STAT 542 / CS 598: Project 2

Fall 2019, by Prathamesh(satpute3), Vivek(vivekg3) and Athul(as81)

Due: Monday, Dec 16 by 11:59 PM Pacific Time

[10 Points, half a page] Project description and summary.

Goal

Goal of this project is to identify malignent moles from benign moles. Provided image dataset has 150 benign and 150 malignent moles with varying resolutions. Some of the images has moles with hairs, markings on the microscope for measurement.

Approach

For Q1 we resized the images to 100x100 resolution and used that all pixel information to perform classification.

For the first model we choose Lasso with best lambda which provided minimum misclassification error by doing 10 fold cross-validation and using that model for predicting the results. For the second model we choose to do the PCA as it's an excellent choice when it comes to images. Using the selected principal components, we trained a SVM model. SVM Model is sensitive to parameters chosen. We used tune.svm() for trying different values of gamma and cost. We picked the gamma and cost which gives minimum classification error. For the third model we use PCA features for training Random Forest model. We tried different values of no. of trees and no. of nodes to get the best parameters for the random forest model.

For Q2, as a starting point, we began from the directions provided by the American Academy of Dermatology association, which targeted the ABCD of melanoma detection. Darrell Rigel (n.d. Link here). This provided us the intuition needed to generate the features. ABCDE is an acronym for Asymmetry, Border, Color, Diameter and Evolving respectively. We extracted features which correlate to the attributes like Area, Perimeter, Circularity index, Color in the segmented region. We used Ridge Regression and Random Forest Model on this transformed dataset to see whether it improved the accuracy.

Results

Table 1: Q1 Results

Model	Accuracy
Lasso	72.00%
SVM	70.00%
Random Forest	92.00%

Table 2: Q2 Results

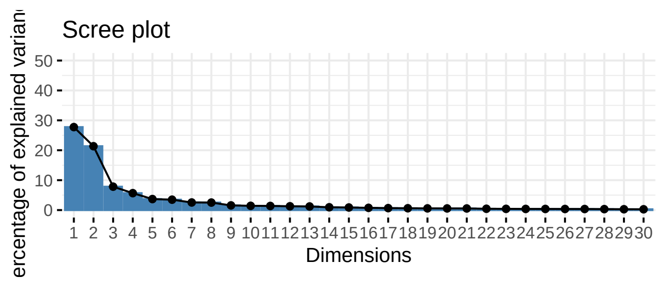
Model	Accuracy
Lasso	74.00%
Random Forest	92.00%

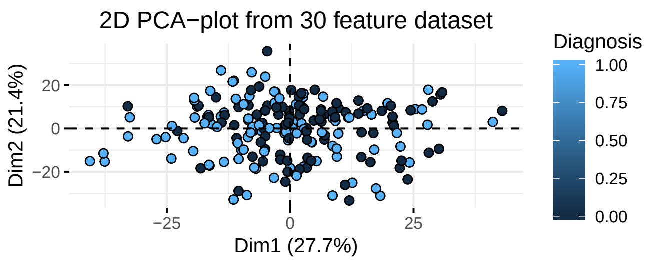
Write here some text analysing result - we have whole page for the point 1 - data processing starts on page 2 as per instructions provided

[5 Points, half a page] Data processing for Question 1.

We have used the EBImage package for processing and getting image data. We have crated the 2 matrix of 150x30000 each for benign and malignant files. We resize each image to 100x100 pixels resized the images to 100 x 100 resolution and then used this resize image data as vector. As this data consits of 3 channels we got maxis with 100x100x3 cols and 150 rows. We used all the pixel information for the lasso model.

We then choose to do the PCA as PCA is an excellent choice when it comes to images because inherently due to its natur, there is spatial corellation among pixel. Instead of using all the pixes, we can signifcantly reduce the number of features which encompass most of the variation. After plotting some information as shown below we choose 27 components which accounts for over 90% of the variation. We split our data in 70/30 train and test data split and perormed SVM and RF on it.



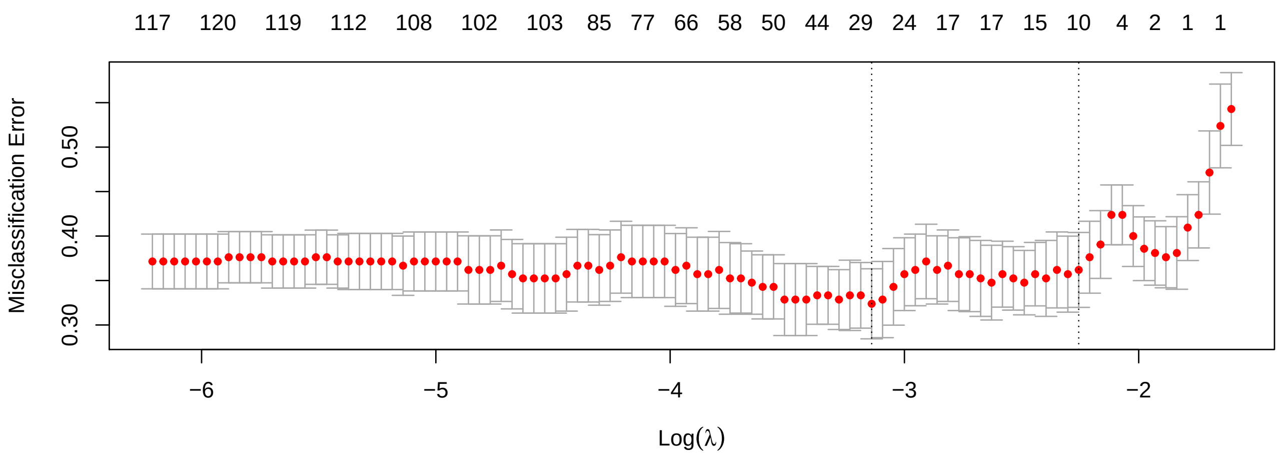


[30 Points, within 5 pages] Classification models based on pixels.

Classification Model 1. Lasso

For performing classification with the lasso we are selecting the family as binomial. We are using all the pixels which we got after resizing the image to 100x100 and by using 3 channels. As we have split the data in 70/30 ratio. The xtrain data is the 210x30000 matrix and ytrain is corrsponding 210 response. xtest is 90x30000 metrix and ytest is corrsponding 90 responses.

```
# use parallel for performace
registerDoMC(cores = 4)
cv_glmnet_model <- cv.glmnet(xtrain, ytrain, parallel = TRUE, alpha=1, nfolds = 10,
                             family="binomial", type.measure = "class")
# selecting the lambda within 1 standard error as best lambda
best_lambda = cv_glmnet_model$lambda.1se
# training with best lambda selected with 10 fold cross validation
glmnet_model <- glmnet(xtrain, ytrain, lambda = best_lambda, alpha=1,
                       family="binomial", type.measure = "class")
# predicting on test data
pred <- predict(glmnet_model, s = best_lambda, newx = xtest, type = "class")
accuracy = mean(ytest == pred) * 100
lasso_results = data.frame("Best lambda" = best_lambda, "Accuracy" = as.numeric(accuracy,3))
```



After performing the 10 fold validation and selecting the best lambda we retrained the model with the selected best lambda and family binomial and type.measure class and useed that model for predicting class on the test data. Accuracy we got is around 72%

Table 3: Lasso Results

Best.lambda	Accuracy
0.1047723	63.33333

Classification Model 2. SVM

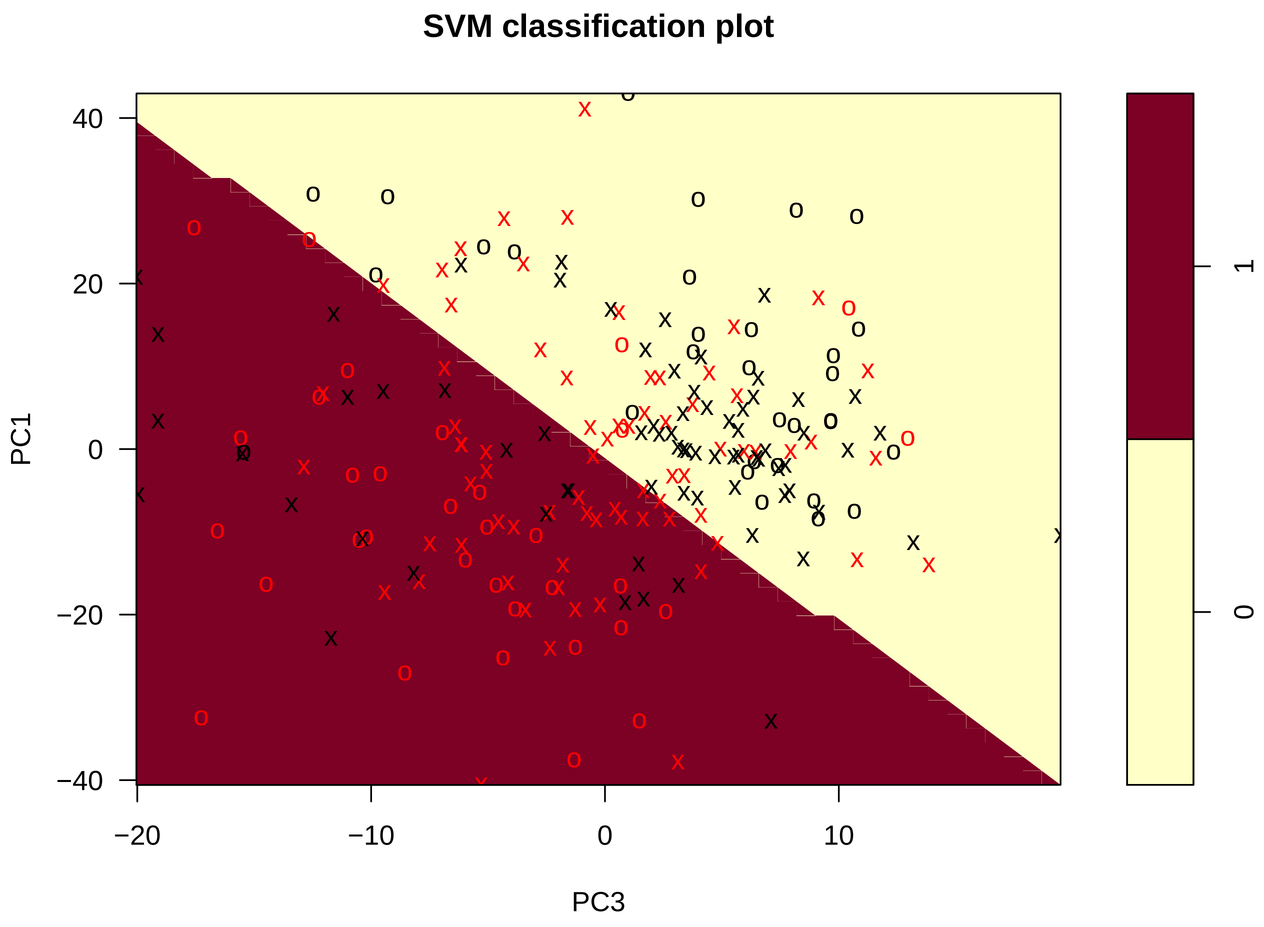
SVM is very sensitive to the choice of parameters. SVM has some parameters that have to be tuned to get better performance. It is very sensitive to the choice of parameters. Used tune.svm() to try out different parameters. It takes in different values of gamma, cost and returns the one with minimum classification error for the 10-fold cross validation. Used the best gamma and cost values to train the best_svm model.

```
  tuned_svm <- tune(svm, train.x=train.data, train.y = train.data[,1], kernel="linear", range=list(cost=10
  best_svm <- svm(as.factor(y)~., data=train.data, kernel="linear",cost=tuned_svm$best.parameters$cost, ga
  summary(best_svm)
```

```
##
## Call:
## svm(formula = as.factor(y) ~ ., data = train.data, kernel = "linear", cost = tuned_svm$best.parameters$
##     gamma = tuned_svm$best.parameters$gamma)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##
## Number of Support Vectors:  146
##
##  ( 74 72 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

SVM for Training data : First two Principle Components

```
plot(best_svm, train.data, PC1 ~ PC3)
```



Non-linear Kernel SVM :-

```
ksvm_model <- ksvm(as.factor(y)~., data=train.data, kernel="rbfdot")
summary(ksvm_model)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

```
all_data_x <- rbind(test.data, train.data[, -1])
all_data_y <- c(pca.test[, 1], train.data[, 1])
svmPred <- predict(best_svm, all_data_x)
confusionMatrix(as.factor(svmPred), as.factor(all_data_y))
```

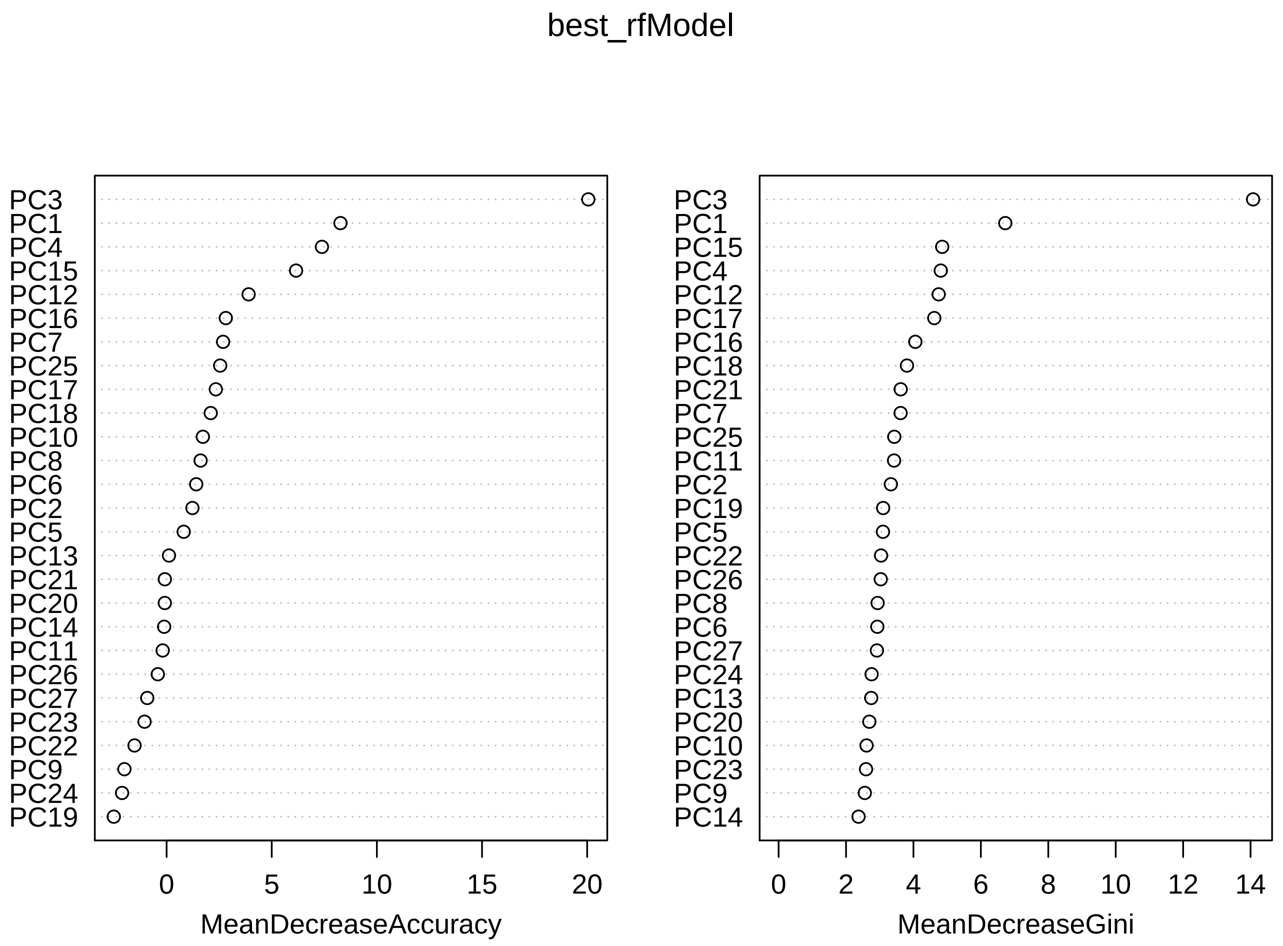
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 119  51
##          1  31  99
##
##                Accuracy : 0.7267
##                  95% CI : (0.6725, 0.7763)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 1.06e-15
##
##                   Kappa : 0.4533
##
##  Mcnemar's Test P-Value : 0.03589
##
##             Sensitivity : 0.7933
##             Specificity : 0.6600
##          Pos Pred Value : 0.7000
##          Neg Pred Value : 0.7615
##              Prevalence : 0.5000
##          Detection Rate : 0.3967
##    Detection Prevalence : 0.5667
##       Balanced Accuracy : 0.7267
##
##        'Positive' Class : 0
##
```

```
ksvmPred <- predict(ksvm_model, all_data_x)
confusionMatrix(as.factor(ksvmPred), as.factor(all_data_y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 131  33
##          1  19 117
##
##                Accuracy : 0.8267
##                  95% CI : (0.779, 0.8678)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6533
##
##  Mcnemar's Test P-Value : 0.07142
##
##             Sensitivity : 0.8733
##             Specificity : 0.7800
##          Pos Pred Value : 0.7988
##          Neg Pred Value : 0.8603
##              Prevalence : 0.5000
##          Detection Rate : 0.4367
##    Detection Prevalence : 0.5467
##       Balanced Accuracy : 0.8267
##
##        'Positive' Class : 0
##
```

Classification Model 3. Random Forest

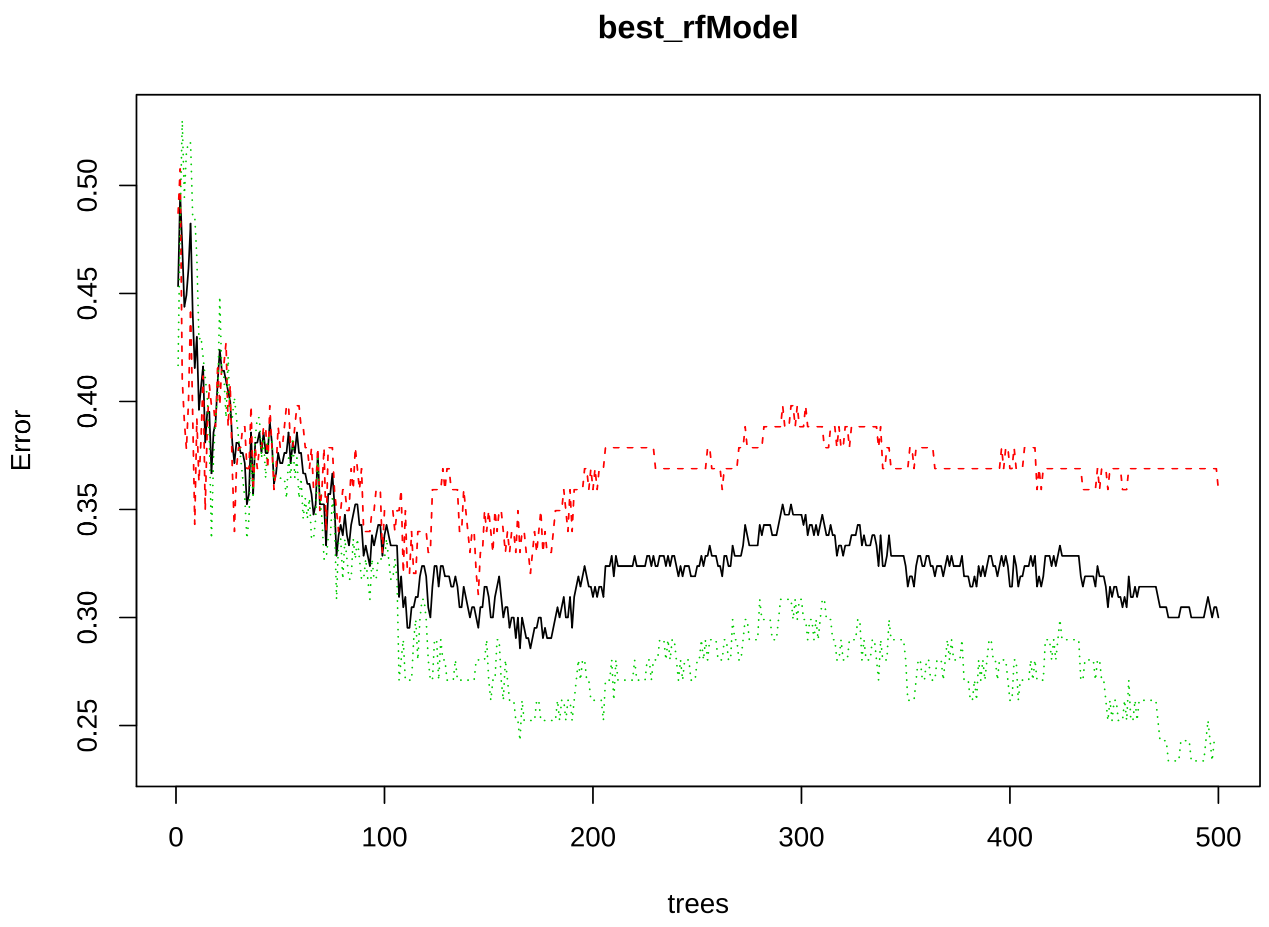
```
varImpPlot(best_rfModel)
```



```r
plot(best_rfModel)
```

Table 4: RandomForest Confusion Matrix

	0	1
0	139	10
1	11	140



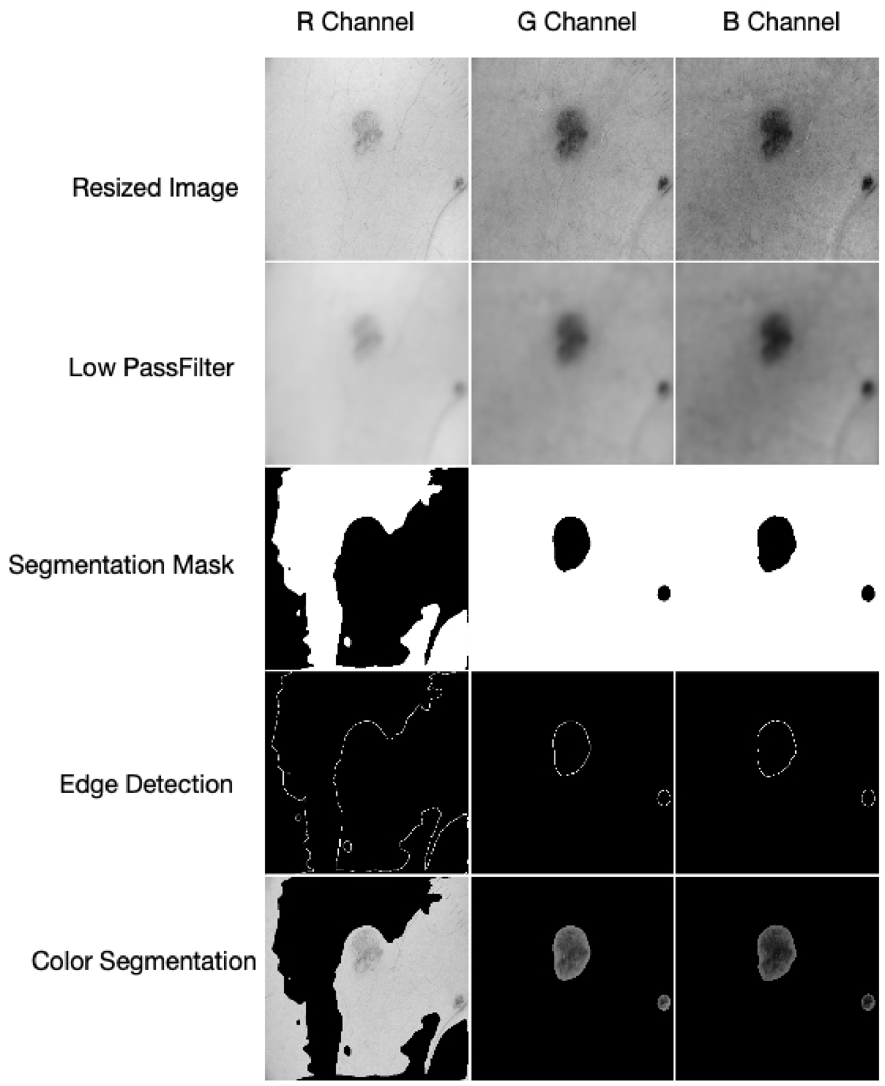
We run a random forest through a grid search through of number of trees (1 to 5000) and nodesize(1 to 50). We find that our best model is with 500 trees and a nodesize of 20. Our accuracy on the entire set is 93.33%

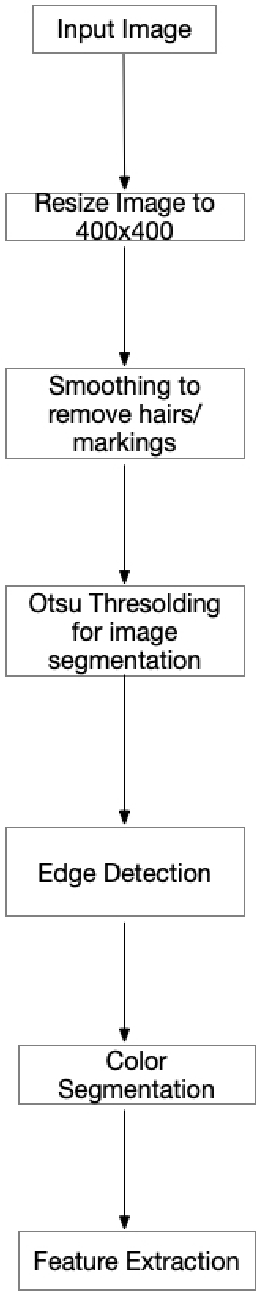
```
yhat.all <- predict(best_rfModel, all_data_x)
temp <- cbind(as.numeric(as.character(yhat.all)), (all_data_y))
all_error <- length(which(temp[, 1] != temp[, 2])) * 100 / nrow(all_data_x)
temp <- confusionMatrix(as.factor(yhat.all), as.factor(all_data_y))
kable(temp$table, caption="RandomForest Confusion Matrix")
```

[10 Points, 1 page] Literature review. You should search and read existing literature and summarize clinically relevant characteristics that could be used for skin cancer image diagnosis. There is no limitation on what type of literature you could use. However, the goal should be motivating your feature engineering approaches from a clinical and analytic point of view. Please give appropriate citations to the literature you read.

The American Academy of Dermatology Association (Darrell Rigel (n.d.)) listes the ABCDE of detecting Melanoma. ABCDE is an acronym for Asymmetry, Border, Color, Diameter and Evolving respectively. Since in this project statement, are given a series of pictures over a lesion over a period of time, we will not be able to create a specific feature for Evolving. Hence, we concentrate on the remaining. A great tools for us is EBImage which is package developed by Pau et al. (2010) which provides us very usable functions for image processing tasks. We use various features of this package for our feature engineering. Amelard, Wong, and Clausi (2012) provides high level features one could use to help engineer our features. Their main work talks about the different way to quantize irregularities. They use a mix of both coarse and fine grain methods to achieve this. Jain, Jagtap, and Pise (2015) Shivangi et al, also talks about a good pipeline to this. Their work diffs in that they introduce quantitative metrics to mark irregulaties in shape. They also take into account the size of the lesion. However to this accurately they must have ensured that the each image is taken from the same distance and focus. We do not have that information regarding our dataset, so we must rely on ration. One of the interesting features proposed in using Circularity Index, it is the ration of (4*pi*Area)/(Perimeter*Perimeter). This is a geat metric becaus this is scale invariant. Of course is Jain, Jagtap, and Pise (2015) and (**???**)(**???**), they do concentrate a fair amount on preprocessing steps like illumination and segmentation.

[10 Points, 1 page] Feature engineering. Motivated by what you have read (or your understanding), process the data in a reasonable way such that the new variables are more intuitive to your collaborator/clinicians. You need to describe clearly what is your data processing criteria and how your variables are calculated.





Our pipeline takes an input image and prepares it for feature extraction. Once, we have our features, we then run it through various classifiers to observe the results. All steps in the preprocessing steps are done in 3 planes. We keep results from all these planes, as the classifier should be able to choose from them. Preprocessing steps : 1. Resize all images to 400 x400 2. Convert Color to Grayscale but also preseve the color image. 3. We run it through a low pass filter to remove hairs and scale markings. 4. Similar to Jain, Jagtap, and Pise (2015), we do automatic thresholding by Otsu in each plane which generate the mask. We apply this mask of the low pass filtered image so that we dont see any small blobs or markings inside the lesion. 5. We apply image detection on this segmented filter. 6. We take the color image and pass it through the segmented image, so we will only see color in the segmented section of the interested area.

Do notice from the image above that the red channel does not yield very good preprocessing results while the green and blue channels are very successful in isolation the regions.

We generate 5 features for each of the channels: 1. Area: We count the number pixels in the mask of segmentation which are 0. 2. Perimeter: In the edge detected image, we count the number of pixels which are 1. 3. After applying the segmentation mask on the color image, we sum the total pixels that are in the segmented area. 4. Regularity Parameter: The ration of area / perimeter 5. Circularity Index: The ratio of (4*pi*area)/(perimeter^2)

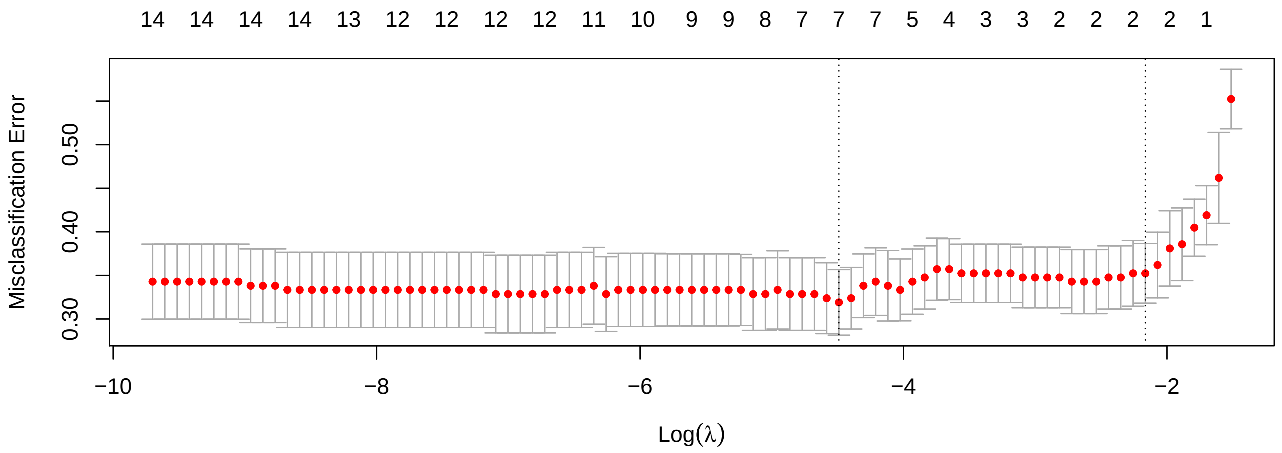
[20 Points, 2 page] Classification models based on new features. Fit two different classification models to identify malignant moles. You can either use the ones from Question 1 or use some new models if you believe they may perform better on the new features. Same requirements of Question 1 apply to this part. Besides, you should focus more on variable selection and interpretation.

```r
extract_features <- function(img_in) {
  resize_w <- 400
  resize_h <- 400
  img_resize <- resize(img_in, resize_w, resize_h)
  img_gray <- img_resize
  colorMode(img_gray) = Grayscale

  #Low Pass filter
  w = makeBrush(size = 31, shape = 'gaussian', sigma = 5)
  img_lp = filter2(img_gray, w)
  img_in_th <- img_lp
  threshold <- otsu(img_in_th)
  img_th = EBImage::combine(mapply(function(frame, th) frame > th, getFrames(img_in_th), threshold, SIMP
  img_th_val <- img_th
  img_th_val[which(img_th) == TRUE] <- 1
  img_th_val[which(img_th) == TRUE] <- 0
  fhi = matrix(1, nrow = 3, ncol = 3)
  fhi[2, 2] = -8
  img_fhi = filter2(img_th_val, fhi)
  img_fhi_col <- filter2(img_gray, fhi)

  #Features based on preprocessing
  area_f <- c(0, 0, 0)
  perimeter_f <- c(0, 0, 0)
  rgb_f <- c(0, 0, 0)
  img_col_thresh <- img_resize * img_th_val

  for (i in 1:3) {
    area_f[i] <- length(which(img_th_val[, , i] == 0)) / (resize_h * resize_w)
    perimeter_f[i] <- length(which(img_fhi[, , i] == 1)) / (resize_h * resize_w)
    rgb_f[i] <- sum(img_col_thresh[which(img_th_val[, , i] == 0)]) / (resize_h * resize_w)
  }

  reg_f <- area_f / perimeter_f
  return(c(area_f, perimeter_f, rgb_f, (area_f/perimeter_f), (4*pi*area_f/perimeter_f^2)))
}
```

Classification Model 1. Lasso

For performing classification with the lasso we are selecting the family as binomial. We are using all the pixels which we got after resizing the image to 100x100 and by using 3 channels. As we have split the data in 70/30 ratio. The xtrain data is the 210x30000 matrix and ytrain is corrsponding 210 response. xtest is 90x30000 metrix and ytest is corrsponding 90 responses.

```r
# use parallel for performace
registerDoMC(cores = 4)
cv_glmnet_model <- cv.glmnet(xtrain, ytrain, parallel = TRUE, alpha=1, nfolds = 10,
                             family="binomial", type.measure = "class")
# selecting the lambda within 1 standard error as best lambda
best_lambda = cv_glmnet_model$lambda.1se
# training with best lambda selected with 10 fold cross validation
glmnet_model <- glmnet(xtrain, ytrain, lambda = best_lambda, alpha=1,
                       family="binomial", type.measure = "class")
# predicting on test data
pred <- predict(glmnet_model, s = best_lambda, newx = xtest, type = "class")
accuracy = mean(ytest == pred) * 100
lasso_results = data.frame("Best lambda" = best_lambda, "Accuracy" = as.numeric(accuracy,3))
```



After performing the 10 fold validation and selecting the best lambda we retrained the model with the selected best lambda and family binomial and type.measure class and useed that model for predicting class on the test data. Accuracy we got is around 72%

Table 5: Lasso Results

Best.lambda	Accuracy
0.1147868	67.77778

Classification Model 2. Random Forest

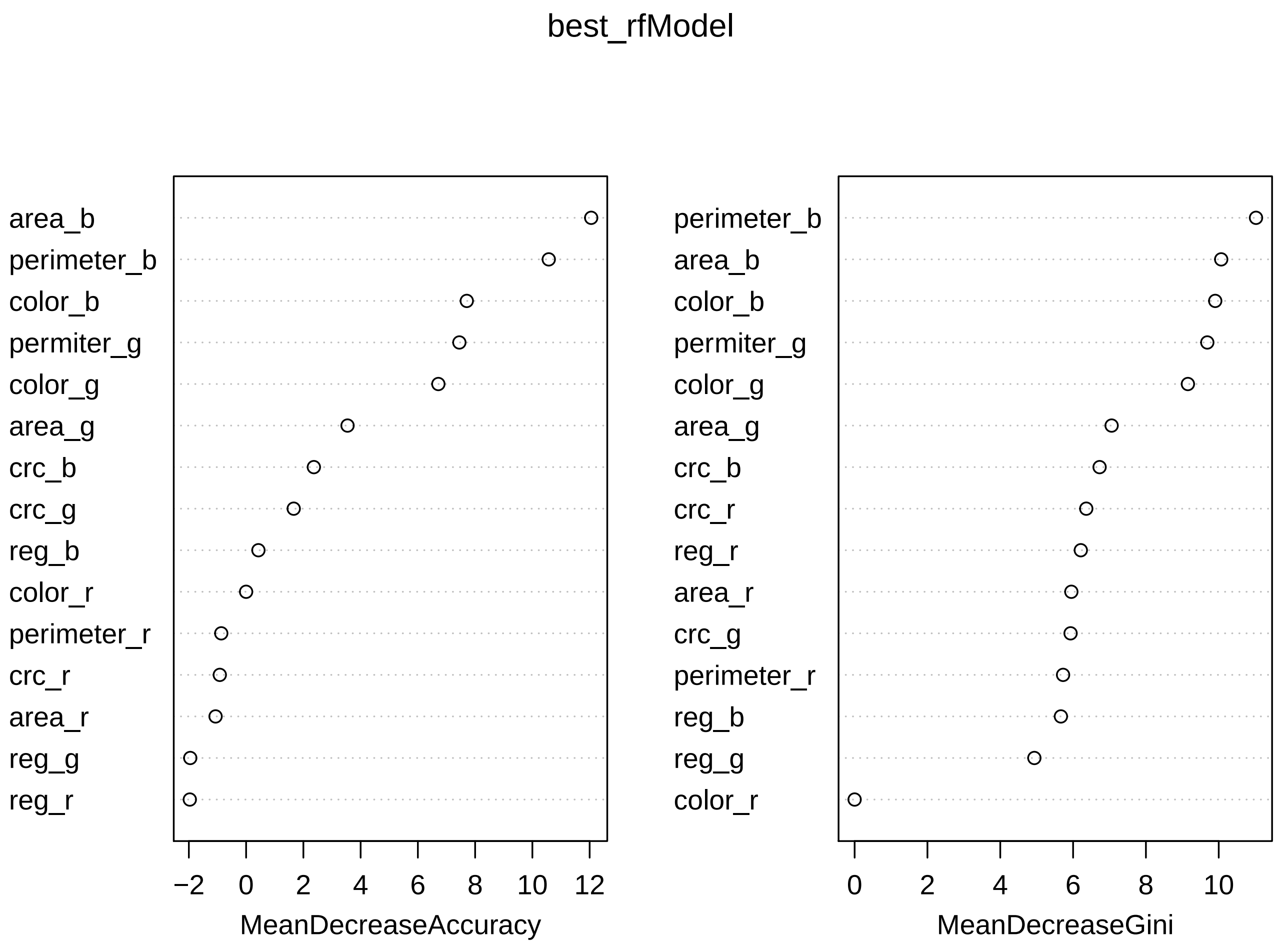
As with question1, we run through a random forest model with a grid search on number of tress and nodesize. The best model was with 100 trees and 5 nodesize with an accuracy of 92%.

```r
#best_rfModel <- randomForest(formula = as.factor(y) ~ ., data = train.data, importance = T, ntree=best_
yhat.all <- predict(best_rfModel, all_data_x)
temp <- cbind(as.numeric(as.character(yhat.all)), (all_data_y))
all_error <- length(which(temp[, 1] != temp[, 2])) * 100 / nrow(all_data_x)
temp <- confusionMatrix(as.factor(yhat.all), as.factor(all_data_y))
kable(temp$table, caption="RandomForest Confusion Matrix")
```

Table 6: RandomForest Confusion Matrix

	0	1
0	137	12
1	13	138

```r
varImpPlot(best_rfModel)
```



By Making understandable features, we see from above that: 1. Area, color and perimeter are key features. If the lesion contains multiple colors, then as Darrell Rigel (n.d.) explained, it is a high chance of cancer. 2. The next set of feature are the regularity and circularity index respectively. 3. Red is the least important channel in the model. We saw during preprocessing as well, the Red plane didn not yield good results as well.

References

Amelard, R, A Wong, and D A. Clausi. 2012. “Extracting Morphological High-Level Intuitive Features (Hlif) for Enhancing Skin Lesion Classification.” In *34th Annual International Conference of the Ieee Engineering in Medicine and Biology Society*. San Diego.

Darrell Rigel, American Academy of Dermatologist Association. n.d. “ABCDE of Melanoma.” https://www.aad.org/diseases/skin-cancer/abcde-of-melanoma.

Jain, Shivangi, Vandana Jagtap, and Nitin Pise. 2015. “Computer Aided Melanoma Skin Cancer Detection Using Image Processing.” *Procedia Computer Science* 48 (December): 736–41. https://doi.org/10.1016/j.procs.2015.04.209.

Pau, Grégoire, Florian Fuchs, Oleg Sklyar, Michael Boutros, and Wolfgang Huber. 2010. “EBImage - an R Package for Image Processing with Applications to Cellular Phenotypes.” *Bioinformatics* 26 (7): 979–81. http://dblp.uni-trier.de/db/journals/bioinformatics/bioinformatics26.html#PauFSBH10.