Group Project: Skin Cancer Diagnostics

Bahman Sheikh, NetID: bahmans2

• Project description and summary

In this project my goal is to construct classification models to identify malignant moles based on images. To do so first I need to extract good features for each class or images. We have 150 labeled images for malignant moles and 150 labeled images for benign images. In the first part of the project after trying different methods I decided to use a simple color statistics feature. For this feature I calculated the mean and standard division of pixels for each red, blue and green channel of images, therefore, I have 6 features for each image. I trained different regression models and Logistic Regression showed the best accuracy (59%). In the second part of the project by reviewing available literatures I found out that Multiple Colors, Border-Shape and Asymmetry are important characteristic of the malignant moles. Based on that I calculated the Hu Moments, Haralick Textures and Color Histogram for each image as new features. These features not only improved the accuracy of the classification models they are very well interpretable and can be related to the moles' Border-Shape and Asymmetry and Multiple Colors, respectively. Using these features Support vector machine (SVM) showed the best accuracy (68%). I used Python programming language to perform my project.

• Data processing

The first step in my project is to process and prepare the data so that it can be analyzed with statistical machine learning algorithms. After reviewing different projects and websites I realized that the most machine learning algorithms (except deep learnings), perform relatively weak on just images pixels. Therefore, according to suggestion of different image analysis I decided to utilize image feature extraction techniques to characterize the contents of the images. So, the steps I took are as follow for question1, and for question2 described later:

○ RGB color channels

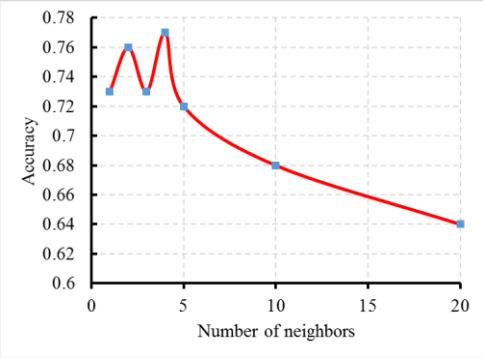
Color Channel Statistics, this feature extraction technique is used to answer mainly the first question. After reviewing available technique in image analysis I realized that statistical classification of images based on RGB channels of image is a primarily, simple and relatively efficient technique. So, to measure the color of the image I extracted the pixel values for each channel of red, green and blue and calculated the mean and standard deviation for each color channel in the image. Therefore, for each image I have 6 features mean and standard deviation of pixels for each Red, Green, and Blue channels. To perform this task I used ***Image*** class from ***PIL*** library and calculated mean and standard division for each channel using ***numpy*** library.

• Regression model analysis

- **KNeighbors**

To train this model I used *KNeighborsClassifier* from *sklearn* library. This model is relatively computationally costly but the good thing is the model just has one parameters to be tuned which is the number of neighbors (*n_neighbors*). To tune the model I only considered the RGB channel pixels feature (Mean and standard deviation of each color) and calculate the accuracy

of the model for different *n_neighbors*. According to the following graph *n_neighbors* = 4 is the optimum value for our KNeighbors model.



- **Random Forest**

To train this model I used *RandomForestClassifier* from *sklearn* library. A random forest fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve accuracy. In this model I used *bootstrap=True*. The model has two input parameters namely *n_estimators* and *max_depth*. To tune the model I only considered the RGB channel pixels feature (Mean and standard deviation of each color) and calculate the accuracy of the model for different *n_estimators*, *max_depth* the results are as following table, according to the table *n_estimators = 100*, *max_dep = 1000* shows the best accuracy.

n_estimator	max_depth	Accuracy
100	default	0.71
10	default	0.71
10	1000	0.74
10	2000	0.68
100	1000	0.76
200	1000	0.66
50	1000	0.76
50	100	0.68

- **Support Vector Machine (SVM)**

To train this model I used *svc* from *sklearn* library. I set the cost value as default and tried different kernel as shown in the below table. As it can be seen SVM with a poly kernel degree 3 shows the highest accuracy.

kernel	degree	Accuracy
linear	1	0.75
poly	2	0.61
poly	3	0.68
sigmoid	1	0.61

- **Logistic Regression**

To train this model I used *LogisticRegression* from *sklearn* library. I employed the following options to tune the model using *multi_class="auto"* :

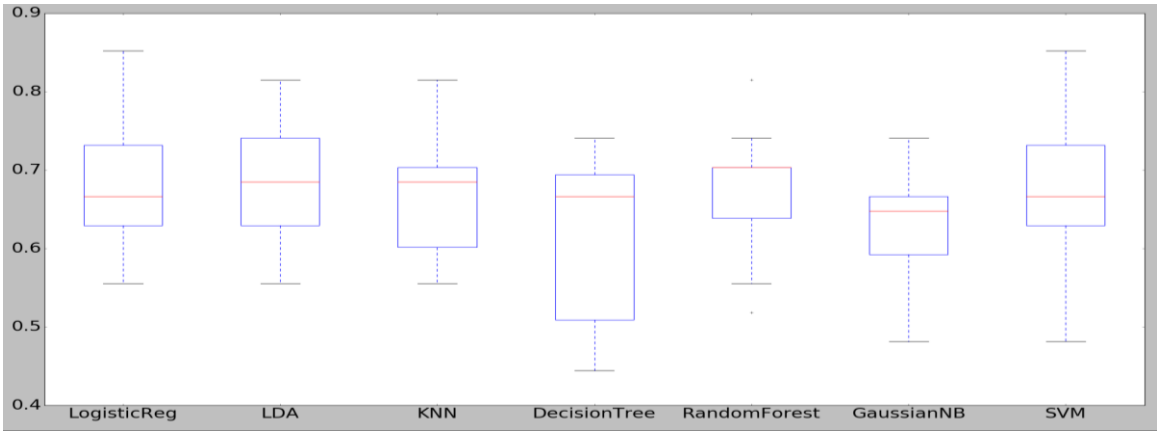
Logistic Regression	Solver	Accuracy
l1 penalty	saga	0.65
l1 penalty	liblinear	0.53
l2 penalty	lbfgs	0.64
without penalty	lbfgs	0.67

- **Linear Discriminant Analysis (LDA)**

To train this model I used *LinearDiscriminantAnalysis* from *sklearn.discriminant_analysis* library. '*svd*', '*lsqr*' and '*eigen*' solvers with and without *shrinkage.* All the solvers have the similar accuracy.

- **Results**

Accuracy with **Color Channel Statistics** feature (RGB pixel) 10-fold validation **train** data.



Model	Test data accuracy
Logistic Regression	0.59
LDA	0.52
KNN	0.55
Decision Tree	0.56
Random Forest	0.53
GaussianNB	0.58
SVM	0.53

As it can be seen Logistic Regression has the best accuracy on the test data. The reason can be the feature vector is relatively short so a simple model like Logistic Regression performed very well.

- **Question 2:**
- **Literature review**

According to "*Melanoma Warning Signs"* article from "*Skin Cancer Foundation"* the **ABCDE**s signs and the **Ugly Duckling** sign can help us to detect melanoma. I briefly explain them in the following ***(Allan C. Halpern, Ashfaq A. Marghoob, Ofer Reiter, (2019). "Melanoma Warning Signs", Skin Cancer Foundation, www.skincancer.org.)***:

***The ABCDEs of melanoma*: A is for Asymmetry**: Most melanomas are asymmetrical. **B is for Border**: Melanoma borders tend to be uneven and may have scalloped or notched edges, while common moles tend to have smoother, more even borders. **C is for Color**: Multiple

colors are a warning sign. While benign moles are usually a single shade of brown, a melanoma may have different shades of brown, tan or black. As it grows, the colors red, white or blue may also appear. **D is for Diameter or Dark**: it's a warning sign if a melanoma is the size of a pencil eraser or larger. **E is for Evolving**: Any change in size, shape, color or elevation of a melanomas may be a warning sign of melanoma.

***The Ugly Duckling*:** Most normal moles are resemble one another, while melanomas stand out like ugly ducklings in comparison. These ugly duckling melanomas or outlier lesions can be larger, smaller, and/or colorful, compared to surrounding moles. Also, isolated lesions without any surrounding moles for comparison are considered ugly ducklings.

- **Feature engineering**

Two features are extremely important to classify Benign and Malignant images: first is the **Multiple Colors** in the mole and the second is the **Border-Shape-Asymmetry** of the mole. I considered the color statistics from in red, blue and green channel in the previous pixel based analysis. In this section to take into consideration the **Border-Shape-Asymmetry** of the image as features **(**A, B features from the above literature) I considered to calculate the Hu Moments and Haralick Textures of the images as new features, also to consider the **Multiple Colors** characteristics of malignant moles I considered to calculate the Color Histogram of the images as a feature in addition to the color pixel statistics. To do so I followed the following steps:

- **Re-Sizing**

After trying different image feature selections, I realized (as also mentioned in the literature) for the following features the performance of the machine learning algorithm increases if each image has the same size, given that for some features it is necessary to have the exact same size. So, first I resized all of the images (benign and malignant) into a fixed size of 600 × 600 pixels. To do so I utilized ***cv2.resize*** function from ***cv2*** library (***OpenCV***).

- **Color Histogram**

Color histogram is a representation of the distribution of colors in an image that quantifies color of the image. To calculate the Color Histogram feature for each image, I used ***cv2.calcHist()*** function from ***cv2*** library (***OpenCV***). The calculated histogram for each image then is normalized and flattened by ***normalize()*** and ***flatten()*** functions and utilized as an image feature.

- **Gray color scale**

To extract the following two image features (Hu Moments and Haralick Textures) the images should be converted to gray color, to do so I used ***cv2.cvtColor*** and ***cv2.COLOR_BGR2GRAY*** from **OpenCV**.

- **Hu Moments**

After resizing all images to a fix pixels size I calculated the Hu-Moments of images as a feature. This feature for each image calculates a set of 7 numbers using central moments that are invariant to translation, scale, and rotation of images, therefore, able to quantify shape of the images. To calculate the H-Moments from each image I used ***cv2.HuMoments()*** function rom ***cv2*** library (***OpenCV***).

- **Haralick Textures**

This feature quantifies texture of images. Image texture is a quantification of the spatial variation of grey tone values. Haralick et al. (1973) suggested the use of gray level co-occurrence matrices (GLCM) (WikiPedia). This method is based on the joint probability distributions of pairs of pixels. To calculate Haralick Texture for each image after converting to gray scale, I used ***features.haralick()*** from ***mahotas*** library.

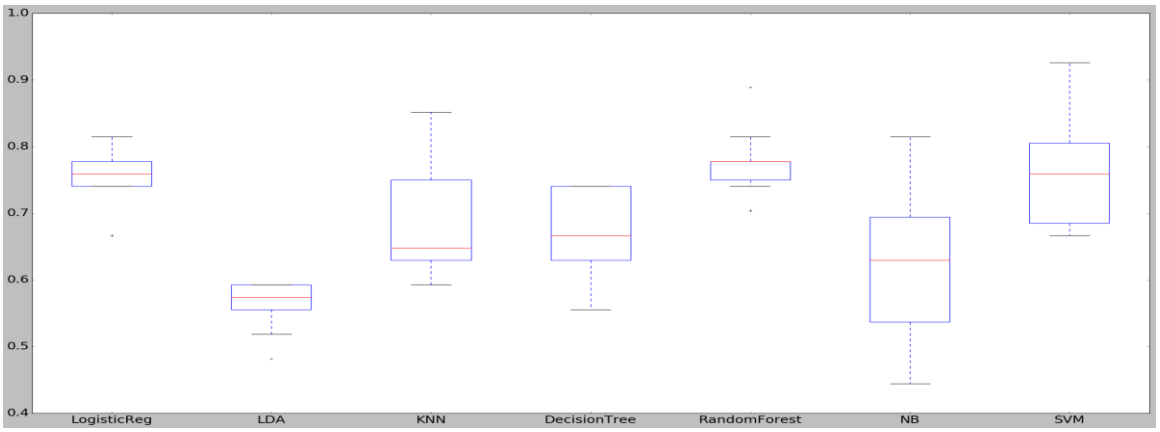
- **Scaling**

I decided to examine the goodness of all of the above features individually and then all the feature together. As using all of the features together may cause one feature dominates the other with respect to its value they should be rescaled and normalized. So I normalized all of the features using ***MinMaxScaler()*** function from ***scikit-learn*** library.

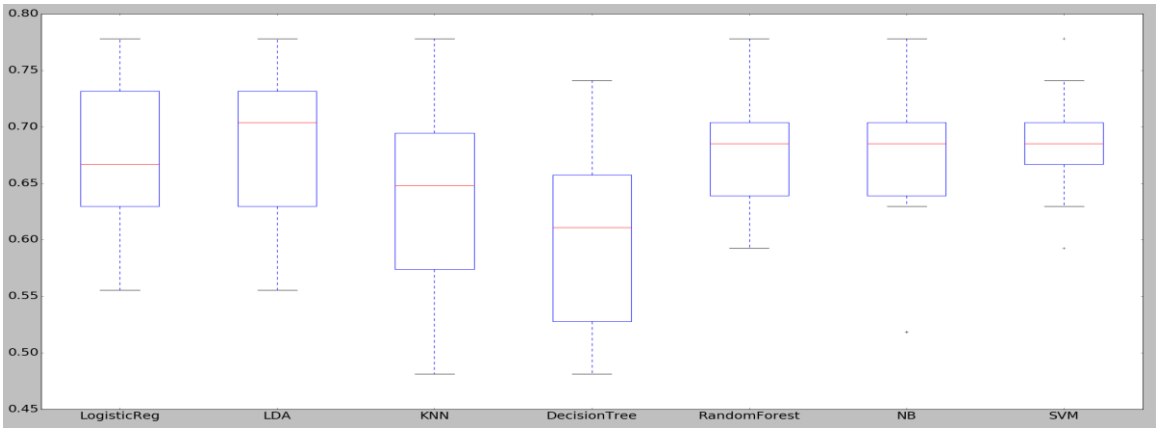
- **Results**

First, I showed the goodness of the each feature individually to predict the train data with 10-fold cross validation using different regression models. Second I used all the feature together to predict the test data accuracy. The procedure to tune the models was similar to the procedure described in the question 1 part so I don't repeat it here again.

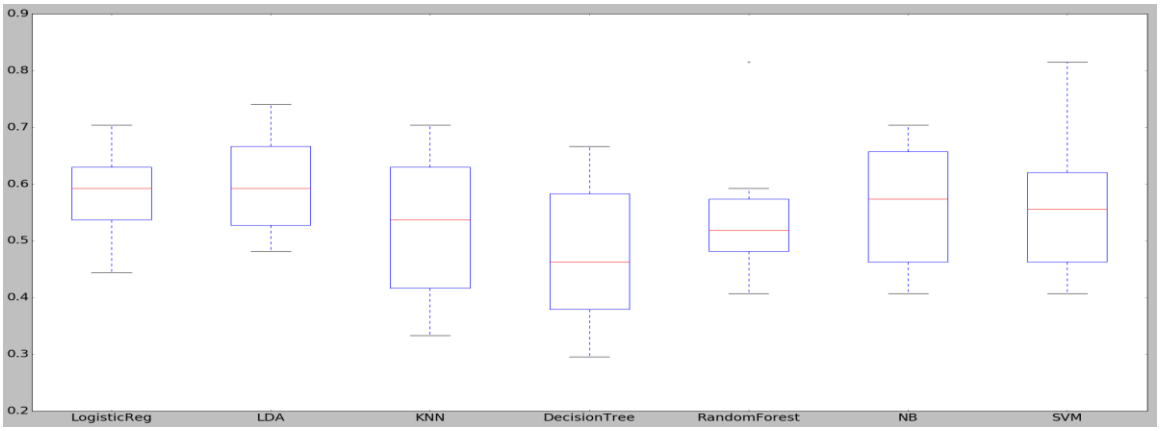
Accuracy with **Color Histogram** feature 10-fold validation **train** data



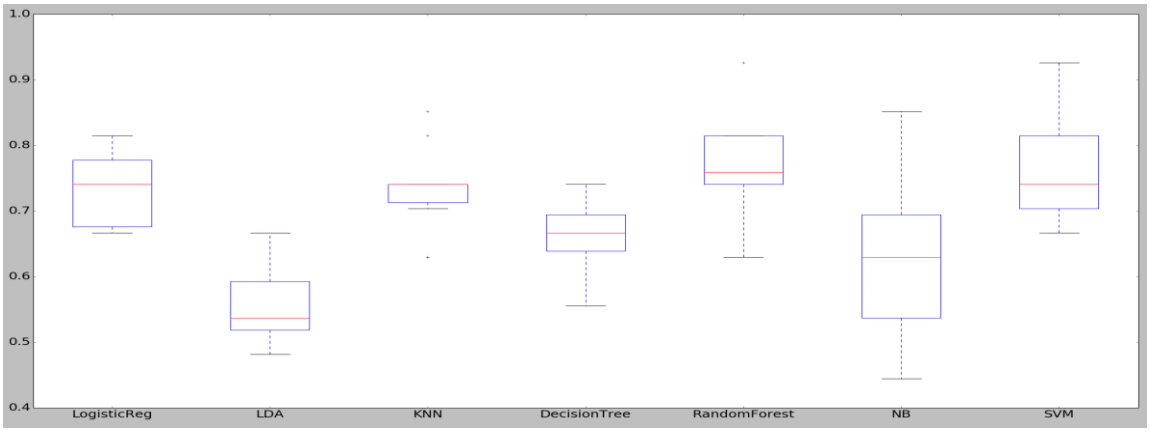
Accuracy with **Haralick Textures** feature 10-fold validation **train** data



Accuracy with **Hu Moments** feature 10-fold validation **train** data



Accuracy with all features 10-fold validation **train** data



Model	Test data accuracy
Logistic Regression	0.64
LDA	0.54
KNN	0.63
Decision Tree	0.64
Random Forest	0.64
Gaussian NB	0.66
SVM	0.68

According to the above results it can be seen **Hu Moments** and **Color Histogram** showed a significant improvement to predict the Malignant images. The main reason can be related to the discussion of the cited literature. As discussed, the Melanoma moles borders tend to be uneven and asymmetry, these characteristics are interpretable and can be captured perfectly with **Hu Moments**. Also, while benign moles are usually a single shade of brown, a melanoma may have different shades of brown, tan or black, and this characteristics also are well understandable/interpretable and can be captured perfectly with **Color Histogram**. As it can be seen using all the features together more advanced models such as SVM showed a better accuracy in compare to using just the color statistics feature in the question 1 where simple Logistic Regression shows a better accuracy.