AIDS Clinical Trials Group Study 175

Práctica APA Curso 2023-2024 Q1

Departamento de Ciencias de la Computación
Grado en Ingeniería Informática - UPC

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Haobin Guo
Xin Lu

Indice

1. Introducción

Este proyecto tiene como objetivo final la implementación de un modelo de machine learning supervisado que es capaz de clasificar, si un paciente que se le ha diagnosticado SIDA fallecerá o no después de un periodo de tiempo de tratamiento.

Nos centraremos en un conjunto de datos obtenidos a partir de un estudio que contiene estadísticas de atención médica e información categórica sobre pacientes a los que se les ha diagnosticado SIDA en US. Este estudio se le llama estudio 175. Los datos son extraídos de UCI Machine Learning Repository.

El trabajo consistirá en un análisis previo y visualización de los datos, seguido de un preprocesado y adaptación de los datos, después, empezamos entrenamos diferentes modelos para obtener el mejor modelo, y finalizamos con una conclusión.

2. Datos

En el conjunto de datos original tenemos las siguientes variables.

pid_num: Identificador del paciente.
cid: Variable objetivo, variable binario que indica si el paciente ha fallecido en el tiempo de observación o no.
time: Duración del tratamiento, en días.
trt: Tratamiento aplicado al paciente. 0 = ZDV only; 1 = ZDV + ddI, 2 = ZDV + Zal, 3 = DDL only
age: Edad del paciente al inicio del estudio.
wtkg: Peso del paciente al inicio del estudio.
hemo: Variable binaria que indica si tiene Hemophilia. Hemophilia es un trastorno genético que dificulta la coagulación de la sangre.
homo: Variable binaria que indica si ha tenido actividad homosexual.
drugs: Variable binaria que indica si ha consumido drogas por vía intravenosas.
karnof: Puntuación Karnofsky. La puntuación Karnofsky es una medida médica que evalúa la capacidad funcional y el estado general de un paciente, utilizando una escala del 0 al 100, donde 100 representa una salud óptima y 0 indica fallecimiento.
oprior: Indica si anteriormente no tuvo tratamiento ZDV.
z30: Indica si en los 30 días previos al inicio del estudio 175 tuvo tratamiento ZDV
zprior: Indica si tuvo tratamiento ZDV antes del inicio del estudio 175.
preanti: Indica el número de días que estuvo sometido a un tratamiento antirretroviral antes del inicio del estudio 175.
race: Color de piel blanco o no.
gender: Género del paciente.
str2: Historial antirretroviral.(0=naive, 1=experienced)
strat: Estratificación del historial de tratamientos antirretrovirales:
    1='Antiretroviral Naive',
    2='Entre 1 y 52 semanas,
    3='> 52 weeks)
symptom: indicador sintomático. (0=asymp, 1=symp)

treat: Indicador de tratamientos. (0=ZDV only, 1=others)
offtrt: Indicador si un paciente estaba fuera del tratamiento antes de las 96 ± 5 semana
cd40: CD4* al inicio del tratamiento
cd420: CD4* a los 20±5 semanas
cd80: CD8* al inicio del tratamiento
cd820: CD8* a los 20±5 semanas

*CD4, tipo de célula que son importantes para evaluar la salud inmunológica en personas con VIH.
*CD8, también son célula crucial del sistema inmunológico y se monitorean en pacientes con VIH para evaluar la respuesta inmune y el progreso del tratamiento.

Vemos que muchas variables indican lo mismo, posteriormente estudiaremos la posibilidad de hacer un tratamiento a estas variables.

Para reducir la complejidad del problema, reduciremos la variable **time** en 13 periodos de tiempo en posteriores análisis.

3. Análisis de datos

Entrando en la exploración mínima de datos, para poder obtener una predicción más eficaz de la nuestra variable objetivo 'cid', modificamos la variable 'time' para el análisis y los entrenamientos posteriores, de manera que 'time' pasa a ser un rango de tiempo, siendo 'time' = 0 el rango de 0 a 99 días, 'time' = 1 el rango de 100 a 199 días y así sucesivamente hasta los 13 rangos.

Hay algunas variables que no aportan ninguna información útil al modelo, por lo que decidimos eliminarlas de nuestro conjunto de datos. Una de ellas es la variable 'pidnum', dado que es un identificador del paciente. Una otra es la variable 'treat', la cual lleva una información repetida que contiene la variable 'trt', también la variable 'zprior' que es la inversa de la variable 'oprior'.

Comprobando los valores perdidos, observamos que no hay ninguno en el nuestro conjunto.

Pasamos a la visualización de los valores de cada variable, en los histogramas de la figura 1 observamos que tenemos varias variables binarias y otras siguen una distribución gaussiana, lo que nos hace pensar en la posibilidad de usar modelos como regresión logística o Máquinas de soporte vectorial.

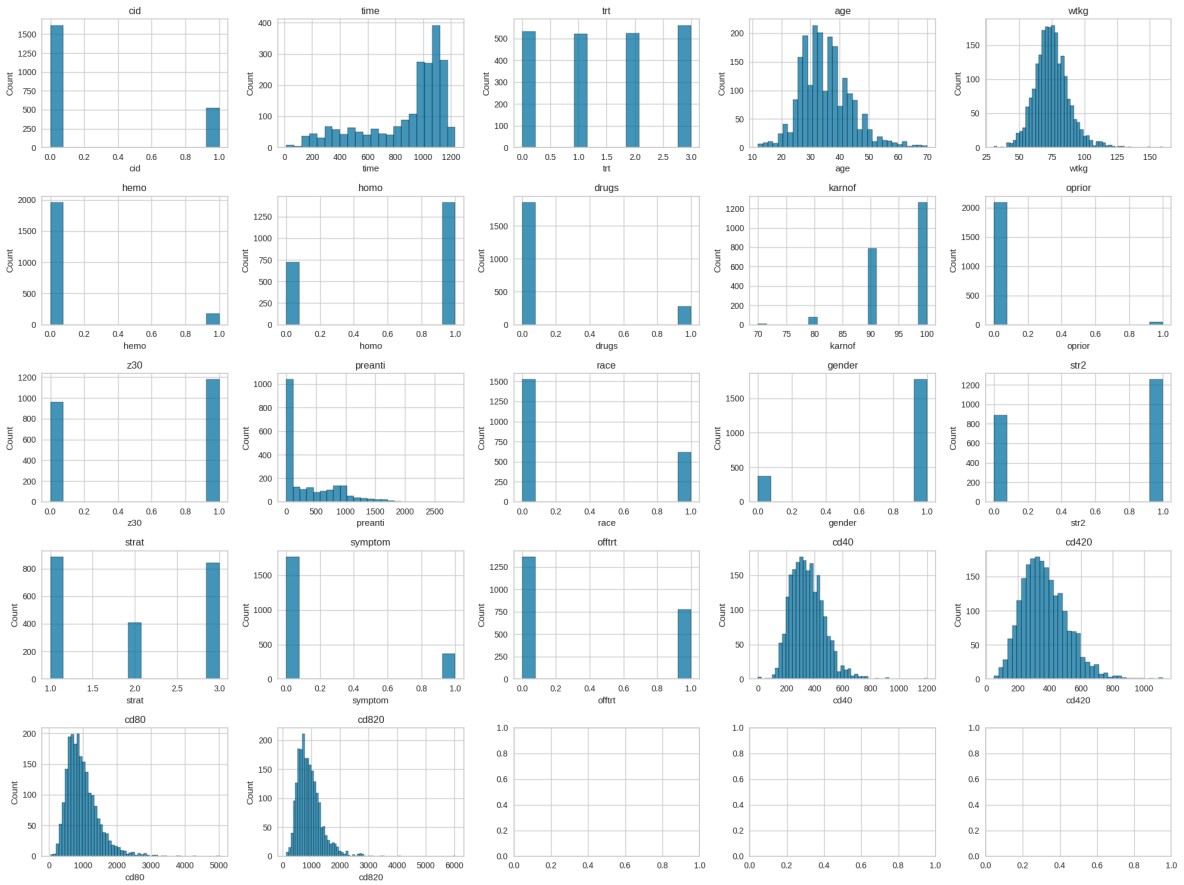

Figura 1: Conjunto de histogramas de las variables

En la siguiente gráfica (figura 2) tenemos la matriz de dispersión entre diferentes variables. Algo destacable de esta gráfica es ver las relaciones entre las variables ‘time’, ‘age’, ‘wtkg’, ‘preanti’, ‘cd40’, ‘cd420’, ‘cd80’ y ‘cd820’. En general no se apreció ninguna relación directa entre nuestra variable objetivo y las otras variables.

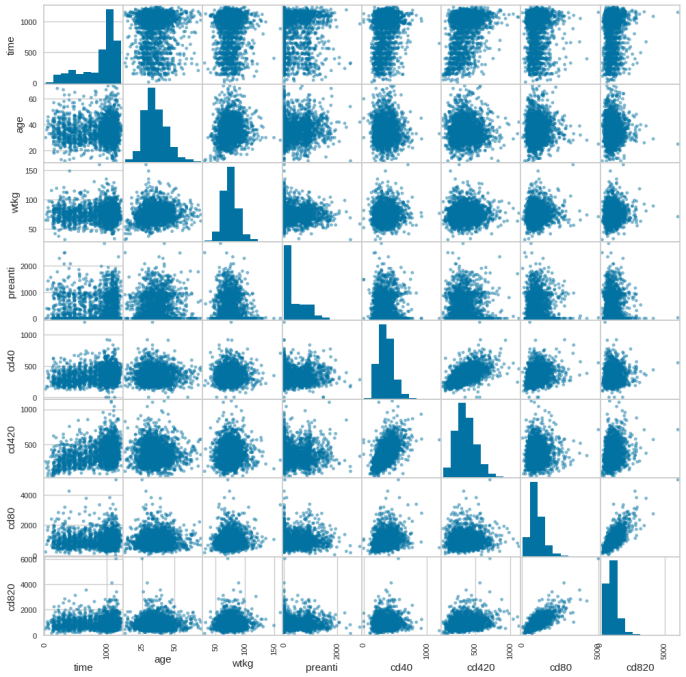

Figura 2: Matriz de dispersión.

En la matriz de correlación que se muestra a continuación (figura 3), vemos que la relación entre las variables en general no es muy alta.

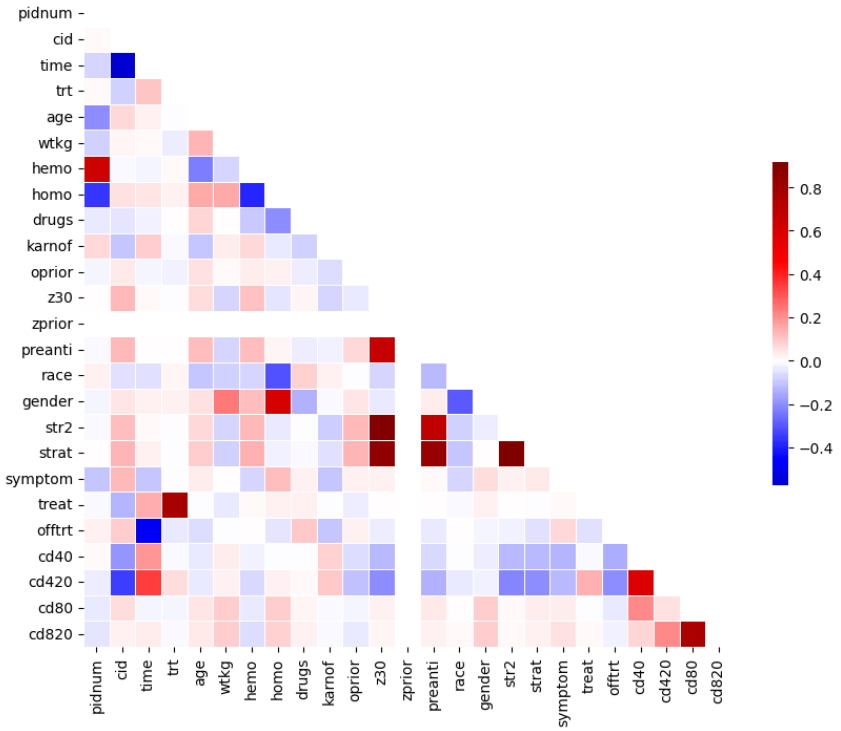

Figura 3: Matriz de correlación.

4. Preproceso de datos

Antes de nada, partimos los datos originales en 2 conjuntos, 80% en el conjunto de entrenamiento y 20% en el conjunto de test.

Hacemos one hot encoding sobre la variable ‘trt’, ‘strat’ y ‘karnof’, para que todos los variables sean binarias. Aplicamos los cambios a la variable ‘time’ mencionados anteriormente.

Decidimos no tratar los outliers porque pensamos que nos puede servir para entrenar nuestros modelos.

Generamos un conjunto de datos estandarizado usando la función StandardScaler, y otro conjunto usando el MinMaxScaler, estos modelos estandarizados se usarán en los correspondientes modelos que vamos a entrenar en posterior.

5. Visualización

PCA

A continuación hacemos un estudio del PCA, usando el conjunto de datos estandarizados con el StandardScaler.

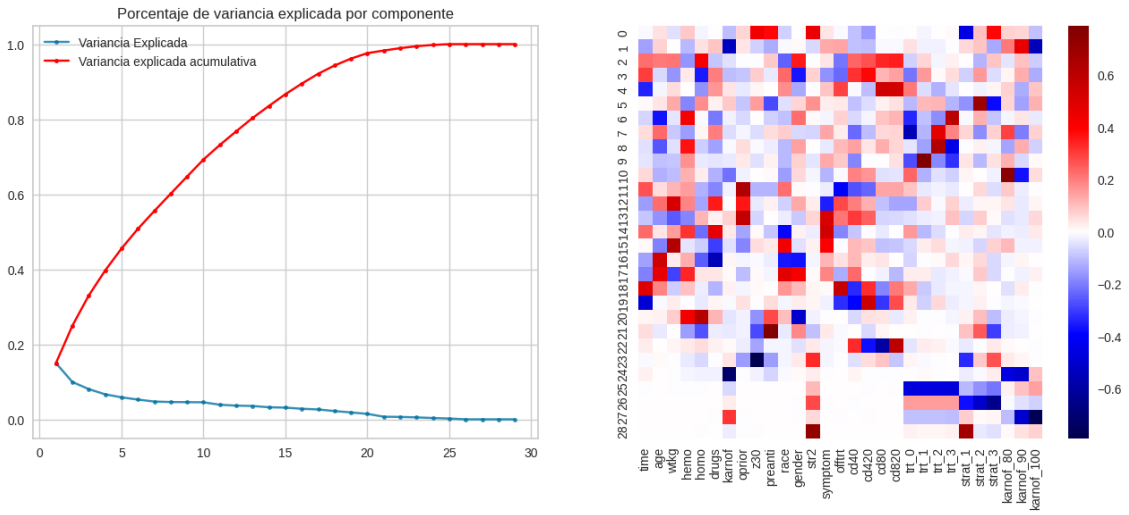

Figura 4: Porcentaje de varianza explicada y matriz de correlación del PCA

Se puede observar que la varianza explicada acumulativa llega a un 80% con 13 componentes, mientras que para llegar al 100% se necesita como mínimo unos 24 componentes, esto quiere decir que con unos 13 componentes se obtiene la mayoría de la información.

En la matriz de correlación muestra que hay algunas variables que están fuertemente correlacionadas. Esto podría afectar a algunos modelos de machine learning, especialmente aquellos que asumen que las características son independientes como la regresión logística.

Al representar los componentes en 2 dimensiones, se puede apreciar que se forman diferentes grupos, pero en estos grupos no se aprecia una separación de la clase de nuestro variable objetivo. Por esto creemos que los modelos no lineales podrían funcionar mejor.

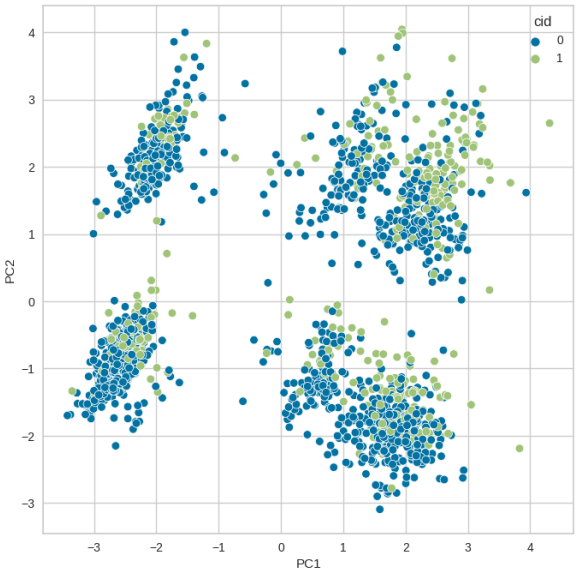

Figura 5: Representación 2D del PCA

6. Entrenamiento

A continuación, se entrenará el nuestro conjunto de datos con diferentes modelos lineales y no lineales, con el objetivo de encontrar el mejor modelo. Por ello se buscará los mejores hiperparametros correspondientes y usará cross validation para evaluar el modelo con 5 particiones.

Modelos lineales

Primero evaluamos el nuestro conjunto de datos con los modelos lineales. Los modelos que hemos elegido son: regresión logística, Naïve Bayes y k-nearest neighbours. Para estos modelos vamos a obtener y comparar sus validaciones cruzadas en conjunto de entrenamiento y de test, visualizamos las matrices de confusión y las correspondientes curvas de ROC. También realizaremos la búsqueda de mejores parámetros para cada modelo.

Regresión logística

Para empezar, vamos a obtener la validación cruzada del modelo de regresión logística. Con este modelo hemos obtenido un resultado bastante bueno, 0.863 en promedio de los puntajes de precisión. Esto significa que el modelo predice correctamente el 86,3% de las etiquetas en el conjunto de datos.

A continuación realizamos una búsqueda exhaustiva de los mejores parámetros para el modelo de regresión logística usando GridSearchCV. Vemos en el resultado con diferentes combinaciones de parámetros se obtiene una validación cruzada variada. Con los parámetros 'C' el valor de regularización a 3.98 y penalización L2, obtenemos un major valor de validación cruzada, de 0.866 aproximadamente, algo mejorado comparando con la validación cruzada que hemos obtenido anteriormente.

	params	mean_test_score	rank_test_score
25	{'C': 3.981071705534969, 'penalty': 'l2'}	0.865582	1
35	{'C': 125.89254117941663, 'penalty': 'l2'}	0.864998	2
33	{'C': 63.0957344480193, 'penalty': 'l2'}	0.864998	3
37	{'C': 251.18864315095772, 'penalty': 'l2'}	0.864418	4
39	{'C': 501.18723362727246, 'penalty': 'l2'}	0.864413	5

Figura 6: Tabla de validación cruzada de la regresión logística con diferente combinación de parámetros

Obteniendo resultados con el conjunto de test, podemos ver en la tabla siguiente donde el valor de acierto es muy similar al anterior, 0.85. Pero destaca que el número de las muestras de la clase 'failure' solo es una cuarta parte de las muestras de la clase 'censoring', esto puede provocar un desequilibrio de los datos y afectar a los resultados de la precisión. Vemos que la clase 'failure' tiene 0.60 de precisión y la clase 'censoring' ha obtenido un 0.94, valor mucho más alto, esto es debido probablemente al problema mencionado anteriormente.

	precision	recall	f1-score	support
censoring	0.94	0.87	0.90	341
failure	0.60	0.77	0.68	87
accuracy			0.85	428
macro avg	0.77	0.82	0.79	428
weighted avg	0.87	0.85	0.86	428

Figura 7: Reporte de clasificación de la regresión logística

Observando la matriz de confusión y las curvas de ROC, vemos que los verdaderos positivos muestra una mayor parte entre otros y las curvas de ROC para las dos clases tienen el mismo valor de AUC a 0.88, de manera que llegan a 1.0 de verdaderos positivos cuando la tasa de falsos positivos es alrededor de 0.7.

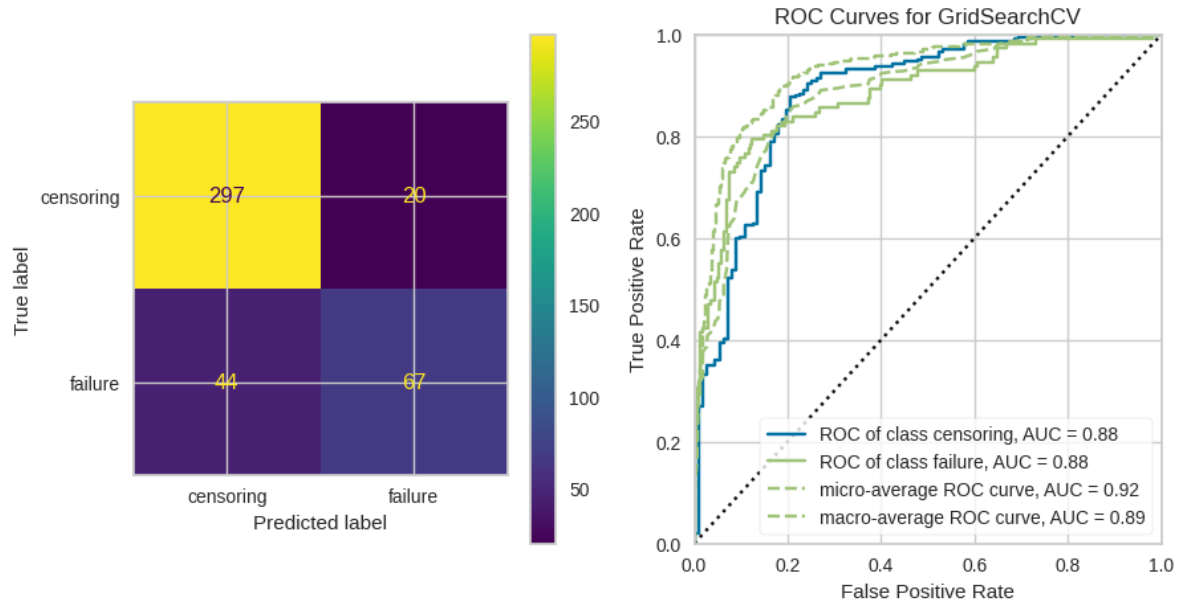

Figura 8: Matriz de confusión y las curvas ROC de la regresión logística

Naïve Bayes (Gaussiano)

En segundo lugar, evaluaremos el nuestro conjunto de datos con el modelo gaussiano (Naïve Bayes) y observaremos los resultados obtenidos.

Empezando con la validación cruzada. Vemos que el resultado no es del todo malo, tenemos un 0.80 aproximadamente.

Ajustando el parámetro ‘var_smoothing’, realizamos una búsqueda para encontrar el mejor parámetro del modelo gaussiano usando GridSearchCV. Obteniendo la tabla anterior de resultados, vemos que cuando el valor de ‘var_smoothing’ sea 1e-6 tenemos el mayor validación cruzada, un 0.832.

	params	mean_test_score	rank_test_score
3	{'var_smoothing': 1e-06}	0.832280	1
4	{'var_smoothing': 1e-05}	0.830525	2
2	{'var_smoothing': 1e-07}	0.810052	3
0	{'var_smoothing': 1e-09}	0.801864	4
5	{'var_smoothing': 0.0001}	0.801287	5

Figura 9: Tabla de validación cruzada de la Naïve Bayes con diferente combinación de parámetros

Obtenemos los resultados del conjunto de test y vemos que ha empeorado ligeramente comparado con el modelo de regresión logística, para este modelo tenemos un 0.83. Y en este caso el número de muestras de la clase 'failure' ha aumentado hasta ser un tercio de la otra clase, como consecuencia la correspondiente precisión también ha aumentado algo.

	precision	recall	f1-score	support
censoring	0.90	0.87	0.89	327
failure	0.63	0.69	0.66	101
accuracy			0.83	428
macro avg	0.77	0.78	0.77	428
weighted avg	0.84	0.83	0.83	428

Figura 10: Reporte de clasificación de la Naïve Bayes

Veamos la matriz de confusión y la gráfica de las curvas de ROC que genera este modelo. Los resultados muestran que las dos clases llegan a obtener la curva ideal cuando la tasa de falsos positivos es cercana a 0.8.

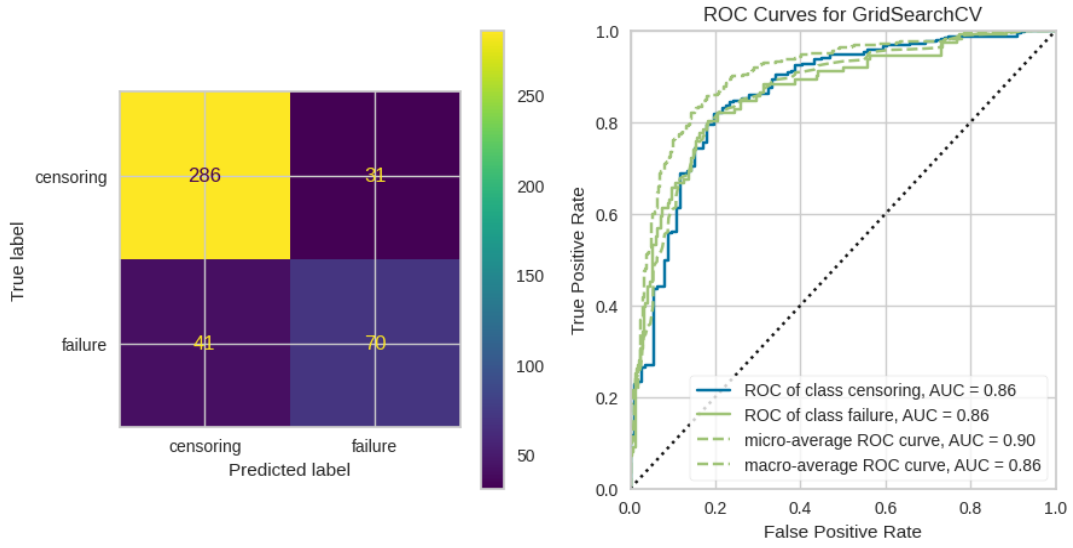

Figura 11: Matriz de confusión y las curvas ROC de la Naïve Bayes

K Nearest Neighbours (KNN)

El último modelo lineal que evaluaremos es el modelo KNN. Como los modelos anteriores, vamos a visualizar primero la validación cruzada. Vemos que la validación cruzada no ha mejorado respecto a los anteriores modelos, tenemos un 0.75 en este caso.

Para este modelo vamos a usar el conjunto de datos estandarizado con MinMaxScaler. Hacemos una exploración de hiperparametros, usando diferentes combinaciones de parámetros ‘leaf_size’, ‘metric’, ‘n_neighbours’ y ‘weights’. Los resultados mostrados para las 5 mejores combinaciones tienen la misma validación cruzada, que es 0.794.

	params	mean_test_score	rank_test_score
54	{'leaf_size': 5, 'metric': 'l1', 'n_neighbors': 7, 'weights': 'distance'}	0.793705	1
126	{'leaf_size': 20, 'metric': 'l1', 'n_neighbors': 7, 'weights': 'distance'}	0.793705	1
18	{'leaf_size': 1, 'metric': 'l1', 'n_neighbors': 7, 'weights': 'distance'}	0.793705	1
90	{'leaf_size': 10, 'metric': 'l1', 'n_neighbors': 7, 'weights': 'distance'}	0.793705	1
162	{'leaf_size': 30, 'metric': 'l1', 'n_neighbors': 7, 'weights': 'distance'}	0.793705	1

Figura 12: Tabla de validación cruzada de la KNN con diferente combinación de parámetros

Respecto al reporte de clasificación, vemos que las muestras de la clase ‘failure’ ha reducido a 46 y obtiene una precisión muy baja, de un 0.27. Por otro lado, tiene un acierto similar a la validación cruzada obtenida anteriormente, 77%.

	precision	recall	f1-score	support
censoring	0.95	0.79	0.86	382
failure	0.27	0.65	0.38	46
accuracy			0.77	428
macro avg	0.61	0.72	0.62	428
weighted avg	0.88	0.77	0.81	428

Figura 13: Reporte de clasificación de la KNN

Visualizando la matriz de confusión y la gráfica de las curvas de ROC de este modelo. A partir de los resultados obtenidos, observamos que AUC de las dos clases es 0.80 y no se obtienen el 100% de verdaderos positivos al final de la curva.

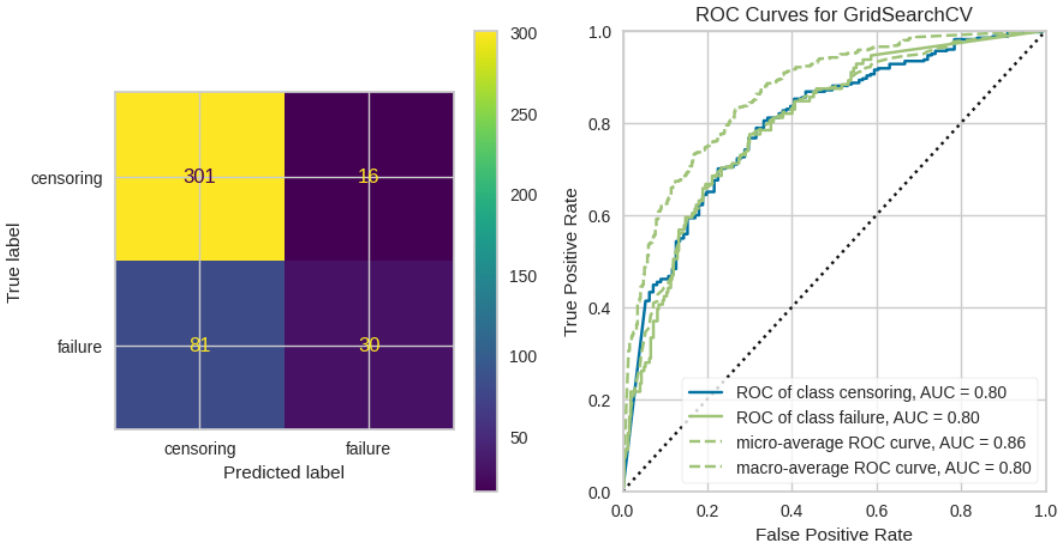

Figura 14: Matriz de confusión y las curvas ROC de la KNN

Modelos no lineales

Ahora vamos a probar con los modelos no lineales. Empezaremos por el modelo principal no lineal, MLP, luego usaremos el modelo SVM con kernel RBF por su eficacia en espacios con características no lineales y por último el modelo de Gradient Boosting por su precisión.

Perceptron multicapa (MLP)

Empezamos con el modelo perceptrón multicapa (MLP), en este modelo usamos los datos estandarizados con StandardScaler. Utilizamos la función BayesSearchCV para encontrar los mejores parámetros para el modelo sobre el conjunto de entrenamiento. En la siguiente tabla podemos ver como los mejores parámetros son con una función de activación Logistic, un alpha de 0.0001, utilizando una tamaño de 50 para la capa oculta, con un learning rate adaptive, con un learning rate init de 0.01, un momentum de 0.85 y el n_iter_no_change de 50, con estos parámetros el resultado medio es de 0.882, bastante altos la verdad, pero no para modelos con fines médicos.

A continuación, aplicamos el modelo con los mejores parámetros sobre el conjunto de test.

En los resultados obtenidos, vemos que la precisión de la clasificación sobre la clase ‘failure’ es bastante baja, un 0.7 de los 93 del support, mientras que la precisión de la clase ‘censoring’ es más alta un 0.95. En las curvas de ROC vemos que con un 0.6 de false positive ya se consigue el 100% de los true positive, algo bastante positivo.

	precision	recall	f1-score	support
censoring	0.95	0.90	0.93	335
failure	0.70	0.84	0.76	93
accuracy			0.89	428
macro avg	0.83	0.87	0.85	428
weighted avg	0.90	0.89	0.89	428

Figura 15: Reporte de clasificación MLP

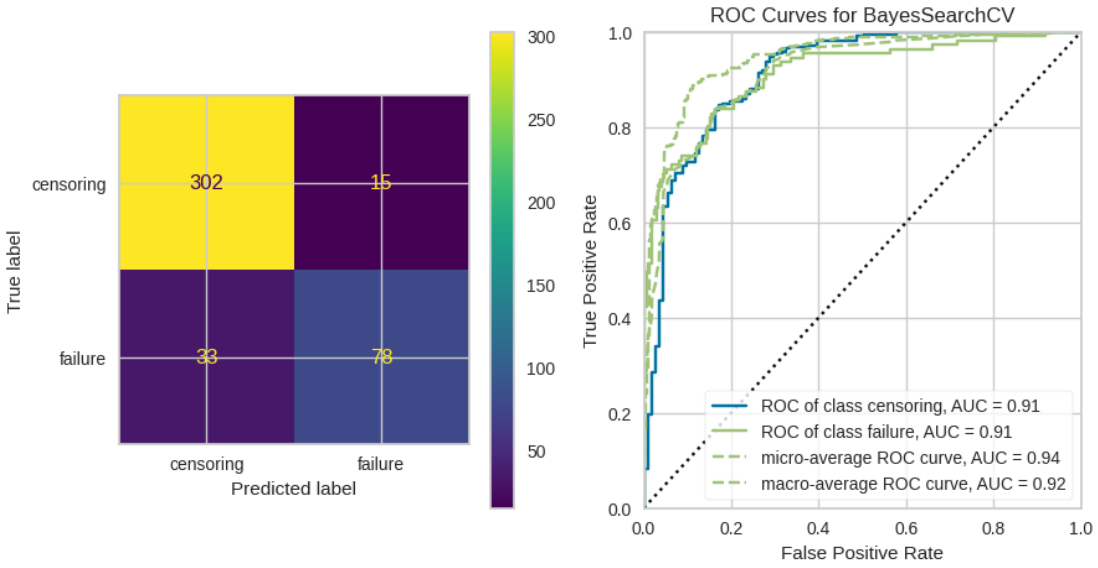

Figura 16: Matriz de confusión y las curvas ROC de MLP

En general este modelo es mejor que los modelos lineales anteriores.

Máquinas de soporte vectorial (SVM)

Seguimos con el modelo de máquinas de soporte vectorial con el kernel RBF, en este caso usando el conjunto de datos estandarizado con el MinMaxScaler. En la búsqueda de los mejores parámetros, encontramos con un valor de regularización 'C' de 125.89 y una gamma en 'auto', que da un resultado medio de 0.8784, algo peor que en el modelo anterior.

Aplicamos el modelo y los mejores parámetros sobre el conjunto de test. Los resultados generales obtenidos son ligeramente peores que el modelo anterior, aunque la precisión sobre la clase 'failure' es un poquito más baja.

	precision	recall	f1-score	support
censoring	0.95	0.89	0.92	337
failure	0.68	0.82	0.74	91
accuracy			0.88	428
macro avg	0.81	0.86	0.83	428
weighted avg	0.89	0.88	0.88	428

Figura 23: Reporte de clasificación SVM

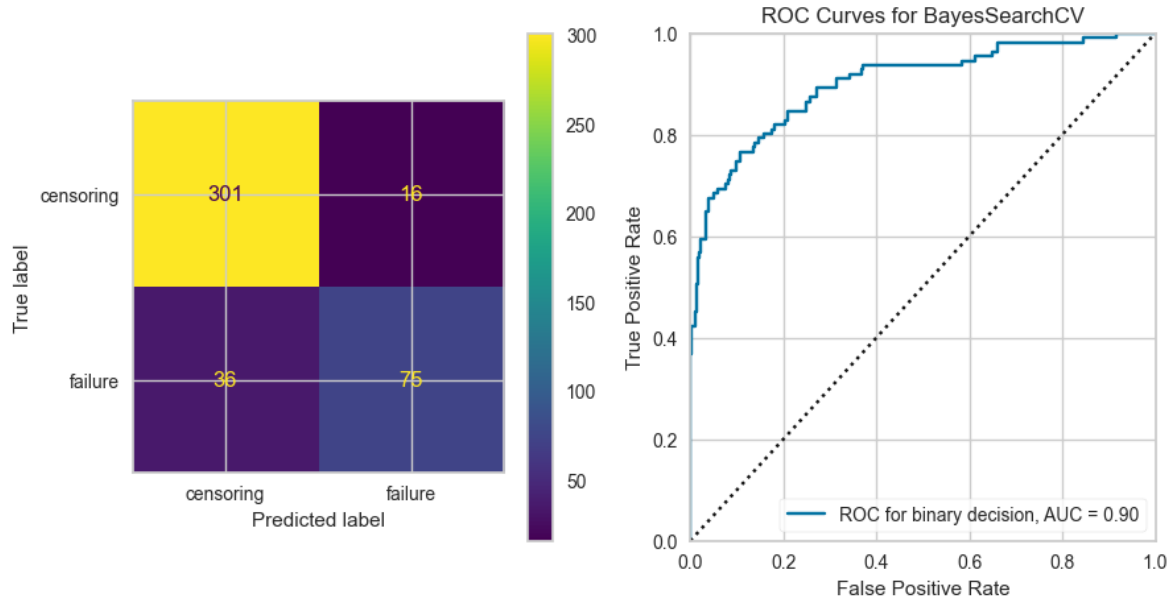

Figura 17: Matriz de confusión y las curvas ROC de SVM

Gradient boosting

Finalmente usamos el modelo de gradient boosting, con datos estandarizado con el StandardScaler, como anteriormente se ha hecho, buscamos los mejores parámetros para este modelo.

Los mejores parámetros encontrado es usando el criterio de squared_error, un learning_rate de 0.5, usando log_loss como función de pérdida, con profundidad máxima de 2 , con un

min_sample_leaf de 5 y un n_estimators de 25, el resultado medio obtenido es de 0.890708, un resultado similar al MLP.

Aplicamos este modelo y los parámetros sobre el conjunto de test. Vemos que el resultado general es ligeramente mejor que el MLP, en la curva de ROC vemos que esta vez la clase 'failure' tiene casi un 100% de False positive para llegar al 100% de los true positive, algo que no pasaba en el MLP.

	precision	recall	f1-score	support
censoring	0.96	0.90	0.93	337
failure	0.70	0.86	0.77	91
accuracy			0.89	428
macro avg	0.83	0.88	0.85	428
weighted avg	0.90	0.89	0.90	428

Figura 18: Reporte de clasificación Gradient boosting

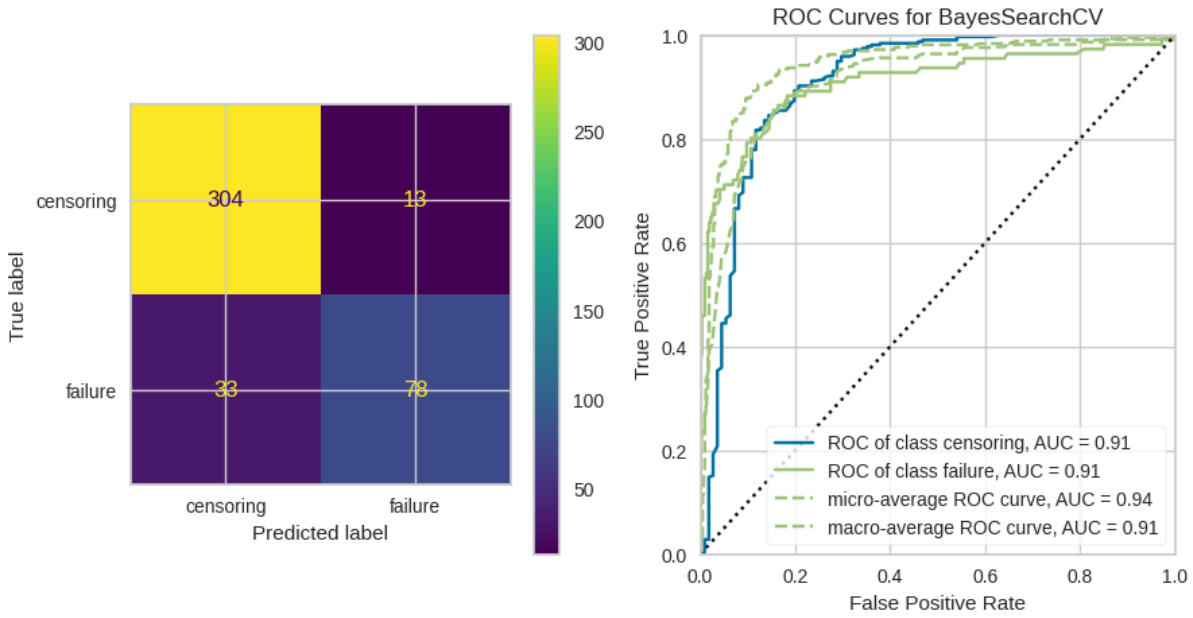

Figura 19: Matriz de confusión y las curvas ROC de Gradient boosting

7. Conclusiones

	train acc	test acc	precision score (W)	recall score (W)	f1 score (W)
Logistic	0.865582	0.850467	0.844812	0.850467	0.844132
GNB	0.832280	0.831776	0.827534	0.831776	0.829114
KNN	0.793705	0.773364	0.752743	0.773364	0.736987
MLP	0.882536	0.887850	0.885210	0.887850	0.884451
SVM	0.878434	0.878505	0.875280	0.878505	0.874348
Gradient boosting	0.890708	0.892523	0.890424	0.892523	0.888846

Figura 20: Resumen de modelos

Después de probar varios modelos, vemos que los mejores modelos para este conjunto de datos son los no lineales. Encabezado por el modelo de Gradient boosting con casi un 90% de precisión. Los otros 2 modelos no lineales le siguen de cerca el MLP con un f1 score de 0.88 y el SVM con un f1 score 0.87. Los modelos lineales en general son un poco peores que los no lineales, ya que como se puede ver en el PCA, las clases eran difíciles de separar.

En conclusión, si tuviera que elegir un modelo entre estos 6, elegimos el Gradient boosting, simplemente porque es el que rinde mejor, con un f1 score de 0.89, algo por encima del segundo, MLP con un f1 score de 0.88.

Pero esto no quiere decir que es un buen modelo para nuestro problema, ya que en casi todos los modelos la precisión para la clase ‘failure’, el que más nos interesa, es muy baja, el más alto es un 0.7 en el Gradient boosting. A parte de esto, el desequilibrio de la clase minoritario ‘failure’ de algunos modelos se ve afectado a la precisión y el rendimiento.

8. Extra

El proyecto no se podría decir que ha sido un fracaso total, pero esperábamos mejores resultados. En un principio esperábamos que los resultados fueran más de un 90%, pero no se ha conseguido con los modelos elegidos.

Hay que decir que también que no teníamos muchos datos, y estos datos tampoco aportan mucha información. El estudio 175 era de hace 2 décadas, ahora con los métodos y tecnologías disponibles creemos que se puede obtener muchos y mejores datos, para hacer un estudio sobre este tema.

El proyecto se ha hecho por 2 personas, empezando con el análisis de datos juntos y luego dividiéndose los modelos a entrenar, cuando se tuvo los modelos entrenados, se puso en común los resultados y se sacaron las conclusiones.

Todo el código, las tablas y gráficos están disponibles en el archivo .ipynb adjunto. Podéis seguir las instrucciones del README para ejecutar el código. (Los resultados pueden cambiar ligeramente por cada ejecución).

9. Bibliografía

S. Hammer, D. Katzenstein, M. Hughes, H. Gundacker, R. Schooley, R. Haubrich, W. K. Henry, M. Lederman, J. Phair, M. Niu, M. Hirsch, T. Merigan. 1996. AIDS Clinical Trials Group Study 175. https://archive.ics.uci.edu/dataset/890/aids+clinical+trials+group+study+175

Profesores de APA. Facultad de Informática de Barcelona. 2023. Material laboratorio. https://sites.google.com/upc.edu/aprenentatge-automatic/p%C3%A1gina-principal/laboratorio?authuser=1