**TRABAJO PRÁCTICO N°4 - CLASIFICACIÓN Y REGULARIZACIÓN DE DESOCUPACIÓN USANDO LA EPH**

Big Data y Aprendizaje Automático para Economistas

**GRUPO N°1: Pilar Valdez y José Antonio López**

**Parte I: Análisis de la base de hogares y tipo de ocupación**

Trabajo sobre las Bases de Datos

Las bases de datos tenian dos formatos diferentes, .dta y .xls, con Phyton pudimos trabajar esos diferente formatos en un solo programa.

Por otro lado, las variables con respuestas categóricas en el año 2004 se cargaban como lista de datos en la Base, en tanto en 2024 se cargan como numericas. De manera que realizamos la unificación con Jupyter. Podemos mencionar que este trabajo se hizo sobre variables como Estado, Nivel de Educación, Sexo, etc.

Una vez realizado lo anterior se juntaron las cuatro Bases de Datos anteriores en un solo dataframe. Para hacer un Merge entre las bases de Individuos y la de Hogares utilizamos las columnas CODUSU y Número de Hogar.

Presentamos a continuación los histogramas de ITF Ingreso total familiar, IPCF Ingreso Per capita familiar y Nivel Educativo.

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

En Relación a las variables que esperamos sean buenos predictores de si una persona está Desocupada, esperamos que las principales sean Nivel de educación, Nivel de Educación, Genero y tamaño del Hogar, a continuación se presenta análisis para cada una.

Tabla

Descripción generada automáticamente

Gráfico, Gráfico de barras

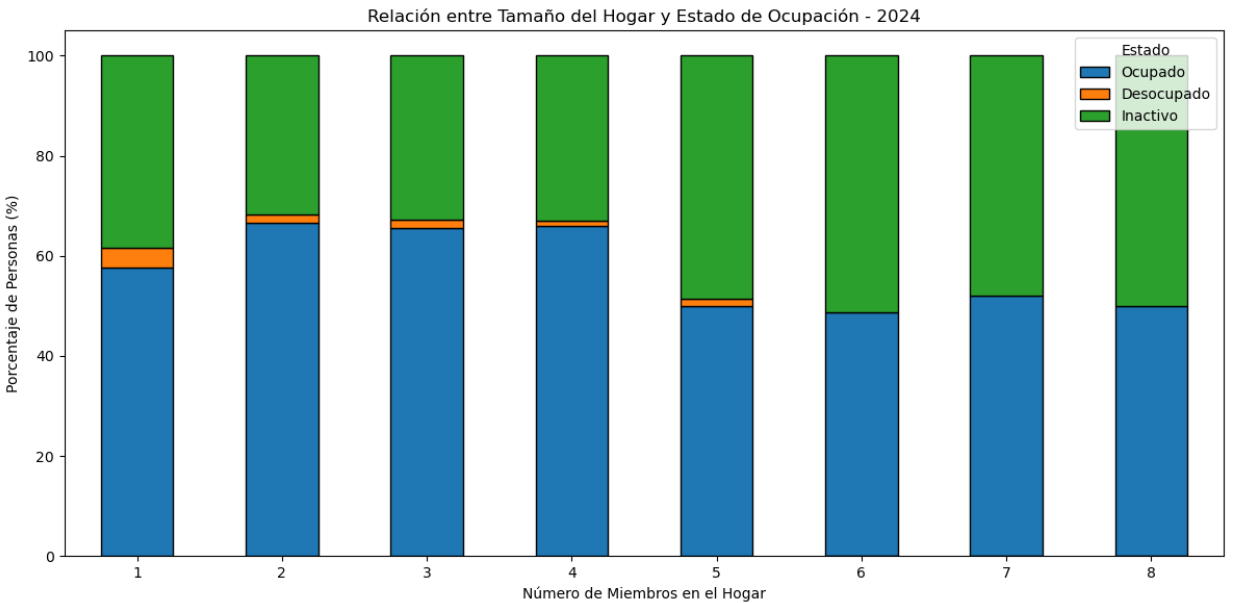
Descripción generada automáticamente

Gráfico, Gráfico de barras

Descripción generada automáticamente

Gráfico, Gráfico de barras

Descripción generada automáticamente



**Parte II Clasificación y Regularización**

**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.

**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.

**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.

**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.

A continuación, los resultados:

Gráfico, Gráfico de líneas

Descripción generada automáticamenteGráfico, Gráfico de rectángulos

Descripción generada automáticamente

L1 (Lasso) - Exactitud: 0.9986

Gráfico, Gráfico de líneas

Descripción generada automáticamenteGráfico, Gráfico de rectángulos

Descripción generada automáticamenteL1 (Lasso) - AUC: 0.9879

L2 (Ridge) - Exactitud: 0.9390

L2 (Ridge) - AUC: 0.8326

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**

- 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.