Instituto Tecnológico y de Estudios Superiores de Monterrey

Evidencia: Reporte formal
Actividad: Diagnóstico por Computadora

BI2009B Procesamiento de imágenes médicas para el diagnóstico (Gpo 201)

Equipo #4

Maxine Annel Pacheco Ramírez	A01551933
Dacia Martínez Díaz	A01733799
Kristen Aideé Pérez Alvarez	A00829551
Allison Hernández Sánchez	A01366543
René Jahaziel Rangel Yáñez	A00826919

Asesor: Dr. José Tamez

Monterrey, NL. Abril , 2022

I. INTRODUCCIÓN

El Deep Learning utiliza una combinación de entradas de datos, pesos y sesgos para tratar de emular el cerebro humano. Estas piezas trabajan juntas para reconocer, categorizar y caracterizar elementos en los datos correctamente. Por otro lado, la diferencia entre el Deep Learning y Machine Learning radica en la forma en que cada algoritmo aprende. El Deep Learning automatiza la mayor parte de la parte del proceso de extracción de características, permitiendo el uso de conjuntos de datos más grandes. Mientras que el Machine Learning trata de encontrar sentido, donde parece no haberlo; se puede usar un proceso de decisión creando una predicción o clasificación basada en datos de entrada, donde la predicción del modelo se evalúa utilizando una función de error y también puede ser utilizado para encontrar la diferencia entre el ejemplo conocido y la estimación del modelo se reduce mediante un procedimiento de optimización del modelo [1] [2].

II. MARCO TEÓRICO

NB - Naive Bayes

Un clasificador Naive Bayes es un modelo de aprendizaje automático probabilístico que se utiliza para tareas de clasificación [3]. Un clasificador es un modelo de aprendizaje automático que se utiliza para discriminar diferentes objetos en función de ciertas características. El quid del clasificador se basa en el teorema de Bayes:

$$P(A\ B) = \frac{P(B\ A)P(A)}{P(B)}$$

Tipos de clasificadores Naive Bayes:

- Multinomio de Bayes ingenuo: Esto se usa comúnmente para resolver problemas de categorización de documentos, como determinar si un documento encaja en las categorías de deportes, política o tecnología. La frecuencia de los términos incluidos en el documento es una de las características/predictores empleados por el clasificador.

- Bernoulli Naive Bayes: Similar a Naive Bayes multinomial, pero usando variables booleanas como predictores. Los parámetros que usamos para pronosticar la variable de clase solo aceptan respuestas de sí o no, como si una palabra aparece en el texto o no.

- Cuando los predictores toman un valor continuo en lugar de un valor discreto, asumimos que los valores se muestrean a partir de una distribución gaussiana.

KNN

El método de k-nearest neighbors, a menudo conocido como KNN o k-NN, es un clasificador de aprendizaje supervisado no paramétrico que realiza clasificaciones o predicciones sobre la agrupación de puntos de datos individuales en función de la cercanía. Se puede usar tanto para problemas de regresión como de clasificación, sin embargo, se emplea más comúnmente como una técnica de clasificación, basada en la idea de que los

puntos comparables se pueden descubrir juntos. Los problemas de regresión usan un concepto similar al de los problemas de clasificación, pero en este caso, se toma el promedio de los k-nearest neighbors para hacer una predicción sobre una clasificación [4].

SVM

Support Vector Machine es una técnica de aprendizaje automático supervisado que se puede utilizar para resolver problemas de clasificación y regresión. Sin embargo, se emplea principalmente para resolver dificultades de categorización. Cada elemento de datos se representa como un punto en un espacio n-dimensional (donde n es el número de características que tiene), siendo el valor de cada característica el valor de una determinada coordenada en el algoritmo SVM [5].

AdaBoost

AdaBoost, también llamado Adaptive Boosting, es una técnica de aprendizaje automático que se utiliza como método de conjunto. El algoritmo más común que se usa con AdaBoost son los árboles de decisión con un nivel, es decir, árboles de decisión con solo una división. Lo que hace este algoritmo es que construye un modelo y otorga pesos iguales a todos los puntos de datos. Luego asigna pesos más altos a los puntos que están mal clasificados, y posteriormente todos los puntos que tienen pesos más altos tienen más importancia en el próximo modelo. Seguirá entrenando modelos hasta que se reciba un error mínimo [6].

GBM

Una Gradient Boosting Machine o GBM combina las predicciones de múltiples árboles de decisión para generar las predicciones finales. Los nodos en cada árbol de decisión toman un subconjunto diferente de características para seleccionar la mejor división, lo que indica que los árboles individuales no son todos iguales y, por lo tanto, pueden capturar diferentes señales de los datos. Cada nuevo árbol toma en cuenta los errores o errores cometidos por los árboles anteriores. Por lo que, cada árbol de decisión sucesivo se construye sobre los errores de los árboles anteriores. Así es como los árboles en un algoritmo de máquina de aumento de gradiente se construyen secuencialmente [7].

Segmentaciones

El fin de utilizar Machine Learning para segmentar es generar un sistema al que se le de una imagen y este arroje una imagen con las regiones u objetos de interés segmentados. Este método permite automatizar la segmentación de imágenes, ahorrando una gran cantidad de tiempo, lo cual resulta muy útil cuando se busca segmentar muchas imágenes y/o las imágenes contienen una gran cantidad de elementos de interés.

Existen distintos problemas a la hora de segmentar imágenes automáticamente por medio de inteligencia artificial, uno de ellos es que los objetos o regiones de interés varían entre imágenes en cuanto a su apariencia, ubicación, brillo, contraste, etc., por lo que es necesaria una enorme cantidad de datos para entrenar a los modelos de aprendizaje a fin de que se logre segmentar correctamente cada imagen a pesar de las diferencias que se pueden presentar entre estas.

Estos modelos no siempre logran segmentar correctamente todas las imágenes, detectando objetos o regiones de interés donde no los hay o, viceversa, no detectando estos a pesar de estar presentes. Esta situación es muy delicada cuando se habla de diagnósticos médicos, generando falsos positivos o falsos negativos según el caso. Un ejemplo de esto sería entrenar un modelo de aprendizaje a detectar melanomas en imágenes médicas, por lo que sí el modelo no detecta la presencia de un melanoma en una imagen donde existe presencia de este, se estaría hablando de un falso negativo y, por el contrario, sí el modelo detecta una región y la segmenta señalando la presencia de un melanoma en una imagen donde realmente no lo hay, se habla de un falso positivo.

III. METODOLOGÍA

Para la segmentación y el entrenamiento de los modelos machine learning fue necesario descargar los documentos del código a utilizar y la base de datos de imágenes del siguiente link:

https://www.dropbox.com/sh/94w0lgobp4fspsi/AABdANxAsebLVSyNSLcHWwCKa?dl=0

Después de realizar la descarga de estos archivos se realizaron las segmentaciones de los lunares y se extrajeron sus características utilizando MATLAB (Anexo 1). Dentro de las características que se extrajeron se encuentran las siguientes:

1. Señal: Media, standard deviation, skewness y kurtosis
   Estas características son datos estadísticos de cada segmentación.
2. Forma: Volumen, superficie, etc.
   Características relacionadas con la geometría de las manchas.
3. Textura: GLCM y Haralick features
   Las características de la textura de Haralick se calculan a partir de una matriz de co-ocurrencia de nivel de gris (GLCM), una matriz que cuenta la co-ocurrencia de los niveles de gris vecinos en la imagen. El GLCM es una matriz cuadrada que tiene la dimensión del número de niveles de gris N en la región de interés.
4. Fractal dimension
   Es una medida de qué tan compleja es una figura autosimilar.

Los datos se importaron a R.Studio y se construyó un Heatmap para identificar posibles clusters e inferir qué características tienen mayor peso en la decisión.

Posterior al análisis del Heatmap, se crearon 5 modelos de Machine Learning (Naive Bayes, KNN, SVM, AdaBoost y GBM) para predecir la existencia de melanomas a partir de las características que se extrajeron de cada imagen.

Para evaluar el desempeño de cada modelo se realizó el análisis de las curvas Receiver Operating Characteristic (ROC), con estas curvas nos fue posible ilustrar la sensibilidad y especificidad que presentan los posibles puntos de corte de un test diagnóstico.

Esta curva ROC nos brinda una respuesta característica que nos permite tomar una decisión adecuada al hacer uso de machine ling.

Con la finalidad de obtener esas curvas fue necesario el uso del código en R.studio (Anexo 2) donde para cada uno de los métodos utilizados fue necesario seleccionar la siguiente configuración:

```
trainFraction = 0.70,
repetitions = 100,
classSamplingType = "Ba",
```

Donde el 0.70 representa el porcentaje de los datos datos que son utilizados para entrenar al modelo, ya que con este porcentaje se logra disminuir el tiempo del entrenamiento y se tiene una buena cantidad de datos para analizar. En cuanto al 'Ba' de la variable 'class Sampling Type' se refiere a un entrenamiento balanceado, es decir que entran 20 y 20 muestras al entrenamiento o 40 y 40 sin crear perturbaciones permitiendo así de igual manera un menor tiempo de simulación.

IV. RESULTADOS Y DISCUSIÓN

En las Figuras [1-6] se muestran ejemplos de las segmentaciones que se realizaron de las imágenes de lesiones de Melanoma, Nevus y Queratosis Seborreica.

El canal verde, normalmente contiene mayor información dado que es la que normalmente llega de mejor manera a la superficie. Posteriormente, se quitan los vellos de la imagen con morfología matemática para después hacer una estandarización de las imágenes y crear una máscara. Después, se encuentra el mínimo de los tres canales de la imágen y se coloca una máscara en el área en el que está la lesión, para después erosionarse para ver únicamente la piel con cáncer y se asume que lo que se encuentra fuera es únicamente piel sana.

De igual forma, las Figuras [1-6] se pueden observar la segmentaciones de las diferentes condiciones dermatológicas, donde se aprecian las imágenes crudas, seguidas de las imágenes obtenidas al procesarse como se mencionó anteriormente. La imagen final obtenida indica la región de control, mientras que la anterior a esta indica la región con cáncer, segmentándose correctamente en algunos casos y en algunos otros no siendo exitosa la segmentación.

Skin Lesions - Melanoma

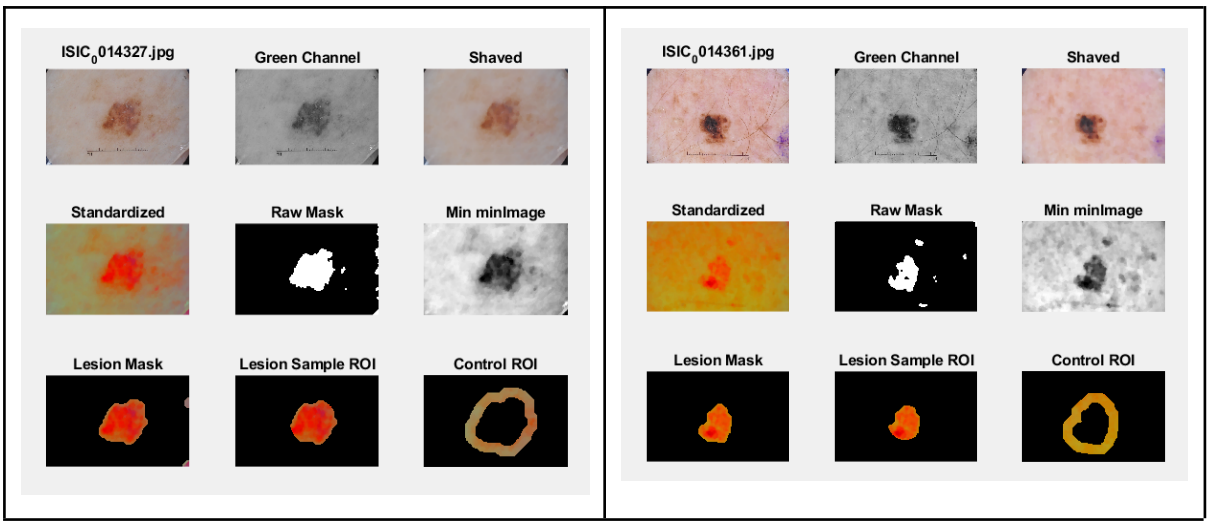

Figura 1. Ejemplos de buenas segmentaciones de Melanoma

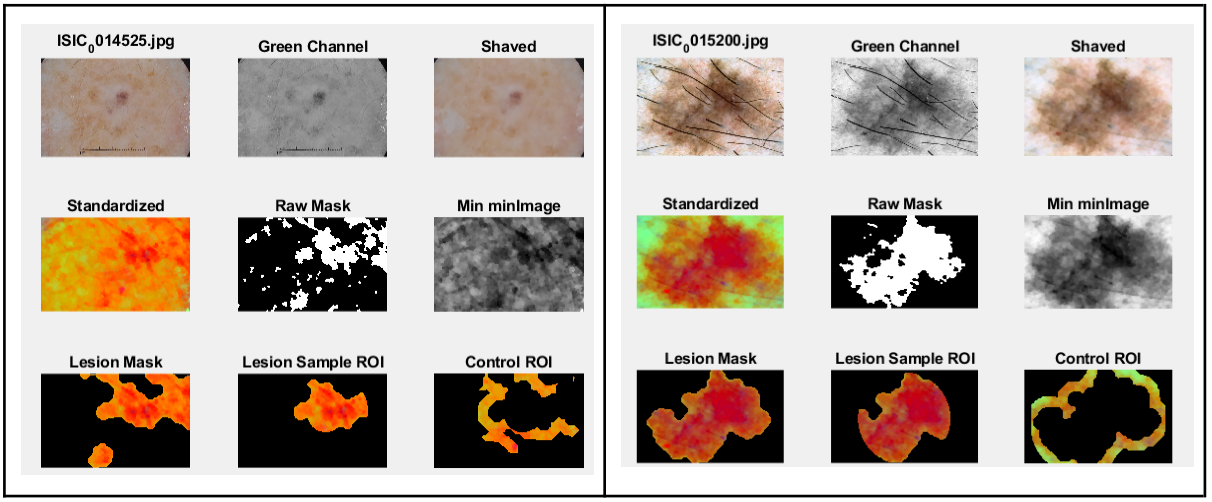

Figura 2. Ejemplos de malas segmentaciones de Melanoma

Skin Lesions - Nevus

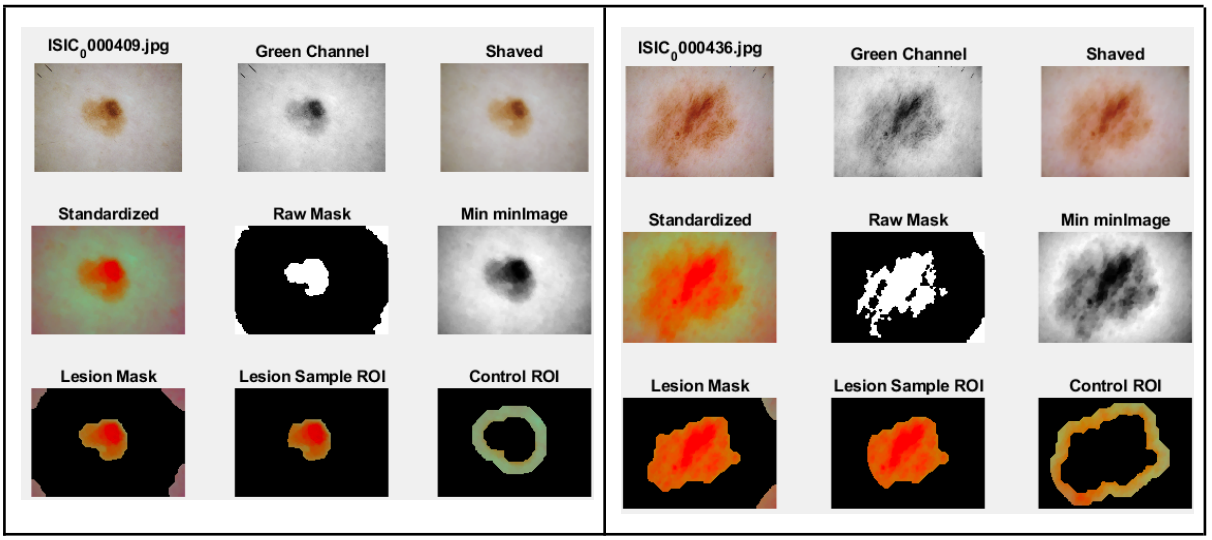

Figura 3. Ejemplos de Buenas segmentaciones de Nevus

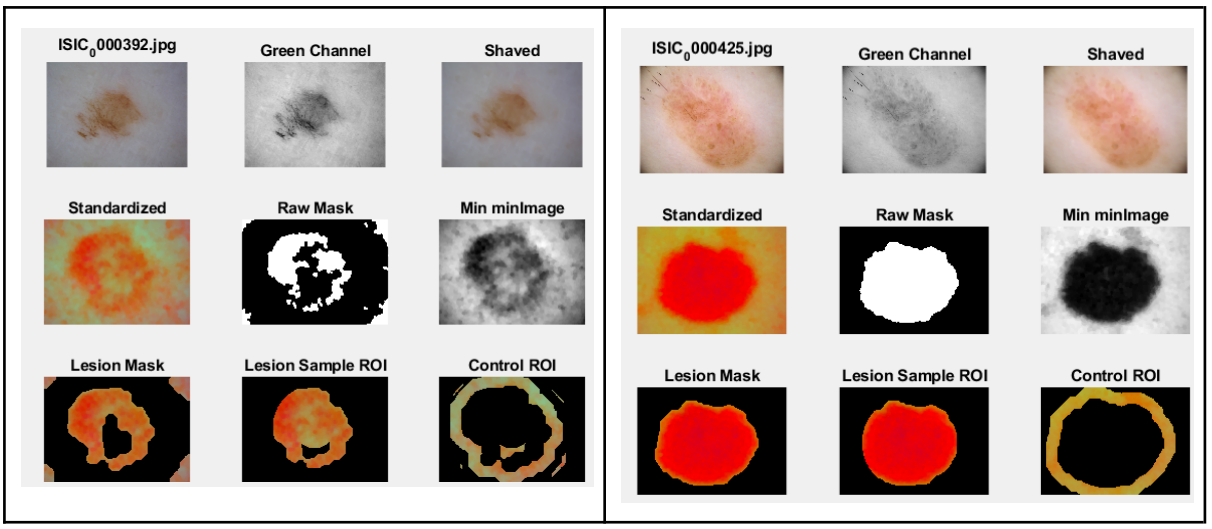

Figura 4. Ejemplos de Buenas segmentaciones de Nevus

Skin Lesions - seborrheic_keratosis

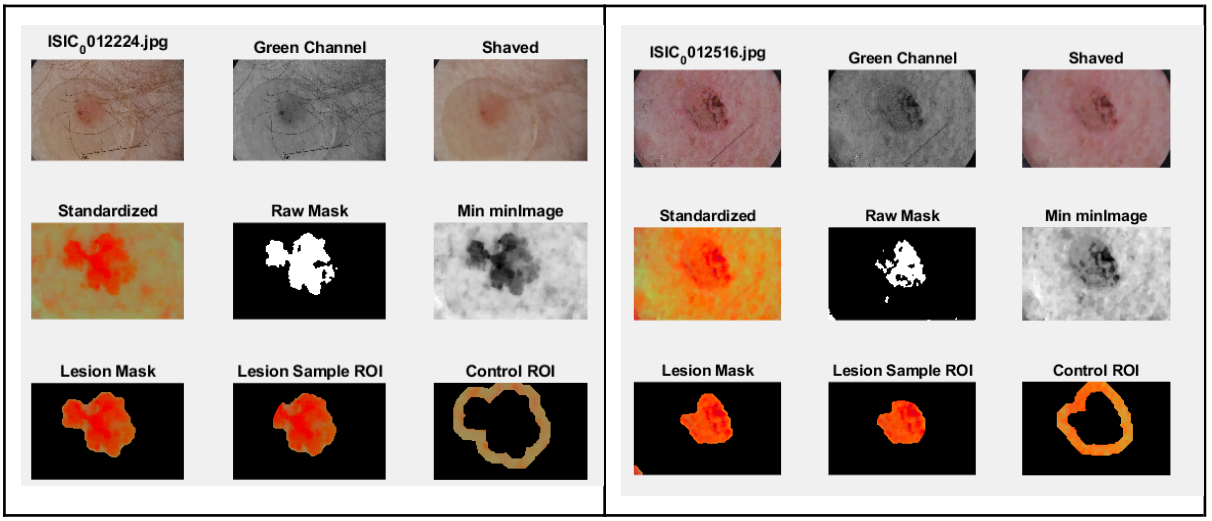

Figura 5. Ejemplos de buenas segmentaciones de Skin Lesions - Seborrheic

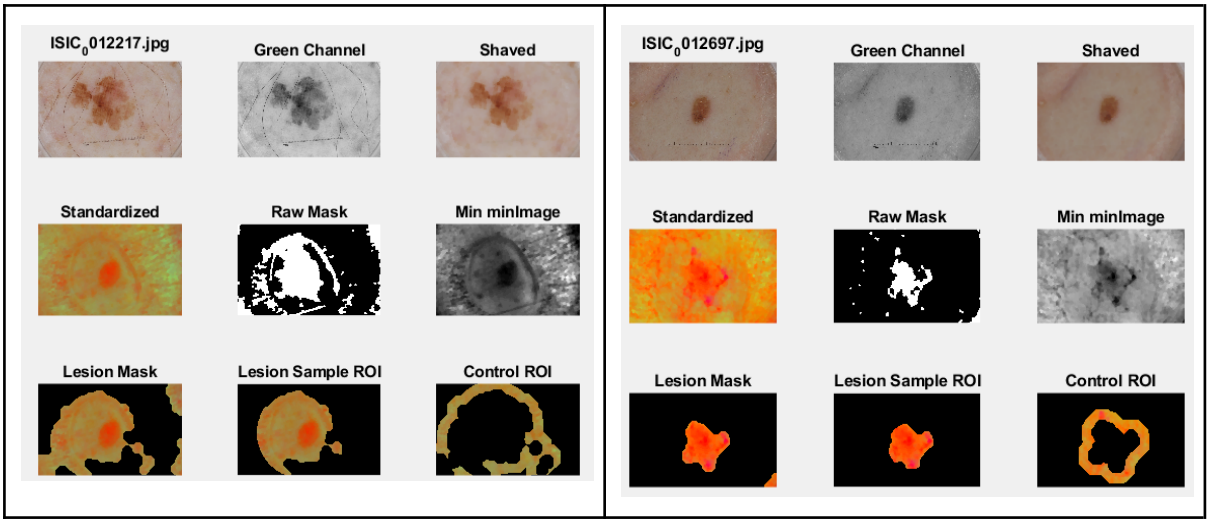

Figura 6. Ejemplos de malas segmentaciones de Skin Lesions - Seborrheic

La Figura 7 muestra el Heatmap que incluye los datos analizados. En esta representación visual se busca obtener clusters de diferentes colores que ayuden a identificar los lugares en los que se presenta cada clasificación. Se puede observar que en el Heatmap obtenido no se presentan patrones específicos. Además, el dendograma de esta representación no muestra una correcta agrupación de los datos. Idealmente, una buena representación visual en el Heatmap tendría clusters de los colores que se encuentran en la normalización de color key, que indicarían la intensidad del melanoma de acuerdo a las etiquetas (0 = no melanoma, 1 = melanoma).

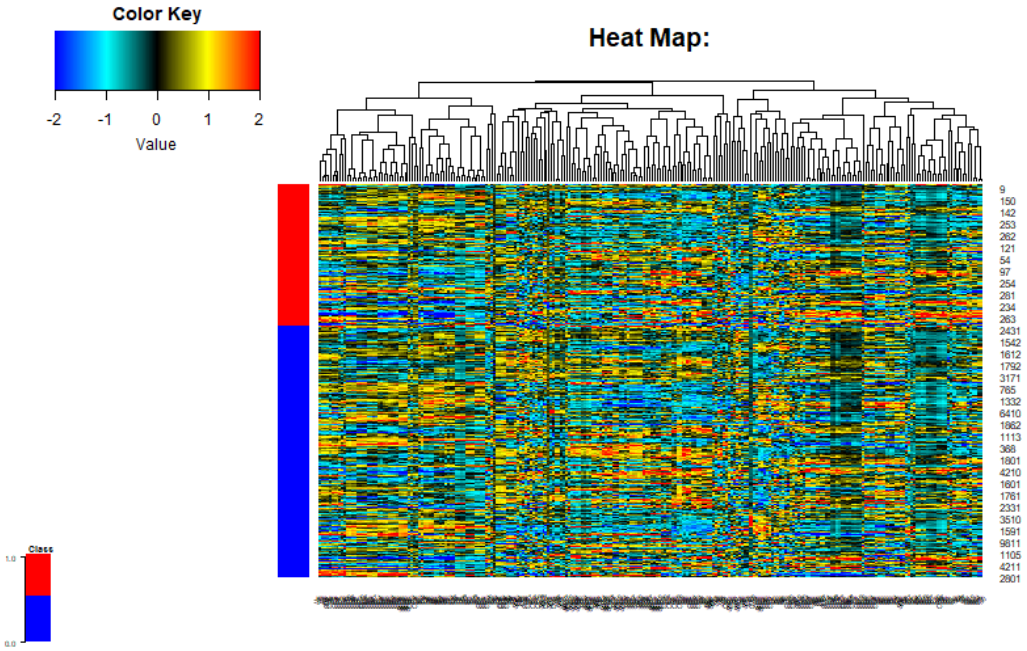

Figura 7. Heatmap de los datos analizados

Un ejemplo de un Heatmap que representa correctamente cada clasificación se muestra en la Figura 8, en el que se pueden observar claramente los clusters de datos agrupados.

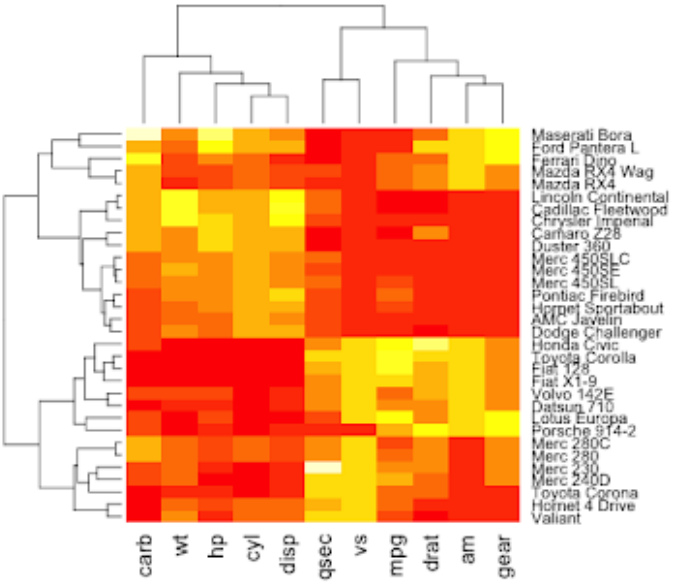

Figura 8. Heatmap con clusters visibles

Al correr el código de R.Studio se obtuvieron resultados para cada curva ROC dependiendo del método utilizado como se puede notar en las Figuras [9-13].

En la Figura 9 se observa la curva ROC del método KNN. En esta se curva se puede apreciar que el AUC es igual a 0.816 y, observando la línea verde graficada se puede identificar que este modelo es más sensible (sensibilidad ≈ 0.84) qué especifico (especificidad ≈ 0.79). Este tipo de modelos en el que se tiene una mayor sensibilidad, se utilizan cuando se busca detectar a todos los pacientes enfermos aunque se consideren pacientes sanos como enfermos [8]; un ejemplo de esto sería el detectar a pacientes con melanoma, en el que se tomaría como positivo cualquier lunar o mancha anormal, esto con el fin de identificar a todos lo que pudieran tener riesgo de cáncer de piel.

Al analizar la matriz de confusión de este método fue posible observar que el método si cuenta con una buena cantidad de verdaderos negativos, sin embargo podemos notar que de igual manera tiene una cantidad considerable de falsos positivos que si esto no se llega a detectar antes de realizar una clasificación o tomar una decisión que podría someter al paciente ya sea a una cirugía o a un procedimiento que no sea tan necesario o sea algo muy drástico tomando en cuenta la condición de este paciente. El método KNN muestra un valor de F1=0.492, este dato muestra una relación directamente proporcional con el desempeño del método, conociendo que entre mayor sea el valor de F1 mejor desempeño mostrará el método, al igual que se puede considerar como una medida de precisión y recuperación de acuerdo a un umbral en ocasiones cuando la curva ROC no sea suficiente o clara para tomar una decisión o realizar alguna clasificación [9][10].

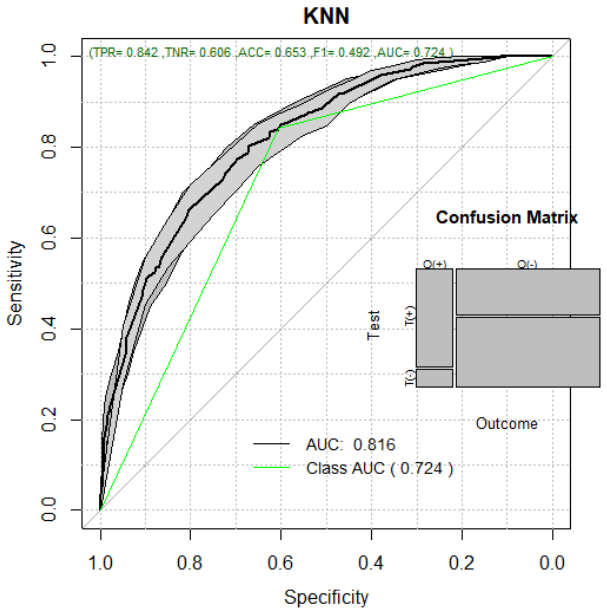

Figura 9. Curva ROC con el método KNN

En la Figura 10 se puede analizar el método de clasificación-regresión de 'Vector Support Machines (SVM)' donde podemos notar con la gráfica ROC un AUC con valor de 0.816, es decir que si se llega a utilizar este método para tomar una decisión es importante para asegurar que la mejor opción o decisión fue seleccionada el pasar por otros procesos para la correcta selección ya que de lo contrario se puede estar propenso a cometer algún error. En cuanto a la ubicación del punto de corte en este método se logra encontrar que se tiene una mayor sensibilidad (≈ 0.79) qué especificidad (≈ 0.83), de manera que esto puede afectar en la decisión tomada al poder cometer el error de que una persona sin la necesidad de tener una cirugía sea intervenido.

Por otro lado, en la matriz de confusión de este modelo se puede observar que se tiene una buena cantidad de verdaderos positivos, sin embargo se tiene de igual manera una cantidad considerable de falsos negativos, lo que podría afectar significativamente si un paciente que en realidad tiene una enfermedad que requiere cirugía es detectado como negativo en el diagnóstico, pudiendo causar una afectación severa en el paciente por no ser tratado adecuadamente. Además, al analizar el valor F1 de este modelo, se puede decir que su desempeño no es el más adecuado comparado con los demás modelos, ya que este fue de 0.588. [11]

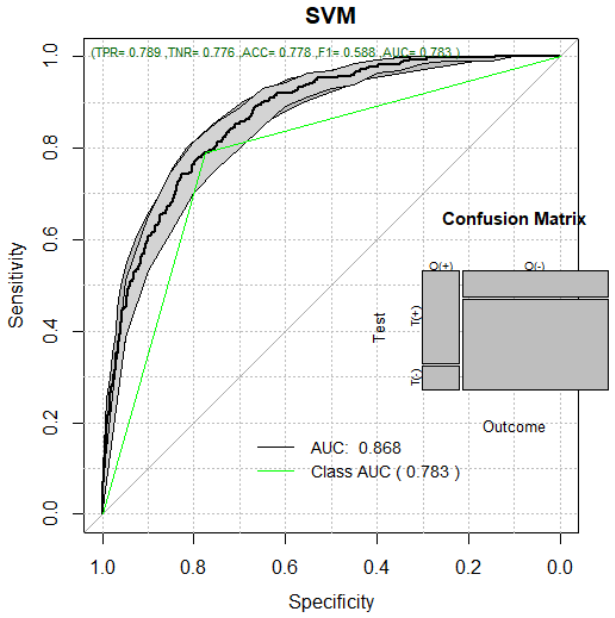

Figura 10. Curva ROC del modelo Vector Support Machines

En la Figura 11 se presenta la curva ROC del método de Naive Bayes donde se tiene un valor de AUC de 0.868, en esta curva se puede observar que de igual manera se tiene un mayor número de sensibilidad sobre la especificidad, con esto en mente no es posible o es una decisión de riesgo el confiar ciegamente al resultado que arroje este método para la clasificación.

En la matriz de confusión de este modelo se puede observar que al realizar un diagnóstico con este, se tiene una cantidad considerable de falsos negativos y también una gran cantidad de falsos positivos, lo que tendría muchas consecuencias si se realiza un diagnóstico de alguna enfermedad que requiere de un tratamiento que puede dañar otros órganos o una enfermedad que debe ser tratada con cirugía y pacientes sanos son diagnosticados como positivos a la enfermedad y se les trata con algo que no requieren. Sin embargo, el método podría utilizarse si lo que se quiere es detectar a todos los pacientes enfermos en alguna población, sin importar que pacientes sanos sean considerados como positivos a la enfermedad. Por otro lado, el valor de F1 de este modelo fue de 0.527, indicando que, tal y como se mencionó anteriormente, el desempeño de este no es bueno, comparado con los demás métodos.

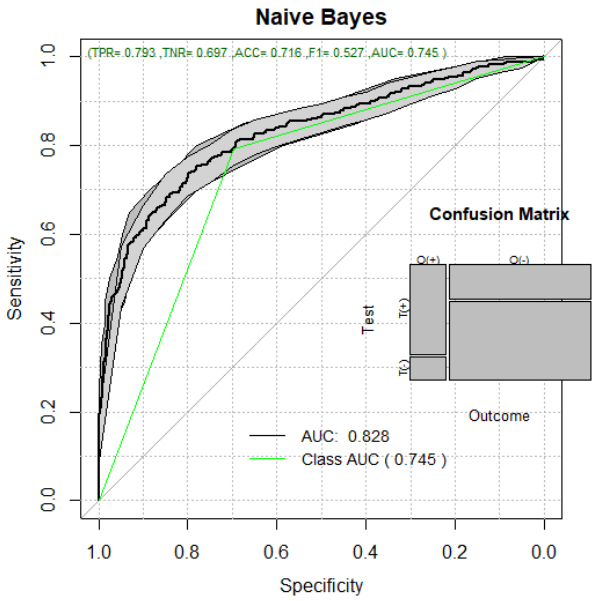

Figura 11. Curva ROC del modelo Naive Bayes [NB]

En la Figura 12 se muestra la curva ROC del método GBM, al realizar un diagnóstico con este modelo se observa un AUC igual a 0.929, este es un método un tanto más sensible (≈ 0.89) que específico (≈ 0.82). Aunque se tiene un poco menos de especificidad, este modelo es confiable para identificar tanto pacientes positivos como negativos a una enfermedad determinada; por ejemplo, al hacer un diagnóstico de un tumor cerebral, no se correría un riesgo tan elevado de intervenir con cirugía a alguien que no la requiere. En cuanto a la matriz de confusión presentada por este modelo es posible notar que una buena cantidad de verdaderos negativos son detectados, y un buen número de verdaderos positivos haciendo que este sea un buen método para la toma de decisiones conociendo que nos dará predicciones que beneficiarán al paciente como lo son que en el caso que el paciente no cuente con el padecimiento que se está detectando sea etiquetado como negativo y en caso de que si lo tenga sea etiquetado como positivo, logrando así una correcta detección y clasificación para actuar correctamente ante las situaciones presentadas. El valor F1 del método de GBM es igual a 0.678, siendo así un número alto a comparación de los presentes por los modelos anteriores logrando así que este método cuente con una buena medida de decisión y recuperación y un desempeño superior a los modelos anteriores.

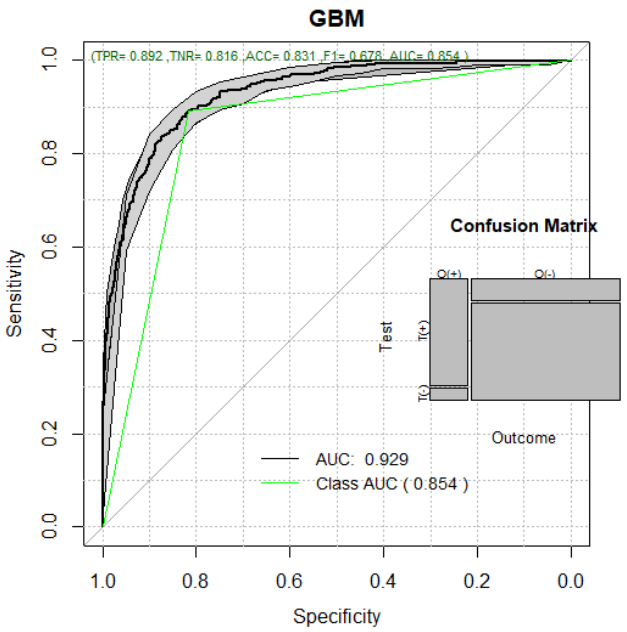

Figura 12. Curva ROC del modelo GBM

En la Figura 13 se muestra el método de Adaboost. Un concepto importante a recordar es que este método entra de las categorías de blackbox, donde el cómo este toma la decisión es un proceso muy rebuscado, y es por esto que las personas no se enfocan en el cómo este obtiene el resultado que revela, sino que lo que observan es el resultado final que arroja el método de Adaboost, es importante señalar que este método cuenta con un AUC igual a 0.940, con este resultado y teniendo en cuenta que al tener un valor de AUC de uno ya no es necesario el pasar por otro análisis para tomar la decisión ya que es 100% correcta esta, se puede argumentar que el método de Adaboost es el método más fiable para tomar una decisión en cuanto a la clasificación. Al notar el comportamiento plasmado en la gráfica de especificidad contra sensibilidad es posible notar cómo este método es altamente específico y altamente sensible generando así un área bajo la curva (AUC) muy cercana a 1. En otras palabras este método es muy confiable debido a las características previamente mencionadas sin embargo el lograr entender el mecanismo utilizado para llegar al resultado que presenta sea incomprensible o muy difícil de comprender debido a que pertenece a los métodos black box.

Además, en la matriz de confusión de este método se puede observar que hay una cantidad muy pequeña de falsos negativos y de falsos positivos, siendo esto beneficioso al momento de realizar un diagnóstico con este ya que se evitaría el tratar a pacientes sanos con medicamentos o procedimientos que en realidad no requieren, así como también, se disminuirían los pacientes que no son tratados aún y cuando realmente sí tienen la enfermedad. Igualmente, el valor de F1 (0.687) demuestra que, en comparación con los demás métodos, Adaboost cuenta con el mejor rendimiento y es el modelo más adecuado para realizar algún diagnóstico.

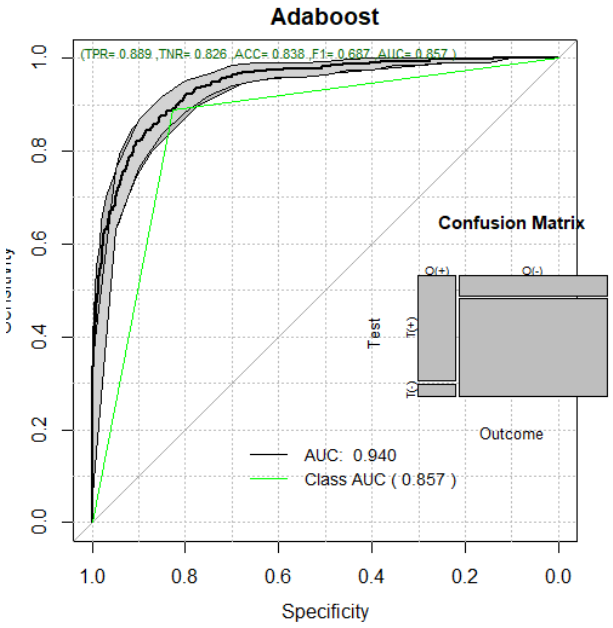

Figura 13. Curva ROC del modelo Adaboost

V. CONCLUSIÓN

La segmentación de imágenes es una rama del procesamiento digital de imágenes que tiene numerosas aplicaciones en el campo del análisis de imágenes. El campo del análisis de imágenes médicas ha estado en crecimiento y la segmentación de los órganos, enfermedades o anomalías en las imágenes médicas se ha vuelto cada vez más esencial. La segmentación de imágenes médicas ayuda a controlar el crecimiento de enfermedades como tumores, controlar la dosis de medicamentos y la dosis de exposición a las radiaciones. En cuanto a las segmentaciones en MATLAB pudimos notar que así como existen buenas segmentaciones de igual manera existen segmentaciones que no se logran realizar de manera perfecta debido a fallas en el programa, umbrales o métodos de detección. En cuanto a los modelos entrenados en estudio se demostró que el mejor AUC fue el Adaboost. A través de este análisis cuantitativo y cualitativo, consideramos que es un excelente acercamiento al procesamiento y segmentación de imágenes, sin embargo consideramos que para tener un mejor desempeño a futuro, es necesario entrenar la red con una mayor cantidad de datos para que así se tengan mejores valores de especificidad y sensibilidad.

VI. REFERENCIAS

[1] IBM Cloud Education. (n.d.). ¿Qué es machine learning? IBM. Retrieved May 4, 2022, from https://www.ibm.com/mx-es/cloud/learn/machine-learning
[2] IBM Cloud Education. (2021). What is deep learning? IBM. Retrieved May 4, 2022, from https://www.ibm.com/cloud/learn/deep-learning
[3] Gandhi, R. (2018, May 17). Naive Bayes classifier. Medium. Retrieved May 4, 2022, from https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

[4] What is the K-nearest neighbors algorithm? IBM. (n.d.). Retrieved May 4, 2022, from https://www.ibm.com/topics/knn
[5] SVM: Support Vector Machine Algorithm in machine learning. Analytics Vidhya. (2021, August 26). Retrieved May 4, 2022, from https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
[6] AdaBoost algorithm - A complete guide for beginners. Analytics Vidhya. (2021, September 15). Retrieved May 4, 2022, from https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/
[7] Boosting algorithms in machine learning. Analytics Vidhya. (2020, April 20). Retrieved May 4, 2022, from https://www.analyticsvidhya.com/blog/2020/02/4-boosting-algorithms-machine-learning/
[8] Espino Y Sosa, S., Godines Enríquez, M., & Ramírez Calvo, J. (2009). MEDICINA BASADA EN EVIDENCIAS: PRUEBA DIAGNÓSTICA. Retos y Redes, 2. https://www.inper.mx/descargas/pdf/Articulo-2_MBE.pdf
[9] *Explicación detallada de F1 Score, P-R, ROC, AUC - programador clic*. (2022). Explicación detallada de F1 Score, P-R, ROC, AUC. Recuperado 2022, de https://programmerclick.com/article/86531223630/
[10] *Machine-learning — Puntuación F1 vs ROC AUC*. (2017, 25 mayo). desarrollo web br bd. Recuperado 2022, de https://www.desarrollo-web-br-bd.com/es/machine-learning/puntuacion-f1-vs-roc-auc/832423594/
[11] González Ferrer, Vielka & Rodríguez, Milagros & Cervantes, Julio. (2011). Curvas Receiver Operating Characteristic y matrices de confusión en la elaboración de escalas diagnósticas. Revista eSalud. 7.

ANEXOS

1. Código utilizado en Matlab

Link al código en github: https://github.com/DaciaMtz/Imagenologia/blob/main/Melanoma%20Segmentation%20Code/loadAndProcessQuantitateImage.m

```
%% Structural Elements
filterDisk30 = strel('disk',3);
filterDisk0 = strel('disk',5);
filterDisk = strel('disk',25);
largeDisk = strel('disk',50);
filterDisk2 = strel('disk',7);
filterDisk3 = strel('disk',10);
numGLCMBins = 40;
%% Filename
quantiativeLesionData = [];
quantiativeControlData = [];
imagename = strcat(melanomapath,imageFile);
%% Image Loading
f=imread(imagename);
figure(1)
subplot(3,3,1)
```

```
imshow(f)
title(imageFile);
subplot(3,3,2)
imshow(f(:,:,2),[])
title("Green Channel");
Gmedian = median(double(f(:,:,2))/255.0,'all');
sz = size(f);
f=double(imresize(f,[512,512*sz(2)/sz(1)]))/255;
f = imclose(f,filterDisk0);
f = imopen(f,filterDisk0);
subplot(3,3,3)
imshow(f,[])
title("Shaved");
f = imcrop(f,[32 32 (512*sz(2)/sz(1)-64) 447 ]);
fcp = f;
RImage = f(:,:,1);
GImage = f(:,:,2);
BImage = f(:,:,3);
Gcenter = imcrop(GImage,[32 32 (448*sz(2)/sz(1)-64) 383 ]);
Ggain = 1.00/Gmedian;
f(:,:,1) = (RImage-GImage)*Ggain + 0.5;
f(:,:,2) = (GImage-Gmedian)*Ggain + 0.5;
f(:,:,3) = (BImage-RImage)*Ggain + 0.5;
fcenter = imcrop(fcp,[32 32 (448*sz(2)/sz(1)-64) 383 ]);
for channel = 1:3
   fcp(:,:,channel) = fcp(:,:,channel)-min(min(fcenter(:,:,channel)));
   fcp(:,:,channel) = fcp(:,:,channel)./max(max(fcp(:,:,channel)));
end
minImage = min(fcp,[],3);
figure(1)
subplot(3,3,4)
imshow(f,[])
title("Standardized");
maskthr = 0.75*(median(minImage,'all') + graythresh(minImage))/2.0;
mask = minImage <= maskthr;
subplot(3,3,5)
imshow(mask,[])
title('Raw Mask');
maskMorph = imdilate(mask,filterDisk3);
maskMorph = imclose(maskMorph,filterDisk3);
maskMorph = imopen(maskMorph,filterDisk);
finalmask = imclose(maskMorph,filterDisk);
%imshow(finalmask,[])
finalmaskD = imdilate(maskMorph,filterDisk2);
controlmask = imdilate(finalmaskD,filterDisk);
controlmask = imdilate(controlmask,filterDisk) & not(finalmaskD);
%%
subplot(3,3,6)
imshow(minImage,[0,max(max(minImage))])
title('Min minImage');
subplot(3,3,7)
lesionimage = f.*finalmask ;
imshow(lesionimage,[]);
title('Lesion Mask');
templateRadius =  sum(sum(finalmask))/3;
templateRadius2 = sum(sum(finalmask))/12;
sz = size(finalmask);
[X,Y] = meshgrid(1:sz(2),1:sz(1));
templateMx = 0.7*mean(X(finalmask)) + 0.3*sz(2)/2;
templateMy = 0.7*mean(Y(finalmask)) + 0.3*sz(1)/2;
templateMx2 = 0.5*mean(X(finalmask)) + 0.5*sz(2)/2;
templateMy2 = 0.5*mean(Y(finalmask)) + 0.5*sz(1)/2;
finalmask = finalmask & (((X - templateMx).^2 + (Y - templateMy).^2) < templateRadius);
finalmask = imopen(finalmask,filterDisk3);
finalmask = imclose(finalmask,filterDisk);
if (templateRadius2 < 900 )
   templateRadius2 = 900;
end
finalmask = finalmask | (((X - templateMx2).^2 + (Y - templateMy2).^2) < templateRadius2);
controlmask = controlmask & not(finalmask) & imdilate(imdilate(finalmask,largeDisk),largeDisk);
lesionimage = f.*finalmask ;
```

```
subplot(3,3,8)
imshow(lesionimage,[]);
title('Lesion Sample ROI');
controlimage = f.*controlmask;
subplot(3,3,9)
imshow(controlimage)
title('Control ROI');
%% Extract Lession features
dy = [-1 -2 -1; 0 0 0; 1 2 1]/8;
dx = dy';
dy2 = [-1 -2 -1; 0 0 0; 0 0 0; 0 0 0; 1 2 1]/16;
dx2 = dy2';
maskarea = sum(sum(finalmask));
for channel = 1:3

   LesionData = f(:,:,channel);
   minv = min(min(LesionData));
   adLesionData = LesionData - minv;

%    imshow(LesionData,[]);
   histoData = reshape(LesionData(finalmask),1,[]);
   temp_mean = mean(histoData);
   temp_m2 = moment(histoData,2);
   temp_m3 = moment(histoData,3);
   temp_m4 = moment(histoData,4);

   masked_m2 = sqrt(temp_m2);
   masked_m3 = power(abs(temp_m3),1/3).*sign(temp_m3);
   masked_m4 = power(temp_m4,1/4);
   dq = quantile(histoData,[0.01,0.05,0.25,0.5,0.75,0.95,0.99]);
   cov = log(10000.0*masked_m2 + 1.0);
   q90cov = log(10000.0*(dq(6)-dq(2)) + 1.0);
   if (abs(temp_mean) > 0)
       cov = log(10000.0*masked_m2/abs(temp_mean) + 1.0);
   end
   if (abs(dq(4)) > 0)
       q90cov = log(10000.0*(dq(6)-dq(2))/abs(dq(4)) + 1.00);
   end
   [N,edges] = histcounts(histoData,32);
   N = N/sum(N);
   N = N.*log(N);
   TF = isnan(N);
   N(TF) = 0;
   entropy = -sum(N);

   quantiativeLesionData =
[quantiativeLesionData,temp_mean,masked_m2,masked_m3,masked_m4,entropy,cov,q90cov];


   volume = sum(sum(adLesionData(finalmask)));
   grad = (abs(conv2(LesionData,dx,'same')) + abs(conv2(LesionData,dy,'same')))/2;
   grad2 = (abs(conv2(LesionData,dx2,'same')) + abs(conv2(LesionData,dy2,'same')))/2;
   gradsum = sum(sum(grad(finalmask)));
   gradsum2 = sum(sum(grad2(finalmask)));
   surface = 128*gradsum + maskarea;
   compactness = 512*volume/surface^(3/2);
   quantiativeLesionData =
[quantiativeLesionData,log(1+volume),log(1+surface),compactness,gradsum/volume,gradsum2/gradsum];
   %%%%%%%%%%%%%%%% Temperature GLCM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   glcm_1 =
graycomatrix(LesionData(finalmask),'NumLevels',numGLCMBins,'GrayLimits',[-1.0,1.50],'Offset',[0 2;
-2 2; -2 0; -2 -2],'Symmetric',true);
   glcm_2 =
graycomatrix(LesionData(finalmask),'NumLevels',numGLCMBins,'GrayLimits',[-1.0,1.50],'Offset',[0 4;
-3 3; -4 0; -3 -3],'Symmetric',true);
   glcm_3 =
graycomatrix(LesionData(finalmask),'NumLevels',numGLCMBins,'GrayLimits',[-1.0,1.50],'Offset',[0 8;
-6 6; -8 0; -6 -6],'Symmetric',true);
   glcm_4 =
graycomatrix(LesionData(finalmask),'NumLevels',numGLCMBins,'GrayLimits',[-1.0,1.50],'Offset',[0 16;
-11 11; -16 0; -11 -11],'Symmetric',true);
```

```
    glcm_1 = sum(glcm_1,3);
    glcm_2 = sum(glcm_2,3);
    glcm_3 = sum(glcm_3,3);
    glcm_4 = sum(glcm_4,3);
    GLCM_stats_1 = graycoprops(glcm_1);
    GLCM_stats_2 = graycoprops(glcm_2);
    GLCM_stats_3 = graycoprops(glcm_3);
    GLCM_stats_4 = graycoprops(glcm_4);
    quantiativeLesionData =
[quantiativeLesionData,GLCM_stats_1.Contrast,GLCM_stats_1.Correlation,GLCM_stats_1.Energy,GLCM_stats
_1.Homogeneity];
    quantiativeLesionData =
[quantiativeLesionData,GLCM_stats_2.Contrast,GLCM_stats_2.Correlation,GLCM_stats_2.Energy,GLCM_stats
_2.Homogeneity];
    quantiativeLesionData =
[quantiativeLesionData,GLCM_stats_3.Contrast,GLCM_stats_3.Correlation,GLCM_stats_3.Energy,GLCM_stats
_3.Homogeneity];
    quantiativeLesionData =
[quantiativeLesionData,GLCM_stats_4.Contrast,GLCM_stats_4.Correlation,GLCM_stats_4.Energy,GLCM_stats
_4.Homogeneity];
    GLCMSlope =
[abs(log(1/2*(1.001-GLCM_stats_1.Correlation)/(1.001-GLCM_stats_2.Correlation)))/log(2)];
    GLCMSlope =
[GLCMSlope,abs(log(1/2*(1.001-GLCM_stats_2.Correlation)/(1.001-GLCM_stats_3.Correlation)))/log(2)];
    GLCMSlope =
[GLCMSlope,abs(log(1/4*(1.001-GLCM_stats_1.Correlation)/(1.001-GLCM_stats_3.Correlation)))/log(4)];
    GLCMSlope =
[GLCMSlope,abs(log(1/2*(1.001-GLCM_stats_3.Correlation)/(1.001-GLCM_stats_4.Correlation)))/log(2)];
    GLCMSlope =
[GLCMSlope,abs(log(1/4*(1.001-GLCM_stats_2.Correlation)/(1.001-GLCM_stats_4.Correlation)))/log(4)];
    GLCMSlope =
[GLCMSlope,abs(log(1/8*(1.001-GLCM_stats_1.Correlation)/(1.001-GLCM_stats_4.Correlation)))/log(8)];
    meanGLCMSlope = mean(GLCMSlope);
    quantiativeLesionData = [quantiativeLesionData,meanGLCMSlope];
    GLCMSlope = [abs(log(1/2*GLCM_stats_1.Contrast/GLCM_stats_2.Contrast))/log(2)];
    GLCMSlope = [GLCMSlope,abs(log(1/2*GLCM_stats_2.Contrast/GLCM_stats_3.Contrast))/log(2)];
    GLCMSlope = [GLCMSlope,abs(log(1/4*GLCM_stats_1.Contrast/GLCM_stats_3.Contrast))/log(4)];
    GLCMSlope = [GLCMSlope,abs(log(1/2*GLCM_stats_3.Contrast/GLCM_stats_4.Contrast))/log(2)];
    GLCMSlope = [GLCMSlope,abs(log(1/4*GLCM_stats_2.Contrast/GLCM_stats_4.Contrast))/log(4)];
    GLCMSlope = [GLCMSlope,abs(log(1/8*GLCM_stats_1.Contrast/GLCM_stats_4.Contrast))/log(8)];
    meanGLCMSlope = mean(GLCMSlope);
    quantiativeLesionData = [quantiativeLesionData,meanGLCMSlope];
    GLCMSlope = [abs(log(2*GLCM_stats_1.Energy/GLCM_stats_2.Energy))/log(2)];
    GLCMSlope = [GLCMSlope,abs(log(2*GLCM_stats_2.Energy/GLCM_stats_3.Energy))/log(2)];
    GLCMSlope = [GLCMSlope,abs(log(4*GLCM_stats_1.Energy/GLCM_stats_3.Energy))/log(4)];
    GLCMSlope = [GLCMSlope,abs(log(2*GLCM_stats_3.Energy/GLCM_stats_4.Energy))/log(2)];
    GLCMSlope = [GLCMSlope,abs(log(4*GLCM_stats_2.Energy/GLCM_stats_4.Energy))/log(4)];
    GLCMSlope = [GLCMSlope,abs(log(8*GLCM_stats_1.Energy/GLCM_stats_4.Energy))/log(8)];
    meanGLCMSlope = mean(GLCMSlope);
    quantiativeLesionData = [quantiativeLesionData,meanGLCMSlope];
    GLCMSlope =
[abs(log(1/2*(1.001-GLCM_stats_1.Homogeneity)/(1.001-GLCM_stats_2.Homogeneity)))/log(2)];
    GLCMSlope =
[GLCMSlope,abs(log(1/2*(1.001-GLCM_stats_2.Homogeneity)/(1.001-GLCM_stats_3.Homogeneity)))/log(2)];
    GLCMSlope =
[GLCMSlope,abs(log(1/4*(1.001-GLCM_stats_1.Homogeneity)/(1.001-GLCM_stats_3.Homogeneity)))/log(4)];
    GLCMSlope =
[GLCMSlope,abs(log(1/2*(1.001-GLCM_stats_3.Homogeneity)/(1.001-GLCM_stats_4.Homogeneity)))/log(2)];
    GLCMSlope =
[GLCMSlope,abs(log(1/4*(1.001-GLCM_stats_2.Homogeneity)/(1.001-GLCM_stats_4.Homogeneity)))/log(4)];
    GLCMSlope =
[GLCMSlope,abs(log(1/8*(1.001-GLCM_stats_1.Homogeneity)/(1.001-GLCM_stats_4.Homogeneity)))/log(8)];
    meanGLCMSlope = mean(GLCMSlope);
    quantiativeLesionData = [quantiativeLesionData,meanGLCMSlope];
end
quantiativeLesionData = [quantiativeLesionData,Ggain,Gmedian];
allquantiativeLesionData = [allquantiativeLesionData;quantiativeLesionData];
%% Extract Control features
maskarea = sum(sum(controlmask));
for channel = 1:3
```

```
   ControlData = f(:,:,channel);
   minv = min(min(ControlData));
   aControlData = ControlData - minv;
%    imshow(ControlData,[]);
   histoData = reshape(ControlData(controlmask),1,[]);
   temp_mean = mean(histoData);
   temp_m2 = moment(histoData,2);
   temp_m3 = moment(histoData,3);
   temp_m4 = moment(histoData,4);

   masked_m2 = sqrt(moment(histoData,2));
   masked_m3 = moment(histoData,3);
   masked_m3 = power(abs(masked_m3),1/3).*sign(masked_m3);
   masked_m4 = power(moment(histoData,4),1/4);
   dq = quantile(histoData,[0.01,0.05,0.25,0.5,0.75,0.95,0.99]);
   cov=0;
   q90cov=0;
   if (abs(temp_mean) > 0)
       cov = log(10000.0*masked_m2/abs(temp_mean) + 1.0);
   end
   if (abs(dq(4)) > 0)
       q90cov = log(10000.0*(dq(6)-dq(2))/abs(dq(4)) + 1.0);
   end
   [N,edges] = histcounts(histoData,32);
   N = N/sum(N);
   N = N.*log(N);
   TF = isnan(N);
   N(TF) = 0;
   entropy = -sum(N);

   quantiativeControlData =
[quantiativeControlData,temp_mean,masked_m2,masked_m3,masked_m4,entropy,cov,q90cov];
   volume = sum(sum(aControlData(controlmask)));
   grad = (abs(conv2(ControlData,dx,'same')) + abs(conv2(ControlData,dy,'same')))/2;
   grad2 = (abs(conv2(ControlData,dx2,'same')) + abs(conv2(ControlData,dy2,'same')))/2;
   gradsum = sum(sum(grad(controlmask)));
   gradsum2 = sum(sum(grad2(controlmask)));
   surface = 128*gradsum + maskarea;
   compactness = 512*volume/surface^(3/2);
   quantiativeControlData =
[quantiativeControlData,log(1.0+volume+1),log(1.0+surface),compactness,gradsum/volume,gradsum2/grads
um];

   %%%%%%%%%%%%%%%%  GLCM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   glcm_1 =
graycomatrix(ControlData(controlmask),'NumLevels',numGLCMBins,'GrayLimits',[-1.0,1.50],'Offset',[0
2; -2 2; -2 0; -2 -2],'Symmetric',true);
   glcm_2 =
graycomatrix(ControlData(controlmask),'NumLevels',numGLCMBins,'GrayLimits',[-1.0,1.50],'Offset',[0
4; -3 3; -4 0; -3 -3],'Symmetric',true);
   glcm_3 =
graycomatrix(ControlData(controlmask),'NumLevels',numGLCMBins,'GrayLimits',[-1.0,1.50],'Offset',[0
8; -6 6; -8 0; -6 -6],'Symmetric',true);
   glcm_4 =
graycomatrix(ControlData(controlmask),'NumLevels',numGLCMBins,'GrayLimits',[-1.0,1.50],'Offset',[0
16; -11 11; -16 0; -11 -11],'Symmetric',true);
   glcm_1 = sum(glcm_1,3);
   glcm_2 = sum(glcm_2,3);
   glcm_3 = sum(glcm_3,3);
   glcm_4 = sum(glcm_4,3);
   GLCM_stats_1 = graycoprops(glcm_1);
   GLCM_stats_2 = graycoprops(glcm_2);
   GLCM_stats_3 = graycoprops(glcm_3);
   GLCM_stats_4 = graycoprops(glcm_4);
   quantiativeControlData =
[quantiativeControlData,GLCM_stats_1.Contrast,GLCM_stats_1.Correlation,GLCM_stats_1.Energy,GLCM_stat
s_1.Homogeneity];
   quantiativeControlData =
[quantiativeControlData,GLCM_stats_2.Contrast,GLCM_stats_2.Correlation,GLCM_stats_2.Energy,GLCM_stat
s_2.Homogeneity];
```

```
   quantiativeControlData =
[quantiativeControlData,GLCM_stats_3.Contrast,GLCM_stats_3.Correlation,GLCM_stats_3.Energy,GLCM_stat
s_3.Homogeneity];
   quantiativeControlData =
[quantiativeControlData,GLCM_stats_4.Contrast,GLCM_stats_4.Correlation,GLCM_stats_4.Energy,GLCM_stat
s_4.Homogeneity];
   GLCMSlope =
[abs(log(1/2*(1.001-GLCM_stats_1.Correlation)/(1.001-GLCM_stats_2.Correlation)))/log(2)];
   GLCMSlope =
[GLCMSlope,abs(log(1/2*(1.001-GLCM_stats_2.Correlation)/(1.001-GLCM_stats_3.Correlation)))/log(2)];
   GLCMSlope =
[GLCMSlope,abs(log(1/4*(1.001-GLCM_stats_1.Correlation)/(1.001-GLCM_stats_3.Correlation)))/log(4)];
   GLCMSlope =
[GLCMSlope,abs(log(1/2*(1.001-GLCM_stats_3.Correlation)/(1.001-GLCM_stats_4.Correlation)))/log(2)];
   GLCMSlope =
[GLCMSlope,abs(log(1/4*(1.001-GLCM_stats_2.Correlation)/(1.001-GLCM_stats_4.Correlation)))/log(4)];
   GLCMSlope =
[GLCMSlope,abs(log(1/8*(1.001-GLCM_stats_1.Correlation)/(1.001-GLCM_stats_4.Correlation)))/log(8)];
   meanGLCMSlope = mean(GLCMSlope);
   quantiativeControlData = [quantiativeControlData,meanGLCMSlope];
   GLCMSlope = [abs(log(1/2*GLCM_stats_1.Contrast/GLCM_stats_2.Contrast))/log(2)];
   GLCMSlope = [GLCMSlope,abs(log(1/2*GLCM_stats_2.Contrast/GLCM_stats_3.Contrast))/log(2)];
   GLCMSlope = [GLCMSlope,abs(log(1/4*GLCM_stats_1.Contrast/GLCM_stats_3.Contrast))/log(4)];
   GLCMSlope = [GLCMSlope,abs(log(1/2*GLCM_stats_3.Contrast/GLCM_stats_4.Contrast))/log(2)];
   GLCMSlope = [GLCMSlope,abs(log(1/4*GLCM_stats_2.Contrast/GLCM_stats_4.Contrast))/log(4)];
   GLCMSlope = [GLCMSlope,abs(log(1/8*GLCM_stats_1.Contrast/GLCM_stats_4.Contrast))/log(8)];
   meanGLCMSlope = mean(GLCMSlope);
   quantiativeControlData = [quantiativeControlData,meanGLCMSlope];
   GLCMSlope = [abs(log(2*GLCM_stats_1.Energy/GLCM_stats_2.Energy))/log(2)];
   GLCMSlope = [GLCMSlope,abs(log(2*GLCM_stats_2.Energy/GLCM_stats_3.Energy))/log(2)];
   GLCMSlope = [GLCMSlope,abs(log(4*GLCM_stats_1.Energy/GLCM_stats_3.Energy))/log(4)];
   GLCMSlope = [GLCMSlope,abs(log(2*GLCM_stats_3.Energy/GLCM_stats_4.Energy))/log(2)];
   GLCMSlope = [GLCMSlope,abs(log(4*GLCM_stats_2.Energy/GLCM_stats_4.Energy))/log(4)];
   GLCMSlope = [GLCMSlope,abs(log(8*GLCM_stats_1.Energy/GLCM_stats_4.Energy))/log(8)];
   meanGLCMSlope = mean(GLCMSlope);
   quantiativeControlData = [quantiativeControlData,meanGLCMSlope];
   GLCMSlope =
[abs(log(1/2*(1.001-GLCM_stats_1.Homogeneity)/(1.001-GLCM_stats_2.Homogeneity)))/log(2)];
   GLCMSlope =
[GLCMSlope,abs(log(1/2*(1.001-GLCM_stats_2.Homogeneity)/(1.001-GLCM_stats_3.Homogeneity)))/log(2)];
   GLCMSlope =
[GLCMSlope,abs(log(1/4*(1.001-GLCM_stats_1.Homogeneity)/(1.001-GLCM_stats_3.Homogeneity)))/log(4)];
   GLCMSlope =
[GLCMSlope,abs(log(1/2*(1.001-GLCM_stats_3.Homogeneity)/(1.001-GLCM_stats_4.Homogeneity)))/log(2)];
   GLCMSlope =
[GLCMSlope,abs(log(1/4*(1.001-GLCM_stats_2.Homogeneity)/(1.001-GLCM_stats_4.Homogeneity)))/log(4)];
   GLCMSlope =
[GLCMSlope,abs(log(1/8*(1.001-GLCM_stats_1.Homogeneity)/(1.001-GLCM_stats_4.Homogeneity)))/log(8)];
   meanGLCMSlope = mean(GLCMSlope);
   quantiativeControlData = [quantiativeControlData,meanGLCMSlope];
end
allquantiativeControlData = [allquantiativeControlData;quantiativeControlData];
```

2. Código utilizado en R.Studio

Link al código en github:
https://github.com/DaciaMtz/Imagenologia/blob/main/Melanoma%20Segmentation%20Code/MelanomaVsSeborrheic_.Rmd

```
title: "MelanomavsSeborreic"
author: "Jose Tamez"
date: "4/9/2021"
output: html_document
editor_options:
  chunk_output_type: console
---
```{r setup, include=FALSE}
```
```

```
knitr::opts_chunk$set(collapse = TRUE, warning = FALSE, message = FALSE,comment =
"#>")

```
Melanoma vs Seborreic_Keratosis
```{r functions,echo = FALSE }

library("FRESA.CAD")
library("e1071")
library("fastAdaboost")
library(gbm)
library(caret)
```

```{r traincontrol}
tunningctrl <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 3
)
noTuningControl <- trainControl(method = "none")
```
Loading data sets
```{r FRESA Map, results = "hide", dpi=300, fig.height= 6.0, fig.width= 8.0}
op <- par(no.readonly = TRUE)
MelanomaFeatures <-
read.csv("D:/MATLAB/SKINCCANCER/MatlabScripts/MelanomaLesionFeatures.csv",
header=FALSE)
SeborrheicFeatures <-
read.csv("D:/MATLAB/SKINCCANCER/MatlabScripts/SeborrheicLesionFeatures.csv",
header=FALSE)
NevusFeatures <-
read.csv("D:/MATLAB/SKINCCANCER/MatlabScripts/NevusLesionFeatures.csv",
header=FALSE)
MelanomaControlFeatures <-
read.csv("D:/MATLAB/SKINCCANCER/MatlabScripts/MelanomaControlFeatures.csv",
header=FALSE)
SeborrheicControlFeatures <-
read.csv("D:/MATLAB/SKINCCANCER/MatlabScripts/SeborrheicControlFeatures.csv",
header=FALSE)
NevusControlFeatures <-
read.csv("D:/MATLAB/SKINCCANCER/MatlabScripts/NevusControlFeatures.csv",
header=FALSE)

colnames(MelanomaControlFeatures) <-
paste("C",colnames(MelanomaControlFeatures),sep="")
colnames(SeborrheicControlFeatures) <-
paste("C",colnames(SeborrheicControlFeatures),sep="")
colnames(NevusControlFeatures) <- paste("C",colnames(NevusControlFeatures),sep="")

sum(is.na(MelanomaFeatures))
sum(is.na(SeborrheicFeatures))
sum(is.na(NevusFeatures))
sum(is.na(MelanomaControlFeatures))
sum(is.na(NevusControlFeatures))

channel1 <- c(1:32)
channel2 <- c(33:64)
channel3 <- c(65:96)
```

```
MMelanomaFeatures <- (  MelanomaFeatures[,channel1] +
                        MelanomaFeatures[,channel2] +
                        MelanomaFeatures[,channel3] )/3.0
SMelanomaFeatures <- (  abs(MelanomaFeatures[,channel1] - MMelanomaFeatures) +
                        abs(MelanomaFeatures[,channel2] - MMelanomaFeatures) +
                        abs(MelanomaFeatures[,channel3] - MMelanomaFeatures) )/3.0

colnames(MMelanomaFeatures) <- paste("M",colnames(MMelanomaFeatures),sep="")
colnames(SMelanomaFeatures) <- paste("S",colnames(SMelanomaFeatures),sep="")

MelanomaFeatures <-
cbind(MelanomaFeatures,SMelanomaFeatures/(0.01+abs(MMelanomaFeatures)))

MNevusFeatures <- (  NevusFeatures[,channel1] +
                        NevusFeatures[,channel2] +
                        NevusFeatures[,channel3] )/3.0
SNevusFeatures <- (  abs(NevusFeatures[,channel1] - MNevusFeatures) +
                        abs(NevusFeatures[,channel2] - MNevusFeatures) +
                        abs(NevusFeatures[,channel3] - MNevusFeatures) )/3.0

colnames(MNevusFeatures) <- paste("M",colnames(MNevusFeatures),sep="")
colnames(SNevusFeatures) <- paste("S",colnames(SNevusFeatures),sep="")

NevusFeatures <- cbind(NevusFeatures,SNevusFeatures/(0.001+abs(MNevusFeatures)))

MSeborrheicFeatures <- (  SeborrheicFeatures[,channel1] +
                          SeborrheicFeatures[,channel2] +
                          SeborrheicFeatures[,channel3] )/3.0
SSeborrheicFeatures <- (abs(SeborrheicFeatures[,channel1] - MSeborrheicFeatures) +
                        abs(SeborrheicFeatures[,channel2] - MSeborrheicFeatures) +
                        abs(SeborrheicFeatures[,channel3] -
MSeborrheicFeatures))/3.0
colnames(MSeborrheicFeatures) <- paste("M",colnames(MSeborrheicFeatures),sep="")
colnames(SSeborrheicFeatures) <- paste("S",colnames(SSeborrheicFeatures),sep="")

SeborrheicFeatures <-
cbind(SeborrheicFeatures,SSeborrheicFeatures/(0.001+abs(MSeborrheicFeatures)))

MMelanomaControlFeatures <- (  MelanomaControlFeatures[,channel1] +
                               MelanomaControlFeatures[,channel2] +
                               MelanomaControlFeatures[,channel3] )/3.0

SMelanomaControlFeatures <- (abs(MelanomaControlFeatures[,channel1] -
MMelanomaControlFeatures) +
                             abs(MelanomaControlFeatures[,channel2] -
MMelanomaControlFeatures) +
                             abs(MelanomaControlFeatures[,channel3] -
MMelanomaControlFeatures))/3.0
colnames(MMelanomaControlFeatures) <-
paste("M",colnames(MMelanomaControlFeatures),sep="")
colnames(SMelanomaControlFeatures) <-
paste("S",colnames(SMelanomaControlFeatures),sep="")


MelanomaControlFeatures <-
cbind(MelanomaControlFeatures,SMelanomaControlFeatures/(0.001+abs(MMelanomaControlF
eatures)))
```

```
MNevusControlFeatures <- (  NevusControlFeatures[,channel1] +
                              NevusControlFeatures[,channel2] +
                              NevusControlFeatures[,channel3] )/3.0
SNevusControlFeatures <- (abs(NevusControlFeatures[,channel1] -
MNevusControlFeatures) +
                            abs(NevusControlFeatures[,channel2] -
MNevusControlFeatures) +
                            abs(NevusControlFeatures[,channel3] -
MNevusControlFeatures))/3.0
colnames(MNevusControlFeatures) <-
paste("M",colnames(MNevusControlFeatures),sep="")
colnames(SNevusControlFeatures) <-
paste("S",colnames(SNevusControlFeatures),sep="")

NevusControlFeatures <-
cbind(NevusControlFeatures,SNevusControlFeatures/(0.001+abs(MNevusControlFeatures))
)


MSeborrheicControlFeatures <- (  SeborrheicControlFeatures[,channel1] +
                                 SeborrheicControlFeatures[,channel2] +
                                 SeborrheicControlFeatures[,channel3] )/3.0
SSeborrheicControlFeatures <- (abs(SeborrheicControlFeatures[,channel1] -
MSeborrheicControlFeatures) +
                               abs(SeborrheicControlFeatures[,channel2] -
MSeborrheicControlFeatures) +
                               abs(SeborrheicControlFeatures[,channel3] -
MSeborrheicControlFeatures))/3.0
colnames(MSeborrheicControlFeatures) <-
paste("M",colnames(MSeborrheicControlFeatures),sep="")
colnames(SSeborrheicControlFeatures) <-
paste("S",colnames(SSeborrheicControlFeatures),sep="")

SeborrheicControlFeatures <- cbind(SeborrheicControlFeatures,

SSeborrheicControlFeatures/(0.001+abs(MSeborrheicControlFeatures)))

CtrDiff <- MelanomaFeatures[,1:ncol(MelanomaControlFeatures)] -
MelanomaControlFeatures;
colnames(CtrDiff) <- colnames(MelanomaControlFeatures)
MelanomaFeatures <- cbind(MelanomaFeatures,CtrDiff)

CtrDiff <- NevusFeatures[,1:ncol(MelanomaControlFeatures)] - NevusControlFeatures;
colnames(CtrDiff) <- colnames(NevusControlFeatures)
NevusFeatures <- cbind(NevusFeatures,CtrDiff)

CtrDiff <- SeborrheicFeatures[,1:ncol(MelanomaControlFeatures)] -
SeborrheicControlFeatures;
colnames(CtrDiff) <- colnames(SeborrheicControlFeatures)
SeborrheicFeatures <- cbind(SeborrheicFeatures,CtrDiff)

MelanomaFeatures <- MelanomaFeatures[complete.cases(MelanomaFeatures),]
NevusFeatures <- NevusFeatures[complete.cases(NevusFeatures),]
SeborrheicFeatures <- SeborrheicFeatures[complete.cases(SeborrheicFeatures),]

MelanomaFeatures$Class <- rep(1,nrow(MelanomaFeatures))
NevusFeatures$Class <- rep(0,nrow(NevusFeatures))
SeborrheicFeatures$Class <- rep(0,nrow(SeborrheicFeatures))
```

```
MelanomaSeborrheicNevus <- rbind(MelanomaFeatures,SeborrheicFeatures,NevusFeatures)
#MelanomaSeborrheicNevus <- rbind(MelanomaFeatures,SeborrheicFeatures)
#MelanomaSeborrheicNevus <- rbind(MelanomaFeatures,NevusFeatures)
table(MelanomaSeborrheicNevus$Class)
```

```
## The Heatmap
```{r, dpi=300, fig.height= 6.0, fig.width= 8.0}
par(op)

boxplot(MelanomaSeborrheicNevus)

hm <- heatMaps(Outcome = "Class",
#                data = MelanomaSeborrheicNevus,
                data = MelanomaSeborrheicNevus[1:900,],
                title = "Heat Map:",Scale = TRUE,
                hCluster = "col",cexRow = 0.75,cexCol = 0.5,srtCol = 45)

par(op)
```
Learning Melanoma with KNN
```{r}
cvKNN <- randomCV(MelanomaSeborrheicNevus,"Class",
                KNN_method,
                trainFraction = 0.70,
                repetitions = 100,
                classSamplingType = "Ba",
                featureSelectionFunction = univariate_KS,
                featureSelection.control = list(pvalue=0.05,limit= -1),
                kn=5
                )

```
Plot performance
```{r}
performace <- predictionStats_binary(cvKNN$medianTest,"KNN")
par(op)

```

```{r}
cvSVM <- randomCV(MelanomaSeborrheicNevus,"Class",
                e1071::svm,
                asFactor = TRUE,
                trainFraction = 0.70,
                repetitions = 100,
                classSamplingType = "Ba",
                featureSelectionFunction = univariate_KS,
                featureSelection.control = list(pvalue=0.05,limit= -1),
                probability=TRUE
                )

```
SVM performance
```{r}
performace <- predictionStats_binary(cvSVM$medianTest,"SVM")
par(op)

```
```

Learning Melanoma with Naive Bayes

```{r}
cvNB <- randomCV(MelanomaSeborrheicNevus,"Class",
                NAIVE_BAYES,
                trainFraction = 0.70,
                repetitions = 100,
                classSamplingType = "Ba",
                featureSelectionFunction = univariate_KS,
                featureSelection.control = list(pvalue=0.05,limit=0.1),
#                pca=FALSE
                )

```

Plot performance

```{r}
performace <- predictionStats_binary(cvNB$medianTest,"Naive Bayes")
par(op)

```

Learning Melanoma with Adaboost

```{r}
cvADA <- randomCV(MelanomaSeborrheicNevus,"Class",
                adaboost,
                trainFraction = 0.70,
                repetitions = 100,
                classSamplingType = "Ba",
                nIter=50)

```

Plot performance

```{r}
performace <- predictionStats_binary(cvADA$medianTest,"Adaboost")
par(op)
```

```{r}
cvgbm <- randomCV(MelanomaSeborrheicNevus,"Class",
                train,
                asFactor = TRUE,
                trainFraction = 0.70,
                repetitions = 100,
                classSamplingType = "Ba",
                method = "gbm",
                trControl = tunningctrl,
#                preProc = c("center", "scale"),
                verbose = FALSE
              )
```

```{r}
performace <- predictionStats_binary(cvgbm$medianTest,"GBM")
par(op)

```