Gráfico, Histograma

Descripción generada automáticamente

Gráfico, Histograma

Descripción generada automáticamente

Gráfico, Histograma

Descripción generada automáticamente

Gráfico, Histograma

Descripción generada automáticamente

Gráfico, Gráfico de barras

Descripción generada automáticamente

Gráfico, Gráfico de barras

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Gráfico, Gráfico de barras

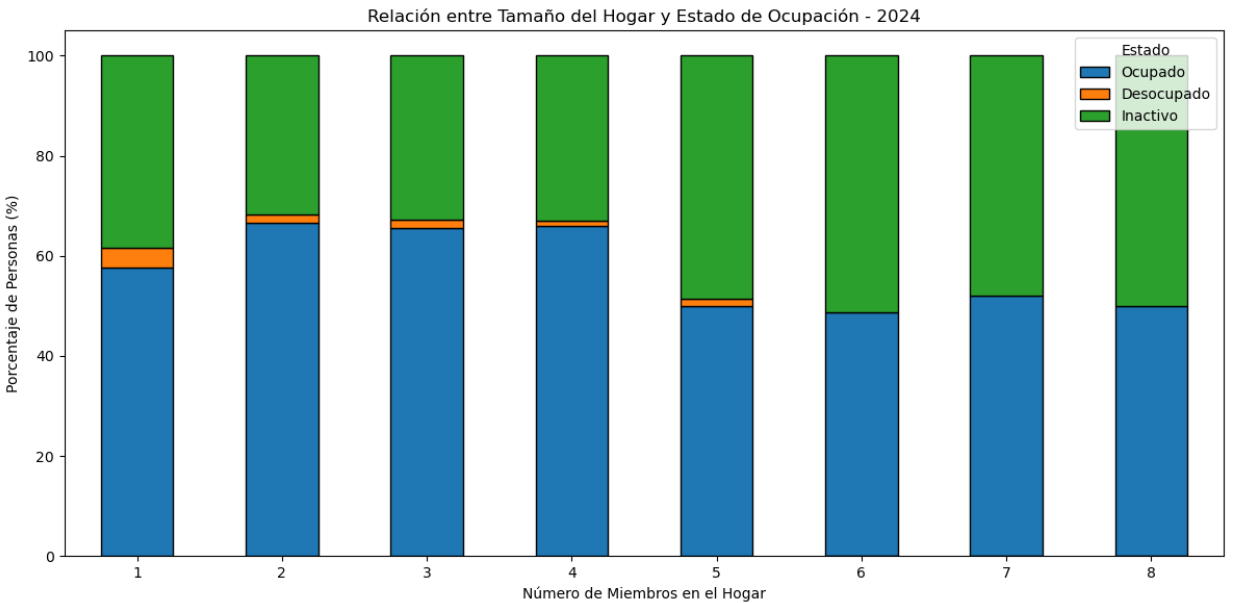
Descripción generada automáticamente

Gráfico, Gráfico de barras

Descripción generada automáticamente

Gráfico, Gráfico de barras

Descripción generada automáticamente



Parte 2

**Respuesta detallada a tus preguntas**

**1. Selección de λ por validación cruzada y por qué no usar el conjunto de prueba**

**Selección de λ por validación cruzada:**

La validación cruzada es una técnica que permite estimar el rendimiento de un modelo en datos no vistos, evitando el sobreajuste que podría ocurrir si utilizamos el conjunto de prueba para ajustar los hiperparámetros (como λ).

El proceso consiste en:

1. **Dividir los datos de entrenamiento:** Se divide el conjunto de entrenamiento en *k* subconjuntos (folds).
2. **Entrenar y evaluar:** Para cada valor de λ:
   * Se utiliza *k-1* folds para entrenar el modelo.
   * El fold restante se utiliza para evaluar el modelo y calcular una métrica de desempeño (por ejemplo, el error cuadrático medio).
3. **Calcular la métrica promedio:** Se calcula el promedio de la métrica de desempeño sobre todas las iteraciones.
4. **Seleccionar el mejor λ:** Se selecciona el valor de λ que obtuvo la mejor puntuación en la validación cruzada.

**¿Por qué no usar el conjunto de prueba?**

* **Sobreajuste:** Si utilizamos el conjunto de prueba para ajustar los hiperparámetros, estaríamos "entrenando" el modelo en los datos de prueba, lo que podría llevar a una sobreestimación del rendimiento del modelo en datos nuevos.
* **Propósito del conjunto de prueba:** El conjunto de prueba se utiliza únicamente para obtener una estimación imparcial del rendimiento final del modelo una vez que se ha seleccionado el mejor conjunto de hiperparámetros.

**2. Implicancias de k en la validación cruzada**

* **k pequeño:** Si k es muy pequeño, la varianza de la estimación del error puede ser alta, ya que cada fold contiene una proporción relativamente grande de los datos. Esto puede dificultar la selección del mejor modelo.
* **k grande:** Si k es muy grande, el tiempo de cálculo aumenta significativamente, ya que se deben entrenar muchos modelos. Además, si el conjunto de datos es pequeño, cada fold puede contener muy pocos datos, lo que puede afectar la estabilidad de las estimaciones.
* **k = n:** Si k es igual al número de muestras, cada fold contiene una sola muestra. Esto se conoce como "leave-one-out cross-validation" y puede ser computacionalmente costoso, especialmente para conjuntos de datos grandes. En este caso, el modelo se estima n veces.

**3. Penalidad L1 (LASSO) y L2 (Ridge) en regresión logística**

La penalización L1 (LASSO) tiende a reducir algunos coeficientes a cero, lo que puede conducir a la selección de características. La penalización L2 (Ridge) reduce el tamaño de todos los coeficientes, pero generalmente no los hace cero.

Para implementar estas penalizaciones en Python, utilizamos el parámetro penalty='l1' o penalty='l2' en el objeto LogisticRegression.

**4. Barrido de λ y generación de gráficos**

Para realizar un barrido de λ y generar los gráficos solicitados, puedes utilizar un bucle for para iterar sobre los diferentes valores de λ y calcular las métricas de desempeño para cada valor. Luego, puedes utilizar las librerías seaborn y matplotlib para crear los gráficos.

**Código de ejemplo (sin datos específicos):**

Python

import numpy as np

from sklearn.model\_selection import GridSearchCV

from sklearn.linear\_model import LogisticRegression

import seaborn as sns

import matplotlib.pyplot as plt

# ... (tu código existente)

# Valores de lambda

lambdas = np.logspace(-5, 5, 11)

# Diccionario para almacenar los resultados

results = {'lambda': [], 'penalty': [], 'mean\_cv\_error': [], 'mean\_coef\_zero': []}

for penalty in ['l1', 'l2']:

for l in lambdas:

# Crear el modelo

logreg = LogisticRegression(penalty=penalty, C=1/l, solver='liblinear')

# Validación cruzada

cv\_results = cross\_validate(logreg, X\_train, y\_train, cv=10, scoring='accuracy')

# Calcular la proporción de coeficientes cero (solo para LASSO)

if penalty == 'l1':

coef\_zero = np.mean(np.abs(logreg.coef\_) < 1e-6)

else:

coef\_zero = np.nan

# Almacenar los resultados

results['lambda'].append(l)

results['penalty'].append(penalty)

results['mean\_cv\_error'].append(1 - np.mean(cv\_results['test\_score']))

results['mean\_coef\_zero'].append(coef\_zero)

# Crear un DataFrame con los resultados

results\_df = pd.DataFrame(results)

# Gráfico de box plot para el error de predicción

sns.boxplot(x='lambda', y='mean\_cv\_error', hue='penalty', data=results\_df)

# Gráfico de línea para la proporción de coeficientes cero (LASSO)

sns.lineplot(x='lambda', y='mean\_coef\_zero', data=results\_df[results\_df['penalty'] == 'l1'])

Usa el código [con precaución](/faq#coding).

**Explicación:**

* **Bucle for:** Iteramos sobre los diferentes valores de λ y las penalizaciones L1 y L2.
* **Validación cruzada:** Utilizamos cross\_validate para realizar la validación cruzada y calcular el error de validación.
* **Proporción de coeficientes cero:** Para LASSO, calculamos la proporción promedio de coeficientes que son cercanos a cero.
* **Almacenamiento de resultados:** Almacenamos los resultados en un DataFrame para facilitar la visualización.
* **Gráficos:** Utilizamos sns.boxplot para visualizar la distribución del error de validación para cada valor de λ y sns.lineplot para visualizar la proporción de coeficientes cero en función de λ para LASSO.

**Importante:** Este es un ejemplo básico. Puedes personalizarlo aún más agregando más métricas, utilizando diferentes visualizaciones y explorando otros hiperparámetros.

**¿Necesitas más ayuda con alguna parte específica del código o con la interpretación de los resultados?**

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Vamos a interpretar los resultados de ambos modelos, el Lasso y el Ridge, en términos de sus hiperparámetros, precisión y las métricas de clasificación.

### Modelo Lasso

1. \*\*Mejores hiperparámetros\*\*:

- `{‘model\_\_C’: 10000.0}` indica que el valor óptimo de `C` (que controla la regularización) es 10000. Un valor alto de `C` reduce la regularización, permitiendo que el modelo se ajuste mejor a los datos de entrenamiento.

2. \*\*Exactitud en test\*\*:

- La precisión en el conjunto de prueba es de \*\*0.9993\*\*, o \*\*99.93%\*\*. Esto muestra que el modelo clasifica correctamente el 99.93% de las observaciones de prueba.

3. \*\*Reporte de clasificación\*\*:

- \*\*Clase 0\*\* (etiqueta de la clase mayoritaria):

- `Precision`, `Recall`, y `F1-score` son \*\*1.00\*\* en cada métrica, indicando que el modelo clasificó todos los ejemplos de esta clase sin errores.

- \*\*Clase 1\*\* (etiqueta de la clase minoritaria):

- La `Precision` es \*\*1.00\*\* y el `Recall` es \*\*0.99\*\*, lo cual significa que el modelo identificó casi todos los ejemplos de la clase 1.

- \*\*Accuracy\*\* (Exactitud), \*\*Macro avg\*\*, y \*\*Weighted avg\*\* son prácticamente 1.00, indicando un rendimiento sobresaliente en ambas clases.

4. \*\*Matriz de confusión\*\*:

- `[[1308, 0], [1, 85]]`: El modelo clasificó correctamente los 1308 casos de la clase 0 y 85 de los 86 de la clase 1, con solo un falso negativo en la clase 1.

### Modelo Ridge

1. \*\*Mejores hiperparámetros\*\*:

- `{‘model\_\_C’: 0.000774263682681127}` indica que el valor óptimo de `C` es \*\*0.00077\*\*, lo cual aplica una mayor regularización, restringiendo la complejidad del modelo.

2. \*\*Exactitud en test\*\*:

- La exactitud en el conjunto de prueba es de \*\*0.9390\*\* o \*\*93.9%\*\*, que sigue siendo buena pero inferior a la del modelo Lasso.

3. \*\*Reporte de clasificación\*\*:

- \*\*Clase 0\*\*:

- `Precision`, `Recall`, y `F1-score` son cercanos a 1.00, indicando que el modelo maneja bien la clase mayoritaria.

- \*\*Clase 1\*\*:

- `Precision` es \*\*0.67\*\*, mientras que el `Recall` es de solo \*\*0.02\*\*. Esto significa que el modelo identificó correctamente muy pocos ejemplos de la clase 1 (solo el 2%), por lo que su capacidad para capturar esta clase es baja.

- \*\*Macro avg\*\* y \*\*Weighted avg\*\*:

- Las métricas muestran la baja capacidad del modelo para clasificar adecuadamente ambas clases, en especial la clase minoritaria.

4. \*\*Matriz de confusión\*\*:

- `[[1307, 1], [84, 2]]`: El modelo clasificó correctamente casi todos los casos de la clase 0, pero solo 2 de los 86 casos de la clase 1, resultando en 84 falsos negativos.

### Comparación e Interpretación General

- \*\*Modelo Lasso\*\* es mucho más preciso y capaz de identificar ambas clases, con una precisión cercana al 100%.

- \*\*Modelo Ridge\*\* presenta una alta precisión para la clase mayoritaria (0) pero es ineficaz para la clase minoritaria (1) debido a la alta regularización.

- Esto indica que el modelo Lasso, con menor regularización, se adapta mejor a la complejidad de los datos, logrando capturar ambas clases de manera precisa.

Por tanto, en este caso, el modelo Lasso es claramente la mejor elección.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Gráfico, Gráfico de barras, Histograma

Descripción generada automáticamente

**Elección de λ\lambdaλ por validación cruzada y no por el conjunto de prueba**

La elección del hiperparámetro λ\lambdaλ en modelos LASSO y Ridge (penalización L1 y L2) mediante el conjunto de prueba podría sesgar la evaluación del modelo. Al ajustar λ\lambdaλ sobre los datos de prueba, estaríamos usando esos datos para mejorar el modelo en lugar de usarlos exclusivamente para evaluar su rendimiento general. Esto violaría el principio de separación entre entrenamiento y prueba, y podría llevar a una sobreestimación del rendimiento del modelo en nuevos datos. Por eso, la validación cruzada es ideal para elegir el mejor valor de λ\lambdaλ, pues permite probar y ajustar el modelo en múltiples divisiones del conjunto de entrenamiento, y reserva el conjunto de prueba para una evaluación final objetiva.

**Validación cruzada: implicancias de usar un kkk muy pequeño o muy grande**

1. **Valor de kkk pequeño**:
   * Implica tener pocas divisiones (por ejemplo, k=2k=2k=2 o k=3k=3k=3).
   * Da lugar a una alta variabilidad en los resultados, ya que cada subdivisión de entrenamiento puede ser bastante diferente en composición.
   * Puede conducir a un modelo que es menos generalizable y puede sobreajustarse a las divisiones específicas de los datos.
2. **Valor de kkk grande**:
   * Reduce la variabilidad, ya que usa muchas más particiones de los datos (más evaluaciones).
   * Sin embargo, el tiempo de entrenamiento aumenta, ya que el modelo debe ajustarse muchas veces.
   * Cuando k=nk = nk=n (validación cruzada de tipo "Leave-One-Out"), el modelo se entrena nnn veces, usando todas las muestras menos una en cada iteración. Este método es más exhaustivo, pero muy costoso en términos de tiempo para conjuntos grandes de datos.

**Implementación de la penalidad L1 y L2 en regresión logística**

Vamos a entrenar una regresión logística con penalización L1 (Lasso) y L2 (Ridge) usando LogisticRegression de sklearn. Luego, evaluaremos los modelos a través de la matriz de confusión, la curva ROC, el AUC y la exactitud en cada uno de los años.

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

L1 (Lasso) - Exactitud: 0.9986

L1 (Lasso) - AUC: 0.9879

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

L2 (Ridge) - Exactitud: 0.9390

L2 (Ridge) - AUC: 0.8326

Ahora Separo por años

Gráfico

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

L1 (Lasso) - Año 2004 - Exactitud: 1.0000

L1 (Lasso) - Año 2004 - AUC: 1.0000

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

L2 (Ridge) - Año 2004 - Exactitud: 0.9987

L2 (Ridge) - Año 2004 - AUC: 1.0000

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

L1 (Lasso) - Año 2024 - Exactitud: 0.9984

L1 (Lasso) - Año 2024 - AUC: 1.0000

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

L2 (Ridge) - Año 2024 - Exactitud: 0.9635

L2 (Ridge) - Año 2024 - AUC: 0.8580

Gráfico

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente