PRÁCTICA PROCESOS GAUSSIANOS

Máster DATCOM

Extracción de Características en Imágenes

Autor
Alberto Armijo Ruiz

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

14 de marzo de 2019

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Procesos Guassianos

Los Procesos Gaussianos se tratan de un modelo que utiliza la aproximación bayesiana para calcular una distribución a posteriori sobre los datos; a diferencia de una Regresión Bayesiana normal, este define una transformación sobre los datos utilizando un kernel, de forma que los datos utilizados en la distribución a posteriori no tienen porque estar en un espacio lineal.

Esta aproximación bayesiana utiliza un modelo generativo para ajustar mejor los parámetros del modelo de forma que tengan en cuenta pequeñas variaciones en los datos. Para ello se define lo siguiente:

- Una distribución sobre los datos y el estado del mundo (las etiquetas en un problema de clasificación, valores en el espacio en un problema de regresión). $P(y|X) = Norm[X^T\phi, \sigma^2 I]$ donde ϕ son parámetros que se estiman.
- Una distribución sobre los parámetros que se van a optimizar para ofrecer mejores resultados (distribución a priori). $P(\phi) = Norm[0, \sigma^2 I]$
- Una distribución sobre los parámetros teniendo en cuenta los datos y el estado del mundo (distribución a posteriori); dicha distribución se calcula mediante la Regla de Bayes. $P(\phi|X, y) = \frac{P(y|X,\phi)P(\phi)}{P(y|X)}$

Una vez queremos predecir el estado del mundo de nuevos datos bastará con generar sus distribuciones a través de su distribución y la distribución definida por los parámetros optimizados.

La diferencia en los Procesos Guassianos y la distribución a posteriori definida arriba es que en vez de utilizar los datos X, se utiliza un transformación sobre estos definida por un kernel K, de forma que lo que utilizamos es $K[X, X]$, la cuál es una matriz con los datos transformados. Algunos de los kernels que puede utilizar son el lineal, polinómico, radial (Gaussiana), etc ...

Capítulo 2

Software Utilizado para el desarrollo de la práctica

El software elegido para desarrollar esta práctica ha sido **GPflow** y el lenguaje de programación **Python**. Este paquete permite crear modelos de Procesos Gaussianos utilizando **tensorflow** para calcular la optimización de los parámetros. Para crear un modelo como el que se pide en la práctica debemos utilizar el modelo *gpflow.models.VGP*; este modelo calcula un proceso gaussiano variacional, que es lo que se ha estudiado en la asignatura. A dicho modelo deberemos de pasarle los datos de train, la etiquetas de los datos, el kernel que utilizará el modelo y la distribución de probabilidad de las etiquetes (likelihood).

El parámetro *likelihood* del modelo será en nuestro caso siempre un objeto de la clase *gpflow.likelihoods.Bernoulli()*, dicha función utiliza internamente una distribución igual que la logística definida en la práctica. Los kernel utilizados son *gpflow.kernels.RBF* y *gpflow.kernels.Linear*, a cada uno de los kernels se le debe indicar el número de dimensiones que tienen los datos; al kernel radial (RBF) se le debe pasar también los parámetros *variance* y *lengthscale* iniciales.

Una vez se ha creado el modelo, se deben optimizar los parámetros para ajustar el modelo a los datos, para ello se debe utilizar la función *gpflow.train.ScipyOptimizer().minimize*; a esta función se le debe pasar el modelo y opcionalmente el número de iteraciones máximo que debe realizar, en la práctica se ha utilizado un máximo de 250 iteraciones ya que a partir de dicho número los resultados obtenidos no mejoraban.

Adicionalmente, se han utilizado las librerías **sklearn**, **numpy**, **scipy.io**, **seaborn**, **pandas**, **matplotlib** para la lectura de datos, dibujar las diferentes gráficas y el cálculo de métricas.

Capítulo 3

Resultados Experimentales

3.1. Resultados Experimentos

En este apartado se comentarán los resultados obtenidos por los modelos generados como ensamble de Procesos Gaussianos sobre los diferentes folds de test.

3.1.1. Núcleo Gaussiano

Fold 1

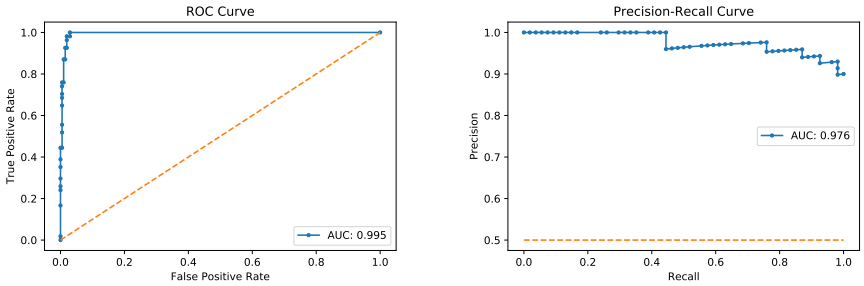

Figura 3.1: Curva Roc y Curva Precision-Recall Fold 1 kernel radial

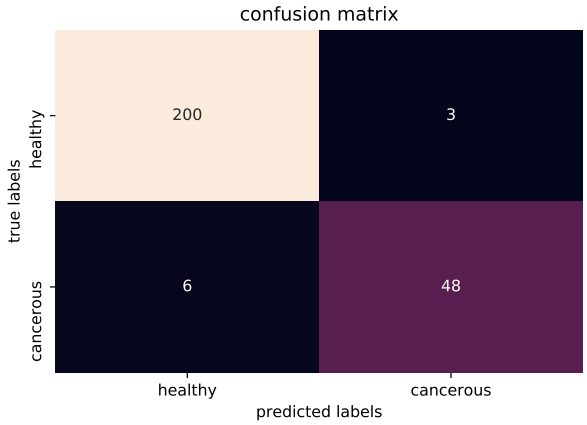

Figura 3.2: Matriz de confusión para Fold 1 kernel radial

acc	**precision**	**specificity**	**recall**	**f1-score**
0.964981	0.941176	0.985222	0.888889	0.914286

Cuadro 3.1: Resultados obtenidos Fold 1 kernel radial

Para este fold se obtienen buenos resultados, obteniendo buenos resultados en *Accuracy* y *F1-score*. Si nos fijamos en la *curva ROC* podemos ver que se obtienen muy buenos resultados, obteniendo un valor de AUC del 99.5 %, con este valor tan alto podemos decir que está clasificando bastante bien ambas clases. La *curva Precision-Recall* también obtiene buenos resultados, obteniendo un buen valor para su AUC también; por lo que se puede ver, se puede ver que hay un buen equilibrio entre *Falsos Positivos* y *Falsos Negativos*, siendo el número de *Falsos Negativos* algo mayor que los *Falsos Positivos*. Si nos fijamos en la matriz de confusión se puede ver que los resultados de las curvas son correctos, viendo que este modelo solamente ha fallado en 9 casos de 257 instancias que tiene este fold; aún siendo buenos resultados, sería mejor que el número de Falsos Positivos fuera mayor que el de *Falsos Negativos*, ya clasificar una imagen como cancerosa aunque después no lo sea a que clasificar una imagen como sana y que sí lo sea por las consecuencias que esto puede tener para un paciente en un caso real.

Fold 2

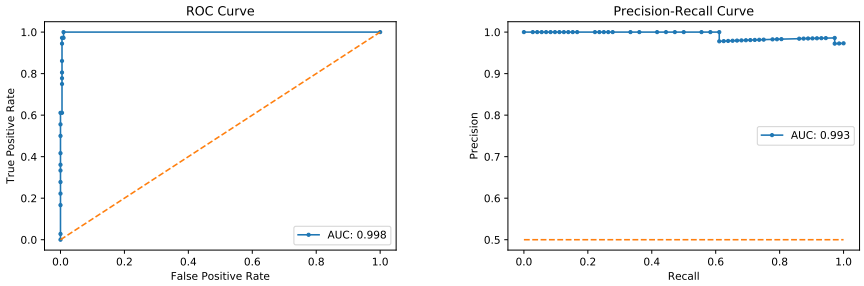

Figura 3.3: Curva Roc y Curva Precision-Recall Fold 2 kernel radial

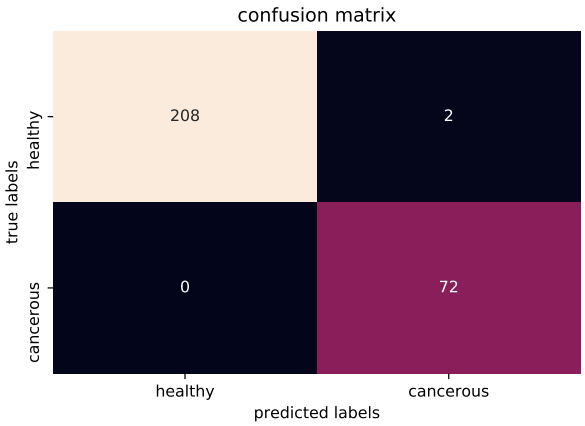

Figura 3.4: Matriz de confusión para Fold 2 kernel radial

acc	**precision**	**specificity**	**recall**	**f1-score**
0.992908	0.972973	0.990476	1.000000	0.986301

Cuadro 3.2: Resultados obtenidos Fold 2 kernel radial

Para este segundo modelo podemos ver mejores resultados todavía que para el modelo anterior, si nos fijamos en las curvas los valores de sus AUC son casi 1. Si nos fijamos en las medidas o en la tabla podemos ver que los resultados de la predicción son casi perfectas también, viendo que todas ellas son mayores que el 97 %. Si nos fijamos en la matriz de confusión podemos ver que el clasificador no ha predicho ningún *Falso Negativo*, por lo que este modelo ha conseguido diferenciar mejor entre las imágenes con tejido canceroso y tejido sano y por lo tanto es un mejor modelo que el anterior.

Fold 3

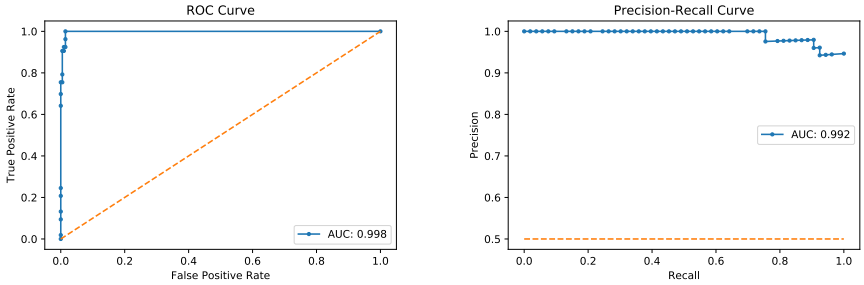

Figura 3.5: Curva Roc y Curva Precision-Recall Fold 3 kernel radial

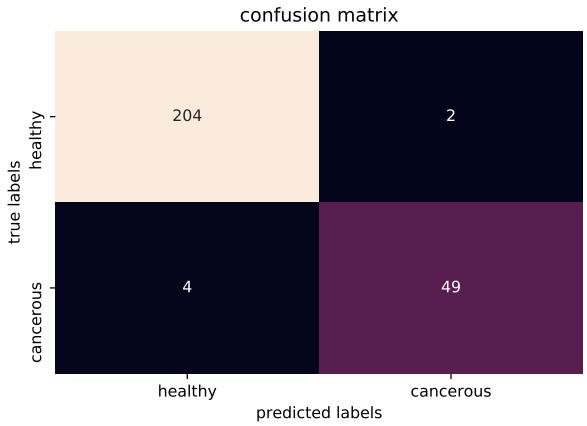

Figura 3.6: Matriz de confusión para Fold 3 kernel radial

acc	**precision**	**specificity**	**recall**	**f1-score**
0.976834	0.960784	0.990291	0.924528	0.942308

Cuadro 3.3: Resultados obtenidos Fold 3 kernel radial

Este tercer modelo, al igual que los anteriores obtiene buenos resultados, valores en las curvas casi perfectos, valores bastante altos en todas las métricas que se han calculado, etc... La diferencia entre este modelo y el anterior es que este obtiene un peor valor para el *Recall* y por lo tanto sí que ha clasificado imágenes de tejido canceroso como tejido sano y es peor modelo que el modelo anterior.

Fold 4

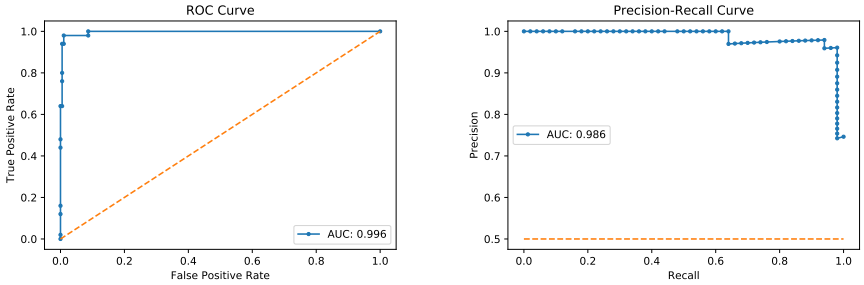

Figura 3.7: Curva Roc y Curva Precision-Recall Fold 4 kernel radial

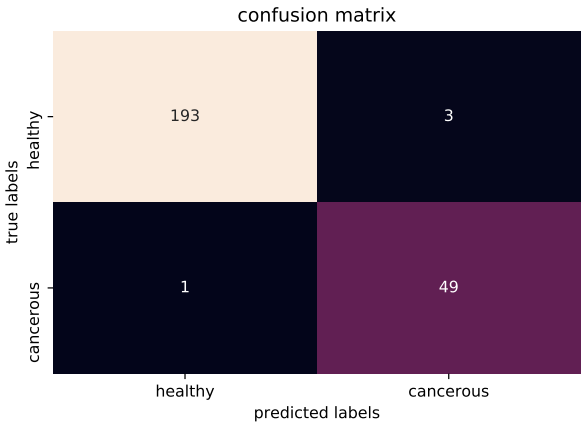

Figura 3.8: Matriz de confusión para Fold 4 kernel radial

acc	**precision**	**specificity**	**recall**	**f1-score**
0.983740	0.942308	0.984694	0.980000	0.960784

Cuadro 3.4: Resultados obtenidos Fold 4 kernel radial

De este modelo se puede destacar su curva de *Precision-Recall*, aunque el valor de *AUC* sea bastante bueno, se pude ver que al final la precisión desciende bastante a diferencia del resto modelo. Si nos fijamos en la matriz de confusión se puede ver que el número de *Falsos Negativos* es menor que el número de *Falsos Positivos*; de ahí que el valor de *Recall* sea mayor que el valor de *Precision*.

Fold 5

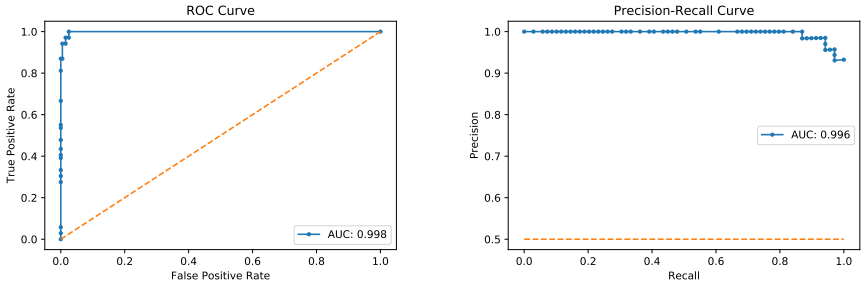

Figura 3.9: Curva Roc y Curva Precision-Recall Fold 5 kernel radial

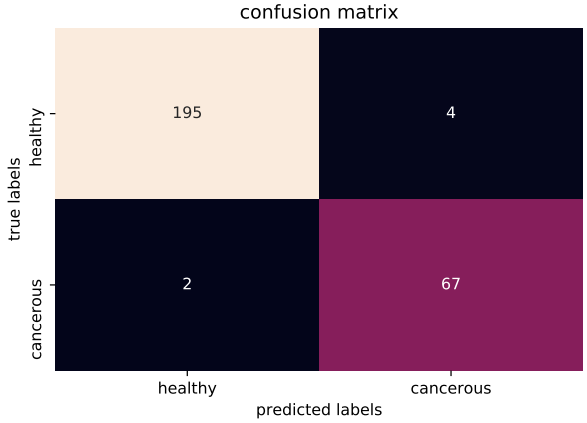

Figura 3.10: Matriz de confusión para Fold 5 kernel radial

acc	**precision**	**specificity**	**recall**	**f1-score**
0.977612	0.943662	0.979899	0.971014	0.957143

Cuadro 3.5: Resultados obtenidos Fold 5 kernel radial

Para este último modelo se pueden ver resultados parecidos al resto de modelos en la métricas, buenos valores en las curvas; manteniendo un buen equilibrio entre *Precision* y *Recall*; obteniendo un número de *Falsos Negativos* menor que *Falsos Positivos*.

3.1.2. Núcleo Lineal

Fold 1

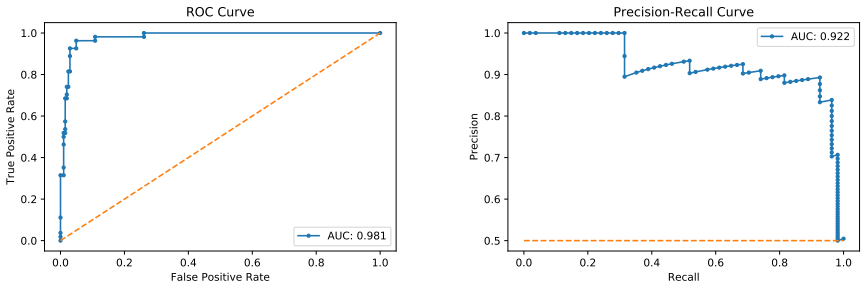

Figura 3.11: Curva Roc y Curva Precision-Recall Fold 1 kernel lineal

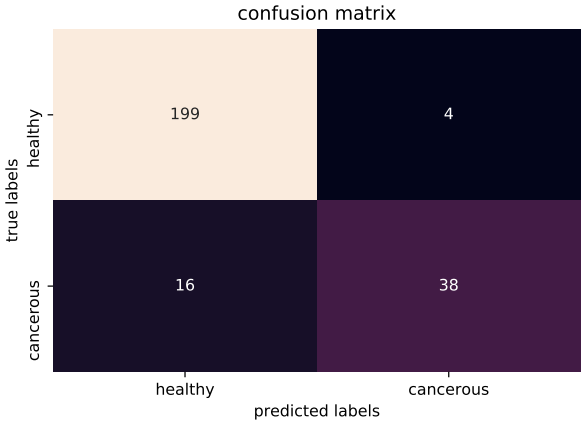

Figura 3.12: Matriz de confusión para Fold 1 kernel lineal

acc	**precision**	**specificity**	**recall**	**f1-score**
0.926070	0.888889	0.975369	0.740741	0.808081

Cuadro 3.6: Resultados obtenidos Fold 1 kernel lineal

Para este primer modelo podemos ver valores bastante buenos en los *AUC* de las curvas, estando ambos por encima del 90 %; se puede destacar la *curva Precision-Recall* que desciende bastante. Si nos fijamos en la matriz de confusión o en las medidas obtenidas se puede ver un número de *Falsos Negativos* mucho mayor que el de *Falsos Positivos*, esto también se refleja en los valores de *Precision* y *Recall* y explica la forma de la *curva Precision-Recall*.

Fold 2

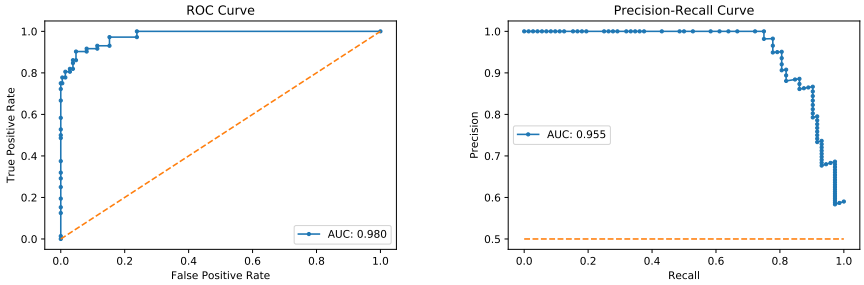

Figura 3.13: Curva Roc y Curva Precision-Recall Fold 2 kernel lineal

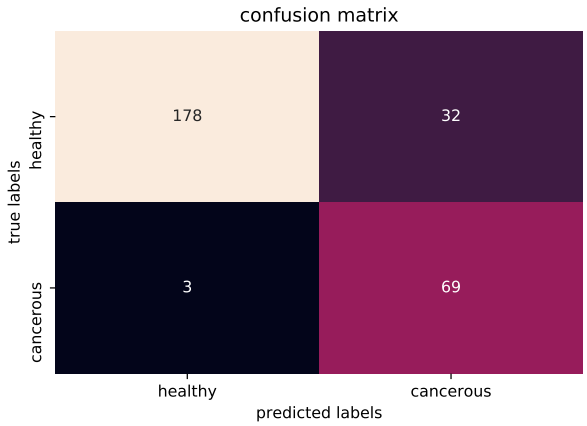

Figura 3.14: Matriz de confusión para Fold 2 kernel lineal

acc	**precision**	**specificity**	**recall**	**f1-score**
0.872340	0.673077	0.838095	0.972222	0.795455

Cuadro 3.7: Resultados obtenidos Fold 2 kernel lineal

Para este segundo modelo podemos ver resultados justamente contrarios a los del modelo anterior, el número de *Falsos Positivos* es mucho mayor que el de *Falsos Negativos*. Por ello este modelo no es bueno ya que no es capaz de diferenciar entre tejido sano y tejido canceroso, aunque es mejor que el modelo anterior porque se equivoca normalmente clasificando imágenes sanas como cancerosas y no al revés.

Fold 3

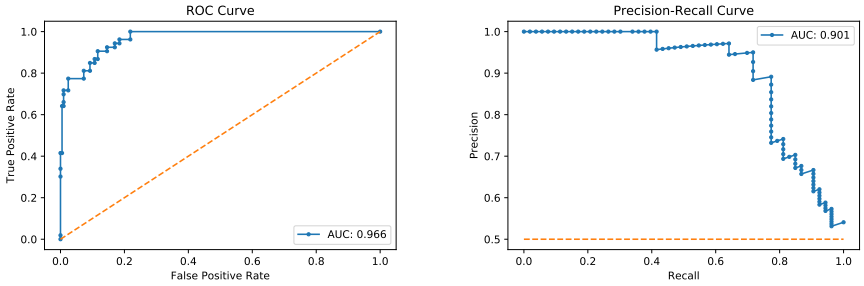

Figura 3.15: Curva Roc y Curva Precision-Recall Fold 3 kernel lineal

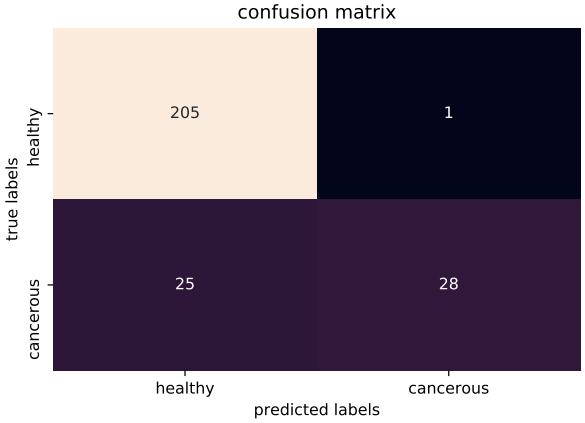

Figura 3.16: Matriz de confusión para Fold 3 kernel lineal

acc	**precision**	**specificity**	**recall**	**f1-score**
0.899614	0.965517	0.995146	0.528302	0.682927

Cuadro 3.8: Resultados obtenidos Fold 3 kernel lineal

Este modelo también es bastante malo, ya que aunque los valores de *Accuracy* o *AUC* es las curvas calculadas no sean malos, su valor de *Recall* es demasiado bajo y por ello el valor de *F1-score* también; por lo tanto este modelo no sería interesante para predecir.

Fold 4

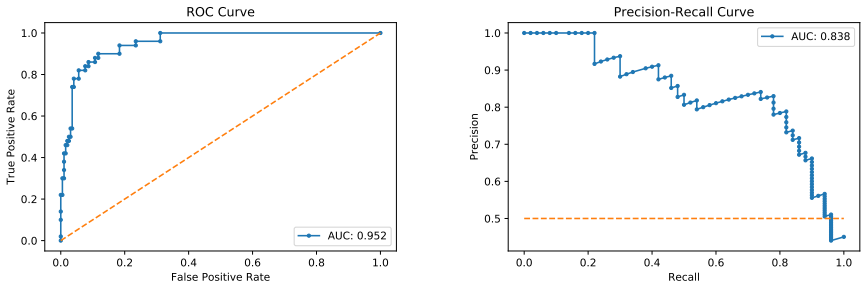

Figura 3.17: Curva Roc y Curva Precision-Recall Fold 4 kernel lineal

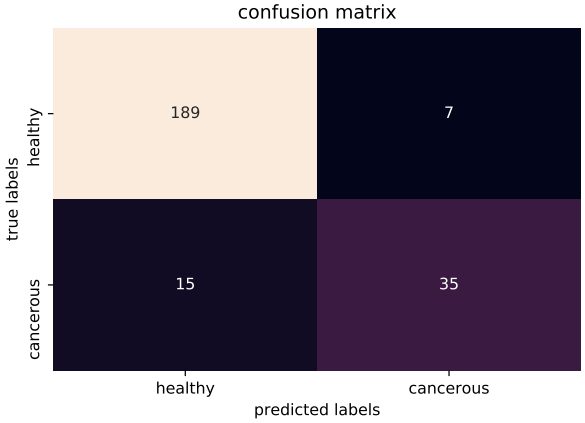

Figura 3.18: Matriz de confusión para Fold 4 kernel lineal

acc	**precision**	**specificity**	**recall**	**f1-score**
0.910569	0.833333	0.964286	0.700000	0.760870

Cuadro 3.9: Resultados obtenidos Fold 4 kernel lineal

De este modelo se puede destacar el valor de *AUC* de la *curva Precision-Recall*, siendo este el más bajo obtenido. Aunque los valores obtenidos no sean demasiado malos y no destaque por equivocarse demasiado al clasificar cualquiera de las clases, no es igual de bueno que el modelo generado con el kernel radial para este mismo conjunto de datos.

Fold 5

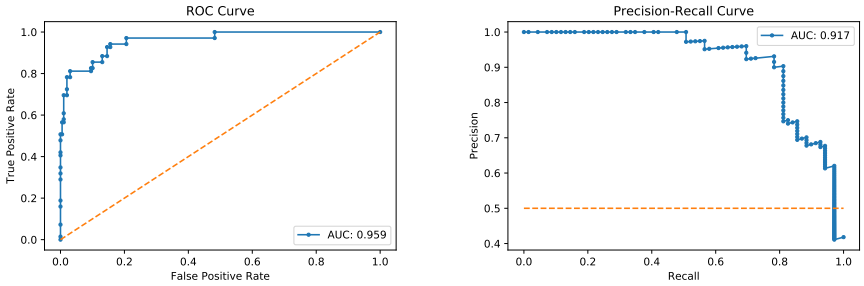

Figura 3.19: Curva Roc y Curva Precision-Recall Fold 5 kernel lineal

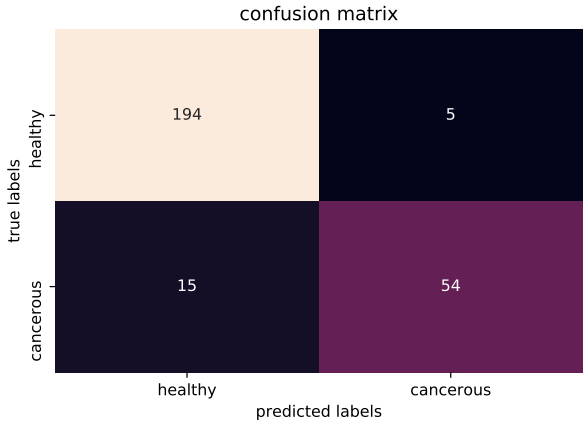

Figura 3.20: Matriz de confusión para Fold 5 kernel lineal

acc	**precision**	**specificity**	**recall**	**f1-score**
0.925373	0.915254	0.974874	0.782609	0.843750

Cuadro 3.10: Resultados obtenidos Fold 5 kernel lineal

Este modelo obtiene relativamente buenos resultados; el valor de *Recall* es algo bajo y esto se refleja también en la matriz de confusión. Al igual que el resto de modelos, la *curva Precision-Recall* disminuye mucho conforme el valor de *Recall* aumenta.

3.2. Clasificación de nuevos datos

Si se quisiera clasificar nuevos datos, bastaría con utilizar el modelo seleccionado y hacer predicción sobre este nuevo dato con dicho modelo. Para ello, sería necesario

guardar el modelo ya entrenado para no tener que repetir el proceso de training. Una vez obtenida la predicción, si sabemos cual es la clase real de dicho nuevo dato; o hay un experto que pueda saber la clase real de dicho dato, se puede comprobar si la predicción del modelo es correcta. Si el modelo que se ha entrenado ha encontrado realmente las características importantes que diferencian las clases la predicción de este modelo debería de ser correcta. Si se prueban nuevos datos y las predicciones son erróneas; se debe volver entrenar el modelo añadiendo dichos nuevos datos; ya que es posible que los datos que se hayan utilizado para generar el modelo no represente todo el espacio de ejemplos que puede haber para ese clasificador.

Para el caso que se estudia en esta práctica, se utilizaría el segundo modelo entrenado con kernel radial. Antes de realizar la predicción, se debe particionar las imágenes en bloques de 2048x2048 y se calculan sus descriptores LBP uniforme invariante y se generan los histogramas; una vez hecho esto se puede utilizar el clasificador para predecir la clase de dicho bloque. Una vez se ha obtenido la predicción, si se dispone de un anatomopatólogo se le debería de preguntar si dicha predicción es correcta.

3.3. Experimentos adicional

Para aumentar el número de datos que tenemos en el modelo, se podría utilizar algún método de oversamplig, por ejemplo *BLSMOTE*. *BLSMOTE* es una modificación del algoritmo *SMOTE*, dicho algoritmo genera nuevas instancias a partir de la combinación de dos instancias de los datos; dichas instancias son un dato aleatorio del dataset y otro dato cercano al primero. La diferencia entre *SMOTE* y *BLSMOTE* es que este último solamente selecciona datos que se encuentran en la periferia entre una clase y otra.

Utilizar este método puede ser interesante por lo que se ha observado durante los experimentos realizados, con los modelos de kernel lineal se obtenían modelos peores que con los modelos con kernel radial. Gracias a esto podemos saber que los datos no son linealmente separables o existen algunas partes en las que los datos de clases diferentes están entremezclados y por ello se la frontera entre un dato y otro no está bien definida. Dado esto, al generar nuevos datos solamente por la periferia de la clase minoritaria podemos mejorar el ajuste del modelo con los datos.

Una vez hecho oversampling sobre el conjunto de los datos solamente se construiría un conjunto de train y otro de test; con el conjunto de train se entería un único modelo de Proceso Gaussiano. Tras esto se comprobaría la calidad del modelo obtenido haciendo la predicción con los datos de test y se obtendrían las diferentes métricas.