# STAT 542 / CS 598: Group Project: Skin Cancer Diagnostics

**Ashwin Amrutphale (ashwina4) (Team Lead)**
**Pratik Patwari (patwari3)**
**Rohit Bansal (rbansal3)**

# 1. Project Description and Summary

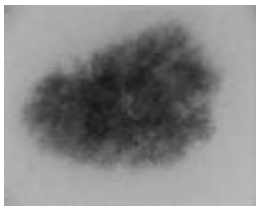
## 1.1 Goal

Construct classification models for identifying malignant moles based on the pixels (RGB colors) of these images and propose new data processing/feature engineering approaches that improve the interpretability and model accuracy. This project compares pixel-based approach with feature-based approach for the image and compare results for the same.

## 1.2 Approach

In the first pixel-based approach, we have used EBIMAGE library in r to read the image and then resize the image into 100 x 100 pixel for resampling and for reducing channels. For the image classification, 3 classifier models were developed using these pixels as features which were 1. **Random Forest** 2. **KNN** 3. **SVM**

**Original image**



**After processing**

In 2nd approach, feature based extraction was used. We extracted ABCD features. **ABCD** (Asymmetry, Border Irregularity, Color, Diameter) **features** are features that dermatologists commonly use. The significance of these features is that, based on them, a dermatologist decides whether an observed skin lesion is a malignant or not. Once these features were extracted, we used these features for image classification. We used two models which were **1) Logistic Regression 2) Classification Tree.**

## 1.3 Results

As shown below, feature based classification has more accuracy as compared to 1st approach

**Approach 1 (Pixel Based):**

| Classification Model | Accuracy |
|---|---|
| Random Forest | 72.2% |
| KNN | 72.2% |
| SVM | 73.3% |

**Approach 2 (Features Based):**

| Classification Model | Accuracy |
|---|---|
| Logistic Regression | 63.33 |
| Classification Tree | 68.89 |

# 2. Data Processing for Question 1

EBIMAGE library was used to perform Data Processing and Analysis on the benign and malignant images. Images were resized to 100x100 and turn them into greyscale so that we can load them into R easily and reduce training time**.** Each image was turned into a vector of length 30K, with each element representing the value in a pixel. The values of the pixels from the resized and greyscale images where then stored into an array. The benign class was labeled as 0 whereas malignant class was labelled 1.

```
lable <- c(rep(0,150),rep(1,150))
imageData <- rbind(benign_data,malignant_data)
```

This data was then randomly split into a training and test set in the ratio 70:30 respectively.

We also tried to get the patterns in the image using PCA and use those patterns as the features. For that we generated more data using rotation and flipping of the images. After doing so, we did not see any significant improvement for our prediction and model interpretability. Thus, we decided to use each pixel value as a feature. As a result, we have 30K features for each image.

```
extract_feature <- function (dir_path, width, height, is_benign = TRUE, add_label = FALSE) {
  img_size <- width*height
  ## List images in path
  images_names <- list.files(dir_path)
  ## This function will resize an image, turn it into greyscale
  feature_list <- pblapply(images_names, function(imgname) {
    img <- readImage(file.path(dir_path, imgname))      ## Read image
    img_resized <- resize(img, w = width, h = height)      ## Resize image
    dim(img_resized)
    img_vector <- imageData(img_resized)
    return(img_vector)
  })
  feature_matrix <- do.call(rbind, feature_list)    ## bind the list of vector into matrix
  feature_matrix <- as.data.frame(feature_matrix)
  names(feature_matrix) <- paste0("pixel", c(1:img_size)) ## Set names
  return(feature_matrix)
}
```

# 3. Classification Models Based on Pixels

After data processing was done, next step is to process the classification model on pixels being used as features and to calculate the accuracy of the models. Three classifiers were then chosen: K- Nearest Neighbor (KNN), Random Forest and Support Vector Machines (SVM).

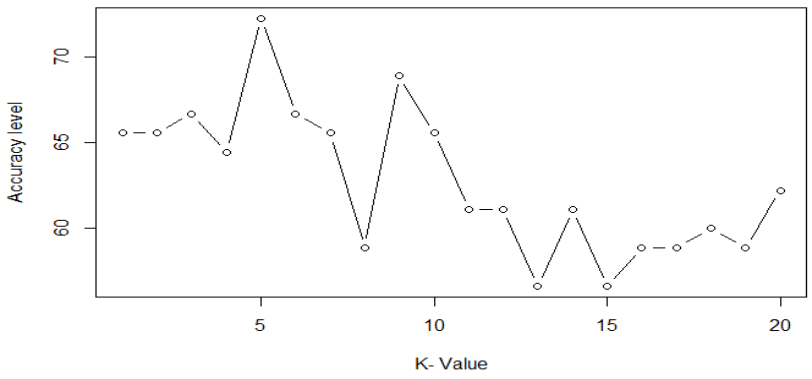
## KNN

KNN classification was used as the 1st classification model and it was fitted with transformed train data and test data. The performance of the model was measured on accuracy, recall, f-1 score and precision. For KNN algorithm, the tuning parameters are ‘k’ value and number of ’features/attributes selection. We have used k

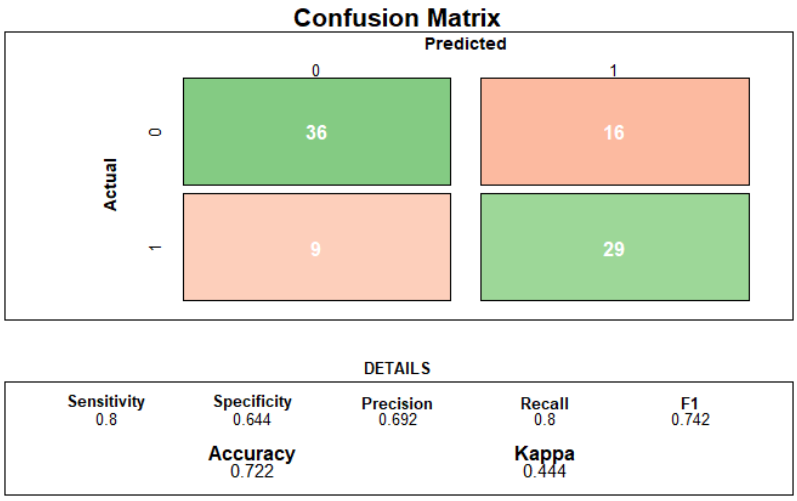
1:20 to find perform model tuning. We found that at K=5 we get the best performance. Its performance was 72% accuracy, 0.8 recall, 0.742 f-1 score and 0.692 precision.

```
library(class)
i=1                          # declaration to initiate for loop
optm=c(1:20)
model = c(1:20)
# running model from k = 1 to 20
for (i in 1:20)
{
  knn.model = knn(train=train_images, test=test_images, cl=as.factor(train_labels), k=i)
  optm[i] = (100 * sum(as.factor(test_labels) == knn.model)/NROW(test_labels))
  cat(i,'=',optm[i],'\n')       # to print % accuracy
  if (i == 1)
  {
    optm_max = optm[i] # find the max accuracy out of 20
  }
  if(optm[i] > optm_max && i>1)
  {
     model_max = knn.model # find the max accuracy out of 20
     optm_max = optm[i]
  }
}
# Plots
plot(optm, type="b", xlab="K- Value",ylab="Accuracy level")  # to plot % accuracy wrt to k-value
t = confusionMatrix(model_max, as.factor(test_labels))
draw_confusion_matrix(t)
```

As shown in below chart, at k=5, maximum accuracy achieved which is 72.2%, after that, it seems increasing K increases the classification but reduces success rate.



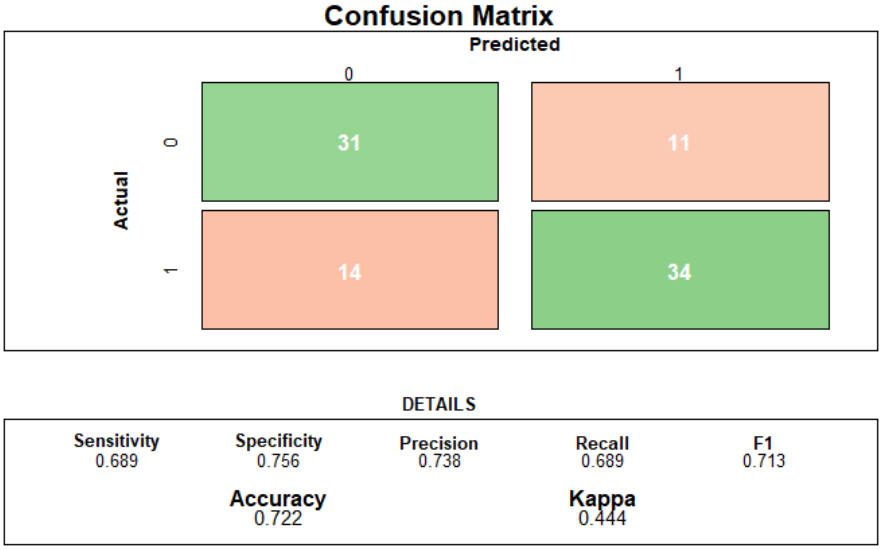

Confusion matrix was created to describe the performance of a classification model. it predicted benign correctly 36 times and malignant correctly 29 times. It misclassified benign as malignant 16 times and vice versa 9 times.




## Random Forest

Radom forest has been used as seconds classification model. it was also fitted with the train set to predict the test set. For Random forest, the tuning parameters ntree and random state was used to tune the mode. We used ntree =500 for random forest.

```
library(randomForest)

library(ggplot2)

rf <- randomForest(x = train_images, y = as.factor(train_labels)

                   , xtest=test_images, ytest=as.factor(test_labels),keep.forest=TRUE,ntree = 500, rando
m_state = 0)

print(paste("Random Forest Accuracy:", (sum(ifelse(rf$test$predicted == as.factor(test_labels),1,0)) / l
ength(rf$test$predicted)) * 100))

# Plots

t = confusionMatrix(rf$test$predicted, as.factor(test_labels))

draw_confusion_matrix(t)
```




As described above, the performance of the model was measured on accuracy, recall, f-1 score and precision. Its performance was 72.2% accuracy, 0.689 recall, 0.713 f-1 score and 0.738 precision.

Confusion matrix was created to describe the performance of a classification model on a set of test data.  it predicted benign correctly 31 times and malignant correctly 34 times. It misclassified benign as malignant 11 times and vice versa 14 times.

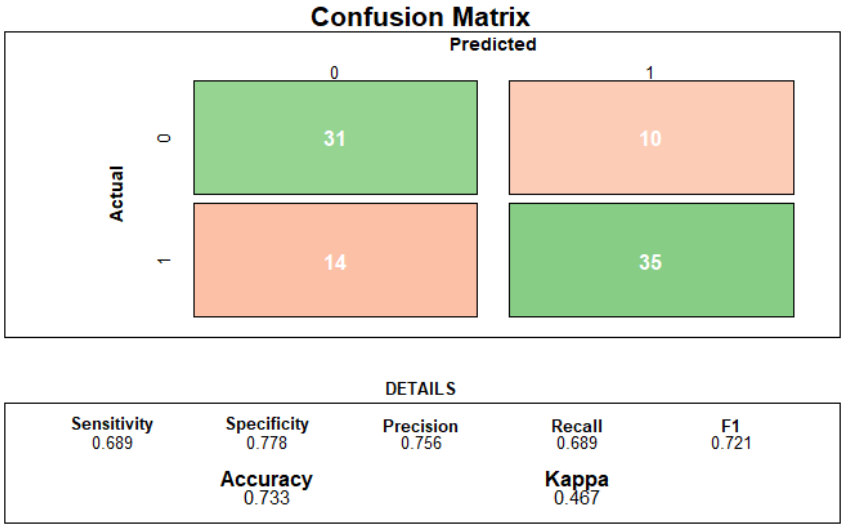
## SVM

SVM has been used as third classification model. It also used pixel values as features on train data to fit a model and then used test data to predict.

```
library(e1071)

svm.model <- svm(x=train_images,y=as.factor(train_labels),kernel = 'radial', type = 'C-classification',
cost=10, scale=FALSE)

y_pred = predict(svm.model, newdata = test_images)

# Plots

t = confusionMatrix(y_pred, as.factor(test_labels))

draw_confusion_matrix(t)
```

The performance of the model was also measured on accuracy, recall, f-1 score and precision. For Random forest, the tuning parameters ntree and random state was used to tune the mode. Its performance was 73.3% accuracy, 0.689 recall, 0.721 f-1 score and 0.756 precision

Confusion matrix was created to describe the performance of a classification model**.**  it predicted benign correctly 31 times and malignant correctly 35 times. It misclassified benign as malignant 10 times and vice versa 14 times.



## Interpretations

Below table outlines the summary of classifier on Accuracy, Precision, Recall and F-1 Score. SVM performed best among all the model tested while KNN and Random Forest were mostly equal in performance.

| Classifier | Accuracy % | Precision | Recall | F-1 Score |
|---|---|---|---|---|
| **KNN** | 72.2% | 0.692 | 0.8 | 0.742 |
| **Random Forest** | 72.2% | 0.738 | 0.689 | 0.713 |
| **SVM** | 73.3% | 0.756 | 0.689 | 0.721 |

Based on the confusion matrix as well, we can say that SVM is best predictor for benign as it predicted correctly 35 times and KNN is best predictor for malignant as it predicated correctly 36 times.

There were two major observations. First, we found that the pattern extraction using PCA does not help much in this case. All the classification models are able to perform better by treating each pixel as an individual feature. This is in contrast to the theoretical understanding as PCA helps in identifying the patterns. Theoretically, we intend to find the different pattern library in the images and then want to use those as features and make prediction based on that. The reason of such a behavior is that we have very limited data (150 images of each type). Generating more data by flipping and rotations does not help in this scenario as pattern library remains pretty small.

Second, the model can be fine tune based on the intention of the study. For example, we might want to avoid false negative compare to false positive as in the case of false negative the patient went undiagnosed. At the same time declaring someone having cancer while he is not (false positive) is equally bad. Thus, the model tuning depends on the use case. There might be a use case where we are ok, with false positive as there exist more sophisticated medical exams. In such a scenario model will be tuned differently then the way we have done. We have not given any preference to any scenario over other i.e. False Positive and False Negative are considered equally bad.

# 4. New Feature Engineering Approaches

## 4.1 Literature review

To convince a medical doctor we can't go with image pixel approach. Even pattern identified via PCA will not work. We have to use some features which are used in medical field. We did some study and found that dermatologist use a system called **ABCDE system.** In this system, they identify few attributes of the scar/mole and based on that they take their decision. So, we tried to implement the same using our classification models.

**ABCDE System**

For clinical presentation most of the dermatologist use ABCDE system. In this system, we have features based on 5 attributes namely – **A**symmetry, **B**order Irregularity, **C**olor, **D**iameter, and **E**volution. Based on the picture of the mole we can extract the first 4 attributes and can use them as features. Based on these features, a dermatologist decides whether an observed skin lesion is a benign or not. The ABCDE system followed by the doctors has been mathematically modeled and implemented in this project. The details of these attributes are as follows:

**Asymmetry:** It is an important feature for detection of skin cancer. It is represented in terms of symmetry Index. In order to find this feature, it is necessary to find the true axis of symmetry around which the image of lesion area is folded. This was done by first finding the center of rotation of the image. The center of image is taken as the point placed in the middle row and middle column of a rectangular region that encompasses the image. The image is then folded at different axes, each passing through the center. A total of 18 axes at increments of10◦, for a total of 180◦, were tried. The true axis of symmetry is identified as the one at which the two halves, when folded, result in maximum overlap. In that situation the area of both the pieces will be same. We use this asymmetry as a feature to predict the nature of the mole. The more asymmetric the mole is higher is the chance of it being malignant.

**Border Irregularity:** It is the most important feature for the identification of benign vs malignant nature of mole is the shape of the lesion. A regular border is usually benign while an irregular border indicates malignant nature. To calculate regularity, we get the parameter of the mole and divide it by 4*pi*area. The resultant number provides the border irregularity coef. Higher the number more chances are of it being malignant.

**Color:** It is another important feature in identifying the nature of a mole is color. It can be identified by using the statistical features i.e., mean, median and standard deviation of each plane (red, green and blue). We are identifying 10 color cluster using K-Means and then we get the distance between those clusters. If the distance is more than 0.5 then we consider those are a different color cluster otherwise we consider it as a same color cluster. Based on the above phenomenon we get the total number of distinct color clusters. This is used as a feature. This tells us how many distinct colors cluster a mole has. The higher the number of clusters more chances are of it being malignant.

**Diameter:** We get the diameter of the mole by getting the average radius and multiplying it by 2. Note, diameter changes with distance between the capturing device and the lesion. Thus, for fair comparison, the distance between camera and the lesion should remain the same for all images in a dataset. We are assuming that the images provided to us have been captured using this precaution already.

**Evolution:** This is something we don't have as this is related to user information and can't be extracted from image.

## 4.2 Feature Engineering

We extracted **A**symmetry, **B**order Irregularity, **C**olor, **D**iameter using following algorithm:

**Asymmetry**

- Step1: Get the image shape and split it in half
- Step 2: Take the area of both the half and compare the difference. Scale it against total area of image.
- Step 3: Rotate the image by 10 degree and perform the step1 and step 2 again.
- Step 4: Repeat the step 3 18 times. Get the lowest scaled area difference. This will work as asymmetry coef. Lower this number more symmetric the image is.

```
img_resized <- resize(img, w = width, h = height)
grayimg <- channel(img_resized, "gray")
x = grayimg > otsu(grayimg)
asym = c(rep(0.0,18))
for (i in 1:18)
{
  x1 = rotate(x,i*10)
  imageshape <- computeFeatures.shape(x1)
```

```
    img1 <- x1[0:width/2,0:height]
    img2 <- x1[width/2:width,0:height]
    imgshape1 <- computeFeatures.shape(img1)
    imgshape2 <- computeFeatures.shape(img2)
    asym[i] <- abs(imgshape1[1]-imgshape2[1])/imageshape[1]
  }
  imageshape <- computeFeatures.shape(x)
  #split images in two parts
  a <- min(asym)
```

**Border Irregularity**

- Step1: Get the perimeter of the shape
- Step2: Divide it by 2*pi*r (where r is average radius). The resultant number will give border irregularity coef. Higher the number more irregular it is.

```
  b <- (imageshape[2]^2) / (4 * pi * imageshape[1])
```

**Color**

- Step1: Get the 10 different color cluster
- Step2: Get the distance between centroid of those clusters
- Step3: If the distance is more than 0.5 than treat it as different color cluster else consider it same
- Step4: Count all the different color cluster. This count will be used as feature

```
  lower=c(0, 0, 0)
  upper=c(0.1, 0.1, 0.1)
  kmeans01 <- colordistance::getKMeanColors(file.path(dir_path, imgname),
                                            lower = lower, upper = upper, n = 10,
                                            plotting = FALSE)
  kmeansDF <- colordistance::extractClusters(kmeans01)
  dis<- dist(as.matrix(kmeansDF[,!(names(kmeansDF) %in% c('Pct'))]), method = "euclidean")
  c <- sum(ifelse(dis[1:10]>0.5,1,0))
```

**Diameter**

- Step1: Get the average radius
- Step2: Multiply it by 2 to get the diameter

```
d <- imageshape[3] * 2
```

## 4.2 Classification Models

We used the extracted features **A**symmetry, **B**order Irregularity, **C**olor, **D**iameter in our classification models. For better interpretability and understanding we used **1) Logistic regression 2) Classification Tree.**

**Model1: Logistic Regression**

We used this model as we wanted to demonstrate each attributes significance in the classification model. We started with a large model with all the quadratic terms and then stepwise optimized it using AIC and BIC. We found that the best accuracy is achieved when we used linear full model.

```
lm1 = glm(label~., data = train_img, family=binomial)
x = ifelse(predict(lm1, newdata=test_img, type="response")>0.5,1,0)
```

```
sum(x==test_img$label)*100/90
```

Using linear full model, we got accuracy for **63.33%.** When we observed to coefficients here is what we found:

```
summary(lm1)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.858665   0.753564   1.139   0.2545
a            1.529772   1.215210   1.259   0.2081
b            0.168080   0.081187   2.070   0.0384 *
c            0.112672   0.087014   1.295   0.1954
d           -0.017614   0.006842  -2.574   0.0100 *
```

Based on this it can be observed that:

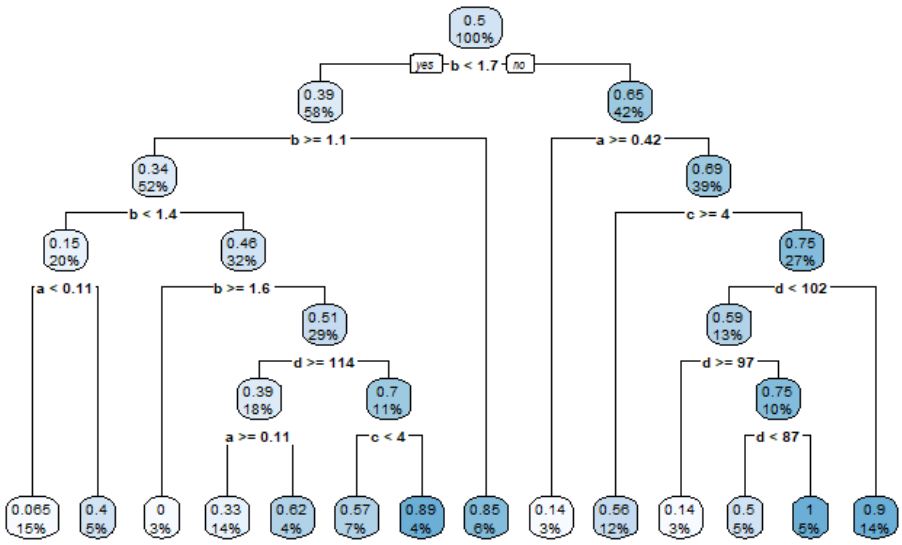
a) Diameter and border irregularity are the two most significant parameters.
b) The higher the asymmetry the more chances of the mole being malignant.
c) The higher the border irregularity the more chances of mole being malignant.
d) More Color cluster means more chances of mole being malignant.
e) Given all other parameter remains same the smaller diameter of mole makes it more dangerous

**Model1: Classification Tree**

The other most intuitive model for this problem was Classification Tree. Thus, we used that.

```
library("tree")
library(rpart)
library(rpart.plot)
trfit= rpart(label ~ ., data=train_img, control = list(maxdepth = 20))
trfit
rpart.plot(trfit)
```

We started with the model with the max depth for 20 and got following tree:



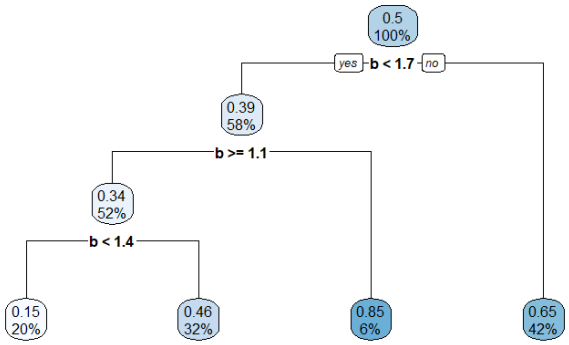
Based on the above tree we get following observation:

a) Border irregularity is the most significate attribute and the values (1.7, 1.1, 1.4. 1.6) are crucial decision points
b) Higher the Border irregularity more chances of the mole being malignant.
c) If border irregularity is high and image is not symmetric then more chances are that mole is malignant. Only exception is if number of color cluster are less and the diameter is also less than .97
d) The impact of number of colors cluster depends on the other parameters.

After this we identified the appropriate cp value which turned out to be 0.036923. Using the cp value, we pruned the tree and performed prediction on the test dataset. We got the accuracy of **68.89** which is higher than what we got using logistic regression.

```
printcp(trfit)
tr1 = prune(trfit, cp = 0.036923)
res = ifelse(predict(tr1, test_img,
       type = "vector",
       )>.5,1,0)

sum(res==test_img$label)/90
```

The pruned tree has following structure:




After pruning it suggests that only significant parameter is border irregularity. If it is between 1.1 to 1.4 then only there is a chance of mole being benign else it will be malignant.

# References


https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4739011/#B11

https://www.ncbi.nlm.nih.gov/books/NBK481857/