 2}\ =\ 3\ distances\ =\ d1\ vector$
        ii. $||Point\ 2\ -\ 3\ centroids||_2^{\ 2}\ =\ 3\ distances\ =\ d2\ vector$
        iii. $||Point\ 3\ -\ 3\ centroids||_2^{\ 2}\ =\ 3\ distances\ =\ d3\ vector$
        iv. $||Point\ 4\ -\ 3\ centroids||_2^{\ 2}\ =\ 3\ distances\ \ d4\ vector$
        v. $d = [d1\ d2\ d3\ d4]$
    b. Output *all* distances into an array *d* and find the lowest distances for each set of points
        i. Point 1 $argmin\ (d1\ in\ d)$
        ii. Point 2 $argmin\ (d2\ in\ d)$

iii. Point 3 $argmin\ (d3\ in\ d)$
iv. Point 4 $argmin\ (d4\ in\ d)$

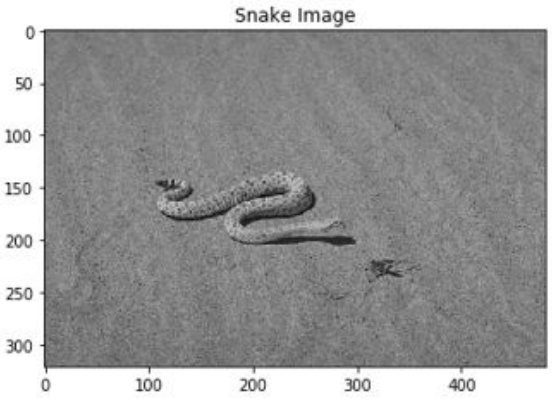
c. Uses the indices of that lowest distance (for each point) to determine indices of nearest centroid for each point
   i. (Index of Points where shortest distance was found) I= $location[argmin\ (d1, d2, d3, 4\ in\ d)]$
      1. This is because the index of distance is based on the points
d. For n = 0:k clusters:
   i. When n = any index in $location[argmin\ (d1, d2, d3, 4\ in\ d)]$
      1. n=1, when one of the minimal distances has the same index as n, the mean of all the points near the cluster will be averaged for the centroid
   ii. New centroids = mean(points(index for where distance was shortest))

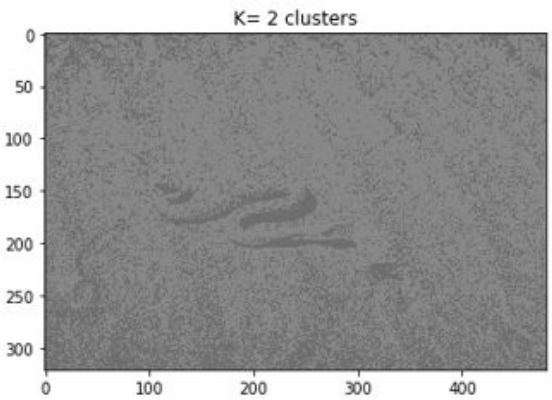
5. Summary: Find points with lowest distances from a particular centroid and average those points to get a new cetroid.
6. Picking a cluster label for a point is simply minimizing the distances of a distance vector contain how far a single point is from each centroid.
7. Repeat Process until centroids remain unchanged.

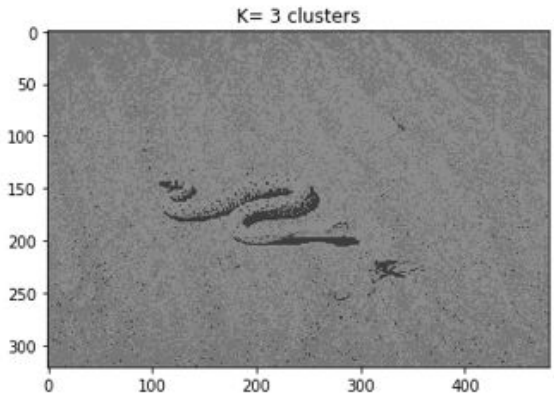
The image data is brought into the k means algorithm such that the k value and the initial clusters are predetermined. The output is the a vector of cluster labels equal to the area of the image (number of pixels) ranging from 0 o k. This is because each image pixel is classified in accordance to the number of clusters selected. Based on the k value, the image is shown with k segments such that an increase in k produces a clearer image:

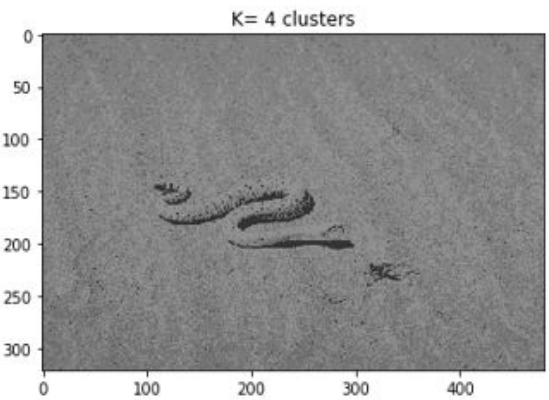


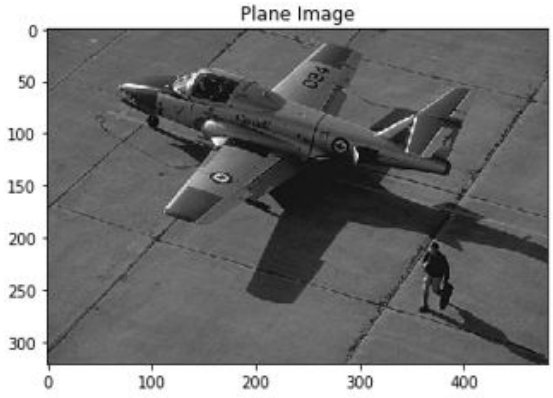
Edward Mina ML Exam 2

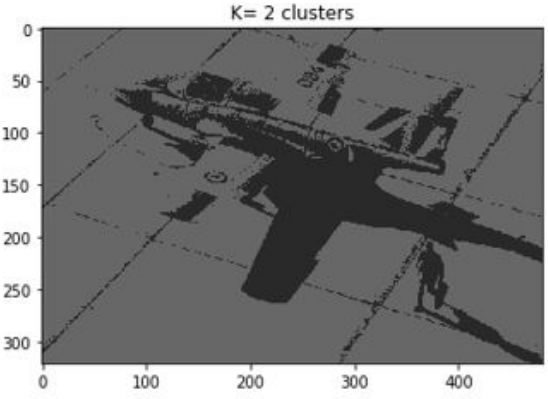


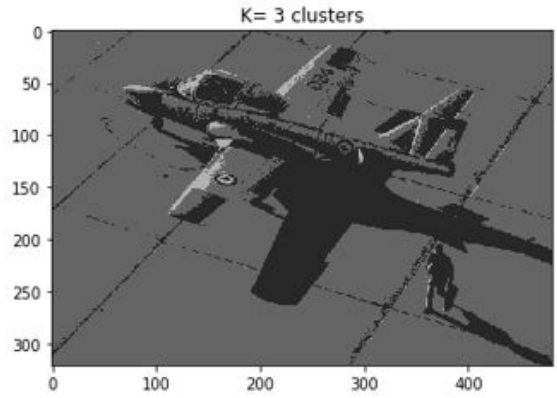


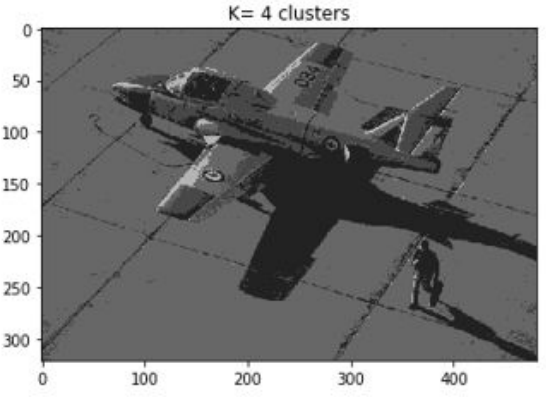












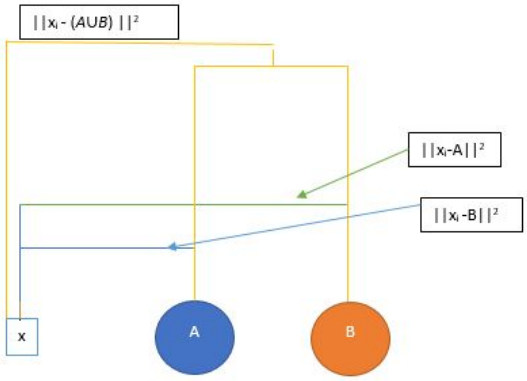
Edward Mina ML Exam 2

Explanation of (Ward) Hierarchical Clustering and Implementation

K means clustering differs in that it produces simple or flat partition which are cluster sets with no structure or organization within them. On the other hand, hierarchical clustering takes into account sub and super clusters often depicting relationships in a tree or dendrogram. More specifically this is known as agglomerative clustering, grouping smaller cluster into larger ones (“Bottom Up Approach”) . The algorithm starts with every single point as its own cluster continues until only one cluster remains. Then clusters that are closer are merged. Such an algorithm is deterministic and thus often does not need any random initial condition like k-means. However, in the case of the image segmentation, the number of clusters restricts the number of overall relationships found. A dendrogram of the image converges until there exists k “superclusters” instead of one representing the whole image. The hierarchical model used, ward linkage, is popular in image segmentation, and is shown below :

$$\Delta(A,B) = \sum_{i \in A \cup B} \|\vec{x}_i - \vec{m}_{A\cup B}\|^2 - \sum_{i \in A} \|\vec{x}_i - \vec{m}_A\|^2 - \sum_{i \in B} \|\vec{x}_i - \vec{m}_B\|^2$$

Where some delta is the merging cost of combining clusters. This linkage method uses a distance metric such that the distance between cluster A and B is how much the euclidean distances will increase after the clusters are merged. This is different from simply taking the Euclidean distance from the two clusters, and is more effective.

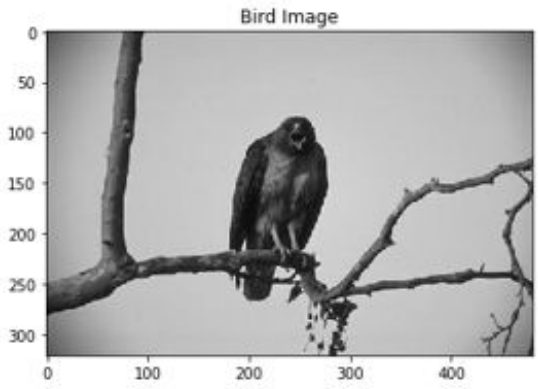


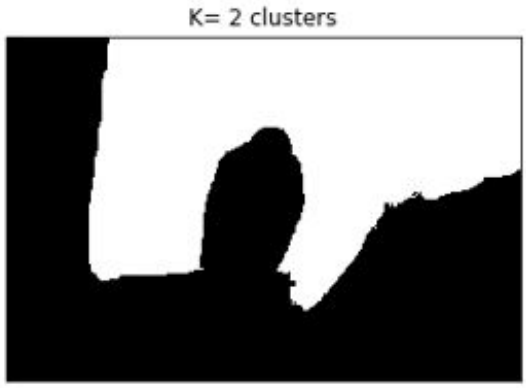
1. Set distance metric as Euclidean with ward linkage
2. Computer a matrix of distances for each each cluster
    a. Initially, all individual points are their own clusters
3. Consider all possible merges
4. Find the best (closest with ward linkage) pair to merge
    a. Argmin (Ward) Euclidean Distance for all existing clusters at the point
5. Converge when k super clusters are formed

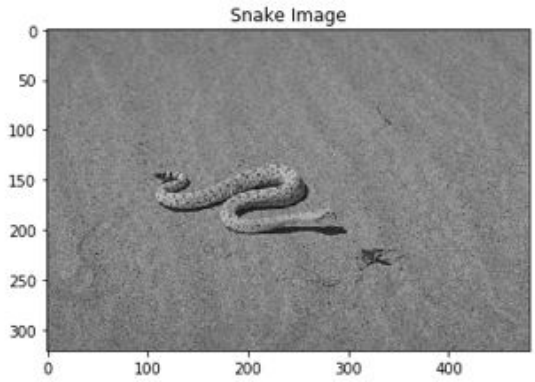
Note that the number of dendrograms with n leafs = $(2n-3)!/[(2^{(n-2)})\,(n-2)!]$

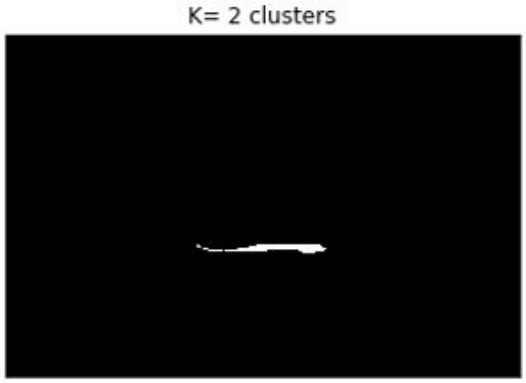
In each image set, as with k means, each pixel is labelled with respect to the number of clusters and shown below:

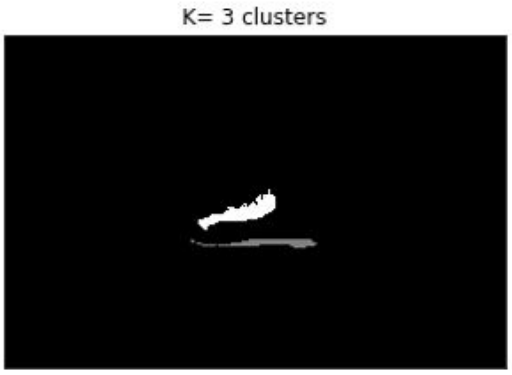
Edward Mina ML Exam 2

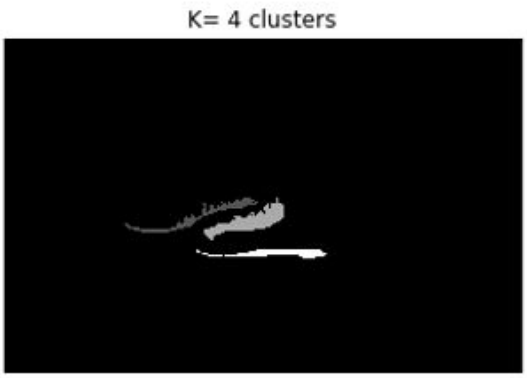




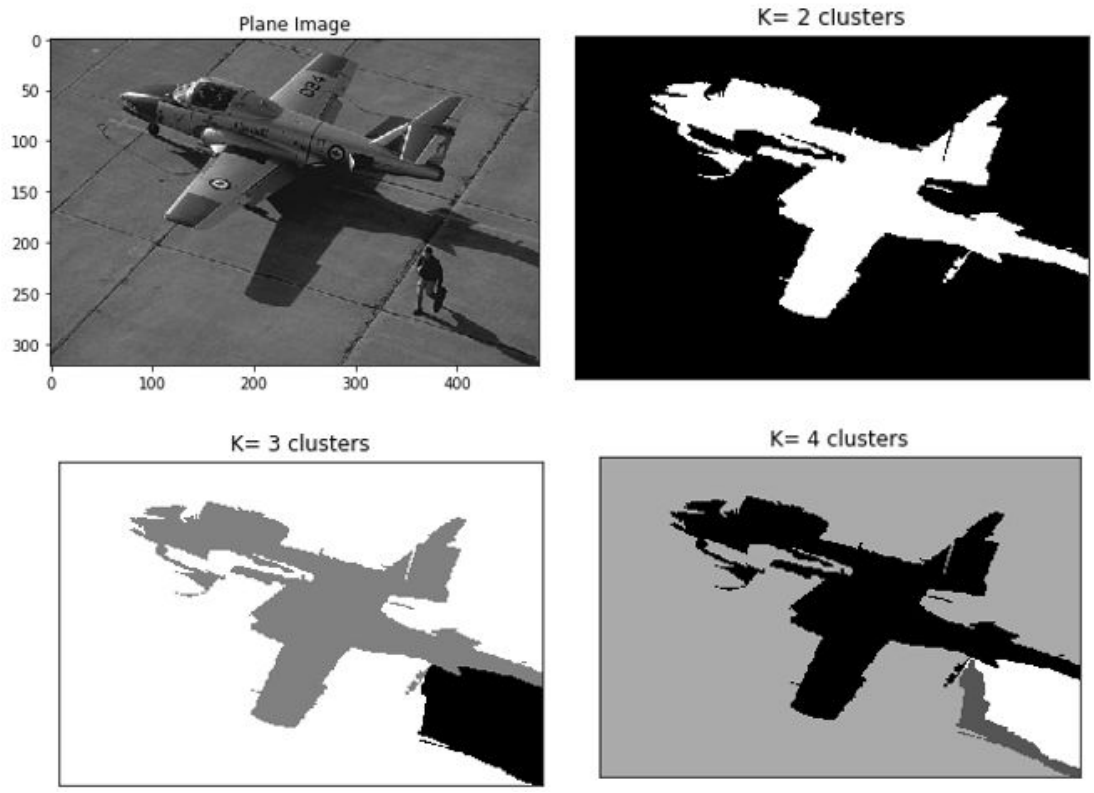








Edward Mina ML Exam 2



Question Discussion: Looking at the 3 image groups being classified, a Bird, Snake, and a Plane, the direct result is that for these images, hierarchical clustering has worked as a better segmenter. Images produced by the K-means technique are blurred and not clearly separated in identifiable clusters. For instance the first image of a Bird has three clear identifiable clusters to the human eye: the bird, the background, and the branch. The K- means algorithm struggles with finding these clusters irrespective on the user parameters k and outputs mixed segments. On the other hand, the clustering method approximates the clusters much better as can be seen with k=3 and 4. Since ward clustering accounts for sub and superclusters (in dendrogram structure), this may explain why the model works better since k means does not consider cluster structure. Independent of the method used however, each image improves in cluster and image formation as the number of user defined clusters increases.

Problem 2:
Dataset Previews:
*Note each section is not the whole data set but only snippet.

Dataset 1: Classification of Cancer as Benign or Malignant given 11 Features about Cell Anatomy
Classes= [B,M]
Size= 569 x 11

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

Dataset 2: Binary Classification of Glass Type Given 8 Features of What Amount of a Particular Element is Found
Class/Type= [1,2]

Edward Mina ML Exam 2

Size= 146 x 10

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.00	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.00	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.00	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.00	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.00	1
5	1.51596	12.79	3.61	1.62	72.97	0.64	8.07	0.00	0.26	1

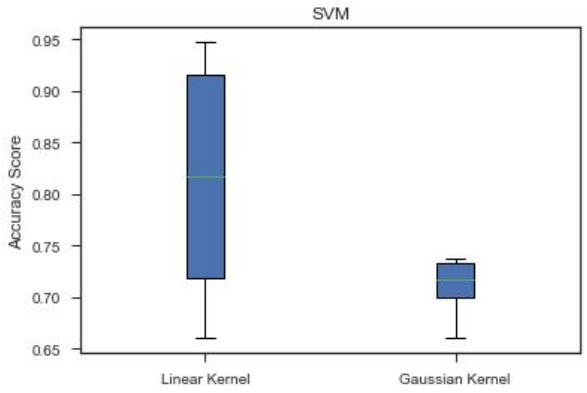
Dataset 3: Iris Classification for only 2 of the 4 different Class (Setosa and Versicolour) given 4 features on Pedal and Sepal Anatomy
Classes= [1,2]
Size= 100 x 5

	Sepal Length	Sepal Width	Petal Length	Petal Width	Class
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

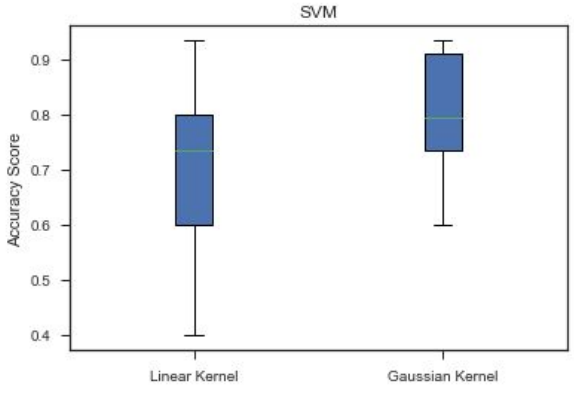
Accuracy scores are developed from Pythons SVM Sklearn Machine Learning Library.
Support Vector Machine Scores per Model

	Model 1: Linear SVM	Model 2: Radial Basis Function (Gaussian Kernel)
Dataset 1: Benign or Malignant Cancer Classification	81.27%	72.57%
Dataset 2: Glass Type (2 Classes)	71.04%	77.66%
Dataset 3: Flower Type (2 Classes)	96.49%	97.00%

Model Box and Whisker Plot of Data as per type of SVM Model.

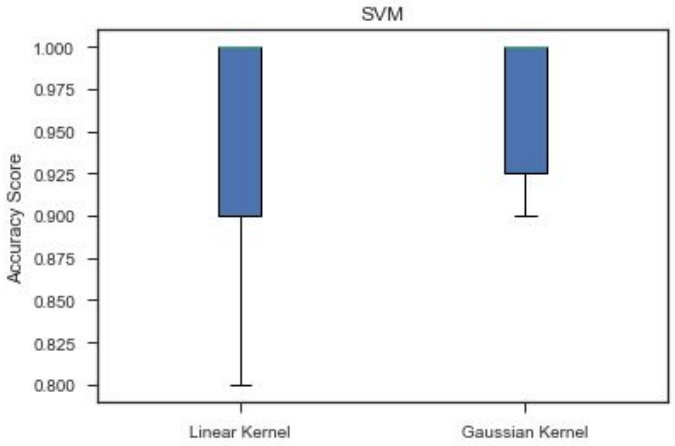


Dataset 1 Cross Validation Scores per Model



Dataset 2 Cross Validation Scores per Model

Edward Mina ML Exam 2



Dataset 3 Cross Validation Scores per Model

Implementation and Discussion:
The above figures arrive from 3 separate datasets with two different classes available for binary classification. The datasets are as follows: breast cancer data which can be classified as B (“Benign”) or M (“Malignant”) given 11 features on cell anatomy; glass class 1 or class 2 (not specified in data) where each class is based on the mixture of elements provides as the 8 feature vectors; iris classification of 1 (“Setosa”) or 2 (‘Versicolour”) given sepal and petal features. This binary class sets were classified using support vector machine/classifier under a k=10 fold cross validation.

Support Vector Machine is a popular machine learning technique that separates data such that the distance between the boundary and the nearest point is maximized. This can be interpreted as having the “widest street” possible around a boundary and the clusters surrounding points. To compute this maxima, the lagrangian of the boundary width at its associate conditions is computed. What is learned is that the maximization of the width of the sample vector is directly dependent on the dot product of pairs of samples. This is important to note because of the incases where it is difficult to to introduce a separating hyperplane, the data points can be transformed into another space. This is done by some similarity kernel function K(x,y) where K solves for the dot product of transformed x and y vectors. The two most popular kernels are those that are used in the questions are linear and gaussian kernels (shown below).

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$

1. Linear Kernel function

$$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\frac{\left\| \mathrm{x}_i - \mathrm{x}_j \right\|^2}{2\sigma^2} \right)$$

2.) Gaussian Kernel Function

The linear kernel function behave well on “fatter” data sets where there are many features relative to the number of samples. This kernel isn’t considered a kernel since it is simply the dot product, however it generally avoids overfitting by using a simple separating boundary function. The Gaussian kernel, also known as Radial Basis Function, does better with “skinnier” data sets with fewer features since it is constrained by a variance term that risks overfitting. With these different SVM kernel functions, a k=10 cross fold validation was done. Cross validation simply means splitting the data set into 10 different paired training and testing folds and training/testing the model with each fold iteration. Cross validation can be expanded in to a combination of k and k-1 validations which is constructed using 2 for loops. The first loop separates the data set in to 10 (k) folds with 9 (k-1) training folds) and 1 validation fold. These 9 folds are then taken into another loop (k-1 validation) which cross validates that section of the data until the optimal parameters are computed. In code this is the equivalent of seeing what parameters in SVM (kernel and C) output the max score. Note that C, the penalty parameter was kept constant for the sake of simplicity. With these optimal

parameters, a the outer loop trains the model in a k-fold validation. Cross validation is an excellent technique into optimally training your data, and decrease the risk of over defining/fitting.

Simple K folds:

1. Fold Length = (length of data)/k (Split the data into k folds of a certain fold length)
2. For 0 to k (From 0 to the number of Folds)
    a. While length of some empty vector named fold < Fold Length (Go through all empty spots in fold vector)
        i. Index = random integer from 0 to (length of data)
        ii. Fold = vector appending value of dataset(index) (fill the fold vector with random data points from the whole dataset)
    b. Move onto the next fold

With the first data set, the gaussian kernels seems to be overfitting the dataset a points giving it a less than 10% accuracy than the linear kernel. Likewise, the linear kernel is has accuracy scored that range upward of 93% (mean of 80%) compared to the max value of 74% in the Gaussian kernel. However, for both datasets 2 and 3 (having fewer samples), the Gaussian kernel performs better with similar variance in each of the k=10 fold validation. This is expected since Gaussian kernels generally perform better than linear kernels.