Aprenentatge Automàtic (APA)
Práctica

Aplicación del Aprendizaje Automático en la detección de pacientes con Hipotiroidismo

Grau en Enginyeria Informàtica

Pol Casacuberta Gil
Marta Granero I Martí

Departament de Ciències de la Computació
Universitat Politècnica de Catalunya

2022/2023 1Q
Enero, 2023

Índice

1. Introducción

Hoy en día, muchos de los habitantes de este mundo estamos familiarizados con el estilo de vida acelerado y ajetreado que a menudo nos deja exhaustos y de lo más agotados. Pero para quienes padecen hipotiroidismo, un trastorno médico muy común en el que la glándula tiroides no produce suficientes hormonas tiroideas, la fatiga es sólo uno de los muchos retos a los que se enfrentan a diario.

Las hormonas tiroideas desempeñan un papel crucial en la regulación del metabolismo y la producción de energía del cuerpo (Dewangan, A., Shrivas, A., & Kumar, P., 2016), y una deficiencia de estas hormonas puede provocar una amplia gama de síntomas, como aumento de peso, piel seca, sensibilidad a las bajas temperaturas y entre muchos otros síntomas que pueden afectar tanto a las mujeres como a los hombres, aunque según (Shrivas, A.K., & Ambastha, P.K.,

2017) el trastorno tiroideo es una de las enfermedades más frecuentes entre las mujeres, en la que estas son más propensas a desarrollar hipotiroidismo.

Desgraciadamente, muchas veces debido al alto ritmo de vida que llevamos algunas personas, restamos importancia a sentirnos cansados. Es por esto que esta enfermedad crónica muchas veces no se suele diagnosticar correctamente y puede tener importantes repercusiones en la calidad de vida de una persona. Si dejamos pasar los síntomas, esta enfermedad a largo plazo puede provocar una serie de complicaciones de salud, como por ejemplo: colesterol alto, cardiopatías, entre otras complicaciones, como por ejemplo, el desarrollo de ciertas lesiones nerviosas. Asimismo, puede empeorar enfermedades preexistentes como puede ser la diabetes y la depresión.

Pero la buena noticia es que el hipotiroidismo puede tratarse fácilmente con medicación. La terapia de sustitución de la hormona tiroidea puede ayudar a restablecer el metabolismo normal del organismo y eliminar los síntomas de la enfermedad. Actualmente, el diagnóstico se realiza mediante un examen exhaustivo y numerosos análisis de sangre juntamente con la realización de ecografías (Moon Jae Hoon, & Steinhubl, S.R., 2019) en la parte de debajo de la prominencia laríngea(nuez), ya que esta puede inflamarse o agrandarse.

Al ser una enfermedad que afecta a millones de personas a lo largo del mundo, nos pareció que tratar este tema de gran interés en esta práctica era un acierto, puesto que ser capaces de dar con la forma de poder detectar y clasificar cuadros clínicos de pacientes que tengan o estén en una fase temprana en el desarrollo de esta enfermedad puede ser de gran ayuda tanto médica como para la calidad de vida de los pacientes.

Así pues, en esta práctica desarrollaremos varios modelos de clasificación usando distintos algoritmos de ML para resolver el problema real de detección de pacientes con hipotiroidismo a través de su un cuadro clínico con rates, índices y hormonas tiroideas para poder clasificar a los pacientes que tienen o no una reducida actividad de la glándula tiroides.

Esperamos que nuestro modelo final ayude a los profesionales médicos a predecir y utilizar este clasificador para posteriores estudios y diagnósticos. Así pues, el objetivo principal de esta corta investigación es utilizar un algoritmo de aprendizaje automático para diagnosticar dado el cuadro clínico de un paciente si tiene una disfunción de la glándula tiroidea.

Los resultados que hemos conseguido son buenos y son acordes a los resultados obtenidos por la literatura que hemos leído, en general el acierto de los modelos que se han elaborado tienen una calidad muy buena, con una precisión con el mejor modelo no lineal del 99.86% usando el modelo de Random Forest y una precisión del 98.41% con el modelo lineal de SVM lineal.

Asimismo, anticipamos que se puede encontrar en este trabajo. En la sección 2 tenemos una breve descripción de los enfoques populares que existen en la literatura juntamente con sus resultados usando conjuntos de datos parecidos al que hemos usado. En la sección 3 explicamos cómo hemos procedido en la exploración de nuestros datos, tanto en el preprocesamiento, la selección de las características, como tratamos los outliers, los *missing values*, la codificación de las variables y la visualización de estos. En la sección 4 explicamos cómo hemos elaborado el protocolo de remuestreo. En la sección 5 explicamos los métodos lineales que hemos escogido y sus resultados correspondientes. Seguimos en la sección 6 explicando los métodos no-lineales de la misma forma que en la sección 5. Seguidamente, en la sección 7 escogemos el modelo final que es el mejor clasificador para este problema. Procedemos con la sección 8 dónde tratamos la

interpretabilidad, notamos que ya hemos hablado de la interpretabilidad de cada modelo en su apartado correspondiente, pero ahora nos adentramos en sus diferencias y explicaremos algunos ejemplos interesantes. A continuación, en el apartado 9 exponemos nuestras conclusiones y reflexiones finales del trabajo. Finalmente, en la última sección se puede encontrar la bibliografía dónde se ha citado la información usada para documentar y dar fundamento a las ideas expuestas, juntamente con varias referencias adicionales.

2. Estudios previos

Hemos realizado una extensa búsqueda en la literatura para saber cómo se ha abordado esta tarea de clasificación y hemos podido ver que la mayoría de trabajos populares que usan un conjunto de datos parecido al que estamos usando, están basados en el aprendizaje automático supervisado. La amplia mayoría de ellos, se basan en estos conjuntos de datos, que podemos encontrar en el repositorio de aprendizaje automático de la UCI[1]. Estos conjunto de datos, tienen distintas formas, más o menos actualizadas, con conjuntos que tienen más de dos clases a predecir, o que tienen ya separados los conjuntos de entrenamiento y test, con más o menos instancias, etc, pero todos ellos se parecen entre sí, se usan en el mismo campo de dominio y se parecen al que hemos considerado para hacer la práctica.

Por el contrario, el conjunto de datos que hemos escogido, está extraído de la plataforma OpenML (Open Machine Learning)[2]. Hemos considerado escoger este, ya que no hemos encontrado ninguna notebook existente en Kaggle que lo tratase y ninguno de los papers de la literatura lo mencionaba o trataba directamente con él. Cabe notar que tanto los conjuntos en varias de sus formas que se encuentran en la UCI y el nuestro tampoco tienen muchas diferencias. Divergen en el número de columnas, 21 o 26, a diferencia de las 30 columnas del dataset que hemos escogido, además del número de instancias de que disponen los del UCI y el nuestro.

Cabe notar, que en la literatura sólo hemos encontrado un trabajo de investigación que estudia y considera usar el aprendizaje no supervisado para la clasificación usando *ANNs* y juntamente con el algoritmo de clustering K-means, véase (Mahurkar, K. K., & Gaikwad, D. P, 2017).

Cabe destacar que en la práctica, la mayoría de trabajos populares, obtienen muy buenos resultados, con unos modelos sorprendentemente casi perfectos. Estos trabajos de investigación han probado varios clasificadores, como siguen: *KNN, Naïve Bayes, Random Forest, SVM* (*Linear Kerne*l)*, Gradient Boosting Classifier, MLP, XGBoost, LDA, QDA, Extra-Trees y CatBoost*, entre muchos otros…

Algunos de los modelos que más coinciden en las calidades en la mayoría de papers han sido los modelos de *Random Forest* con una precisión de alrededor del 95-99%, el modelo de *Extra-Trees*, alrededor del 90-99%, *CatBoost* y *XGBoost* alrededor del 95%, LDA alrededor del 94%, *KNN* alrededor del 80-85% y la SVM con un kernel lineal con una precisión muy parecida a los modelos de *Random Forest*, cerca del 90-98%.

Nos gustaría destacar que entre la literatura también hemos encontrado ciertas discrepancias a la hora de comparar las calidades de los mismos clasificadores. En algunos obtenían casi todos

[1] Lo mantiene la Universidad de California, Irvine, y es uno de los repositorios de datos de aprendizaje automático más citados en este campo. Este repositorio alberga una colección de distintos datasets que se utilizan para la investigación en el aprendizaje automático.
[2] Plataforma abierta dónde se encuentran varios conjuntos de datos.

clasificadores perfectos (Moharekar, T. T., Vadar, M. S., Pol, U. R., Bhaskar, P. C., & Moharekar, M. T., 2022), y en otros ya se podían apreciar diferencias más notables entre modelos, véase (Anika Shama, M.B. Hossain, A. Adhikary, et al., 2022)

3. Exploración de los datos

En esta sección explicaremos cómo exploramos y preprocesamos los datos que estamos usando mediante el dataset *hypothyroid* que podemos encontrar en la plataforma abierta OpenML. Además, mostraremos distintas visualizaciones de nuestros datos.

3.1 Descripción del conjunto de datos

Nuestro conjunto de datos está compuesto por 3772 muestras, y 30 columnas, 29 sin incluir la variable objetivo. Entre las columnas predictoras, encontramos tanto variables categóricas binarias como variables numéricas. Este dataset no viene preprocesado, ya que encontramos números valores NaN y con valores faltantes en la mayoría de muestras del conjunto de datos. Asimismo, este conjunto de datos no se ha generado sintéticamente.

3.2 Selección y extracción de características

Empezaremos a tratar las 29 características que nos ayudarán o nos intentarán dar información sobre el problema principal que estamos tratando, definir si una paciente presenta o no un cuadro clínico positivo o negativo en hipotiroidismo.

Así que nos tocará seleccionar las características que en efecto nos sirven. Entonces, para preprocesar sólo los datos necesarios, eliminamos todas las columnas no útiles para reducir la cantidad de ruido y datos irrelevantes que el modelo tiene que procesar.

En concreto, reducimos nuestro problema a uno con sólo 26 columnas. Ya que eliminamos tres columnas por diferentes razones. Extraemos dimensionalidad al dataset, eliminando las columnas:

- **TBG**: tiene todos sus valores a NaN
- **TBG measured**: todos sus valores son 0, por lo tanto tiene sentido que no tengamos valores para TBG.
- **referral source**: tiene hasta 5 valores únicos: SVHC,other,SVI,STMW,SVHD. Aun así, decidimos eliminarla del conjunto de datos, ya que creemos que añadiría una complejidad innecesaria y confusión, pues más de la mitad de sus valores tienen la categoría “other”, lo que nos suscita que esta característica no es lo suficientemente específica y en este caso saber la fuente de derivación para el análisis no será de gran relevancia como característica en el conjunto de datos.

Entre las características comenzamos nuestro trabajo con las 26 siguientes:

- **age:** Edad de los pacientes.
- **sex:** Sexo de los pacientes, toma los valores categóricos: F (femenino), M(masculino), ‘?’(indefinido).
- **on thyroxine:** Se refiere a si la persona está tomando tiroxina. Toma valores: f (falso), t (cierto).
- **query on thyroxine:** Se refiere a que la persona está considerando iniciar un tratamiento con tiroxina, o que actualmente está tomando tiroxina y tiene preguntas al respecto, o que

ha experimentado efectos secundarios u otros problemas con la tiroxina y está buscando más información. Toma valores: f (falso), t (cierto).

- **on antithyroid medication:** Se refiere a si la persona está tomando medicación para tratar una glándula tiroides hiperactiva. Toma valores: f (falso), t (cierto).
- **sick:** Se refiere a si la persona está sufriendo una enfermedad o tiene síntomas. Toma valores: f (falso), t (cierto).
- **pregnant:** Se refiere a si la persona está embarazada. Toma valores: f (falso), t (cierto).
- **thyroid surgery:** Se refiere a si la persona ha sido operada de la glándula tiroides. Toma valores: f (falso), t (cierto).
- **I131 treatment:** Se refiere al tratamiento con yodo radiactivo (I131), que suele utilizarse para tratar afecciones tiroideas como el hipertiroidismo (glándula tiroides hiperactiva). Toma valores: f (falso), t (cierto).
- **query hypothyroid:** Sospecha o posible diagnóstico potencial de hipotiroidismo. Toma valores: f (falso), t (cierto).
- **query hyperthyroid:** Sospecha o posible diagnóstico de hipertiroidismo. Toma valores: f (falso), t (cierto).
- **lithium:** Se refiere al litio, un medicamento utilizado para tratar el trastorno bipolar y otras enfermedades mentales. Toma valores: f (falso), t (cierto).
- **goitre:** Se refiere a una inflamación anormal de la glándula tiroides, que puede ser signo de una afección tiroidea. Toma valores: f (falso), t (cierto).
- **tumor:** Se refiere a la presencia de un tumor tiroideo. Toma valores: f (falso), t (cierto).
- **hypopituitary:** Se refiere a una deficiencia en la glándula pituitaria, que puede afectar a la función tiroidea. Toma valores: f (falso), t (cierto).
- **psych:** Se refiere a una afección psicológica o psiquiátrica. Toma valores: f (falso), t (cierto).
- **TSH measured:** Se refiere a si se ha medido la hormona TSH. La TSH es una hormona producida por la hipófisis y ayuda a regular la función tiroidea. Toma valores: f (falso), t (cierto)
- **TSH:** Se refiere al nivel de la hormona estimulante del tiroides (TSH) en el organismo.
- **T3 measured:** Se refiere a si se ha medido la triyodotironina (T3). La T3 es una hormona tiroidea producida por la glándula tiroides. Toma valores: f (falso), t (cierto)
- **T3:** Se refiere al nivel de triyodotironina (T3) en el organismo.
- **TT4 measured:** Se refiere a si se ha medido la tiroxina total (TT4). TT4 es una medida de la cantidad total de tiroxina (una hormona tiroidea) en el organismo. Toma valores: f (falso), t (cierto)
- **TT4:** Se refiere al nivel de tiroxina total (TT4) en el organismo.
- **T4U measured:** Se refiere a si se ha medido la absorción de tiroxina (T4). La absorción de T4 es una medida de la cantidad de tiroxina que absorben las células del organismo. Toma valores: f (falso), t (cierto)
- **T4U:** se refiere al nivel de absorción de tiroxina (T4) en el organismo.
- **FTI measured:** Se refiere a si se ha medido el índice de tiroxina libre (ITL). El ITF es una medida de la cantidad de tiroxina libre (una hormona tiroidea) en el organismo. Toma valores: f (falso), t (cierto)
- **FTI:** Se refiere al nivel de índice de tiroxina libre (FTI) en el cuerpo.

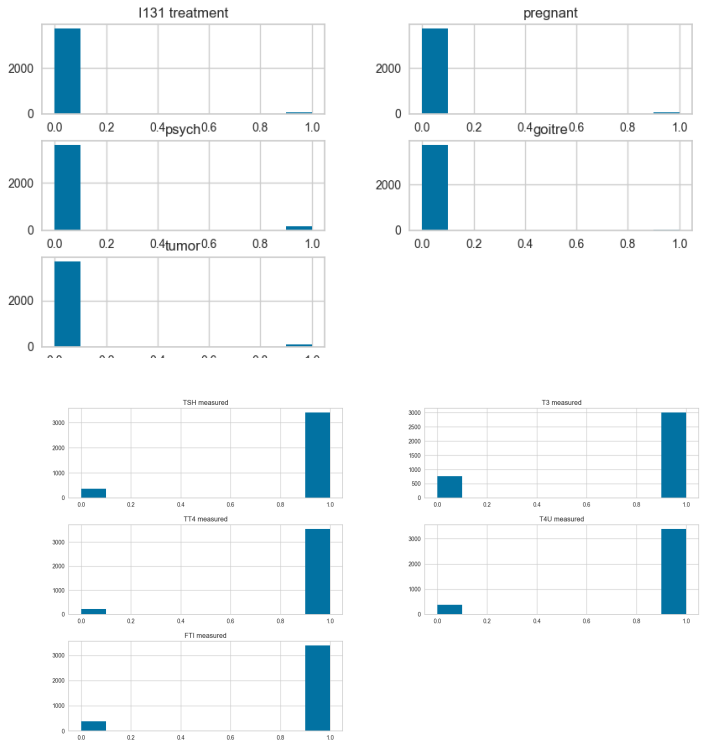
Y la variable objetivo:

- **binaryClass:** Ésta toma valores categóricos: (P, N) en función de sí un paciente es positivo en Hipotiroidismo (P), o N en caso contrario.

3.3 Codificación de variables

Hemos codificado todos los valores categóricos t y f por los valores binarios 1 y 0, respectivamente, para todas las características categóricas binarias. Estas representan la gran mayoría de las características que tenemos en nuestro conjunto de características predictoras. Además, también hemos hecho lo mismo para la variable ***sex,*** codificando mujeres con el valor binario 0, y hombres con el valor binario 1. Asimismo, hemos codificado la variable respuesta **binaryClass,** codificando el valor P, por el valor binario 1 y el valor N por el valor binario 0.

Cabe notar que la mayoría de las características binarias de las que disponemos no tienen un equilibrio correcto entre muestras verdaderas y muestras falsas, estando estas clases muy desequilibradas, como podemos ver en los histogramas a continuación.



Por un lado, las características que no indican si una hormona ha sido medida, predominan las muestras falsas, y se intercambian las tornas, en las clases que miden las hormonas, que ahora predominan las muestras verdaderas.
Asimismo, también tenemos un desequilibrio importante en la variable respuesta, teniendo una proporción de muestras del **92%/8%** para muestras verdaderas(P) y falsas(N). Concretamente: 3481 muestras de la clase P, y 291 de la clase N.

3.4 Missing values

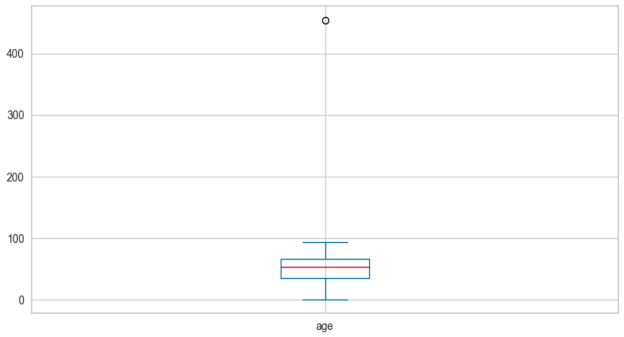
Una vez llegado a este punto, nos queda buscar los valores que faltan en las características de las que disponemos. Vemos que nos faltan muchos valores para la variable ***sex,*** que hemos detectado mediante el valor categórico '?'. Simplemente, hemos decidido asignar los valores faltantes de la categoría '?', suponiendo que se tratan de pacientes mujeres, ya que tenemos un valor muy superior para esta categoría.

Por otro lado, también disponemos de una gran cantidad de valores NaN para las variables que representan a las hormonas: **TSH, T3, TT4, T4U, FTI**, en este caso, debido a nuestra falta de nuestro conocimiento experto, hemos decidido asignar a los valores faltantes de cada columna el valor medio de cada columna respectiva. Para cada columna, el valor medio de esa columna se calcula utilizando la función `mean()`.

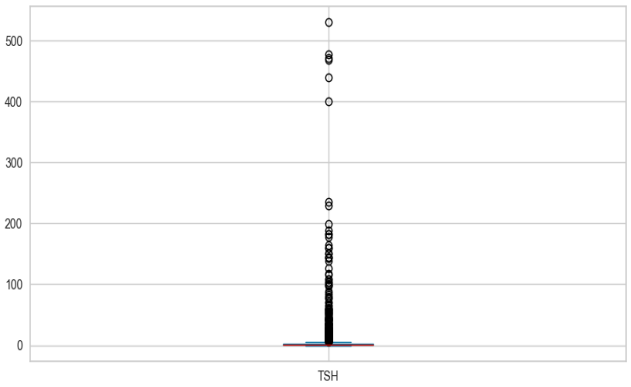
3.5 Valores atípicos

En nuestro conjunto de datos hemos encontrado un outlier en el atributo numérico de ***age.***

Lo tratamos simplemente eliminando la fila en la que se encuentra, ya que disponemos de suficientes datos en nuestro conjunto. Realizando este ajuste, obtendremos un nuevo boxplot, sin este valor que queda por encima de 400.

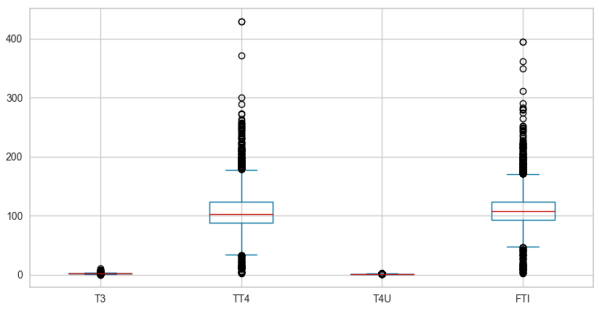


Asimismo, también hemos podido observar que al realizar el boxplot de la característica numérica ***TSH,*** hemos obtenido un gran número de outliers. En este caso, no podemos proceder igual que como hemos hecho con la variable ***age,*** ya que estaríamos hablando de eliminar una gran cantidad de filas de nuestro conjunto de datos.
En este caso, al tratarse de una variable que es una hormona que estimula la tiroides y que desempeña un papel clave en la regulación de la función tiroidea, no estamos seguros de que los valores atípicos obtenidos indiquen condiciones anormales o poco saludables. Por ejemplo, algunas personas pueden tener niveles de TSH naturalmente altos o bajos que se encuentran fuera del rango normal, y estos valores pueden no ser necesariamente indicativos de problemas de salud subyacentes. Así, como no estamos seguros de que no pueda ser información relevante, hemos decidido mantener los valores atípicos en el conjunto de datos.



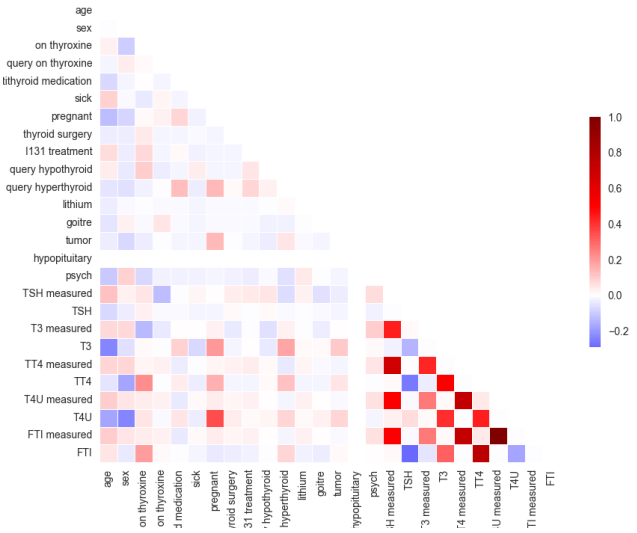
Si fuéramos una persona experta en el tema, lo apropiado sería identificarlos y analizarlos para ver si hay alguna razón subyacente para su existencia. Por ejemplo, podrían deberse a errores de medición, o pueden ser el resultado de algún factor subyacente, o por la toma de medicamentos, o por algún trastorno autoinmune, por determinados medicamentos, o que también sea el caso que las demás variables del conjunto de datos no son capaces de capturar por qué tenemos estos outliers.

Sucede exactamente lo mismo para las demás variables numéricas que tiene valores continuos como son: ***T3, TT4, T4U y FTI.*** Así, que es importante considerar cuidadosamente el contexto y las causas potenciales de



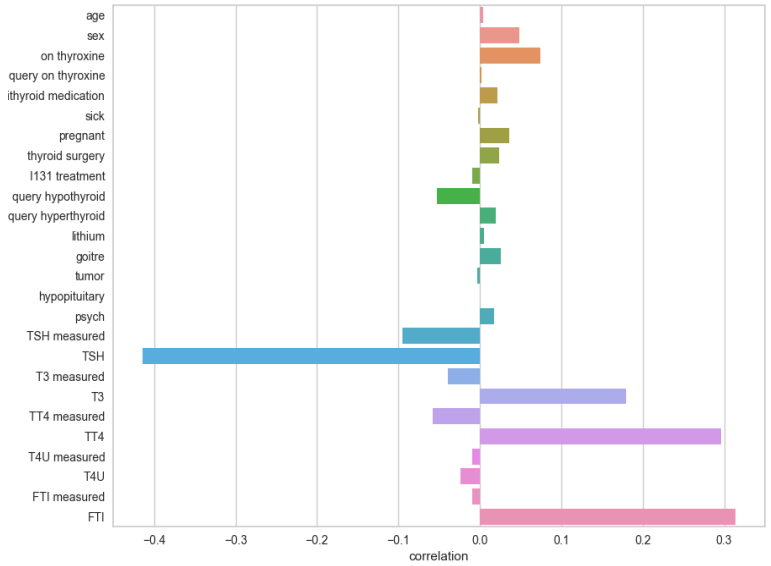
los valores atípicos en estas variables, y tomar decisiones sobre su inclusión o exclusión, basándose en una comprensión profunda de los datos y los fenómenos subyacentes que estamos estudiando.

3.6 Correlación entre las variables predictoras en el conjunto de entrenamiento



Como podemos observar en el heatmap no existe una correlación importante entre la mayor parte de características binarias, ya que podemos ver que en la parte izquierda del *heatmap* predominan los colores pasteles, indicando una correlación que oscila entre los 0.0 ± 0.2.

Por otra parte, en la parte derecha podemos ver que existe una alta correlación entre las variables que representan los distintos tipos de hormonas y las distintas variables que indican si se han medido las respectivas hormonas. También notamos las correlaciones que hay entre la variable **age** y las hormonas **T3** y **T4U**, y entre las variables **T4U** y **pregnant.** Ya que en un embarazo normal, la placenta produce gonadotropina coriónica humana (hCG), que puede estimular la glándula tiroides para que produzca más T4U.
A continuación, comentaremos la relación entre las variables predictoras con la variable objetivo, mediante el coeficiente de correlación de Pearson.



Sin ninguna duda, las variables que más correlación tienen con la variable objetivo, **binaryClass,** son los diferentes tipos de hormonas: **TSH, T3, TT4 y FTI.** Con una fuerte correlación negativa con la hormona TSH y con una importante correlación positiva con las demás hormonas mencionadas. Además, observamos que no tenemos correlación entre la variable **hypopituitary** y la respuesta, como ya sucedía en el heatmap anterior. Esto es debido a que en el conjunto de

entrenamiento todos los valores de esta variable son 0. Notamos también que aparecen correlaciones con las demás variables que tienen una plena lógica. Como es el caso de la variable **sex o pregnant** (Shrivas, A.K., & Ambastha, P.K., 2017) o **query hypothyroid** o la variable **on thyroxine.** También nos ha parecido interesante ver que apenas hay correlación entre la edad y la variable objetivo, o entre la variable **tumor**(ya que se refiere a la presencia de un tumor tiroideo) o **goitre(**inflamación anormal de la glándula tiroides), puesto que podríamos pensar que con las ecografías tomadas tener una inflamación podría ser un signo de tener hipotiroidismo**.** También es interesante, que se puede observar que hay una correlación, positiva aunque pequeña, con la variable **psych(**tener o no afección psicológica o psiquiátrica), ya que como hemos dicho, el hipotiroidismo puede empeorar la depresión y se agrava con el estrés.
Aunque hemos observado estas correlaciones, cabe recordar que la correlación no implica necesariamente causalidad, por el hecho de que es posible que dos variables estén correlacionadas sin que una cause la otra.

4. Protocolo de remuestreo

En primer lugar, particionamos nuestro conjunto de datos en un conjunto de entrenamiento y un conjunto de test en una proporción del **80%/20%** del conjunto de datos totales, respectivamente.

Para llevar a cabo esta partición utilizamos el método `train_test_split`, de la librería de `scikit-learn`. Añadiendo a este método, el parámetro de `stratify=binaryClass` explícitamente.
Este parámetro nos garantiza que en el proceso de remuestreo las proporciones relativas de las diferentes clases en la variable objetivo, `binaryClass`, se mantienen en ambos conjuntos. Este parámetro nos es útil en casos en que las clases de la variable objetivo están desequilibradas, ya que nos afianzará que los conjuntos de entrenamiento y test sean más representativos de la distribución general de los datos.
Asimismo, añadimos un valor para `random_state=42` para obtener las mismas particiones si volvemos a ejecutar el código. Además, tendremos en cuenta el parámetro `shuffle=True`(toma este valor por defecto), en el que se barajarán aleatoriamente los datos antes de ser divididos. Mezclar los datos cuando están ordenados de alguna manera, nos ayuda a crear conjuntos de entrenamiento y test más representativos.

A continuación, utilizamos el conjunto de entrenamiento para seleccionar los mejores hiperparámetros para cada modelo lineal y no-lineal empleado. Para ello, con el objetivo de obtener la mejor precisión y mitigar al mismo tiempo el sobreajuste, utilizamos la validación cruzada `10-fold`, para cada modelo que entrenamos.

Por último, tras seleccionar los mejores hiperparámetros que hemos obtenido para cada modelo, utilizamos el conjunto de test para calcular la precisión final de nuestros modelos.

Notamos que para cada modelo, utilizamos exactamente el mismo conjunto de test, con el fin de realizar una comparación equitativa entre los distintos modelos calculados.

5. Métodos lineales/cuadráticos

5.1 Naïve Bayes

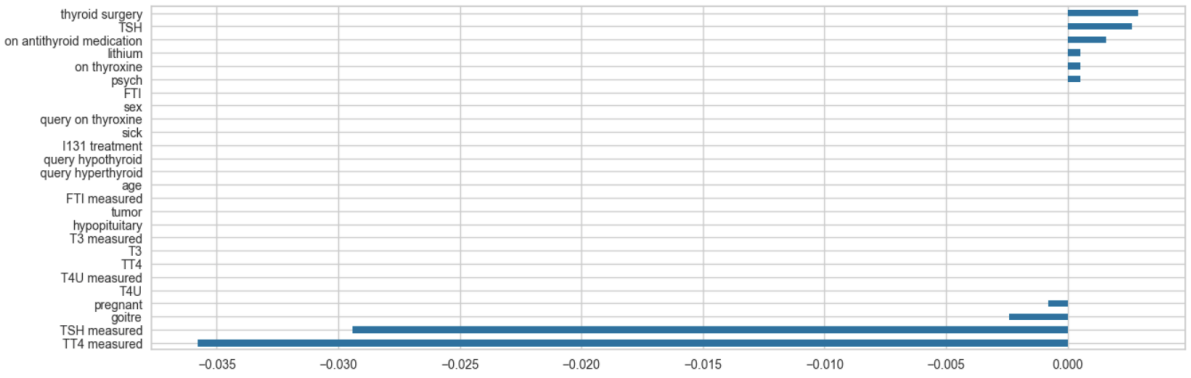
En primer lugar, lo que haremos será establecer una línea base mediante un modelo sencillo para problemas de clasificación, como es Naïve Bayes. De esta forma nos haremos una idea de la dificultad del problema y obtendremos unos primeros resultados de referencia que, esperamos, que mejoren en los siguientes modelos que realicemos.

Después de dividir el conjunto de datos ya preprocesado en un conjunto barajado de entrenamiento y test con una proporción del **80%/20%** respectivamente, hemos normalizado los datos para maximizar la precisión y tener una mejor interpretación del modelo. Hemos normalizado los datos usando `MinMaxScaler()`, ya que las características con las que tratamos no siguen una distribución normal y sobre todo las características de las hormonas, tienen un amplio rango de valores.

Usando este modelo, obtuvimos una precisión de validación cruzada de 0.22. Si nos fijamos en el informe de clasificación vemos que este modelo tiene serias dificultades para predecir la clase 0 (negativo para hipotiroidismo). La precisión para la clase 0 es de 0.98, lo que significa que el modelo es muy bueno para predecir la clase 0 cuando hace una predicción para esa clase. Sin embargo, el recall de esta clase 0 es sólo de 0.09, lo que significa que el modelo sólo predice correctamente una fracción muy pequeña de las muestras reales de clase 0. En cambio, el modelo parece predecir mejor la clase 1 (hipotiroidismo positivo). La precisión para la clase 1 es de 0.15, lo que significa que el modelo no es muy bueno para predecir la clase 1 cuando predice la clase 1, es decir, hace muchas predicciones positivas falsas. Sin embargo, el recall de la clase 1 es de 0.99, lo que significa que el modelo predice correctamente casi todas las muestras de la clase 1.
En general, el modelo tiene dificultades para predecir la clase 0, por lo que su precisión global es relativamente baja 0.22. Notamos que para esta clase únicamente contamos con 58 muestras.

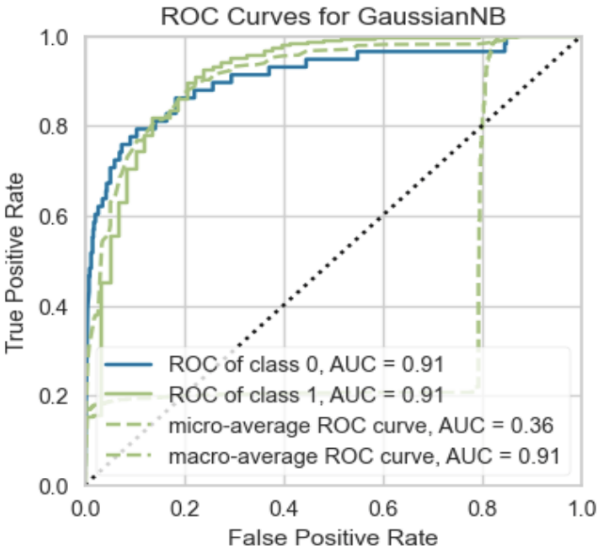
	precision	recall	f1-score	support
0	0.98	0.09	0.16	647
1	0.15	0.99	0.27	108
accuracy			0.22	755
macro avg	0.57	0.54	0.21	755
weighted avg	0.86	0.22	0.18	755

Entonces, analizando los coeficientes del modelo y nuestras características, obtenemos lo que vemos en la siguiente imagen:

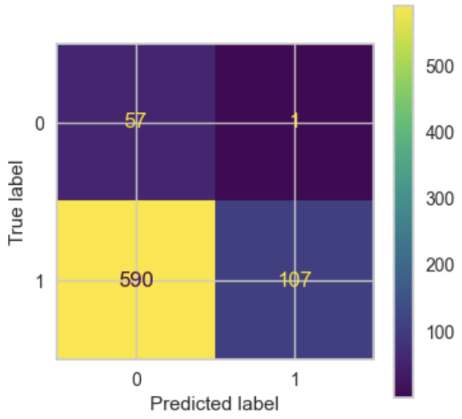


Como podemos observar las features que tienen mayor importancia son **thyroid surgery, TSH** y con una menor importancia las features **TT4 measured**, **TSH measured** y **goitre**. Por lo tanto, nuestro modelo de Naïve Bayes funciona mejor cuando no usa estas últimas features de **TT4** o de **TSH measured**.

Ahora podemos echar un vistazo a las curvas ROC para ambas clases objetivo, siendo la etiqueta 0 negativo en hipotiroidismo, y 1 en caso contrario. Como podemos ver son muy similares y tienen la misma área debajo de la curva, 0.91.



Por último, utilizando el conjunto de test, obtuvimos una precisión del 21,72% con la siguiente matriz de confusión:



5.2 LDA

En la siguiente sección analizaremos nuestro siguiente modelo, LDA o Linear Discriminant Analysis. Para este modelo también dividimos el conjunto de datos ya preprocesado en un conjunto barajado de entrenamiento y test, de **80/20%** respectivamente, y normalizamos los datos para obtener mejores resultados.

En este modelo utilizamos el método `GridSearchCV` de sklearn para encontrar los mejores hiperparámetros para este modelo. Hemos decidido buscar entre cinco hiperparámetros diferentes:

- Solver: hemos probado tres valores distintos: svd, eigen y lsqr
- Shrinkage(técnica de regularización que puede ayudar a evitar el sobreajuste): probando estos valores: None, 'auto', 0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9 y 1
- Número de componentes: probando las dimensiones, None, 1, 2, 3, 4, 5, 8 y 10
- Tolerancia del solver: probando tolerancias de: 1e-2, 1e-3, 1e-4, 1e-5, 1e-6 y 1e-9
- Store covariance(controla si el modelo debe almacenar la matriz de covarianza): True y False

Tras la búsqueda de validación cruzada con el conjunto de entrenamiento, descubrimos que los mejores hiperparámetros son los siguientes:

{'n_components': 1, 'shrinkage': 'auto', 'solver': 'eigen', 'store_covariance': True, 'tol': 0.01}

Con estos hiperparámetros conseguimos una precisión de validación cruzada de 0.9406. Si echamos un vistazo al informe de clasificación, podemos ver que para la clase 0, el modelo tiene una precisión del 0.33, lo que significa que predice correctamente la clase negativa aproximadamente un tercio de las veces que realiza una predicción para esa clase. Asimismo, el recall de la clase negativa en hipotiroidismo es de 0.90, lo que significa que el modelo predice correctamente una gran parte de las muestras negativas reales.

Para la clase positiva, el modelo tiene una precisión perfecta, lo que nos dice que predice correctamente la clase positiva cada vez que realiza una predicción para esa clase. El recall de la clase 1 es de 0.95, lo que significa que el modelo predice correctamente la gran parte de las muestras positivas reales.

Además, la puntuación del f1-score para la clase 0 es 0.48 y para la clase 1 es 0.97. Esto significa que el modelo está haciendo un mejor trabajo a la hora de equilibrar la precisión y la recuperación para la clase 1 en comparación con la clase 0. Eso se debe a que el modelo le cuesta más predecir la clase 0 en comparación con la clase 1, ya que tenemos un gran desequilibrio de las muestras de la clase del 0 y 1.

En general, el modelo está haciendo un buen trabajo en la predicción de ambas clases, con una alta exactitud, precisión y recall. Y sin lugar a dudas funciona mejor que el modelo base de Naïve Bayes.

	precision	recall	f1-score	support
0	0.33	0.90	0.48	21
1	1.00	0.95	0.97	734
accuracy			0.95	755
macro avg	0.66	0.93	0.73	755
weighted avg	0.98	0.95	0.96	755

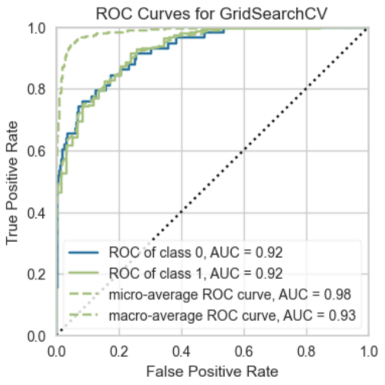
Para analizar los coeficientes del modelo y nuestras características, podemos usar la librería eli5 que nos proporciona herramientas para explicar los modelos de aprendizaje automático y sus predicciones.

Si nos fijamos, el modelo de LDA tenemos que las features que tienen más importancia son las hormonas. De mayor a menor importancia: **TSH, TT4, FTI, T3** y la hormona **T4U**. Por otro lado, las características que tienen menor peso son, como destacamos en la correlación de variables, la variable obviamente encontramos **hypopituitary**, junto con las variables **age** y **I131 treatment**.

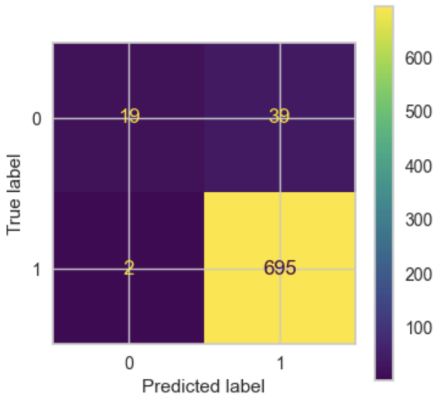
y=1 top features

Weight[?]	Feature
+10.789	TT4
+5.007	FTI
+4.692	T3
+2.739	<BIAS>
+1.786	thyroid surgery
+0.794	lithium
+0.675	sex
+0.648	on antithyroid medication
+0.598	on thyroxine
+0.373	pregnant
+0.342	T4U measured
+0.342	FTI measured
+0.263	goitre
+0.245	psych
+0.214	sick
+0.134	T3 measured
+0.080	age
+0.077	I131 treatment
+0.000	hypopituitary
-0.321	query hyperthyroid
-0.456	tumor
-0.482	query on thyroxine
-0.836	query hypothyroid
-1.128	TSH measured
-1.188	TT4 measured
-4.396	T4U
-39.448	TSH

Si echamos un vistazo a las curvas ROC podemos ver que, de nuevo, son similares, con el mismo AUC, pero ahora con un valor mucho mejor, 0.92.



Por último, utilizando el conjunto de test, obtuvimos una precisión del 94,56% con la siguiente matriz de confusión:



Podemos notar que el error de test y de la validación cruzada son realmente parecidos, ya que nuestro modelo ha sido capaz de generalizar a nuevos datos que aún no había visto.

5.3 KNN

Continuamos viendo modelos, y ahora es el turno del modelo de K-vecinos más cercanos. Para este modelo, hemos procedido de forma análoga al modelo de LDA. Es decir, conjunto de datos preprocesado, 80/20% de proporción en la división de conjuntos y posterior normalización de estos para obtener mejores resultados.

En este modelo también usamos el método `GridSearchCV` de sklearn para encontrar los mejores hiperparámetros para el modelo. Hemos decidido buscar entre cinco hiperparámetros diferentes:

- Número de vecinos: probando los valores de: 1, 2, 3, 4, 5, 7, 9, 11, 13, 15, 20, 25, 30, 40 y 50
- Pesos: probando los pesos de: distance y uniform
- Algoritmo(controla el algoritmo que el modelo debe usar para encontrar los vecinos más cercanos): entre estos cinco tipos, auto, ball_tree, kd_tree y brute
- Métrica: probando las métricas de distancia de: euclidean, manhattan, chebyshev, minkowski, l2 y l1
- Tamaño hoja: probando valores de: 1, 5, 10, 20 y 30

Asimismo, probamos buscar entre un sexto hiperparámetro llamado p. Este hiperparámetro, controla el parámetro de potencia para la métrica de Minkowski. Y a medida que considerábamos un mayor valor de p, nuestro modelo se volvía más sensible al ruido que teníamos en los datos.
Nos dimos cuenta de esto, y decidimos eliminar este hiperparámetro que había añadido complejidad al modelo y resultaba en peores predicciones.

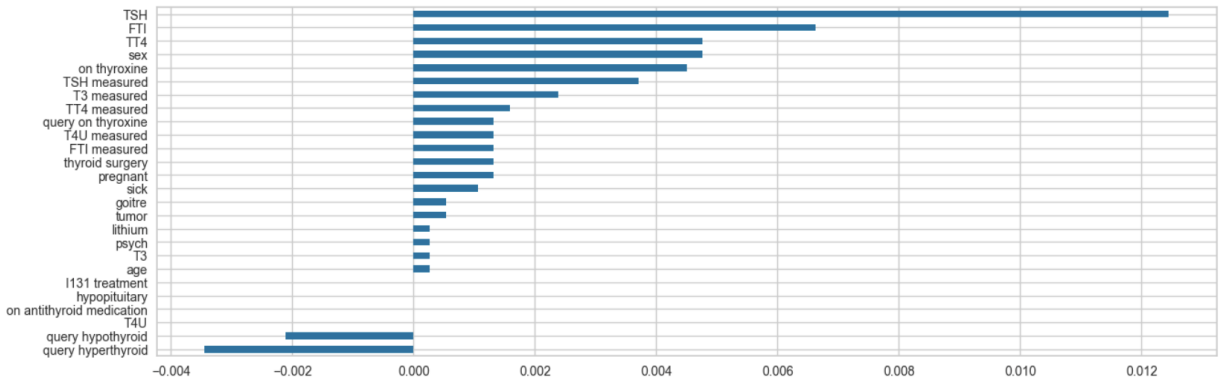
Tras la búsqueda de validación cruzada con el conjunto de entrenamiento, descubrimos que los mejores hiperparámetros eran los siguientes:

{'algorithm': 'brute', 'leaf_size': 1, 'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'distance'}
Con estos hiperparámetros conseguimos una precisión de validación cruzada de 0.939983.

Si echamos un vistazo al informe de clasificación, para la clase 0, el modelo tiene una precisión de 0.24, lo que significa que predice correctamente la clase negativa aproximadamente una cuarta parte de las veces que realiza una predicción para esa clase. El recall de la clase 0 es de 0.78, lo que significa que el modelo predice correctamente una gran parte de las muestras negativas reales.
Para la clase 1, el modelo tiene una precisión del 0.99, lo que significa que predice correctamente la clase positiva, casi siempre que realiza una predicción para esa clase y un recall de 0.94, lo que significa que el modelo predice casi todas las muestras positivas reales.
En comparación con el informe de clasificación del modelo LDA, este modelo tiene una peor precisión y f1-score para ambas clases.

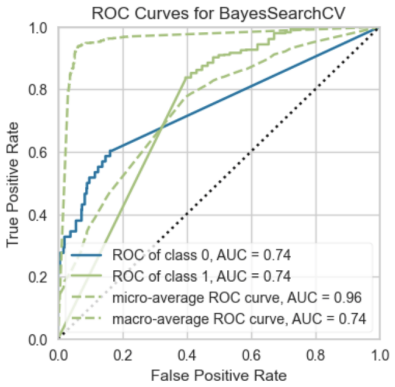
	precision	recall	f1-score	support
0	0.24	0.78	0.37	18
1	0.99	0.94	0.97	737
accuracy			0.94	755
macro avg	0.62	0.86	0.67	755
weighted avg	0.98	0.94	0.95	755

Para analizar la importancia de las características, k-vecinos más cercanos no tiene ponderaciones, pero lo podemos realizar mediante el método de `permutation_importance`.

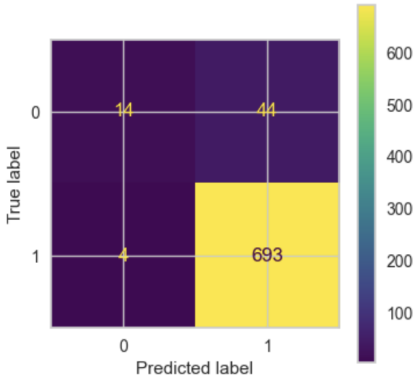


Podemos ver, de nuevo, que las características más importantes son, las hormonas **TSH, FTI, TT4**. También considera que es la variable **sex**, y considerando las variables **query hyperthyroid** y **query hypothyroid,** el modelo de KNN funciona peor.

Si echamos un vistazo a las curvas ROC podemos ver que, de nuevo, son similares, con el mismo AUC, pero ahora con un valor mucho peor al obtenido con el modelo de LDA, 0.74.



Por último, utilizando el conjunto de test, obtuvimos una precisión del 93.64% con la siguiente matriz de confusión:



5.4 SVM lineal

En esta sección, vamos a ver cuál ha sido el rendimiento del modelo que ha sido obtenido con las máquinas de vectores soporte lineales. Hemos procedido de forma análoga con los datos, tal y como hemos hecho para todos los modelos lineales anteriores. Aunque una mejor opción hubiese sido disminuir el tamaño del conjunto de entrenamiento, ya que, por el contrario, el tiempo de cálculo del modelo ha tomado un largo tiempo de cómputo.

Para obtener un buen resultado en este modelo, usamos el método `BayesSearchCV` para encontrar los mejores hiperparámetros y lo hicimos entre cinco diferentes:

- C: Este hiperparámetro es crucial en las SVC, ya que se comporta como un parámetro de regularización en la SVM. Con un mayor valor de C, el modelo será más sensible a los datos de entrenamiento e intentará ajustarse a los datos lo máximo posible. Con un valor menor de C el modelo será menos sensible a los datos de entrenamiento e intentará equilibrar el ajuste de los datos con la simplicidad del modelo. Hemos escogido los valores: 0.1, 1, 10, 100, 1000, 10000, 100000 y 1000000
- Tolerancia: usamos los valores: 1e-3, 1e-4, 1e-5 y 1e-6
- Shrinking: controla si el algoritmo de optimización debe usar la heurística de shrinking para acelerar o no la convergencia, toma valores: True o False
- Gamma: controla la anchura de la función del kernel y determina hasta qué punto se tienen en cuenta en el modelo los puntos alejados del límite de decisión. Cogimos los valores: 1e-3, 1e-4 y 1e-5
- Tamaño de la caché: controla el tamaño de la caché que el modelo SVC debe utilizar para almacenar las evaluaciones que hace el kernel. Toma valores: 200, 500 y 1000.

Descubriendo que los mejores hiperparámetros eran los siguientes:

{'C': 100000.0, 'cache_size': 1000, 'gamma': 0.0001, 'shrinking': True, 'tol': 0.001}

Con estos hiperparámetros el modelo, usando una validación cruzada de 10-fold, obtuvimos una precisión de validación del 97,61%.

Si echamos un vistazo al informe de clasificación, podemos ver que, hasta ahora, este es el mejor modelo lineal de los 4 analizados.

Podemos ver que como veníamos teniendo en los otros modelos, para este también es más fácil clasificar la clase de las personas que tienen hipotiroidismo. Esto es una gran notícia. Ya que nuestro modelo tiene una precisión perfecta para esta clase, y predice correctamente en un 99% las muestras positivas reales. Aun así, también ha mejorado mucho la predicción para la clase de las personas negativas en hipotiroidismo. Como podemos ver, la precisión de los modelos ha ido

escalando, encontrando en este modelo una precisión del 0.83 frente a las precisiones de 0.24 o 0.33 que teníamos en los modelos de LDA y KNN respectivamente. Asimismo, el recall de esta clase también es casi perfecto mejorando con respecto a los otros modelos lineales.

	precision	recall	f1-score	support
0	0.83	0.96	0.89	50
1	1.00	0.99	0.99	705
accuracy			0.98	755
macro avg	0.91	0.97	0.94	755
weighted avg	0.99	0.98	0.98	755

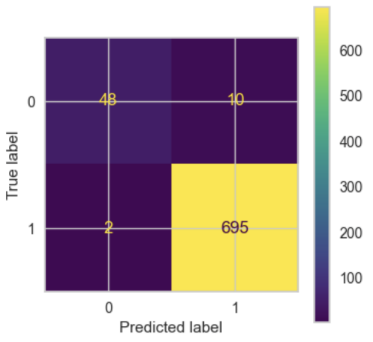
Si vemos cómo distribuye el modelo la importancia de las características, podemos ver que la más importante en valor absoluto es el nivel de la hormona **TSH**. Seguida de la característica **on thyroxine**, de la hormona **TT4**, de la característica **thyroid surgery** y de las hormonas **FTI** y **T4U**. Como ya venían haciendo los otros modelos, la hormona TSH es la que tiene una mayor importancia, dejándonos entrever que sí que parece ser que la correlación que mostraba el coeficiente de Pearson también se refleja en la causalidad de la variable que queremos predecir.

y=1 top features

Weight[?]	Feature
+8.971	on thyroxine
+4.585	TT4
+3.619	thyroid surgery
+2.239	<BIAS>
+1.230	T3
+0.374	I131 treatment
+0.356	age
+0.192	T3 measured
+0.164	lithium
+0.140	sick
+0.094	FTI measured
+0.094	T4U measured
+0.077	on antithyroid medication
+0.040	psych
+0.036	TT4 measured
-0.076	sex
-0.153	query hyperthyroid
-0.258	tumor
-0.298	query on thyroxine
-0.372	TSH measured
-0.539	query hypothyroid
-2.340	T4U
-2.486	FTI
-143.297	TSH

También destacamos que apenas este modelo le ha dado importancia al sexo de los pacientes, ya que tanto KNN como LDA le daban mucha más relevancia. Asimismo, nos gusta que le haya dado un peso importante a la variable **on thyroxine**, puesto que las personas que tienen esta enfermedad toman este medicamento.

Por último, utilizando el conjunto de test, obtuvimos una precisión del 98.41% con la siguiente matriz de confusión:



Como podemos observar, vemos que este modelo funciona realmente bien, ya que como podemos apreciar en la matriz de confusión, el modelo apenas toma decisiones erróneas.

6. Métodos no-lineales

6.1 Random Forest

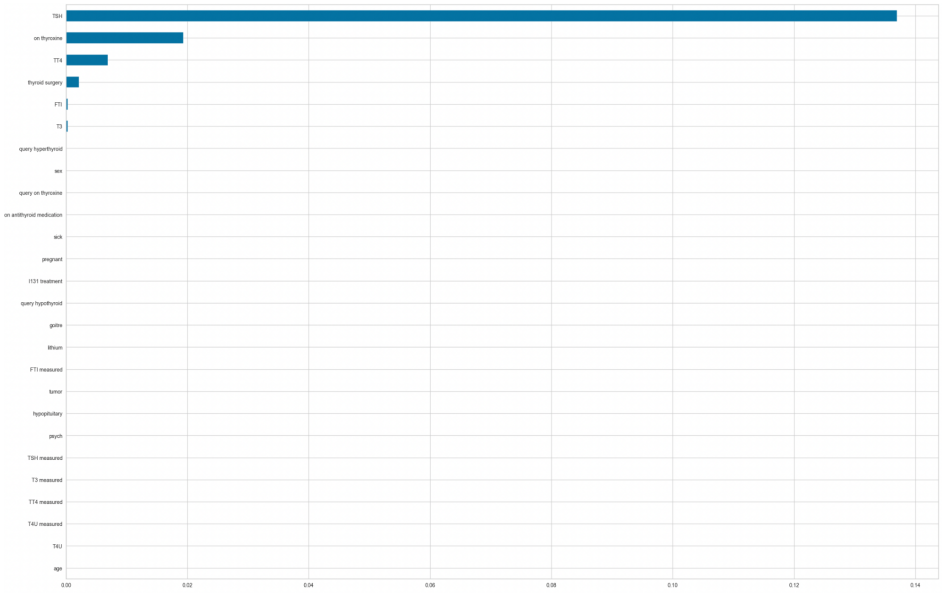
Para nuestro primer modelo no-lineal hemos escogido el Random Forest. El tiempo de entrenamiento de este modelo es bastante corto, por lo que probar de entrenarlo con muchos hiperparámetros posibles no es un gran problema, así como sí que lo ha sido para otros modelos que ya hemos visto anteriormente, como por ejemplo de modelos lineales, el SVM lineal.
Con el objetivo de obtener un buen modelo, hemos usado `BayesSearchCV` para explorar los mejores hiperparámetros para este modelo:

- n_estimators: Este es el número de árboles en nuestro “bosque”. Aumentar el número de árboles generalmente nos conducirá a un mejor resultado, pero también aumentará el tiempo de entrenamiento. Hemos escogido 10 valores entre 200 y 20000
- max_depth: Esta es la profundidad máxima a la que puede llegar cada árbol del bosque, este parámetro es el clásico problema en el tema de machine learning, el sesgo y el sobreajuste. Una profundidad mayor, posiblemente, nos conducirá hacia un sobreajuste, pero una profundidad que no sea suficiente nos dará un modelo posiblemente peor. Los valores que hemos escogido son, None, 10,20,50,100,200,1000.
- max_features: Este es el número de features máximo que se tienen en cuenta a la hora de partir un nodo en el árbol de decisiones. Un número mayor tiene en cuenta más features y va a ser seguramente más preciso para los datos de entrenamiento, pero nos arriesgamos a tener overfitting. Con los valores, None, sqrt, log2
- min_samples_split: Este es el mínimo número de muestras que se requieren para partir un nodo del árbol de decisiones. Aumentar el valor va a hacer que nuestro modelo sea más robusto al overfitting. Le damos los valores posibles 2,10,20,50,100,200,1000,10000.
- criterion: Esta es la métrica que nos va a medir la calidad de un corte, generalmente se suele usar la de Gini, pero también existe la entropía o log_loss. Necesitamos una métrica para poder comparar y ver qué árbol es mejor que otro, sin esto, el algoritmo no podría funcionar. Le damos los valores “gini” y “entropy”
- bootstrap: Este valor nos va a decir si usamos muestras bootstrap o no a la hora de construir los árboles. Le damos los valores “True” y “False”. Este método lo que hace básicamente es coger muestras de nuestro conjunto de muestras grande de forma aleatoria y con **reemplazo**, esto quiere decir que si escoges una muestra, la puedes volver a escoger aunque ya haya salido anteriormente. Entonces tendrías dos o más muestras repetidas.

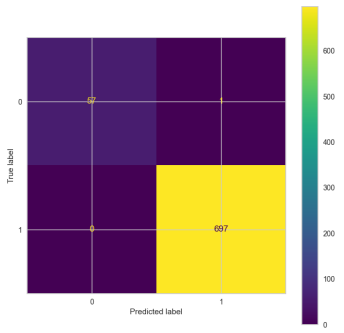
Los mejores paràmetros han sido {'bootstrap': True, 'criterion': 'entropy', 'max_depth': 50, 'max_features': None, 'min_samples_split': 10, 'n_estimators': 6800} con un mean_test_score de 0.997679.
Para este modelo clasificar ambas clases parece bastante fácil, en especial para el caso negativo, pero incluso en el caso positivo encontramos resultados realmente buenos. El recall es perfecto en ambas clases, pero la precisión en el caso de los positivos no llega a ser perfecta.

	precision	recall	f1-score	support
P	0.98	1.00	0.99	57
N	1.00	1.00	1.00	698
accuracy			1.00	755
macro avg	0.99	1.00	1.00	755
weighted avg	1.00	1.00	1.00	755

Para ver cómo distribuye el modelo la importancia de las features hemos usado permutation importance. En este caso vemos que las variables que obtienen la mayor importancia son el **TSH**, seguida por **on thyroxine**, es bueno ver que esta es importante para decidir como ya hemos dicho, **TT4 y thyroid surgery**.



En la matriz de confusión podemos ver claramente que solo hay un ejemplo mal clasificado, unos resultados realmente buenos.



6.2 SVM con kernel polinómico no-lineal

A continuación vamos a ver una alternativa al SVM lineal que habíamos visto anteriormente y que funcionaba considerablemente bien, para ver si usando un kernel polinómico, este funcionara mejor.

Vamos a explorar los mejores hiperparámetros con `BayesSearchCV` de entre los siguientes:

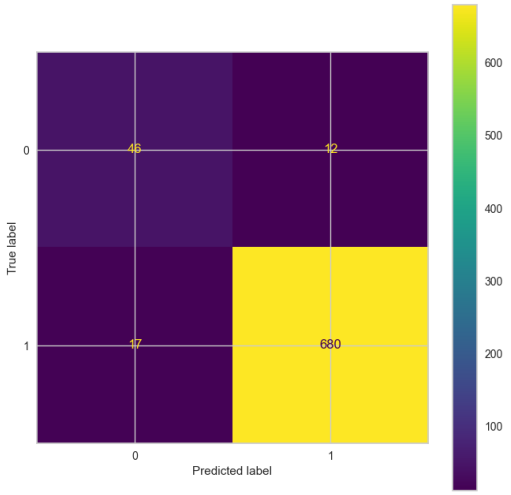
- C: Parámetro de regularización, cómo de importante sea la regularización, es directamente proporcional a la C.
- degree: El grado del polinomio del kernel, debe ser positivo. Le damos los valores 2 y 3.

Para este método hemos usado esos dos parámetros para ver si mejoraba respecto al modelo lineal. De estos, los hiperparámetros que han dado mejor resultado son:
{'C': 10000.0, 'degree': 3} con un mean_test_score de 0.979113.

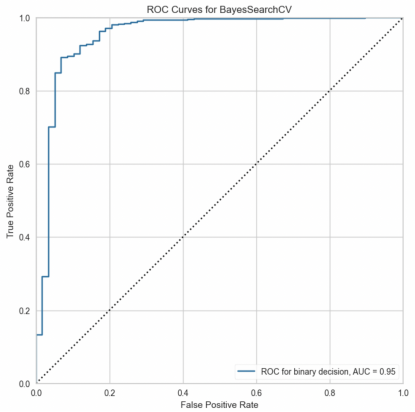
	precision	recall	f1-score	support
0	0.79	0.73	0.76	63
1	0.98	0.98	0.98	692
accuracy			0.96	755
macro avg	0.88	0.86	0.87	755
weighted avg	0.96	0.96	0.96	755

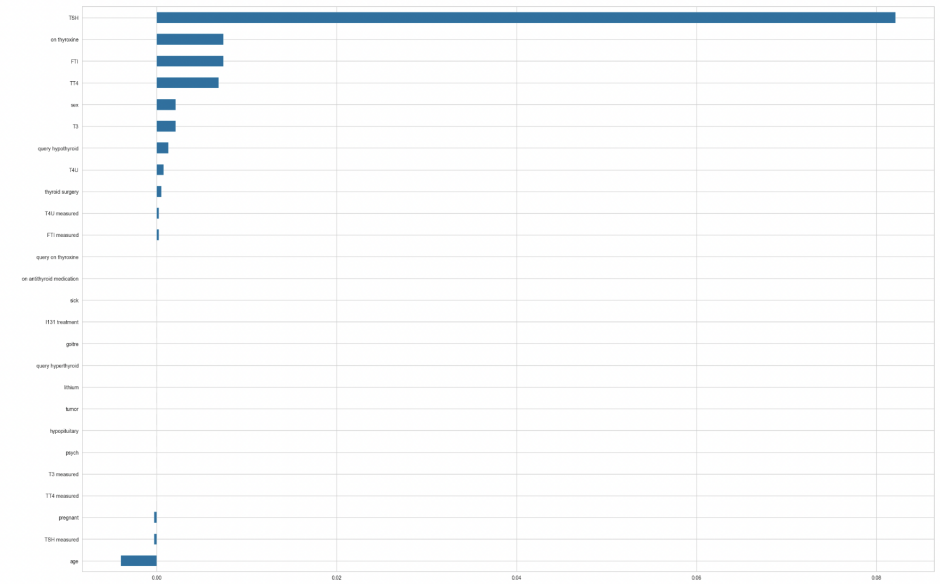
Como podemos apreciar en la tabla anterior, la precisión y el recall han disminuido respecto al SVM lineal, por lo tanto, podríamos pensar que se está sobre ajustando a los datos.

En la matriz de confusión podemos ver que hay dos errores más en la clase 0 y 15 errores más en la clase 1 lo cual supone unos resultados bastante peores.



Para poder ver la importancia de los features, hemos utilizado permutation importance. Claramente, vemos como **TSH** vuelve a ser la más importante seguida por **on thyroxine, FTI y TT4** siendo las más importantes, por la otra parte, con importancias negativas tenemos la edad.





6.3 MLP

Vamos a continuar viendo modelos no-lineales, en este caso veremos las redes neuronales MLP (multi-layer Perceptron)

Vamos a explorar los siguientes hiperparàmetros con `GridSearchCV`:

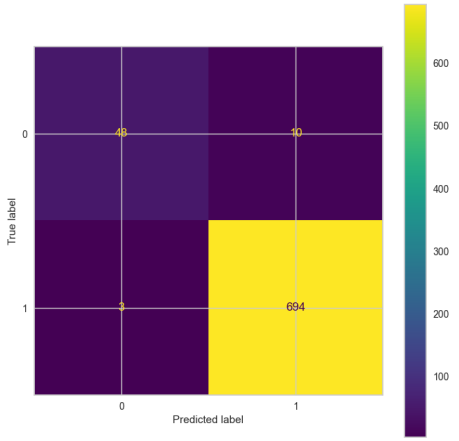
- hidden_layer_sizes: Este parámetro corresponde con tuplas que especifican el número de unidades en cada capa, por ejemplo la dupla (100,100), quiere decir un MLP donde tenemos dos capas, cada una con 100 neuronas. Hemos escogido los siguientes valores, 10, 50, 100, 200, 300 (una sola capa para cada uno).
- activation: Estas son el tipo de funciones que se van a ajustar en nuestras capas ocultas. Puede tomar los valores “relu” “identity” “logistic”.
- alpha: Esta es la fuerza que le vamos a dar al término de regularización para ajustar entre sesgo y overfitting. Le asignamos: 0.0001, 0.001, 0.01.
- momentum: Este parámetro determina la influencia de la actualización del peso anterior en la actualización actual. Se usa para que en el descenso de gradiente, lleguemos a la solución más rápidamente. Es un valor que va entre 0 y 1. Cuando este valor es cercano a 1 (normalmente se va a ajustar cercano a 1) el optimizador va a hacer actualizaciones de mayor peso, cosa que ayuda a converger rápidamente. Sin embargo, un valor demasiado alto puede converger en una solución subóptima o divergir. Toma los valores: 0.95, 0.9, 0.85, 0.8.
- learning_rate_init: Este es el learning rate inicial con el que el modelo va a empezar a aprender. Los valores dados son: 0.001, 0.01, 0.1.
- n_iter_no_change: Es el número máximo de epochs permitidos para no llegar a “tol” , es decir, cuando el resultado no está mejorando al menos por un valor tol, durante n_iter_no_change, entonces consideramos que hemos llegado a converger. Le asignamos valores: 10, 20, 40, 50.
- learning_rate: Este es la forma en que se va a usar el learning rate. “constant” quiere decir que el learning rate va a usar el learning_rate_init hasta el final. “Invscaling” decrementa gradualmente el learning rate a cada paso usando la inversa de una exponencial. “adaptive” Mantiene el learning rate constante mientras vayamos acercándonos a mejores soluciones, cuando en dos epochs consecutivas no mejoramos, decrementa el learning rate. Los valores posibles son “constant” “invscaling” “adaptive”.

Al acabar, los hiperparámetros que han dado mejor resultado han sido los siguientes: {'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': 200, 'learning_rate': 'adaptive', 'learning_rate_init': 0.1, 'momentum': 0.8, 'n_iter_no_change': 40}

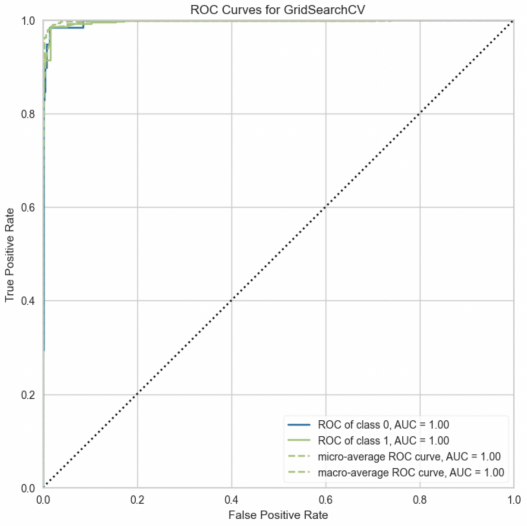
Los resultados que podemos observar en la tabla son bastante buenos, a la par con el SVM lineal; sin embargo, nada que ver con los resultados vistos con el Random Forest con una precisión y recall en ambas clases casi perfectas. En la clase 0 vemos que la precisión es bastante más baja y el recall también es un poco más bajo.

	precision	recall	f1-score	support
0	0.83	0.94	0.88	51
1	1.00	0.99	0.99	704
accuracy			0.98	755
macro avg	0.91	0.96	0.94	755
weighted avg	0.98	0.98	0.98	755

Dando un vistazo rápido a la matriz de confusión vemos que en la clase 1 sólo ha habido 3 errores respecto a los 694 aciertos; sin embargo, en la clase 0 ha habido 10 errores respecto a los 48 aciertos, un error bastante considerable.



Viendo la curva ROC, vemos un resultado muy bueno, y una área bajo la curva casi perfecta, cercana a 1.



6.4 Voting classifier

Vamos a ver el último modelo no-lineal, como ya hemos visto modelos que ya han dado muy buenos resultados, hemos decidido ver si una combinación de estos puede darnos unos resultados aún mejores, este clasificador por votos podría ser la solución.
Se ha decidido utilizar un **Random Forest**, ya que ha dado muy buenos resultados, **MLP** y el **SVM polinómico**.

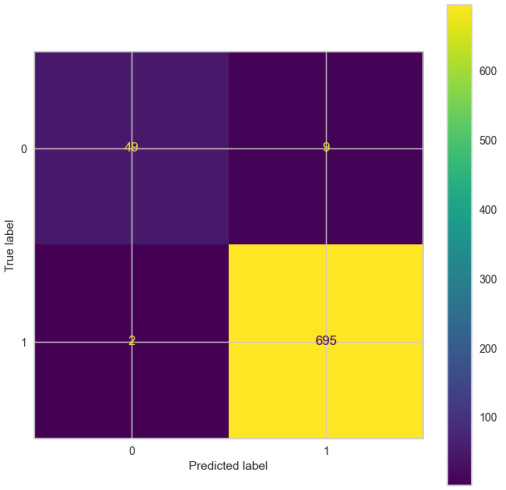
Los parámetros para ajustar este modelo básicamente son los pesos que se le da a cada modelo y le hemos decidido dar el mismo peso a todos los modelos para ver cómo se comportaba de esta manera, puesto que el tiempo de entrenamiento era bastante prohibitivo como para entrenar con muchas distribuciones de pesos.

Como podemos ver, los resultados no son los que esperábamos, pero para ser sinceros era bastante difícil de superar los ya muy buenos resultados del Random Forest, quizás juntar ese modelo tan bueno con otros dos que no lo son tanto, no ha hecho más que tomar decisiones equivocadas y sospechamos que aumentando

	precision	recall	f1-score	support
P	0.84	0.96	0.90	51
N	1.00	0.99	0.99	704
accuracy			0.99	755
macro avg	0.92	0.97	0.95	755
weighted avg	0.99	0.99	0.99	755

el peso que tiene el random forest para decidir acabaría dándole todo el paso para decidir al Random Forest.

También podemos ver la matriz de confusión, donde en la clase 1 ha cometido solo 2 errores, pero con hasta 9 errores en la clase 0 donde tenemos menos muestras, por lo tanto, ese error es aún más notorio.



7. Modelo final elegido

En esta sección seleccionaremos el modelo final para el problema de clasificación de pacientes con un cuadro clínico positivo o negativo en hipotiroidismo.

Para ello, a continuación mostramos los resultados obtenidos(ordenados según la calidad obtenida en el conjunto de test) en todos los modelos mediante la validación cruzada 10-fold y la calidad del conjunto de test mostrando las precisiones obtenidas por cada uno de ellos.

Clasificador	Precisión de la validación cruzada 10-fold	Precisión para el conjunto de test
Random Forest	99.76%	99.86%
Voting Classifier	98.54%	98.54%
SVM lineal	97.61%	98.41%
MLP	98.04%	96.82%
SVM polinómico	97.91%	96.15%
LDA	94.06%	94.56%
KNN	93.99%	93.64%
Naïve Bayes	22.05%	21.72%

Cabe destacar que para escoger el mejor modelo no utilizamos en ningún caso únicamente en las precisiones obtenidas con el conjunto de test para tomar decisiones con respecto a la selección del mejor modelo. Sino que nos fijamos en las f1-scores obtenidas por los modelos, el equilibrio y acierto de cada clase, como asimismo el comportamiento en las predicciones de cada una, las curvas ROC y el posible sobreajuste de que puedan hacer cada uno de los modelos con el dataset. De nada nos serviría un modelo con una buena precisión para una clase, si este no es capaz de predecir con precisión los ejemplos de la otra. Es por esto que buscamos el mejor

modelo global que se comporte de forma adecuada para ambas clases y que su generalización y sobreajuste sean aceptables.

Podemos apreciar que los modelos que han funcionado de una manera más precisa han sido los modelos no-lineales. Estos han sido capaces de capturar mejor los patrones entre nuestros datos, aunque como podemos notar, los modelos lineales no se comportan de una forma mucho peor; así que los modelos más simples tampoco parecen ir mal para nuestro conjunto. Lo que sí que podemos notar es que los modelos lineales no son capaces de capturar los patrones para predecir pacientes que no tengan hipotiroidismo, si nos fijamos en el bajo acierto de los modelos lineales para esta clase.

Destacamos que los modelos que se han comportado de peor manera han sido Naïve Bayes y KNN. Cabe destacar que los modelos de SVM se han comportado de forma similar, obteniendo unas precisiones del 97.62% y del 97.91% usando el kernel lineal y polinomial respectivamente.

Por todas las conclusiones sacadas en cada modelo y vistos los resultados obtenidos, podemos ver que el modelo que mejor se comporta para ambas clases es el modelo de **Random Forest**.
Este modelo obtiene una precisión para el conjunto de test del **99.86%** con unas puntuaciones obtenidas en el *classification report* casi perfectas ya que podemos observar que los f1-score son del 0.99 i del 1.0. Además como podemos ver, una cosa que puede suceder al tratar con árboles es que se produzca un sobreajuste de los datos, pero vemos que nuestro modelo generaliza de forma prácticamente perfecta sin tener problema alguno.

8. Interpretabilidad de dos modelos

Aunque ya hemos tratado la interpretabilidad de cada uno de los modelos por separado, a continuación profundizaremos más en la interpretabilidad de dos modelos. Así pues, explicaremos y analizaremos los resultados del análisis de interpretabilidad de un modelo lineal y un modelo no-lineal. Los modelos escogidos para hacer este análisis son el mejor modelo no-lineal, **Random Forest** y el modelo lineal, **LDA**.

Lo que nos interesará será ver y analizar ejemplos mal clasificados por ambos modelos. Los ejemplos clasificados incorrectamente nos van a permitir ver las flaquezas de ambos modelos viendo las variables que empujan a la muestra hacia la otra clase. Aunque ambos modelos se comporten de forma diferente, comparten que algunas features importantes coinciden, como es la variable **TSH** o **TT4**.

Este último modelo que hemos seleccionado puede sorprender su elección, puesto que difiere del mejor modelo lineal, que es la SVM lineal, pero esta se debe a que nos parece interesante ver cómo este modelo, a diferencia del mejor lineal, da muchos ejemplos mal clasificados para la clase negativa y ver cuál puede ser el motivo que empuja el modelo a clasificar incorrectamente una muestra.

8.1 Random Forest

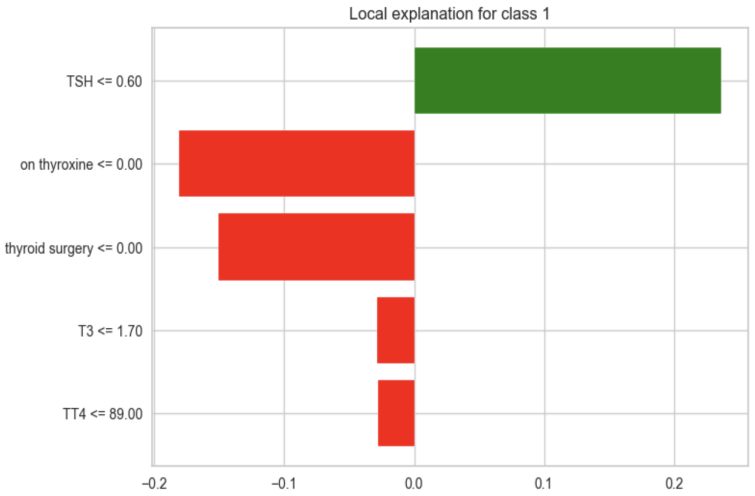
Para poder resumir mejor el ejemplo de nuestras muestras hemos decidido escoger solo las features que se han visto con el método de permutation importance que toman una mayor importancia a la hora de decidir si una muestra es positiva o negativa.

En cuanto a las gráficas que veremos a continuación, los pesos en verde significan que esa variable está empujando la muestra hacia la clase positiva, i.e la clase de los pacientes con hipotiroidismo.

En cambio, los pesos en rojo simbolizan las variables que empujan a la muestra hacia la clase negativa.

True Positive:

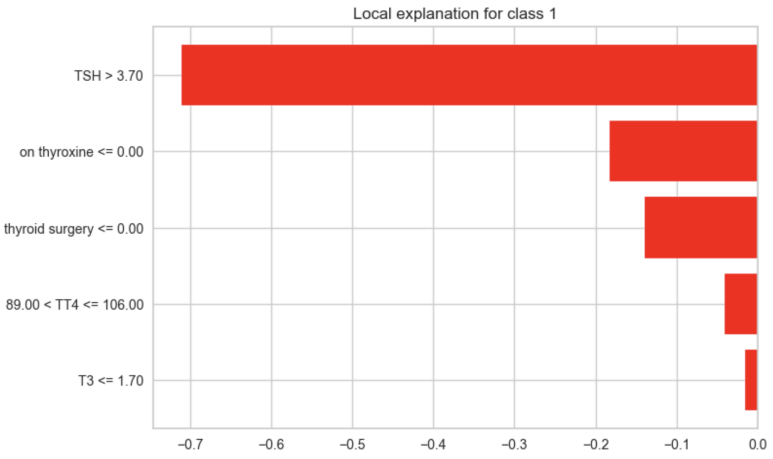
on thyroxine	0.00
thyroid surgery	0.00
TSH	0.40
TT4	88.00



Podemos observar en la gráfica que la variable predictora que nos ha empujado esta muestra hacia la clase positiva es principalmente **TSH**, todas las demás, estaban tirando hacia la clase negativa. Aunque parece ser que se ha clasificado como positiva por muy poco.

True Negative:

on thyroxine	0.00
thyroid surgery	0.00
TSH	7.10
TT4	96.00

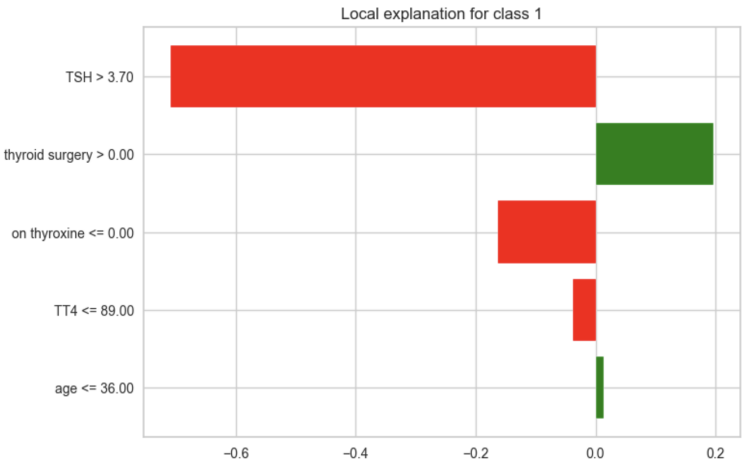


En esta gráfica vemos que todas las features importantes que nos indican que la muestra va a ser negativa y, que, por lo tanto, no ha tenido dificultades para identificarla correctamente donde le corresponde.

False Positive:

Este error seguramente ha sido debido a que la persona ha tenido cirugía de tiroides.

on thyroxine	0.00
thyroid surgery	1.00
TSH	103.00
TT4	65.00



Como vemos aquí, las dos features que tiran hacia la clase positiva son la cirugía de tiroides como se había predicho y la edad, sorprende ver la edad, ya que normalmente la edad influye hacia la clase negativa o a veces ni siquiera es muy importante a la hora de decidir; sin embargo, en este caso vemos cómo ha sido un contribuyente a que tengamos un falso positivo.

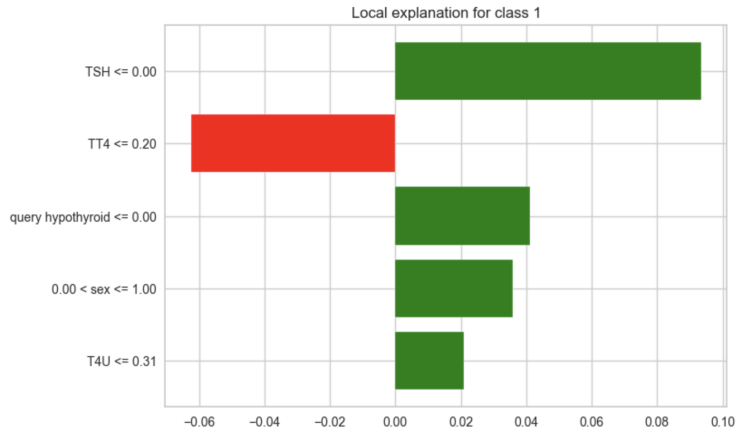
False Negative:
No hay falsos negativos en los datos de test para este modelo, así que podemos concluir que ha aprendido bien cómo no cometer errores en los falsos negativos.

8.2 LDA

Ahora, con el mismo modo de ejecución que con el modelo no-lineal, veremos la importancia de las variables en las decisiones de clasificación tomadas.

True positive:

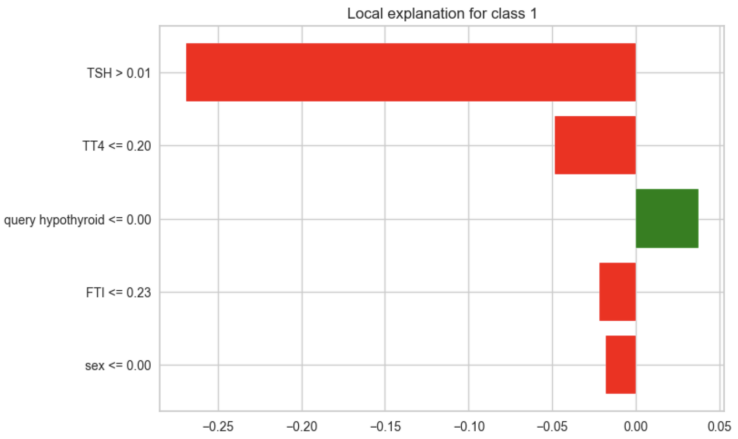
sex	1.000000
query hypothyroid	0.000000
TSH	0.000745
TT4	0.200935



En esta decisión la única variable que podía haber confundido a nuestro modelo es la de **TT4**, pero no ha sido el caso, el resto de variables importantes, empujan nuestro modelo a clasificar la muestra como positiva y ese ha sido el caso.

True negative:

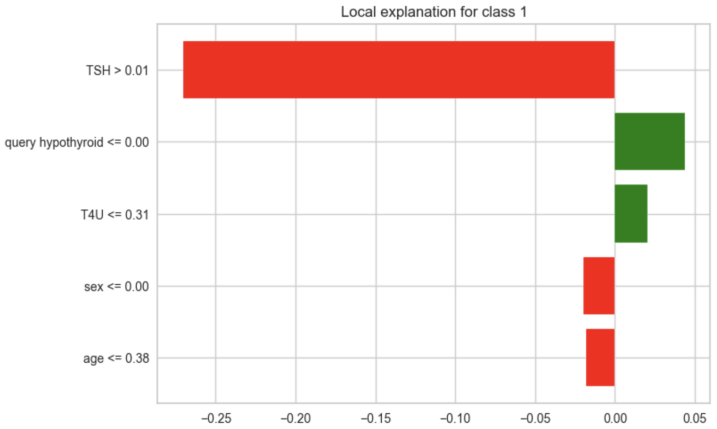
TSH	0.098105
TT4	0.147196
query hypothyroid	0.000000
FTI	0.175573



Ahora vemos cómo ha clasificado correctamente una muestra negativa, la mayor parte de las features con pesos negativas, especialmente una de las más importantes, la **TSH**, la única que tiene pesos positivos es la de **query hypothyroid**.

False Positive:

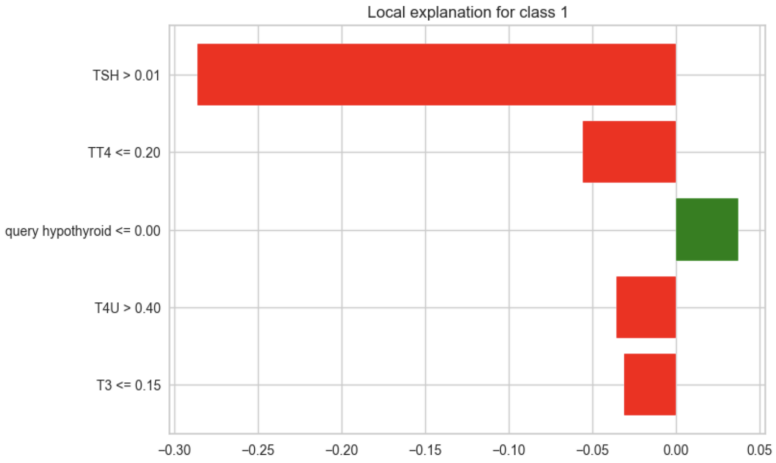
TSH	0.013387
query hypothyroid	0.000000
T4U	0.309179
sex	0.000000



En este caso, incluso con variables como **TSH** que nos lleva hacia la clase negativa, query hypothyroid y **T4U** han conseguido empujar a nuestra variable hacia clasificar la muestra como positiva, erróneamente.

False Negative:

TSH	0.143388
TT4	0.046729
query hypothyroid	0.000000
T4U	0.420290



Esta muestra ha sido clasificada como negativa, ya que los indicios de la mayoría de variables indican que nuestra muestra debería ser negativa, sin embargo, esto no es así, podría ser un outlier o bien podría haber algún error fundamental con el modelo.

9. Conclusiones

Una autoevaluación de éxitos, fracasos y dudas

Estamos muy contentos con los buenísimos resultados obtenidos para los modelos que hemos elaborado. Al principio, al ver el resultado obtenido con Naïve Bayes nos asustamos un poco y pensábamos que no obtendríamos muy buenos resultados con los otros modelos, pero una vez empezamos a leer la literatura y a ejecutar los modelos, vimos que la precisión de estos era concorde a otros resultados previamente obtenidos. Estamos contentos también porque vimos reflejado en nuestro trabajo cómo otros modelos anunciados en la literatura también funcionaban muy bien para nuestro conjunto de datos, como es el caso del modelo de *Extra-Trees*[3]

[3] Se puede consultar este modelo en el notebook correspondiente dónde se encuentran los modelos ejecutados.

Además, al principio también al tratar con un dataset con tantos valores NaN nos entró un poco el pánico, pero los analizamos y vimos cuál era el mejor abordaje para tratarlos. Asimismo, cometimos algunos errores en el preprocesamiento que ya cuándo estábamos ejecutando los modelos nos dimos cuenta y nos forzó a volver atrás.
Adicionalmente, dudamos mucho con la inclusión de ciertas variables en el dataset, pues parecían que no tenían correlación y causalidad alguna con la variable objetivo. Nuestro fracaso en este aspecto ha sido la falta de conocimiento sobre algunas variables predictoras, pues la falta de conocimiento experto hubiese sido un punto de aporte positivo para analizar mejor la interpretabilidad.
Por otra parte, tuvimos algunas dudas con respecto a la misma interpretabilidad, pero las pudimos resolver rápidamente mediante los notebooks de la asignatura, viendo cómo se había abordado la explicabilidad de los modelos.

Conclusiones científicas y personales
Al principio de empezar este trabajo teníamos muchas dudas a la hora de usar el conjunto de datos que obtuvimos de la plataforma de *OpenML*. Éste nos generó muchísimas dudas cuándo hicimos una breve exploración de los datos y vimos el resultado obtenido con el modelo base de Naïve Bayes. En este vimos que faltaban muchos valores y que algunas columnas no aportaban ningún valor en la predicción de la variable objetivo.

Pero creemos que una vez realizados los experimentos, el objetivo de encontrar un modelo capaz de clasificar a los pacientes con hipotiroidismo de los que no ha sido satisfactorio, ya que utilizando nuestro modelo final tenemos un rendimiento muy bueno.

Al ver que hemos sido capaces de tener un modelo con tan buenos resultados, pensamos que nuestro modelo podría tener aplicaciones útiles en la sociedad, y más concretamente en el campo de la salud. Por el hecho de que el hipotiroidismo es una afección médica muy común y tratable y si no se trata, puede tener importantes repercusiones en la salud. Por ejemplo, ayudar al diagnóstico precoz de la enfermedad a los médicos para comenzar antes el tratamiento, lo que podría mejorar la calidad de vida del paciente y prevenir la aparición de problemas de salud más graves. Además, creemos que nuestro modelo podría ayudar a los médicos a identificar a los pacientes con riesgo de padecer la enfermedad y tomar medidas preventivas para reducirla. Como también ayudar a descartar pacientes que parecen tener síntomas de hipotiroidismo.

Posibles extensiones y limitaciones conocidas
Este proyecto opinamos que puede ser extensible de muchas maneras. Para seguir trabajando en este consideramos que sería beneficioso añadir más datos a nuestro conjunto que sean de pacientes que no tienen hipotiroidismo. De esta manera conseguiremos que el modelo, no solo funcione de forma perfecta para el diagnóstico de pacientes con la enfermedad, sino también para persona que no la tengan. Además, que por parte de los médicos no se haga un diagnóstico erróneo de esta. Consideramos que lo que se debería hacer es un mix de distintos datasets que han usado los distintos papers leídos, cruzando datos y haciendo un conjunto de datos más atractivo, con features que tengan más relación con la variable objetivo y/o añadiendo más características que puedan ser de relevancia para esta aplicación. También sería interesante hacer un trabajo de campo en distintas instituciones, centros y hospitales y establecer contacto con personal experto en la materia para perfilar y mejorar nuestro modelo para sacarle más partido a la interpretabilidad.
En cuanto a las limitaciones, cabe destacar el elevado tiempo de ejecución de algunos modelos, lo que nos ha frenado un poco en la búsqueda de mejores hiperparámetros en algunos modelos,

como por ejemplo en el modelo de SVM juntamente con el modelo de VotingClassifier, ya que usa los tres mejores modelos.

10. Bibliografía

Dewangan, A., Shrivas, A., & Kumar, P. (2016). Classification of thyroid disease with feature selection technique. Int J Eng Tech, 2, 128-131.

Mahurkar, K. K., & Gaikwad, D. P. (2017). Normalization using Improvised K-Means applied in diagnosing thyroid disease with ANN. 2017 International Conference on Trends in Electronics and Informatics (ICEI), 579-583. https://doi.org/10.1109/ICOEI.2017.8300768

Shrivas, A.K., & Ambastha, P.K. (2017). An ensemble approach for classification of thyroid disease with feature optimization. International Education and Research Journal, 3.

Moon Jae Hoon, & Steinhubl, S.R. (2019). Digital medicine in thyroidology: A new era of managing thyroid disease. Endocrinol Metab, 34(2), 124-131. https://doi.org/10.3803/EnM.2019.34.2.124

R, Chandan, Vasan, C., MS, Chethan, & S, Devikarani. (2021). Thyroid detection using machine learning. International Journal of Engineering Applied Sciences and Technology, 5, 28. https://doi.org/10.33564/IJEAST.2021.v05i09.028

Aversano, L., Bernardi, M.L., Cimitile, M., Iammarino, M., Macchia, P.E., Nettore, I.C., & Verdone, C. (2021). Thyroid disease treatment prediction with machine learning approaches. Procedia Computer Science, 192, 1031-1040. https://doi.org/10.1016/j.procs.2021.08.106

Islam, S.S., Haque, M.S., Miah, M.S.U., Sarwar, T.B., & Nugraha, R. (2022). Application of machine learning algorithms to predict the thyroid disease risk: an experimental comparative study. PeerJ Computer Science, 8, e898. https://doi.org/10.7717/peerj-cs.898

Tahir Alyas, M. Hamid, K. Alissa, T. Faiz, N. Tabassum, & A. Ahmad. (2022). Empirical method for thyroid disease classification using a machine learning approach. BioMed Research International, 2022, 9809932. https://doi.org/10.1155/2022/9809932

Anika Shama, M.B. Hossain, A. Adhikary, et al. (2022, May 3). Prediction of hypothyroidism and hyperthyroidism using machine learning algorithms [Preprint]. Research Square. https://doi.org/10.21203/rs.3.rs-1486798/v2

Moharekar, T. T., Vadar, M. S., Pol, U. R., Bhaskar, P. C., & Moharekar, M. T. (2022). Thyroid Disease Detection Using Machine Learning and Pycaret. Specialusis Ugdymas, 1(43), 10150-10160.