Universidad de San Andrés Octubre de 2021

Big Data, Aprendizaje y Minería de Datos

Trabajo Práctico N°3: Regularización aplicada a la EPH

Profesor: Walter Sosa Escudero
Asistentes: Belén Michel Torino y Rodrigo Bonazzola

Fecha de entrega: Lunes 25/10 a las 23:59.
Contenidos: analizar el problema del no reporte de ingresos en las encuestas de hogares y como eso afecta la medición de la tasa de pobreza. Se profundizará en el análisis de este tematica con los nuevos métodos vistos en clase para clasificar individuos sin reporte de ingresos.

Modalidad de Entrega

- Al finalizar el trabajo práctico deben hacer un último `commit` en su repositorio de GitHub con el mensaje "Entrega final del tp".
- Asegúrense de haber creado una carpeta llamada TP3. Su reporte (pdf) y código (jupyter notebook) deben estar dentro de esa carpeta.
- También deben completar el link de su repositorio -para que pueda ser clonado y corregido- en esta google sheet
- La última versión en el repositorio es la que será evaluada. Por lo que es importante que:
  - No completen la google sheet hasta no haber terminado y estar seguros de que han hecho el `commit` y `push` a la versión final que quieren entregar. Debido a que se pueden tomar hasta 3 días de extensión a lo largo del curso, no corregireemos sus tareas hasta no ver el link en la google sheet.
  - No hagan nuevos `push` después de haber entregado su versión final. Esto generaría confusión acerca de qué versión es la que quieren que se les corrija.

Reglas de formato y presentación

- El trabajo se debe entregar como Jupyter Notebook. Este debe contener un boton para esconder los bloques de código. De esta forma, al corregir podremos optar por ver tando la versión con código como la versión 'limpia' donde solo esten sus bloques de texto con respuestas, graficos y tablas. La versión 'limpia' debe tener una extensión máxima de 10 páginas al imprimirse en PDF.
- Todos los miembros del equipo deben haber hecho al menos un `commit` durante la producción del TP para asegurar que todos hayan aportado a su resolución.
- Se espera una buena redacción en la resolución del práctico.

Parte I: Análisis de la base de hogares y cálculo de pobreza

Ahora que ya se han familiarizado con la Encuesta Permanente de Hogares (EPH) y con el uso de la tabla de adulto equivalente vamos a complejizar un poco la construcción del indice de pobreza. Lo calcularemos a nivel de hogares, con sus respectivos factores de expansión y realizaremos una limpieza de la base con mayor dedicación.

A continuación complementaremos el trabajo hecho en el TP2 usando la encuesta a nivel de hogares de la EPH:

1. Exploren el diseño de registro de la base de hogar: a priori, ¿qué variables creen que pueden ser muy predictivas de pobreza y que sería muy útil incluir para perfeccionar el ejercicio del TP2?
2. Descarguen la base de microdatos de la EPH correspondiente al primer trimestre de 2021 (la base de hogares se llama `usu_hogar_T121.xls`). Importen los datos de la encuesta de hogar y al igual que en el TP1 conserven sólo las observaciones que corresponden a los aglomerados de Ciudad Autónoma de Buenos Aires o del Gran Buenos Aires.
3. Unan la tabla de la encuesta individual con la de la encuesta de hogar.
4. Generen sus propias funciones para limpiar la base de datos o, si deciden utilizar funciones existentes en paquetes como `numpy` y `pandas`, mencionen cuáles usarán y de qué paquetes son.
5. Limpien la base de datos tomando criterios que hagan sentido, tanto para el tratamiento de los valores faltantes, de los *outliers*, como así tambien decidan qué variables categóricas y strings usarán y transfórmenlas de forma que haga sentido para los ejercicios siguientes. Justifiquen sus decisiones.
6. Presenten estadísticas descriptivas de 5 variables de la encuesta de hogar que ustedes creen que pueden ser relevantes para predecir pobreza.
7. Repitan el inciso 1.2.f del TP2 para construir la columna `adulto_equiv` y la columna `ad_equiv_hogar` (pueden utilizar su código del TP2)
8. Repitan el inciso 1.3 y 1.4 del TP2 para dividir la base en dos dataframes

donde: uno conserve las personas que reportaron ITF (llamada `respondieron`) y la otra conserve las personas que **no** reportaron ITF (llamada `norespondieron`). Además, agreguen a la base `respondieron` una columna llamada `ingreso_necesario` que sea el producto de la canasta básica por `ad_equiv_hogar`.

9. Agreguen a `respondieron` una columna llamada `pobre` que tome valor 1 si el ITF es menor al `ingreso_necesario` que necesita esa familia, y 0 en caso contrario.[1]
10. En el TP2 calcularon los individuos bajo la linea de pobreza. Sin embargo, cuando se habla de pobreza el número más utilizado es el de la tasa de hogares bajo la linea de pobreza. Para calcularlo, utilicen una sola observación por hogar y sumen el ponderador `PONDIH` que permite expandir la muestra de la EPH al total de la población que representa. ¿Cuál es la tasa de hogares bajo la línea de pobreza para el GBA? ¿Se asemeja al que reporta el INDEC en sus informes?

Parte II: Construcción de funciones

El objetivo de esta parte del trabajo es generar código que sea flexible y que esté modularizado (en funciones bien documentadas con docstrings). De esta forma evitarán repetir código y podrán utilizarlo en distintos escenarios (como por ejemplo la Parte III de este TP y sus proyectos personales a futuro).

1. Escriban una función, llamada `evalua_metodo`, que reciba como argumentos un modelo y los datos de entrenamiento y prueba (`X_train, y_train, X_test, y_test`). La función debe ajustar el modelo con los datos de entrenamiento y calcular las métricas que considere necesarias para esta problemática (de mínima debe reportar la matriz de confusión, las curvas ROC y los valores de AUC y de *accuracy score* de cada método). El output de la función debe ser una colección con las métricas evaluadas.
2. Escriban una función, llamada `cross_validation`, que realice validación cruzada con k iteraciones (k-fold CV), llamando a la función del inciso anterior en cada una, pero para las k distintas particiones. La función debe recibir como argumentos el modelo, el valor de k y un dataset (es decir, sólo `X` e `y`). Pueden ayudarse con la función KFold para generar las particiones necesarias.
3. Escriban una función, llamada `evalua_config` que reciba una lista de configuraciones de hiperparámetros[2] (los distintos valores a probar como hiperparámetros podrian codificarse en diccionarios de Python) y utilizando la función `cross_validation` obtenga el error[3] promedio para cada configuración. Finalmente, la función debe devolver la configuración que genere

[1] Notar que esta etiqueta es distinta a la pedida en el TP2, donde, por error, les dimos la indicación confusa de que el 0 correspondía a estar por debajo de la línea de pobreza. Creemos que esta nueva asignación es más intuitiva, dado que les pedimos que la columna se llame `pobre`.

[2] En `scikit-learn`, muchos métodos llaman `penalty` al método de regularización y `C` a la inversa del hiperparámetro λ

[3] utilicen la medición de error que prefieran. Una opción sería el Error Cuadrático Medio

menor error. [4]

4. Escriban una función, llamada `evalua_multiples_metodos` que les permita implementar los siguientes métodos con los hiperparámetros que ustedes elijan. Para la regresión logística, asegúrense de que esta función utilice su función `evalua_config` para optimizar el λ de la regularización. Finalmente, el output de la función debe ser una tabla donde las columnas sean las métricas que hayan evaluado (las que hayan incluido en la función `evalua_metodo`) y las filas sean los modelos (con su configuración de hiperparámetros asociada) que hayan corrido. Asegúrense de que la tabla incluya una columna con nombre del modelo y el valor de los hiperparámetros/configuración [5]:
   - Regresión logística
   - Análisis de discriminante lineal
   - KNN
   - Arbol de decisión
   - Support vector machines (SVM)
   - Bagging
   - Random Forests
   - Boosting

Parte III: Clasificación y Regularización

El objetivo de esta parte del trabajo es nuevamente intentar predecir si una persona es o no pobre utilizando datos distintos al ingreso, dado que muchos hogares son reacios a responder cuánto ganan. Esta vez lo haremos con la base unida de las preguntas de la encuesta de hogar y la encuesta individual. A su vez, incluiremos ejercicios de regularización y de validación cruzada.

1. Eliminen de ambas bases (`respondieron`, `norespondieron`) todas las variables relacionadas a ingresos (en el archivo `codigos_eph.pdf` ver las categorías: ingresos de la ocupación principal de los asalariados, ingresos de la ocupación principal, ingresos de otras ocupaciones, ingreso total individual, ingresos no laborales, ingreso total familiar, ingreso per cápita familiar). Elimine también las columnas `adulto_equiv`, `ad_equiv_hogar` e `ingreso_necesario`. Establezca a `pobre` como su variable dependiente (vector y). El resto de las variables serán las variables independientes (matriz X). Dependiendo de la función que usen, no se olviden de agregar la columna de 1 cuando sea necesario.

[4] **Consejo:** cuanto más genérica construyan la función, luego podrá ser utilizada en más situaciones. Por ahora, la usaremos solo para buscar el λ óptimo cuando utilicemos regularización en la regresión logistica.

[5] **Pista:** Para la regresión logistica, cuando incluyan regularización observen que deberán correr la función `evalua_metodo` dos veces. Una para optimizar los hiperparámetros (con un set de datos para train y otro para validación) y otra para obtener las métricas con el hiperparámetro óptimo (con un set de datos para train y otro para test)

2. Corran la función `evalua_multiples_metodos` con la base `respondieron`. En los próximos incisos profundizaremos en la tarea de regularización, pero en este ejercicio prueben al menos un hiperparámetro para regularizar y al menos un valor de λ.
3. Expliquen cómo elegiría λ por validación cruzada. Detallen por qué no usarían el conjunto de prueba (test) para su elección.
4. En validación cruzada, ¿cuál es el problema de usar un K muy pequeño y uno muy grande? y cuando $K = n$ (con n el número de muestras), ¿cuántas veces se estima el modelo?
5. Realicen un barrido en $\lambda = 10^n$ con $n \in \{-5, -4, -3..., +4, +5\}$ y utilicen 10-fold CV para elegir el λ óptimo en regresión logística con Ridge y con LASSO. ¿Qué λ seleccionó en cada caso? Generen box-plots mostrando la distribución del error de predicción para cada λ. Cada box debe corresponder a un valor de λ y contener como observaciones el error medio de validación para cada partición. Además, para la regularización LASSO, genere un box-plot similar, pero ahora graficando la proporción de variables ignoradas por el modelo en función de λ, es decir la proporción de variables para las cuales el coeficiente asociado es cero.
6. En el caso del valor óptimo de λ para LASSO encontrado en el inciso anterior, ¿qué variables fueron descartadas? ¿Son las que hubieran esperado? ¿Tiene relación con lo que respondió en el inciso 1 de la Parte I?
7. Elijan alguno de los modelos de regresión logistica donde haya probado distintos parametros de regularización y comente: ¿Qué metodo de regularización funcionó mejor Ridge o LASSO? Comente mencionando el error cuadrático medio (ECM).
8. ¿Cuál de todos los métodos evaluados predice mejor? ¿Con qué hiperparámetros? Justifiquen detalladamente utilizando las medidas de precisión que conoce.
9. Con el método que seleccionó, predigan qué personas son pobres dentro de la base `norespondieron`. ¿Qué proporción de los hogares son pobres en esa submuestra?