Barrientos, Dinger, Falaq

Ciencia de Datos
Tercer Trabajo Práctico

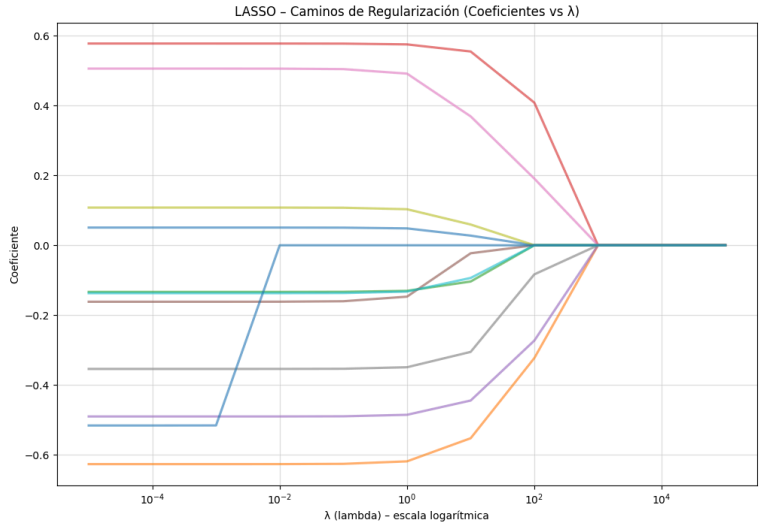
Link al GitHub:

A. Modelo de Regresión Logística con Regularización: Ridge y LASSO

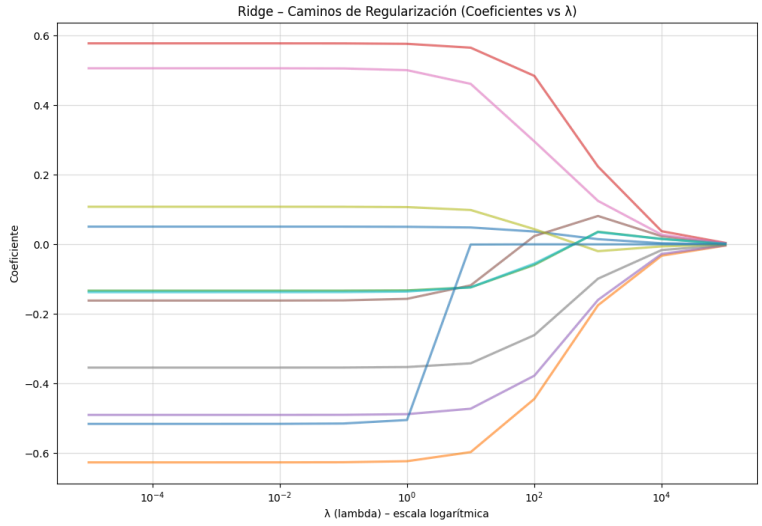
1)

Para este ejercicio construimos los caminos de regularización para LASSO y Ridge usando únicamente la base *respondieron_2025*. Partimos de la matriz de predictores que armamos previamente y aplicamos un preprocesamiento básico (imputación y estandarización) hecho solo sobre la muestra de entrenamiento. Con esa base ya preparada, definimos una grilla de valores de penalización siguiendo lo indicado en la consigna: usamos $\lambda = 10^n$ para valores de n entre −5 y 5, lo que nos permitió cubrir desde penalizaciones muy débiles hasta penalizaciones muy fuertes.

Con esta grilla ajustamos dos modelos de regresión logística: uno con penalidad L1 (LASSO) y otro con penalidad L2 (Ridge). Para cada valor de λ guardamos los coeficientes estimados, de manera de poder observar cómo cambia cada predictor cuando aumenta la fuerza de la regularización. Después convertimos esa información en tablas y armamos los gráficos de "caminos de regularización", uno para LASSO y otro para Ridge.Los resultados muestran comportamientos distintos.



En el gráfico de LASSO, a medida que λ crece, varios coeficientes se van achicando hasta quedar exactamente en cero. Esto refleja que LASSO tiende a seleccionar solo algunas variables y descartar otras cuando la penalización es fuerte.



En cambio, en el caso de Ridge todos los coeficientes se reducen progresivamente, pero ninguno llega exactamente a cero, lo que indica que Ridge suaviza los parámetros sin eliminar variables del modelo. Ambos gráficos se representan en escala logarítmica para que los cambios sean más visibles.

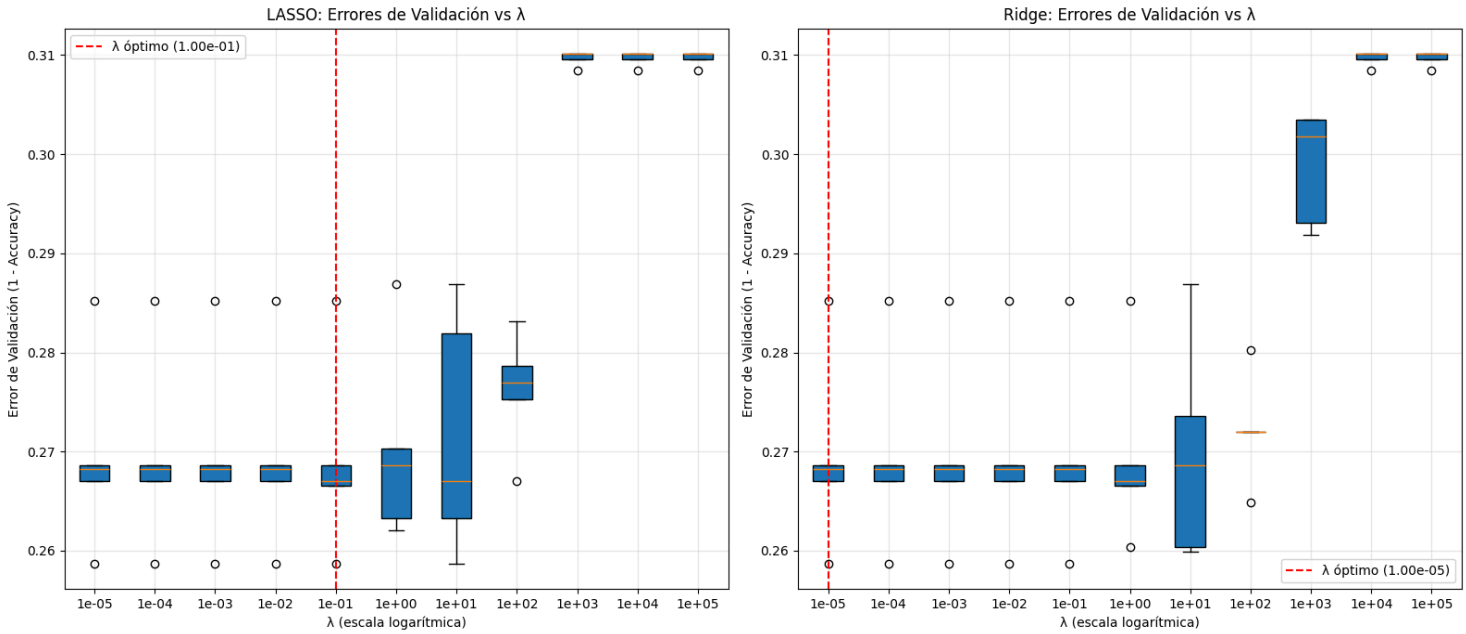
Los comportamientos son distintos, LASSO simplifica el modelo reduciendo el número de predictores activos, mientras que Ridge mantiene todas las variables pero controla su magnitud.

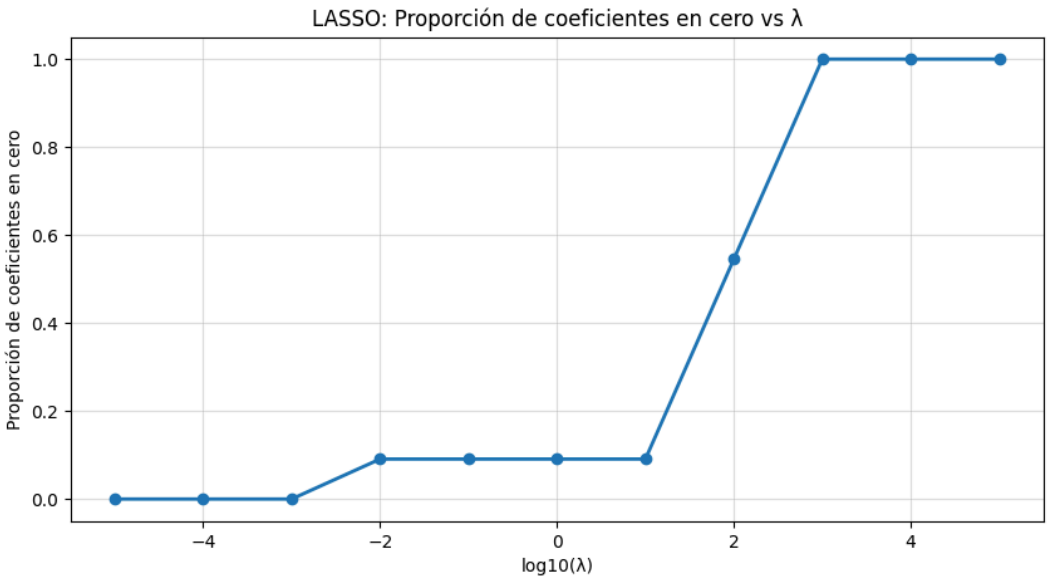
2)

Para elegir la penalidad óptima de LASSO y Ridge usamos *LogisticRegressionCV* con validación cruzada de 5 particiones, aplicando el mismo preprocesamiento del ejercicio anterior y la misma grilla de valores de $\lambda = 10^n$ para valores de n entre −5 y 5. Esto nos permitió evaluar, para cada valor de λ, el error de clasificación promedio en los distintos folds y seleccionar automáticamente el valor que mejor desempeño obtuvo en validación.

En los boxplots de validación puede verse claramente cómo varía el error de predicción (1 − accuracy) para cada valor de λ en ambos modelos. En el caso de **LASSO**, el error se mantiene bastante estable para penalizaciones bajas, pero empieza a aumentar cuando λ crece demasiado. El valor óptimo seleccionado por cross-validation fue $\lambda = 10^{-1}$, que coincide visualmente con una zona donde los errores son bajos y relativamente consistentes entre folds. Cuando la penalización es muy grande (por ejemplo, 10^3 en adelante), el desempeño del modelo empeora y la dispersión de los errores entre particiones aumenta, lo cual sugiere que el modelo se vuelve demasiado restrictivo.

En cambio, para **Ridge** el comportamiento es distinto. El error de validación se mantiene prácticamente plano y bajo para casi todos los valores de λ pequeños e intermedios, y recién empieza a aumentar cuando la penalización es extremadamente alta. Aun así, el valor óptimo elegido por cross-validation fue $\lambda = 10^5$, lo cual refleja que Ridge tolera mucho mejor penalizaciones fuertes sin desestabilizar el modelo. Incluso en ese punto, la variabilidad del error entre folds sigue siendo relativamente baja comparada con LASSO.





El gráfico de proporción de coeficientes en cero ayuda a entender mejor el comportamiento de LASSO. Para valores chicos de λ, prácticamente ningún coeficiente es eliminado; la mayoría se mantiene activa en el modelo. Pero cuando la penalización aumenta (entre 10^2 y 10^3), la proporción de coeficientes en cero crece rápidamente hasta llegar a 1. Esto significa que, bajo penalizaciones muy fuertes, LASSO termina dejando un modelo totalmente vacío, sin predictores. Este patrón coincide con lo que se observa en los boxplots: si LASSO elimina demasiadas variables, la capacidad predictiva empeora.

Los gráficos muestran que ambos modelos responden de manera distinta a la regularización. LASSO encuentra su mejor rendimiento con una penalización moderada, antes de empezar a anular demasiadas variables, mientras que Ridge se mantiene estable incluso con penalizaciones altas y recién muestra deterioro en valores extremos. Esto refuerza la diferencia conceptual entre ambos métodos: LASSO tiende a seleccionar variables y simplificar el modelo, mientras que Ridge prioriza la estabilidad y distribuye la penalización de manera más suave.

3)

Para este ejercicio estimamos tres modelos de regresión logística usando X_train de *respondieron_2025*: un modelo sin penalidad, un modelo con penalidad L1 (LASSO) y otro con penalidad L2 (Ridge), utilizando en los dos últimos los valores óptimos de λ seleccionados mediante cross-validation en el ejercicio anterior. Todos los modelos se ajustaron con el mismo preprocesamiento para asegurar comparabilidad y luego extrajimos los coeficientes finales para cada predictor.

Al comparar los resultados, se observa que la estructura general de los coeficientes no cambia drásticamente entre los tres modelos: las variables que eran relevantes en el modelo sin penalidad mantienen señales y magnitudes similares tanto en LASSO como en Ridge. Esto es especialmente claro en variables como *CH06*, *NIVEL_ED y ESTADO*, que conservan

coeficientes negativos y de magnitud parecida en los tres casos. Lo mismo ocurre con *CH08* y *CAT_INAC*, que mantienen coeficientes positivos en todos los modelos.

Sin embargo, LASSO introduce algunas diferencias. En particular, tiende a reducir aún más los coeficientes pequeños, y en algunos casos los aproxima mucho a cero (por ejemplo, MAS_500_S pasa de -0.51 en el modelo sin penalidad a -0.17 en LASSO). Esto refleja el comportamiento típico de la penalización L1, que empuja a los coeficientes hacia cero y puede dejar fuera a variables con menor poder explicativo. Aun así, en nuestro caso ningún coeficiente llegó exactamente a cero, lo que significa que LASSO no eliminó por completo ninguna variable de la matriz X_train, aunque sí achicó varias de forma importante.

Por otro lado, la penalidad L2 (Ridge) mantuvo prácticamente todos los coeficientes con magnitudes muy similares al modelo sin penalidad. Esto es consistente con el funcionamiento de Ridge, que suaviza los coeficientes pero no los lleva a cero. En nuestro resultado concreto, Ridge no elimina variables y conserva la estructura del modelo base casi sin cambios, solo reduciendo levemente algunos coeficientes (por ejemplo, PP07A o CH03).

La comparación muestra que la regularización no alteró la dirección ni el significado económico de los coeficientes, pero sí afectó su magnitud. LASSO es el que introduce mayor ajuste, reduciendo coeficientes pequeños y acercando algunos a cero, mientras que Ridge preserva más la forma original del modelo.

Tabla de coeficientes (sin redondear):

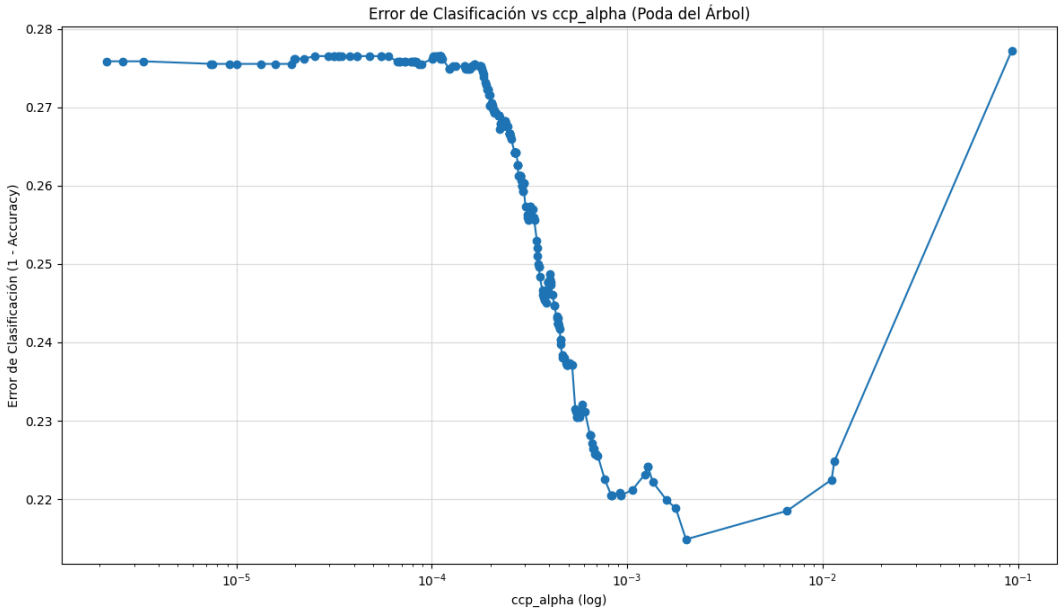
	coef_sin_penalidad	coef_L1	coef_L2
CH04	0.050744	0.050402	0.050744
CH06	-0.626343	-0.625427	-0.626343
CH07	-0.133167	-0.133170	-0.133167
CH08	0.577198	0.576972	0.577198
NIVEL_ED	-0.489998	-0.489393	-0.489998
ESTADO	-0.161606	-0.159833	-0.161606
CAT_INAC	0.505621	0.503684	0.505621
PP07A	-0.354103	-0.353405	-0.354103
PP07C	0.107928	0.107362	0.107928
CH03	-0.137475	-0.136662	-0.137475
MAS_500_S	-0.515759	-0.179472	-0.515759

	coef_sin_penalidad	coef_L1	coef_L2
CH04	0.0507	0.0504	0.0507
CH06	-0.6263	-0.6254	-0.6263
CH07	-0.1332	-0.1332	-0.1332
CH08	0.5772	0.5770	0.5772
NIVEL_ED	-0.4900	-0.4894	-0.4900
ESTADO	-0.1616	-0.1598	-0.1616
CAT_INAC	0.5056	0.5037	0.5056
PP07A	-0.3541	-0.3534	-0.3541
PP07C	0.1079	0.1074	0.1079
CH03	-0.1375	-0.1367	-0.1375
MAS_500_S	-0.5158	-0.1795	-0.5158

B. Árboles

4)

Estimamos un árbol de decisión (CART) y aplicamos poda mediante el hiperparámetro de costo de complejidad, ccp_alpha. Para elegir el valor óptimo utilizamos validación cruzada 10-fold y evaluamos, para cada valor de la grilla, el error de clasificación promedio. Luego graficamos ese error en función de ccp_alpha para visualizar cómo cambia el desempeño del modelo a medida que se incrementa la penalización.



En la parte izquierda, cuando ccp_alpha es extremadamente chico, el árbol prácticamente no se poda y mantiene una estructura muy grande. En esta zona el error de validación es relativamente alto y bastante plano (alrededor de 0.27), lo que indica sobreajuste: el árbol memoriza demasiada estructura del set de entrenamiento y no generaliza bien.

A medida que ccp_alpha aumenta ligeramente (entre 10^{-4} y 10^{-3}) empieza a aparecer una caída muy pronunciada del error. En ese tramo la poda elimina ramas y el modelo mejora notablemente su capacidad de generalización. Este descenso abrupto es el corazón del trade-off entre complejidad y desempeño: el árbol deja de ser tan específico y empieza a capturar únicamente los patrones más estables.

El error llega a su punto mínimo alrededor de valores cercanos a ccp_alpha ≈ 2×10^{-3}, donde observamos el mejor rendimiento del modelo (error cercano a 0.215–0.22). Este es el nivel de poda que logra el mejor equilibrio entre simplicidad y precisión.

Finalmente, cuando ccp_alpha sigue aumentando (entre 10^{-2} y 10^{-1}), el error vuelve a subir con fuerza. Esto refleja subajuste: el árbol queda tan podado que pierde estructura, se vuelve demasiado simple y ya no puede capturar las relaciones entre las variables. El salto final en el error lo confirma.

Entonces, el análisis del gráfico muestra que el árbol sin poda inicial tiene un desempeño limitado por sobreajuste, pero que una poda moderada mejora significativamente la calidad del modelo. El valor óptimo de ccp_alpha se encuentra en la zona intermedia de la grilla.

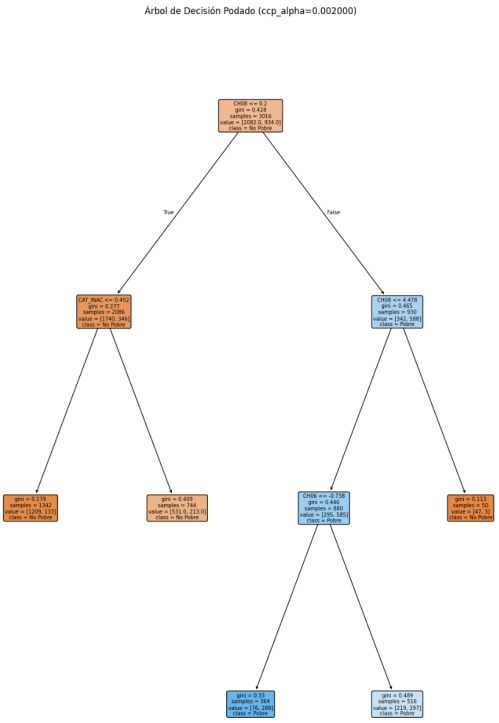
5)

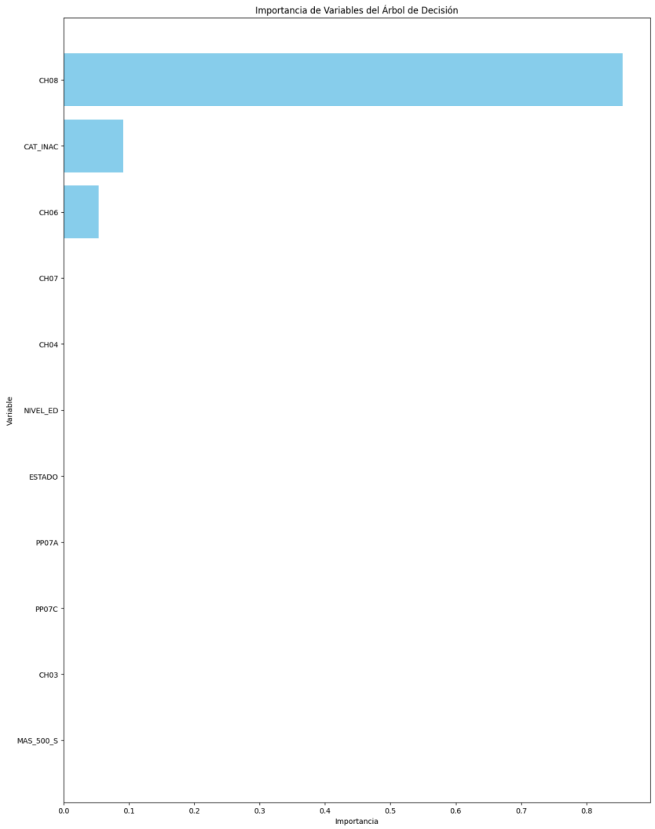
En el panel A se muestra el árbol de decisión final, entrenado con el valor óptimo de ccp_alpha obtenido en el ejercicio anterior (alrededor de 0.002). La poda redujo significativamente la profundidad del árbol, dejando únicamente las divisiones que aportan para la predicción. El árbol resultante es relativamente simple, lo cual coincide con la idea de que una poda moderada mejora la generalización.

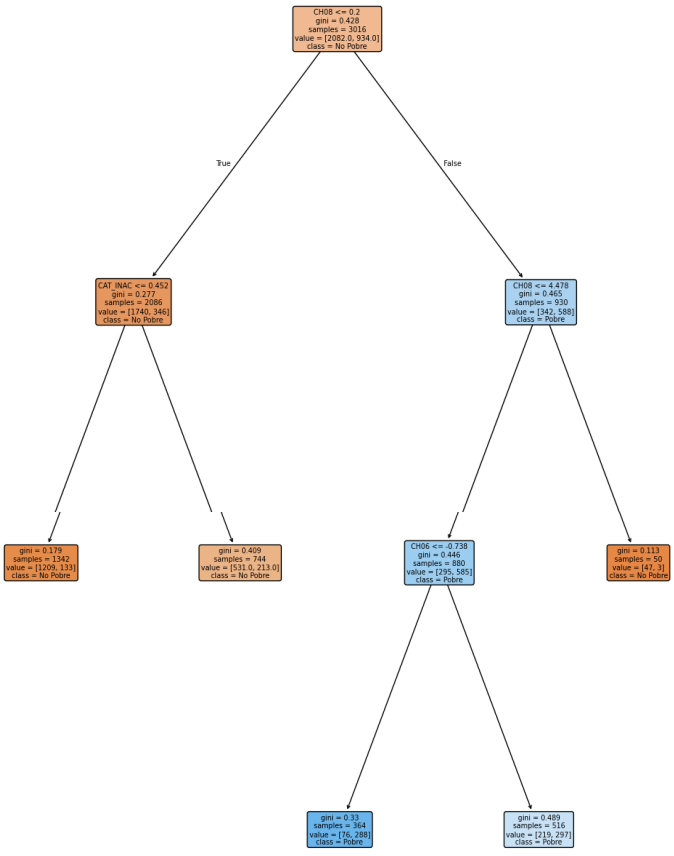
A partir de la estructura del árbol se observa que CH08 es la variable predominante: aparece como la división inicial (root node), lo que indica que es el predictor que más reduce la impureza en la clasificación. Luego, en ramas secundarias, también aparecen CAT_INAC y CH06, lo cual sugiere que estas variables son relevantes pero en menor medida. El resto de las variables no llegan a generar divisiones adicionales, señal de que su aporte marginal es mucho más bajo.

En el panel B se presenta el gráfico de importancia de variables del árbol podado. La evidencia visual es contundente: CH08 concentra más del 80% de la importancia total, mientras que CAT_INAC y CH06 aportan una fracción bastante menor. El resto de los predictores prácticamente no contribuye al modelo. Esta distribución tan concentrada en pocas variables coincide con la estructura del árbol observado en el panel A, que utiliza esas mismas tres variables para producir todas las divisiones.

Finalmente, al comparar estas importancias con los coeficientes del modelo LASSO, se observa un patrón consistente. En LASSO, las variables con menor peso relativo —como MAS_500_S, CH03, PP07C o CH04— son justamente aquellas que el árbol considera irrelevantes y no utiliza en ningún nodo. Asimismo, CH08, que en LASSO presenta uno de los coeficientes más grandes, es también la variable más importante en el árbol. Esto muestra que ambos algoritmos coinciden en qué predictores son los verdaderamente útiles, aunque lo expresen de formas distintas: LASSO los “achica” hacia cero, mientras que el árbol directamente los descarta de su estructura.


Árbol de Decisión Podado (ccp_alpha=0.002000)
CH08 <= 0.2
gini = 0.428
samples = 3016
value = [2082.0, 934.0]
class = No Pobre
True
False
CAT_INAC <= 0.452
gini = 0.277
samples = 2086
value = [1740, 346]
class = No Pobre
CH08 <= 4.478
gini = 0.465
samples = 930
value = [342, 588]
class = Pobre
gini = 0.179
samples = 1342
value = [1209, 133]
class = No Pobre
gini = 0.409
samples = 744
value = [531.0, 213.0]
class = No Pobre
CH06 <= -0.738
gini = 0.446
samples = 880
value = [295, 585]
class = Pobre
gini = 0.113
samples = 50
value = [47, 3]
class = No Pobre
gini = 0.33
samples = 364
value = [76, 288]
class = Pobre
gini = 0.489
samples = 516
value = [219, 297]
class = Pobre


Importancia de Variables del Árbol de Decisión
CH08
CAT_INAC
CH06
CH07
CH04
NIVEL_ED
ESTADO
PP07A
PP07C
CH03
MAS_500_S
Variable
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Importancia


CH08 <= 0.2
gini = 0.428
samples = 3016
value = [2082.0, 934.0]
class = No Pobre
True
False
CAT_INAC <= 0.452
gini = 0.277
samples = 2086
value = [1740, 346]
class = No Pobre
CH08 <= 4.478
gini = 0.465
samples = 930
value = [342, 588]
class = Pobre
gini = 0.179
samples = 1342
value = [1209, 133]
class = No Pobre
gini = 0.409
samples = 744
value = [531.0, 213.0]
class = No Pobre
CH06 <= -0.738
gini = 0.446
samples = 880
value = [295, 585]
class = Pobre
gini = 0.113
samples = 50
value = [47, 3]
class = No Pobre
gini = 0.33
samples = 364
value = [76, 288]
class = Pobre
gini = 0.489
samples = 516
value = [219, 297]
class = Pobre

C. Comparación entre métodos

6)

Para comparar el desempeño de los distintos modelos estimados, calculamos la matriz de confusión (con umbral p > 0.5), la curva ROC y las métricas utilizadas en el TP3. Evaluamos cinco modelos sobre el mismo set de test: regresión logística sin penalidad, KNN (con el K óptimo del TP3), regresión logística con penalidad L1 (LASSO), con penalidad L2 (Ridge) y el árbol de decisión podado. Esta comparación permite analizar no solo la performance predictiva, sino también el trade-off entre interpretabilidad y exactitud.

Desempeño general:

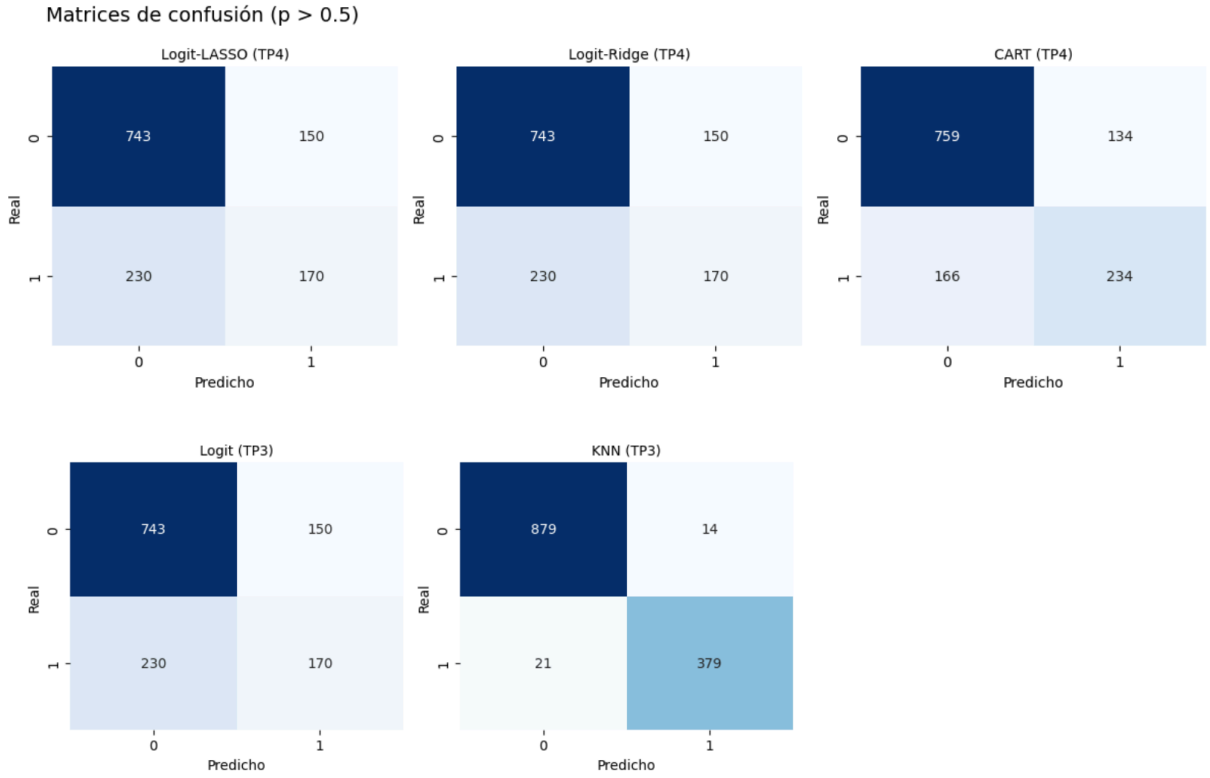
Las métricas muestran diferencias marcadas entre los modelos. Las tres regresiones logísticas (sin penalidad, LASSO y Ridge) presentan exactamente el mismo desempeño: un accuracy de 0.706, error (1 – accuracy) de 0.294 y un AUC de 0.748. Esto indica que, en este caso, las penalizaciones no generaron mejoras en la capacidad predictiva respecto del modelo base.

El árbol de decisión (CART) obtuvo un mejor rendimiento que las regresiones logísticas, especialmente para la clase “pobre”: logró mayor recall (0.586 vs 0.423) y un F1 más alto. Su accuracy total también mejora ligeramente (0.768), y su AUC aumenta a 0.765. Esto muestra que un modelo no lineal puede capturar relaciones que el logit no detecta.

Por otro lado, KNN es el modelo con mejor performance por amplio margen: accuracy de 0.947, error muy bajo (0.053), AUC de 0.997 y valores extremadamente altos de recall y precisión. En términos predictivos, KNN supera al resto de los métodos.

	Accuracy	Error (1-Acc)	Recall (Pobre)	Precision (Pobre)	F1 (Pobre)	AUC
Logit (TP3)	0.7061	0.2939	0.4250	0.5312	0.4722	0.7476
KNN (TP3)	0.9729	0.0271	0.9475	0.9644	0.9559	0.9970
Logit-LASSO (TP4)	0.7061	0.2939	0.4250	0.5312	0.4722	0.7476
Logit-Ridge (TP4)	0.7061	0.2939	0.4250	0.5312	0.4722	0.7476
CART (TP4)	0.7680	0.2320	0.5850	0.6359	0.6094	0.7647

Matrices de confusión:



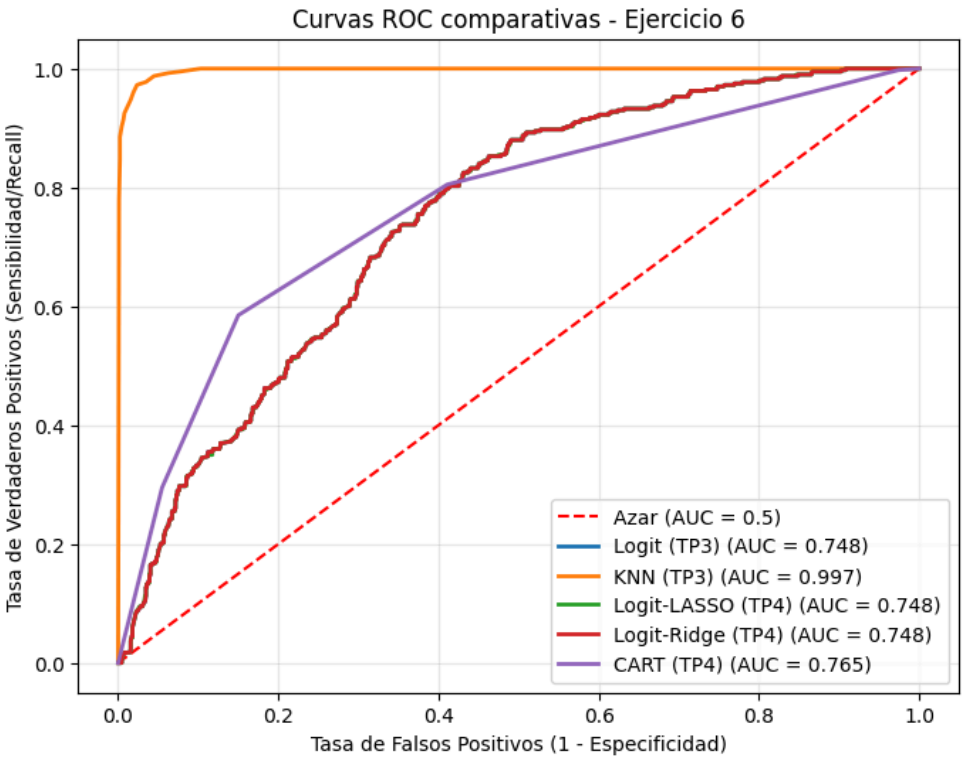
Las matrices permiten observar cómo se distribuyen los aciertos y errores:

Regresiones Logísticas (sin penalidad, LASSO y Ridge)
Presentan exactamente la misma matriz: muchos verdaderos negativos, pero también un número considerable de falsos negativos (230). Esto implica que el modelo tiene dificultades para identificar correctamente a quienes se perciben como “pobres”.

Árbol de Decisión (CART): Reduce tanto los falsos negativos como los falsos positivos en comparación con Logit. Esto se refleja en un mayor recall y mayor F1, lo cual indica una mejora real en la detección de la clase minoritaria.

KNN: Tiene muy pocos errores en ambas clases. La matriz muestra que el modelo prácticamente no confunde observaciones, coherente con el AUC cercano a 1. Esto confirma que es el mejor método en términos predictivos.

Curvas ROC: Las curvas ROC refuerzan las conclusiones anteriores. Las tres versiones del logit se superponen por completo, con un AUC moderado (0.748). CART muestra una curva más alta, con AUC de 0.765, indicando un mejor balance entre sensibilidad y especificidad. KNN se destaca con una curva extremadamente cercana al punto (0,1), reflejando un AUC casi perfecto.



Es importante aclarar que, en el TP3, tuvimos que incluir una variable de ingreso para poder estimar KNN, porque no contábamos con suficientes variables continuas para que el modelo funcionara correctamente. Esta decisión metodológica puede explicar, en parte, el desempeño excepcionalmente alto que obtuvo KNN en esta comparación: el modelo termina utilizando el una variable muy informativa para distinguir entre quienes se perciben “pobres” o “no pobres”, lo que naturalmente mejora la capacidad predictiva. Por lo tanto, si bien KNN es el mejor en términos de exactitud, su ventaja se ve influida por esta restricción del TP3 y debe interpretarse con esa consideracion.

7)

Si miramos solo la performance, el modelo que mejor predice en el test es KNN. Sin embargo, ese resultado está fuertemente influido por el uso de la variable de ingreso, que en el TP3 tuvimos que incluir únicamente para poder estimar el modelo (porque necesitábamos al menos una variable continua). El problema es que, no corresponde usar variables relacionadas al ingreso, ya que la idea es poder predecir la pobreza incluso cuando las personas no reportan su ingreso. Por eso, aunque KNN tenga la mayor accuracy, no sería una opción válida para una política pública de asignación de recursos.

En cambio, el árbol de decisión podado mantiene un rendimiento sólido y, sobre todo, es completamente interpretable: permite ver con claridad qué variables determinan cada división y por qué un hogar es clasificado como vulnerable. Esa transparencia es clave para justificar decisiones cuando se manejan recursos escasos. Por eso, para este tipo de intervención, el árbol podado es el modelo que se elegiría.