2021

Aprendizaje Supervisado

MÁSTER EN BIOLOGÍA COMPUTACIONAL 2021-2022

ASIGNATURA: APRENDIZAJE AUTOMÁTCO

SANDRA ALONSO PAZ

Contenido

Metodología

1. Estudio de los datos de entrada.

Uno de los mayores problemas al que nos enfrentamos a la hora de crear y entrenar un modelo de clasificación es encontrar un conjunto de datos que se adapte a nuestras necesidades.

a. Recolección de datos

Al tratarse de un proyecto académico, el proveedor de los datos es el profesor de la asignatura de "Aprendizaje automático" y responder a las principales preguntas que nos permiten saber si los datos pertenecen a una fuente de datos fiable (¿Quién podría tener los datos que necesito?, ¿Por qué decidieron ponerlo a mi disposición? y ¿Cómo puedo hacerme con estos datos?) resulta sencillo.

Los datos fueron descargados del *Moodle* de la asignatura y se utilizarán como conjunto de datos para la creación y análisis de distintos modelos de clasificación.

b. Tipología del conjunto de datos

Disponemos de un conjunto de datos relacionados con el cáncer colorrectal (CCR), una de las principales causas de muerte y su diagnóstico y tratamiento precoz puede llevar a una recuperación completa. Además, se sabe que cada paciente responde de manera distinta al tratamiento debido a su información génica.

Los datos proporcionados contienen información sobre diferentes SNP de algunos pacientes diagnosticados con esta enfermedad y la categorización de los mismos en función de su buena (R) o mala (NR) respuesta al tratamiento.

Todos los datos han sido numerados del 1 al 53 (número de pacientes) y se compone de 21 características (SNP) y un valor objetivo (respuesta ante el tratamiento). En cada columna podremos encontrar valores del 0 al 3 (MM, WW, WM, MW) que corresponde a los diferentes SNP.

c. Análisis de los datos.

Con el fin de no repetir código en los diferentes Jupyter Notebooks he creado uno aparte donde he podido visualizar en un formato más legible el número total de datos, nombre de sus columnas, su proporción de R y NR y algunas características como la media y la desviación estándar.

Esto me ha permitido ver que, por ejemplo, **la media de los valores que toma NR es considerablemente mayor que los de R**. Esto a la hora de analizar los modelos puede ser un punto clave porque podemos inferir que si el paciente tiene SNP de tipo MM y WW es más probable que tenga una buena respuesta al tratamiento y sin embargo las combinaciones de WM y MW podrían conllevar una mala respuesta al mismo.

2. Lectura y preparación del conjunto de datos de entrada

a. Transformación de atributos

A la vista de los datos descritos en el apartado anterior, podemos observar que todos ellos han sido transformados de un valor categórico (MM, WW, WM, MW) a un valor numérico (0,1,2,3). Sin embargo, la columna que contiene el valor objetivo aún está compuesto por dos valores, R y NR. Para que estos valores sean compatibles con el formato de los modelos de aprendizaje los he binarizado. Este proceso se puede llevar a cabo mediante librerías como *sklearn.preprocessing* y funciones como *OneHotEncoder()*. Sin embargo, al tratarse de únicamente dos valores yo he decidido hacerlo mano recorriendo toda la columna "Target" y creando un Array donde R ha sido representado mediante 0 y NR mediante 1.

b. Análisis de las componentes principales (ACP)

A la hora de analizar cada una de las columnas de las que estaba compuesto nuestro conjunto de datos, barajé la posibilidad de realizar un análisis de las componentes principales. Al tener tan pocos datos de pacientes y tantas características me pareció que necesitaba simplificar el número total de estas. Sin embargo, tras varias pruebas no vi que características altamente significativas y por lo tanto descarté esta opción.

3. Separación del conjunto de datos en datos de entrenamiento y de prueba.

Los algoritmos de Machine Learning aprenden de los datos con los que los entrenamos. A partir de ellos, intentan encontrar o inferir el patrón que les permita predecir el resultado para un nuevo caso. Pero, para poder calibrar si un modelo funciona, necesitaremos probarlo con un conjunto de datos diferente. Por ello, en todo proceso de aprendizaje automático, los datos de trabajo se dividen en dos partes: datos de entrenamiento y datos de prueba o test.

Normalmente el conjunto de datos se suele repartir en un 70% de datos de entrenamiento y un 30% de datos de test, pero se puede variar la proporción según el caso. Lo importante es ser siempre conscientes de que hay que evitar el temido sobreajuste u "overfitting".

Para esta tarea he utilizado la librería *sklearn.model_selection* y su función *train_test_split().* En ella he fijado *X* como el conjunto de características, *Y* como la columna objetivo, el tamaño del conjunto de prueba que en la mayoría de modelos he mantenido en el 30% y por último el *random_state* en 125 para analizar siempre el mismo conjunto de datos en todos los modelos.

4. Seleccionar un modelo y sus parámetros.

Cada modelo cuenta con una serie de parámetros que nos permite variar sus características para ajustar más o menos un modelo. Para consultar estos parámetros he utilizado la guía scikit-learn. Una vez conocidas las características del modelo he utilizado la librería *sklearn.model_selection* y su función *GridSearchCV()* que permite evaluar y seleccionar de forma sistemática los parámetros de nuestro modelo. Indicándole un modelo y los parámetros a probar, puede evaluar el rendimiento del primero en función de los segundos mediante la validación cruzada. Además, podemos obtener el modelo con mejor respuesta mediante consultando su atributo *best_estimator_*

5. Entrenar el modelo con el conjunto de datos de entrenamiento.

Una vez que tenemos el conjunto de datos de entrenamiento y el modelo con mejor respuesta, procederemos a entrenar el modelo mediante la función *fit().*

6. Evaluación del modelo mediante técnicas validación cruzada.

La validación cruzada o cross-validation es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba. El proceso de ajuste visto en el apartado de selección de un modelo y sus parámetros, optimiza las características del modelo para que éste se ajuste a los datos de entrenamiento lo máximo posible.

Si tomamos una muestra independiente como dato de prueba, normalmente el modelo no se ajustará a los datos de prueba igual de bien que a los datos de entrenamiento. Esto se denomina sobreajuste y acostumbra a pasar cuando el tamaño de los datos de entrenamiento es pequeño o cuando el número de parámetros del modelo es grande. En nuestro caso se cumplen ambos requisitos y por lo tanto es necesario aplicar la validación cruzada.

Existen multitud de técnicas de cross-validation. Sin embargo, entre las más utilizadas se encuentran k-fold cross-validation o validación cruzada de k-iteraciones, validación cruzada aleatoria, bootstrapping y la validación cruzada dejando uno fuera o Leave one out.

Aunque no existe un método de validación que supere al resto en todos los escenarios, la elección debe basarse en varios factores. Según algunos autores como cienciadedatos exponen que, si el tamaño de muestra es pequeño, como en nuestro caso, se recomienda emplear repeated k-Fold-Cross-Validation, ya que consigue un buen equilibrio bias-varianza y, dado que no son muchas observaciones, el coste computacional no es excesivo.

En este sentido, primeramente, cree modelos sin *GridSeacrchCV* y probé las distintas técnicas. Pude ver entonces que el valor que mejor ajustada en la mayoría de los casos es 5 (que además coincide con el valor por defecto de

GridSeachCV). Por ello, para que los resultados de todos los modelos sean comparables, he utilizado cv=5 en todos ellos.

7. Selección del modelo con mejores resultados y entrenarlo con el conjunto de datos de entrenamiento.

El mejor modelo es le obtenido mediante el atributo *best_estimator* del resultado de la función GridSearchCV(). Entrenando este modelo con el conjunto de datos de entrenamiento podremos realizar las predicciones y el estudio y análisis de los resultados.

8. Calcular el rendimiento del modelo utilizando el conjunto de datos de prueba.

Para ello utilizaremos el modelo con mejores resultados, obtenido en el apartado anterior, y utilizaremos la función *predict()* para obtener un array compuesto por los valores de la predicción. Para calcular la precisión del modelo comprobaré el set de prueba con los resultados obtenidos y obtendremos el porcentaje de acierto del modelo.

Es importante destacar que un buen modelo no tiene por qué obtener un 100% de precisión, pero sí que es recomendable que sea lo más alta posible.

9. Análisis de los resultados obtenidos.

Gracias a la matriz de confusión y al report proporcionado por la función *classification_report(),* podremos hacer un análisis de la precisión, recall y f1-score.

Es importante hacer hincapié en la distribución de la matriz de confusión. Al tratarse de un software de diagnóstico, no se pueden tomar en cuenta de la misma manera los falsos positivos y los falsos negativos.

- Falso positivo: quiere decir que nuestro modelo ha clasificado a un paciente como que tendrá una mala respuesta al tratamiento. Sin embargo, al compararlo con el resultado real, comprobamos que realmente se podría beneficiar del tratamiento puesto que tendrá una buena respuesta.
- Falso negativo: implica que nuestro algoritmo ha clasificado a ese paciente como que obtendrá una buena respuesta al tratamiento. Sin embargo, al compararlo con el resultado real, comprobamos que realmente tendrá una mala respuesta.

En este sentido, es mejor tratar de **eliminar** casi al completo los **falsos negativos** porque al clasificarlos como buena respuesta, el doctor muy seguramente proporcionará al paciente un tratamiento que finalmente no surtirá efecto y por lo tanto estaremos poniendo en riesgo su salud.

Por otra parte, aunque no es beneficiosos tampoco, contar con falsos positivos no dañará tan dramáticamente la salud del paciente puesto que, aunque no hayan sido considerados aptos para este tratamiento, se podrá buscar otros o en su defecto realizar más pruebas o aplicar otros modelos de clasificación.

En cuanto a la métrica utilizada para comparar la precisión, el recall y la f1-score he utilizado la media ponderada porque no hay los mismos R y NR en la muestra de datos, por lo que es interesante tener en cuenta la proporción de los mismo a la hora de analizar el resultado.

Modelos de clasificación

1. Regresión logística

La regresión logística o Logistic Regression es un algoritmo de clasificación que se utiliza para predecir la probabilidad de una variable dependiente categórica. En este modelo, la variable dependiente es una variable binaria que contiene datos codificados, en nuestro caso R o NR (0,1).

Los resultados de este tipo de modelos son fácilmente interpretables debido a su simplicidad. Sin embargo, este modelo funciona bien cuando hay una gran cantidad de datos y las interrelaciones entre ellos no son muy complejas.

En cuanto a los parámetros del modelo debemos destacar:

- **Solver**: algoritmo utilizado en el problema en cuestión. En este parámetro tuve que descartar los algoritmos *lbfgs*, *sag* y *saga* porque no llegaban a converger.
- **Max_iter:** número máximo de iteraciones permitidas para converger.
- **C:** fuerza inversamente proporcional a la regulación.

De entre todas las combinaciones probadas, el modelo que mejor resultados ha obtenido (rendimiento) es:

LogisticRegression(C=1, max_iter=25, random_state=125, solver='newton-cg')

A la hora de seleccionar el porcentaje de los datos de entrada que deben ser destinados al entrenamiento y al test he podido comprobar que con la proporción 70%-30% el modelo no terminaba de tener suficiente accuracy.

Sin embargo, la precisión del set de entrenamiento era perfecta (1.0). Llegue a la conclusión de que seguramente se estaba produciendo over-fitting. Esto puede deberse a que tiene demasiados datos de entrenamiento o que estos no son muy variados. Como en este caso no disponemos de más datos para hacer el set más diverso, disminuí el número de datos de entrenamiento (65%) para que no pudiese ajustarse tan bien a estos. En consecuencia, aumenté el número de datos de prueba (35%) y los resultados en cuanto a precisión mejoraron.

Tras establecer los mejores parámetros para el modelo y el porcentaje de datos destinados a el entrenamiento y el test obtuve estos resultados:

Set de prueba = 35%, Set de entrenamiento = 65%.

	Set de entrenamiento	Set de prueba
Acuracy	**1.0**	**0.684**
Precision (weighted avg)	**1.0**	**0.8**
Recall (weighted avg)	**1.0**	**0.68**
F1-score (weighted avg)	**1.0**	**0.71**

Verdaderos positivos 10	**Falsos positivos 5**
Falsos negativos 1	**Verdaderos negativos 3**

Matriz de confusión del conjunto de datos de prueba:

- **Verdaderos positivos:** real (R), predicho (R).
- **Verdaderos negativos:** real (NR), predicho (NR).
- **Falsos positivos:** real (R), predicho (NR).
- **Falsos negativos:** real (NR), predicho (R).

Observación: Aunque la accuracy del set de prueba no es muy alta (0.684), vemos en la matriz de confusión que la mayoría de los "fallos" se producen en falsos positivos, es decir, pacientes que fueron clasificados como que tendrían una mala respuesta al tratamiento. Sin embargo, comparándolo con el set de prueba real, comprobamos que sí tendrían una buena respuesta al mismo. No hay apenas falsos negativos que son los que supondrían un riesgo para la salud del paciente.

2. Árbol de decisión

Un árbol de decisión es un modelo predictivo que divide el espacio de los predictores agrupando observaciones con valores similares para la variable respuesta o dependiente.

Para dividir el espacio muestral en subregiones es preciso aplicar una serie de reglas o decisiones, para que cada subregión contenga la mayor proporción posible de individuos de una de las poblaciones. Si una subregión contiene datos de diferentes clases, se subdivide en regiones más pequeñas hasta fragmentar el espacio en subregiones menores que integran datos de la misma clase.

Este tipo de modelos de clasificación son fáciles de entender y de interpretar. Además, requiere poca preparación de datos ya que es capaz de manejar tanto datos numéricos como categorizados. Por otra parte, es considerado uno de los modelos más robustos y funciona correctamente tanto para grades volúmenes de datos como para pequeños sets si los parámetros están bien ajustados.

En cuanto a los parámetros del modelo debemos destacar:

- **Min_samples_leaf**: número mínimo de muestras necesarias para llegar a un nodo hoja.
- **Min_samples_Split:** número mínimo de muestras necesarias para dividir un nodo interno.
- **Max_depth:** profundidad máxima del árbol.
- **Critterion:** función empleada para medir la calidad de una división.
- **Splitter:** estrategia utilizada para elegir la división en cada nodo.

De entre todas las combinaciones probadas, el modelo que mejor resultados ha obtenido (rendimiento) es:

DecisionTreeClassifier(criterion='entropy', max_depth=5, min_samples_leaf=5, random_state=125)

Una vez escogido el árbol y sus parámetros con el que obtenemos el mejor resultado, utilizaremos la regla de 70%-30% para los datos de entrenamiento y prueba respectivamente.

a. Conjunto de datos de prueba = 0.30

Verdaderos positivos 5	**Falsos positivos 7**
Falsos negativos 2	**Verdaderos negativos 2**

	Set de entrenamiento	Set de prueba
Accuracy	**0.83**	**0.43**
Precission (weighted avg)	**0.85**	**0.59**
Recall (weighted avg)	**0.84**	**0.44**
F1-score (weighted avg)	**0.84**	**0.47**

Como se puede observar, aunque la accuracy del set de entrenamiento es relativamente buena, la accuracy obtenida como resultado de la predicción está por debajo del 50%. Parce que el modelo no está aprendiendo correctamente. Aumentaremos el set de entrenamiento para que pueda utilizar más datos para aprender.

b. Conjunto de datos de prueba = 0.20

Verdaderos positivos 5	**Falsos positivos 3**
Falsos negativos 2	**Verdaderos negativos 1**

	Set de entrenamiento	Set de prueba
Accuracy	**0.95**	**0.54**
Precission (weighted avg)	**0.95**	**0.59**
Recall (weighted avg)	**0.95**	**0.55**
F1-score (weighted avg)	**0.95**	**0.56**

En este caso las métricas han mejorado considerablemente, el modelo ha tenido una mayor cantidad de datos de entrenamiento y por lo tanto ha podido ajustar mejor. Sin embargo, como hemos comentado

anteriormente, nuestro objetivo es tratar de eliminar los falsos negativos. Como aún no se ha producido over-fitting, probaremos a reentrenar el modelo con más cantidad de datos.

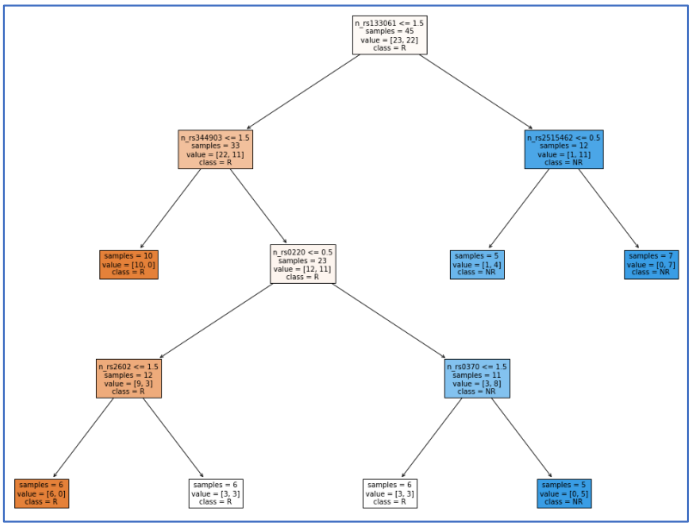
c. Conjunto de datos de prueba = 0.15

Verdaderos positivos 5	**Falsos positivos 2**
Falsos negativos 0	**Verdaderos negativos 1**

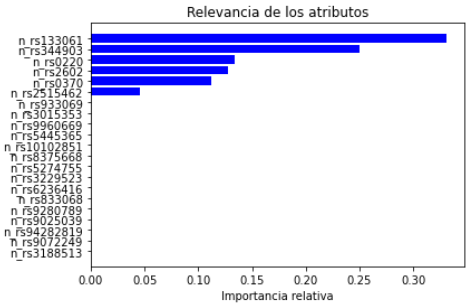
	Set de entrenamiento	Set de prueba
Accuracy	**0.84**	**0.75**
Precission (weighted avg)	**0.86**	**0.92**
Recall (weighted avg)	**0.84**	**0.75**
F1-score (weighted avg)	**0.84**	**0.79**

Una vez ajustado el modelo observamos que la accuracy en el set de entrenamiento ha disminuido, pero la del set de prueba ha mejorado considerablemente. Nos interesa más la accuracy del set de prueba porque nos muestra qué tal predice el modelo para datos que no conoce. Además, hemos conseguido eliminar por completo los falsos negativos.

A continuación, se muestra el esquema del árbol final:



En la imagen podemos ver que la profundidad del árbol es de 4 niveles y que tiene 7 nodos terminales o nodos hoja



Aunque el gráfico anterior es mucho más visual, también se puede conocer la disposición del árbol mediante este esquema que refleja la importancia relativa de cada una de las características.

La característica n_rs133061 es la más significativa y por lo tanto podemos comprobar que está situada en la cabeza del árbol.

Sin embargo, los atributos con relevancia 0 no estarán incluidos en el árbol y por lo tanto no son necesarios para realizar la predicción.

Observaciones: el modelo árbol de decisión es un algoritmo muy polivalente y aún con un pequeño set de datos de entrenamiento es capaz de clasificar más del 75% de los datos correctamente. Además, cabe destacar que gracias a este sencillo ejemplo vemos que es preferible que las accuracies del set de entrenamiento y de prueba estén proporcionadas

3. Bosque aleatorio (RF)

Bosque aleatorio o Random Forest es un método versátil de aprendizaje automático capaz de realizar tanto tareas de regresión como de clasificación. El algoritmo Random Forest ejecuta varios algoritmos de árbol de decisiones en lugar de uno solo. Para clasificar un nuevo objeto basado en atributos, cada árbol de decisión da una clasificación y finalmente la decisión con mayor "votos" es la predicción del algoritmo.

Este modelo a pesar de estar compuesto de varios árboles de decisión (considerados de caja blanca), es considerado un algoritmo de caja negra porque no es tan entendible ni interpretable. Se utiliza sobre todo con grandes cantidades de datos aunque puede ser adaptado para cualquier tipo de set.

En cuanto a los parámetros del modelo debemos destacar:

- **Min_samples_leaf**: número mínimo de muestras necesarias para llegar a un nodo hoja.
- **Min_samples_Split:** número mínimo de muestras necesarias para dividir un nodo interno.
- **Max_depth:** profundidad máxima del árbol.
- **Critterion:** función empleada para medir la calidad de una división.
- **Bootstrap:** estrategia utilizada para elegir la división en cada nodo.
- **N_stimators:** número de árboles que conformarán el bosque

De entre todas las combinaciones probadas, el modelo que mejor resultados ha obtenido (rendimiento) es:

RandomForestClassifier(max_depth=2, min_samples_leaf=2, n_estimators=10, random_state=125)

Una vez escogido el modelo y sus parámetros con el que obtenemos el mejor resultado, utilizaremos la regla de 70%-30% para los datos de entrenamiento y prueba respectivamente.

a. Conjunto de datos de prueba = 0.3

Verdaderos positivos 8	**Falsos positivos 4**
Falsos negativos 0	**Verdaderos negativos 4**

	Set de entrenamiento	Set de prueba
Acuracy	**1.0**	**0.75**
Precission (weighted avg)	**1.0**	**0.88**
Recall (weighted avg)	**1.0**	**0.75**
F1-score (weighted avg)	**1.0**	**0.77**

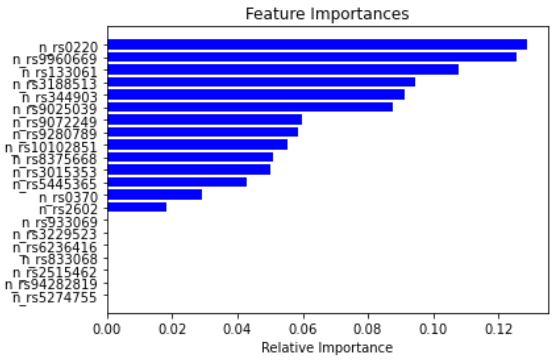
Como podemos observar en la matriz de confusión, con la proporción 70%-30% obtenemos muy buenos resultado en ambos sets de datos. Sin embargo, da la sensación de que el modelo podría haber sufrido over-fitting ya que la accuracy del set de entrenamiento es del 100% y sin embargo la del set de prueba no lo es. Con el fin de evitar este sobre ajuste, disminuiremos el tamaño del set de entrenamiento. Aún así debemos destacar que el objetivo a alcanzar (eliminar los falsos negativos) ha sido cumplido.

b. Conjunto de datos de prueba = 0.35

Verdaderos positivos 11	**Falsos positivos 4**
Falsos negativos 0	**Verdaderos negativos 4**

	Set de entrenamiento	Set de prueba
Acuracy	**0.94**	**0.79**
Precission (weighted avg)	**0.95**	**0.89**
Recall (weighted avg)	**0.94**	**0.79**
F1-score (weighted avg)	**0.94**	**0.81**

Una vez disminuido el tamaño de del conjunto de datos de entrenamiento y por lo tanto aumentado el de prueba, obtenemos una mejor accuracy en el set de prueba y seguimos manteniendo a 0 el número de falsos negativos. Además, ya no hay over-fitting porque hemos disminuido el accuracy en el conjunto de datos de entrenamiento.



A fin de saber qué atributos utiliza el Random Forest para la predicción he creado un gráfico donde se puede ver la importancia relativa de cada uno de los atributos iniciales.

La característica n_rs0220 es la más significativa y por lo tanto estará situada en la cabeza del árbol.

Sin embargo, los atributos con relevancia 0 no estarán incluidos en ningún árbol del bosque y por lo tanto no son necesarios para realizar la predicción.

4. K vecinos más próximos (KNN)

El algoritmo KNN es uno de los algoritmos de clasificación más simples, incluso con tal simplicidad puede dar resultados altamente competitivos. Pertenece al dominio de aprendizaje supervisado y puede ser utilizado para el reconocimiento de patrones, extracción de datos y detección de intrusos.

Es un clasificador robusto y versátil que a menudo se usa como un punto de referencia para clasificadores más complejos como las redes neuronales artificiales y vectores de soporte (SVM).

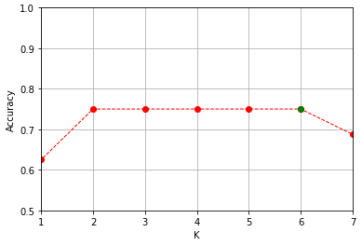
Entre las desventajas de este modelo hay que destacar que es muy sensible al ruido y tiene un procesamiento muy lento si hay muchos datos. Sin embargo, nuestro set de datos no tiene valores innecesarios según ha comprobado el ACP ni tenemos una gran cantidad de datos por lo que este algoritmo podría obtener muy buenos resultados.

Para este modelo, no he hecho uso del GridSearchCV() para elegir individualmente los mejores valores para el modelo (k y métrica). Muestro su análisis a continuación:

a. Elegir el valor K

Este algoritmo permite seleccionar un valor de K. Estos serán los K datos más cercanos al valor que se desea predecir como solución. Es importante seleccionar un valor de K acorde a los datos para tener una mayor precisión en la predicción.

Para ello, he buscado el modelo que mejor precisión tenía entre los 8 primeros k. El resultado fue 7.



b. Estudio de la mejor métrica

I. Modelo con métrica euclídea

Verdaderos positivos 8	**Falsos positivos 4**
Falsos negativos 0	**Verdaderos negativos 4**

	Set de entrenamiento	Set de prueba
Acuracy	**0.86**	**0.75**
Precission (weighted avg)	**0.87**	**0.88**
Recall (weighted avg)	**0.86**	**0.75**
F1-score (weighted avg)	**0.86**	**0.77**

II. Modelo con métrica Manhattan

Verdaderos positivos 10	**Falsos positivos 2**
Falsos negativos 1	**Verdaderos negativos 3**

	Set de entrenamiento	Set de prueba
Acuracy	**0.84**	**0.81**
Precission (weighted avg)	**0.84**	**0.83**
Recall (weighted avg)	**0.84**	**0.81**
F1-score (weighted avg)	**0.84**	**0.82**

Observaciones: Aunque ambos modelos han obtenido una tasa de acierto muy elevada, es algo mejor el modelo Manhattan. Sin embargo, también debemos de tener en cuenta que este modelo contiene un falso negativo. En este punto ambos modelos podrían considerarse equivalentes y dependiendo del caso en que se utilice y el objetivo que tengamos podemos elegir uno u otro.

5. Perceptrón multicapa

El perceptrón multicapa es una red neuronal artificial (RNA) formada por múltiples capas, de tal manera que tiene capacidad para resolver problemas que no son linealmente separables.

El Perceptrón Multicapa no extrapola bien, es decir, si la red se entrena mal o de manera insuficiente, las salidas pueden ser imprecisas. Además, la existencia de mínimos locales en la función de error dificulta considerablemente el entrenamiento, pues una vez alcanzado un mínimo el entrenamiento se detiene, aunque no se haya alcanzado la tasa de convergencia fijada.

En cuanto a los parámetros del modelo debemos destacar:

- **Max_iter**: número máximo de iteraciones permitidas para converger.
- **Activation:** función de activación de la capa oculta
- **Max_fun:** número máximo de llamadas a la función de pérdida.
- **Hidden_layer_sizes:** número de neuronas de la capa oculta.
- **Solver:** algoritmo utilizado para la optimización de pesos. En este parámetro tuve que descartar los algoritmos *sgd* y *adam* porque no llegaban a converger.

De entre todas las combinaciones probadas, el modelo que mejor resultados ha obtenido (rendimiento) es:

MLPClassifier(activation='logistic', hidden_layer_sizes=20, max_fun=1000, random_state=125, solver='lbfgs')

Una vez escogido el modelo y sus parámetros con el que obtenemos el mejor resultado, utilizaremos la regla de 70%-30% para los datos de entrenamiento y prueba respectivamente.

a. Conjunto de datos de prueba = 0.3

Verdaderos positivos 4	**Falsos positivos 8**
Falsos negativos 2	**Verdaderos negativos 2**

	Set de entrenamiento	Set de prueba
Acuracy	**1.0**	**0.37**
Precission (weighted avg)	**1.0**	**0.55**
Recall (weighted avg)	**1.0**	**0.38**
F1-score (weighted avg)	**1.0**	**0.40**

Como podemos observar en la tabla resumen, al aplicar la regla de 70%-30% obtenemos muy buen resultado en la predicción de los datos de entrenamiento, sin embargo, en el set de prueba los resultados son considerablemente bajos. Esto se debe a que el modelo no está aprendiendo correctamente. Además, se está produciendo over-fitting.

Al tratarse del modelo de perceptrón multicapa, no podemos disminuir el tamaño del set de entrenamiento para tratar de gestionar el sobreajuste ya que este tipo de modelos necesita de una gran cantidad de datos para aprender. Por lo tanto, nos centraremos en tratar de corregir la precisión del modelo aumentando ligeramente su set de entrenamiento

b. Conjunto de datos de prueba = 0.2

Verdaderos positivos 5	**Falsos positivos 3**
Falsos negativos 3	**Verdaderos negativos 1**

	Set de entrenamiento	Set de prueba
Acuracy	**1.0**	**0.45**
Precission (weighted avg)	**1.0**	**0.45**
Recall (weighted avg)	**1.0**	**0.45**
F1-score (weighted avg)	**1.0**	**0.45**

Aunque hemos aumentado su set de entrenamiento, la accuracy del set de prueba no ha mejorado. Además, han aumentado los casos de falsos negativos y esto es completamente inaceptable. En este momento tenemos dos líneas de actuación: disminuir aún más el set de prueba o modificar los parámetros del modelo.

c. Conjunto de datos de prueba = 0.1

Verdaderos positivos 4	**Falsos positivos 1**
Falsos negativos 0	**Verdaderos negativos 1**

	Set de entrenamiento	Set de prueba
Acuracy	**1.0**	**0.83**
Precission (weighted avg)	**1.0**	**0.92**
Recall (weighted avg)	**1.0**	**0.83**
F1-score (weighted avg)	**1.0**	**0.85**

En este caso ha mejorado considerablemente el accuracy del set de prueba. Además, hemos eliminado por completo los falsos negativos. Sin embargo, no deberíamos fiarnos de este resultado puesto que seguramente el modelo este sobre ajustado y al tener tan pocos datos en el set de prueba esta tasa de acierto podría ser fruto de la casualidad.

d. Conjunto de datos de prueba = 0.2

Verdaderos positivos 5	**Falsos positivos 2**
Falsos negativos 0	**Verdaderos negativos 1**

	Set de entrenamiento	Set de prueba
Acuracy	**0.97**	**0.75**
Precission (weighted avg)	**0.98**	**0.92**
Recall (weighted avg)	**0.98**	**0.83**
F1-score (weighted avg)	**0.98**	**0.85**

Uno de los parámetros más significativo de este modelo es el número de neuronas en la capa oculta. Si tenemos pocos datos y un gran número de neuronas, es altamente probable que o el algoritmo no pueda llegar a converger o que se produzca un sobre ajuste de determinados valores. Por este motivo he decido reducir el número de neuronas en la capa oculta a tan solo 5.

Los resultados obtenidos confirman que esta es la mejor opción. Hemos eliminado los falsos negativos y el over-fitting y hemos aumentado el porcentaje de acierto en el set de prueba.

Observaciones: Aunque es posible eliminar el over-fitting reduciendo el tamaño del set de entrenamiento, solo lo he conseguido dividiendo los datos a partes iguales entre el set de entrenamiento y de prueba (50%-50%). Sin embargo, el accuracy del set de prueba cae a un 0.43 por lo que el modelo no estaría bien ajustado.

Conclusiones.

Una vez desarrollado y ajustados en la medida de los posible todos los modelos, haré un pequeño análisis comparativo para elegir el mejor de ellos.

Aunque bien es cierto que no hay un modelo perfecto y que funcione en mejor medida para cualquier situación sobre todos los demás, si nos enfocamos en un escenario concreto sí que podemos hacer un ranking comprando los resultados obtenidos de cada uno. En este caso, yo haré un análisis únicamente sobre este caso concreto, **"mejor modelo para predecir la respuesta a un tratamiento contra en cáncer colorrectal".**

A continuación, se muestra una tabla resumen con las principales carteristas y resultados obtenidos de cada uno de los modelos estudiados:

	Accuracy	Promedio entrenamiento-prueba	Posible over-fitting	Número de falsos negativos
Regresión logística	0.68	35-65	Sí	0
Árbol de decisión	0.75	15-85	No	0
Bosque aleatorio	0.79	35-65	No	0
KNN (Euc-Man)	0.75 - 0.81	30-70	No	0-1
Perceptrón multicapa	0.75	20-80	No	0

Cada una de las características mostradas en la tabla son importantes. No es lo mismo comparar un modelo con mucho entrenamiento y poca prueba con uno que no ha necesitado tanto entrenamiento o uno donde se ha preferido aumentar la accuracy a costa del posible aumento de los falsos negativos.

Centrándonos en la mejor precisión, el modelo KNN con métrica Manhattan es el que mejor resultados ha tenido, además no se ha producido sobreajuste y tiene una proporción de sets normal. Sin embargo, produce 1 falso negativo. Al tratarse de un modelo destinado al diagnóstico de una enfermedad tan grave, en mi opinión, esto es inaceptable y por lo tanto el modelo KNN con métrica Manhattan quedaría descartado

El siguiente mejor modelo según su precisión es el Bosque aleatorio o Random Forest. En este modelo no hay síntomas de haberse producido sobreajuste y el número de falsos negativos es 0. Por último, la proporción entrenamiento-prueba es normal. Incluso ha necesitado menos datos para entrenar, lo que deja un set de datos más grande para realizar pruebas. Aún teniendo más datos para clasificar en el set de prueba, ha podido mantener su precisión alta (79%).

Los siguientes modelos con mejor precisión son el árbol de decisión, KNN con métrica euclídea y el perceptrón multicapa. Todos ellos han eliminado por completo los falsos negativos y no presentan síntomas de sufrir over-fitting. Aunque parezcan similares, debemos compara la columna restante, la proporción entrenamiento-prueba. En este sentido observamos que tanto el árbol de decisión como el perceptrón multicapa han necesitado más datos para poder ser entrenados. Esto también se traduce en que el tamaño del set de prueba es menos y podría no ser completamente fiable el resultado. Sin embargo, KNN Euclídea ha obtenido los mismos resultados, pero con menos datos de entrenamiento.

Finalmente, el modelo de **Regresión logística** no ha conseguido alcanzar el 70% de precisión. Además, vemos un claro sobreajuste en el modelo. Aún así ha conseguido eliminar todos los falsos negativos aún teniendo un set de prueba ligeramente mayor que el resto.

Para concluir, los dos mejores modelos son el Bosque Aleatorio y KNN. Teniendo en cuenta que el modelo tendrá repercusión sobre la salud de los pacientes, yo escogería **Bosque Aleatorio** porque elimina por completo los falsos negativos. Sin embargo, si el doctor no se fija únicamente en este modelo, sino que lo complementa con otras pruebas, elegiría KNN Manhattan porque tiene mayor accuracy.

	Accuracy	Promedio entrenamiento-prueba	Posible over-fitting	Número de falsos negativos
Regresión logística	0.68	35-65	Sí	0
Árbol de decisión	0.75	15-85	No	0
Bosque aleatorio	0.79	35-65	No	0
KNN	0.75 - 0.81	30-70	No	1-0
Perceptrón multicapa	0.75	20-80	No	0

Finalmente, me gustaría aclara que quizá con un mayor número de datos podremos sacar mayor partido a algoritmos como el perceptrón multicapa y la regresión logística que, aunque tienen un buen resultado, podría mejorarse.

Bibliografía

https://www.webcir.org/revistavirtual/articulos/diciembre13/uruguay/uru_espanol_a.pdf
https://runebook.dev/es/docs/scikit_learn/modules/cross_validation
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RepeatedKFold.html
https://es.wikipedia.org/wiki/An%C3%A1lisis_de_componentes_principales#:~:text=En%20estad%C3%ADstica%2C%20el%20an%C3%A1lisis%20de,%C2%ABcomponentes%C2%BB)%20no%20correlacionadas.&text=El%20ACP%20se%20emplea%20sobre,y%20para%20construir%20modelos%20predictivos.
https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
https://rpubs.com/Cristina_Gil/PCA
https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html
https://towardsdatascience.com/understanding-8-types-of-cross-validation-80c935a4976d
https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada
https://www.cienciadedatos.net/documentos/30_cross-validation_oneleaveout_bootstrap
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic%20regression#sklearn.linear_model.LogisticRegression
https://empresas.blogthinkbig.com/datos-entrenamiento-vs-datos-de-test/
https://www.cienciadedatos.net/documentos/py06_machine_learning_python_scikitlearn.html
https://stats.stackexchange.com/questions/27730/choice-of-k-in-k-fold-cross-validation
https://www.cienciadedatos.net/documentos/30_cross-validation_oneleaveout_bootstrap
https://moodle.upm.es/titulaciones/oficiales/pluginfile.php/10049178/mod_resource/content/1/Unit1-DataPreProcessing.pdf
https://moodle.upm.es/titulaciones/oficiales/pluginfile.php/10049177/mod_resource/content/2/Unit1-DataPreparation.pdf
https://moodle.upm.es/titulaciones/oficiales/pluginfile.php/10038276/mod_resource/content/2/Unit0-Tema1-Validation.pdf
https://aprendeia.com/aprendizaje-supervisado-logistic-regression/
https://www.maximaformacion.es/blog-dat/que-son-los-arboles-de-decision-y-para-que-sirven/