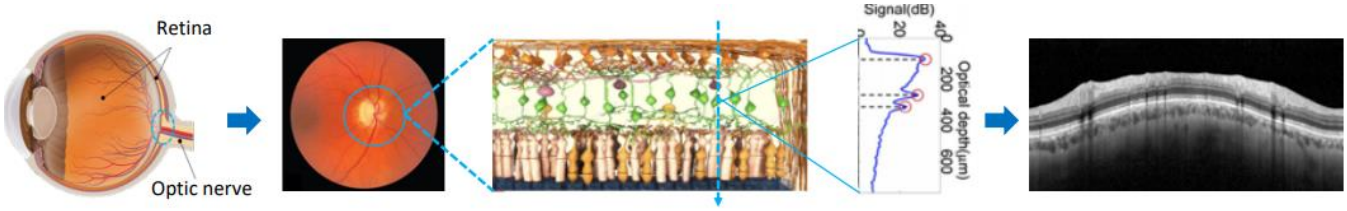
El objetivo es diseñar y desarrollar un **sistema de ayuda al diagnóstico** para **determinar** si un paciente tiene **glaucoma**.

El glaucoma afecta al **nervio óptico** causando **cambios estructurales** y la prueba para detectarlo es a través de muestras de **tomografía por coherencia óptica** (**OCT**) debido a que proporciona **información sobre la deterioración de las capas de fibras de la retina**.

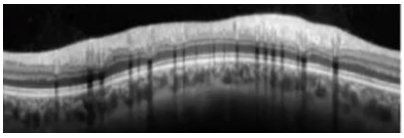
Lo primero que se debe hacer es una recopilación bibliográfica para **conocer el estado del arte**.

Las imágenes con las que se trabajará se han obtenido mediante este proceso:

- Del globo ocular, se obtiene una imagen de fondo de ojo.
- Del disco óptico se obtiene, la capa de fibras de retina, que, leída por columnas, se genera una señal A-scan.
- Del conjunto de señales A-scan, se obtiene la imagen OCT.



Las muestras que tenemos son las siguientes:

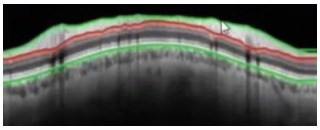
Imagen OCT	Capa de fibras nerviosas de la retina (RNFL)	Región de interés de la retina
		

La **máscara más importante es la RNFL** porque el **grosor de la capa es un buen indicador** de si el paciente sufrirá la enfermedad.
Un **grosor fino indica que el paciente está enfermo**, mientras que un grosor amplio que está sano.

En primer lugar, buscamos **comprender el problema** cargando el dataset para observar el **número de instancias**, **clases** y **frecuencia**.
Instancias (filas): 189 | Clases: **Glaucoma** y **Healthy** | Frecuencia: 93 y 96
De este estudio vemos que las **imágenes OCT están etiquetadas** y que es un dataset **balanceado**.

Para familiarizarnos con las imágenes vamos a superponer las máscaras (RNFL y RETINA) sobre la imagen original. Para ello, se utilizará:

- cv2.findContours() para detectar los contornos de las máscaras.
- cv2.drawContours() para dibujar los contornos.



Utilizamos una **validación cruzada**: kFold(n_splits = 5, shuffle = True, random_state = 42) para hacer la partición de los datos.
Los datos se han particionado: bolsa **train** (glaucoma 77, healthy 75) y bolsa **test** (glaucoma 16, healthy 21).
Por último, se **aleatoriza el orden de las muestra**s en las bolsas para hacerlas más robustas.

En la **extracción de características** se van a obtener estadísticos **unidimensionales de la RNFL** y **bidimensionales de la RETINA**.

Sobre la máscara RNFL se pretende determinar el grosor de la capa de fibras por columnas. Este grosor se halla restando la posición del primer píxel blanco frente al último gracias a que la imagen es binaria y tiene pixeles negros (0) y blancos (255).

En este momento tenemos 768 valores correspondientes a cada una de las columnas thickness_rnfl = [20, 8, …, 8, 20]). Sin embargo, son muchos más valores que el número de instancias y el modelo podría caer en overfitting. Como de esta manera no nos aporta valor vamos a **transformar esa información en diferentes características**.

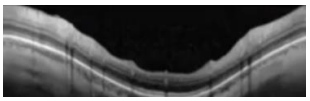
- Basadas en medidas de tendencia central: media, mediana.
- Basadas en medidas de dispersión: desviación estándar.
- Distribución: asimetría, curtosis.
- Otras: mínimo, máximo

El máximo y/o el mínimo, pueden ser interesantes porque un mínimo muy pequeño ej.: 2, podría asociarse a muestras glaucomatosas.

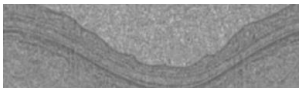
Sobre la máscara RETINA se pretende determinar la boundingbox para aplicárselo sobre la imagen original y extraer dos descriptores de textura. Al haber estudiado el problema previamente, se descartan descriptores de forma, geométricas, etc.

- **Gray-Level Coocurrence Matrix** (**GLCM**): la matriz de concurrencia greycomatrix () extraerá características del cambio de intensidad de los pixeles. La función utiliza un parámetro "angles" que se debe definir a "90" para trabajar en vertical.
  (1) contraste (2) disimilitud (3) homogeneidad (4) ASM (5) energía (6) correlacion
- **Local Binary Patterns** (**LBP**): este extractor de necesita un radio (R) y unos vecinos (P)
  Se han extraído 10 características LBP: LBP1, LBP2, LB3, LBP4, LBP5, LBP6, etc.

CROP RETINA



LBP RETINA



La extracción de características finaliza generando un "**fingerprint**" o matriz de **características para cada imagen** de las extraídas en la **RNFL** y **RETINA** junto con la **clase** en formato numérico (0 = sano, 1 = enfermo)
El fingerprint_train será una matriz (152, 24), es decir, 152 muestras (instancias) y 23 características más la clase para el train.
El fingerprint_test será una matriz (37, 24), es decir, 37 muestras (instancias) y 23 características más la clase para el test.

La **selección de características** consistirá en **determinar cuales de ellas son relevantes para discriminar** entre pacientes.

Para poder determinar que características son relevantes, **estandarizaremos la matriz de train** con StandardScaler() para acotarlos en el mismo rango. El primer paso es entrenar al estandarizar con el fit() y aplicarle el transform() a la matriz de train.

Nota: la estandarización coge cada uno de los valores para una columna, se lo resta a la media de la columna (centrado) y lo divide entre la desviación típica (escalado). Dicha media y desviación se puede extraer del estandarizador.

En este punto **se estudia si los datos siguen una distribución normal de media 0 y desviación típica 1 N(0, 1)**. Este estudio se hace con la función de Kolmogorov-Smirnov (kstest).

Empezamos **definiendo un nivel de significancia** (alpha = 0.01) **o lo que es lo mismo un nivel de confianza** del 99% porque vamos a **elaborar un contraste de hipótesis** para estudiar si la distribución de cada una de las características es normal N(0, 1)

Para hacer ese contraste de hipótesis se **extraer un pvalor a partir de la prueba de kstest para compararlo con el nivel de significancia** y **si el pvalor es menor o igual se rechaza la hipótesis nula y la característica se dice que no sigue una distribución normal**.

Por el contrario, **si el pvalor es mayor al nivel de significancia** no hay evidencias para rechazar la hipótesis nula **y asumiremos que la característica sigue una distribución normal N(0, 1)**.

Si la característica no sigue la distribución normal se asigna un 0 a un nuevo vector. En caso contrario, un 1.

h_norm = [1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Una vez se tiene el vector de distribución, **se estudia la capacidad discriminativa de los atributos en función de su distribución**.

El **ttest_ind** y **mannhwhitneyu** permiten hacer una comparación entre los valores de las características asociados a cada clase.

El **ttest_ind** aplica una comparación de **medias** a las características que seguían una distribución normal (1).

El **mannhwhitneyu** aplica una comparación de **medianas** a las características que no seguían una distribución normal (0).

Empezamos separando las muestras de los pacientes con glaucoma (glaucoma_data) de los sanos (healthy_data).

Para cada característica de h_norm se mira su distribución:

- Si es 0 se extrae el pvalue = mannwhitneyu(glaucoma_data, healthy_data)
- Si es 1 se extrae el pvalue = ttest_ind(glaucoma_data, healthy_data)

Una vez extraído el pvalue se hace un contraste de hipótesis que consiste en estudiar el poder discriminatorio de las características.

H0: independencia entre la característica y la clase.

Para hacer el contraste de hipótesis **se compara el pvalor con el nivel de significancia y si el pvalor es menor o igual se rechaza la H0 y asumimos la dependencia entre la característica y la clase**.

Por el contrario, **si el pvalor es mayor al nivel de significancia** no hay evidencia para rechazar la H0 y asumimos que **la característica y la clase son independientes**.

Si la característica asume dependencia se asigna un 1 a un nuevo vector. En caso contrario, un 0.

h_disc = [1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0]

0: no discrimina y 1: sí distrimina.

Tener tantas características discriminativas indican que hemos elegido bien inicialmente y las que no lo son se deben eliminar.

Tras eliminarlas queda un vector de (152, 18), es decir, 182 instancias y 18 características para el train.

Eliminadas (no discriminatorias): COR, LBP10, LBP2, con, dis

Al visualizar la nueva matriz (sin las características anteriores) con un diagrama de cajas y bigotes, **observamos características correlacionadas.** Esto, indica que nos aportan la misma información y se deben eliminar para tener un modelo más robusto.

Para **estudiar el nivel de independencia entre pares de variables** (si dos variables aportan la misma información) realizamos un **análisis de correlación.** En este caso, **nos interesa la independencia** entre pares de variables.

Para descartar las características correlacionadas, definimos un umbral de descarte (th_cor = 0.9), es decir, descartaremos las que tengan un 90% de similitud. Para ello, utilizamos np.triu() que consiste en formar una matriz triangular superior que nos devolverá esta información.

Las características correlacionadas (a eliminar): mediana, max, E, LBP4, LBP5, LBP6, LBP7, LBP9

La matriz de train final tiene (152, 10), es decir, 150 muestras (instancias) y 10 características que son las que entrenarán al modelo.

En este momento, como sabemos **las características no discriminatorias y las correlacionadas las eliminaremos del conjunto de test y también lo normalizaremos**. La **normalización se tiene que hacer con la media (mu) y desviación típica (sigma)** porque como **la estandarización se ha realizado antes que la selección de atributos** hemos tenido que ir arrastrando la mu y sigma para poder hacer el centrado y escalado.

La última parte consiste en **modelar**, para ello definiremos los modelos de clasificación. Definimos dos para poder compararlos y **cubrir la perspectiva desde un punto de vista lineal y no linea**l.

Regresión logística (LOGR): permite clasificar datos de manera **lineal** y es un algoritmo que **se usa en problemas de clasificación**.

Los parámetros introducidos son:

- penalty: aplica una penalización al algoritmo.
- solver = optimizador del backward propagation y optimizar los pesos.
- max_iter: número máximo de épocas que el algoritmo va a hacer el backward y forward propagation.
- random_state: semilla para hacer reproducible el experimento.

Perceptrón multicapa (MLP): permite clasificar datos de manera **no lineal** basado en de redes neuronales que busca conexiones entre las neuronas durante el forward propagation. Evalúa el error cometido (nivel de exactitud alcanzado) y en base a eso durante el backward propagation actualiza los hiperparámetros de las conexiones de las neuronas para que en la siguiente iteración reduzca el error. **Se utiliza en problemas de clasificación y regresión**.

Los parámetros introducidos son:

- hidden_layer_sizes: definimos el número y tamaño de las capas ocultas ej.: [100,] una capa oculta de 100 neuronas.
- activation: función de activación.
- solver: optimizador del backward propagation y optimizar los pesos.
- batch_size: número de muestras que en cada época hacen el backward y forward propagation (potencias de dos).
- learning_rate: se puede fijar en adaptativo o contaste la tasa de aprendizaje.
- max_iter: número máximo de épocas que el algoritmo va a hacer el backward y forward propagation.
- random_state: semilla para hacer reproducible el experimento.

El parámetro **max_iter** no implica que el algoritmo entrene durante ese "n" total de épocas, sino que **entrenará hasta que converja**.

En este punto se hace la **partición interna de los datos con cross-validation** para ser robusto frente a la aleatorización en las particiones de datos de train y validación con el método de cross_val_score() que me permitirá conocer la **exactitud** que obtenga en cada una de las cinco bolsas de validación.

En primer lugar, **se puede observar que MLP funciona ligeramente mejor que LOGR**, aunque para poder afirmarlo habría que extraer más métricas. Sin embargo, es un **buen indicar porque es conjunto de datos está balanceado y devuelve el número de muestras que acierta frente al total**.

Cuando hemos optimizado todo lo posible los algoritmos, creamos un modelo definitivo al que **le pasamos todos los datos de train**.

Durante el entrenamiento se pueden obtener atributos como:

- Mínimo error cometido
- Número de iteraciones que se ha entrenado el modelo

Finalmente, se almacenan los modelos generados y se carga el conjunto de test para realizar la evaluación:

- LOGR.predict(X_test)
- MLP.predict(X_test)

En este caso, **se evaluarán diversas métricas de clasificación** (a diferencia del entrenamiento que solo se evaluó la precisión).

- Precisión
- Sensibilidad
- F1-Score
- Accuracy
- AUC

Los resultados finales sobre el test respecto a la matriz de confusión nos dicen que de los pacientes que tenían glaucoma los acierta todos y de los que estaban sanos falla dos.

(1) Una de las primeras acciones a realizar al comienza de un proyecto debería de ser:

a. **Llevar a cabo una exhaustiva recisión bibliográfica del estado del arte.**
b. Separar los datos en train y test mediante validación de tipo hold out.
c. Preprocesar los datos y prepararlos para la fase de modelado.
d. Ninguna de las anteriores.

(2) La matriz de correlación es:

a. Un descriptor de características de textura.
b. Una tabla de resultados que permite evaluar el rendimiento de un modelo predictivo enfrentando las predicciones y los valores reales (target).
c. **Una métrica para determinar el nivel de: independencia entre pares de atributos.**
d. Ninguna de las anteriores.

(3) ¿Qué función estadística permite determinar si las variables siguen una distribución N(0, 1), es decir, una distribución normal de media 0 y desviación estándar 1?

a. La prueba de T-Student.
b. La prueba de Analysis of Variance (ANOVA).
c. **La prueba de Kolmogorov-Smirnov.**
d. La prueba de Mann-Whitney U-test.

(4) Selecciona la opción correcta

a. El algoritmo de regresión logística se basa en la creación de diferentes árboles de decisión
b. El algoritmo "random forest" es uno de los más populares dentro de las técnicas de regresión logística.
c. El algoritmo de perceptrón multicapa siempre utiliza una tasa de aprendizaje (learning rate) contaste.
d. **En un algoritmo de perceptrón multicapa se pueden establecer tanto el número de capas ocultas como el número de neuronas dentro de cada capa oculta.**