Trabajo Práctico 3 - Regularización aplicada a la EPH

Gil Deza, Hüppi Lo Prete, Walker

Show code

Parte I

1)

Las variables que pueden servir como predictores para entender si un hogar es pobre o no son: -Variables que representan las características de la vivienda como el acceso al agua, a un baño, materiales de construcción y si está dentro de un barrio de emergencia. -Variables que representan las características habitacionales del hogar como por ejemplo, la cantidad de cuartos destinados para dormir, régimen de tenecia del terreno y qué combustible se utiliza para cocinar. -Variables denominada "Estrategias del hogar", entre ellas se encuentran si cobran un subsidio, si tienen una beca de estudio, si tuvieron que vender sus pertenencias en el último tiempo, menores de edad trabajando, cantidad de personas que viven en el hogar, decil de ingresos, entre otras. Consideramos que esta última variable es muy relevante dado que marca la diferencia entre medir la pobreza a nivel hogar o a nivel individuo debido a que hogares pobres tienden a tener más miembros.

2)

. . .

3)

. . .

Eliminamos todas las columnas duplicadas luego del merge, dado que se encontraban en ambas bases.

. . .

4)

Utilizamos la función dropna del paquete de Pandas para eliminar columnas con observaciones vacias

5)

. . .

. . .

```
MAS_500_x es una variable object
CH05 es una variable object
```

Chequeamos cuál es el contenido de las variables que nos generarían problemas al no ser numéricas

. . .

Observamos que CH05 contiene la fecha de nacimiento de los individuos (con lo cual podríamos borrarla, dado que tenemos su edad en la base también)

MAS_500_x contienen una letra que indica el tamaño del aglomerado. Al habernos quedado con las observaciones de BsAs y Gran BsAs, todas tienen una "S".

Debido a esto, consideramos que podemos deshacernos de tales variables sin mayores problemas.

. . .

Ahora, chequearemos que no queden variables de ingresos o edades con observaciones negativas en nuestra base de datos.

. . .

Eliminamos variables con observaciones negativas

. . .

6)

Las variables que consideramos de interés para predecir la pobreza son: -V5_M: es el monto que recibe por subsidio o ayuda social -T_VI: monto total de ingresos no laborales -V11_M: monto de ingreso por beca de estudio -PP02I: variable binaria que indica si la persona trabajó en algún momento de los últimos 12 meses -IV12_3: variable binaria que indica si la vivienda está ubicada en un barrio de emergencia

```
count     5231.000000
mean       560.357484
std       3209.419657
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max      70000.000000
Name: V5_M, dtype: float64
count      5231.000000
mean       8146.944179
std       17669.813799
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max      104000.000000
Name: T_VI, dtype: float64
count     5231.000000
mean        51.787421
std       1253.600611
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max      60000.000000
Name: V11_M, dtype: float64
count    5231.000000
mean        0.892755
std         0.986800
min         0.000000
25%         0.000000
50%         0.000000
75%         2.000000
max         2.000000
Name: PP02I, dtype: float64
count    5231.000000
mean        1.993691
std         0.079183
min         1.000000
25%         2.000000
50%         2.000000
75%         2.000000
max         2.000000
Name: IV12_3, dtype: float64
```

7)

Tanto en el presente punto como en los dos siguientes recuperamos lo realizado en el TP2

. . .

. . .

. . .

8)

```
La cantidad de personas que no respondieron cuál es su ingreso total familia
r es: 1542
1       435000
2       120000
3        15000
6       210000
8        63000
         ...
5224     22000
5225     16000
5226    160000
5228     53500
5230    282000
Name: ITF_x, Length: 3689, dtype: int64
0       0
4       0
5       0
7       0
14      0
       ..
5218    0
5219    0
5221    0
5227    0
5229    0
Name: ITF_x, Length: 1542, dtype: int64
```

. . .

9)

. . .

10)

. . .

. . .

. . .

```
La suma del PONDIH de los hogares pobres es 1262824

La suma del PONDIH de todos los hogares 4978383

La tasa de hogares bajo la linea de pobreza de AMBA es 25.366148004281712 %
```

La tasa de pobreza que obtenemos se asemeja a la del INDEC. Para el primer semestre de 2022 para AMBA obtenemos una tasa de 25.36%, mientras que en el INDEC la tasa de pobreza para esta misma región es de 28.2%.

Parte 2

1)

1)

Si bien inicialmente habíamos incluido la curva de ROC y una versión más completa de la matriz de confusión, estas nos traían problemas posteriormente por lo que Belén nos sugirió no graficarlas.

2)

3)

4)

Parte 3

1)

. . .

. . .

. . .

. . .

. . .

. . .

. . .

. . .

. . .

. . .

. . .

. . .

2)

. . .

3)

Para elegir el parámetro λ por Cross Validation, dividimos la muestra de entrenamiento en una determinada cantidad de particiones. A continuación, estimamos el modelo elegido con diferentes valores λ donde, para cada uno, lo estimamos tantas veces como particiones se determinen rotando la partición que se deja fuera para poder computar el ECM con la predicción realizada sobre esta. Así, para cada λ computamos el ECM promedio y elegimos la configuración que minimiza ECM promedio. Por otro lado, no elegimos el conjunto de

test para su elección porque, en ese caso, estaríamos dándole a conocer al modelo los valroes de los datos que luego usamos para computar el ECM y así no podríamos chequear las predicciones de nuestro modelo inicial.

4)

Según el valor de K que tomemos este será el tamaño que nos queda para las muestras de entrenamiento y de test. Si K es demasiado pequeño, tendremos una muestra de entrenamiento pequeña que nos dará probablemente un modelo sesgado. Si K es demasiado grande, nos quedará una muestra de test pequeña y además corremos el riesgo de tener un problema de overfit (un modelo que predice muy bien adentro de la muestra, pero mal fuera de ella). Por otro lado, si K = n estaremos estimando el modelo n veces con n - 1 datos, esto se conoce como el procedimiento "leave one out".

5)

. . .

. . .

Elegimos el lamda óptimo con Lasso y Ridge

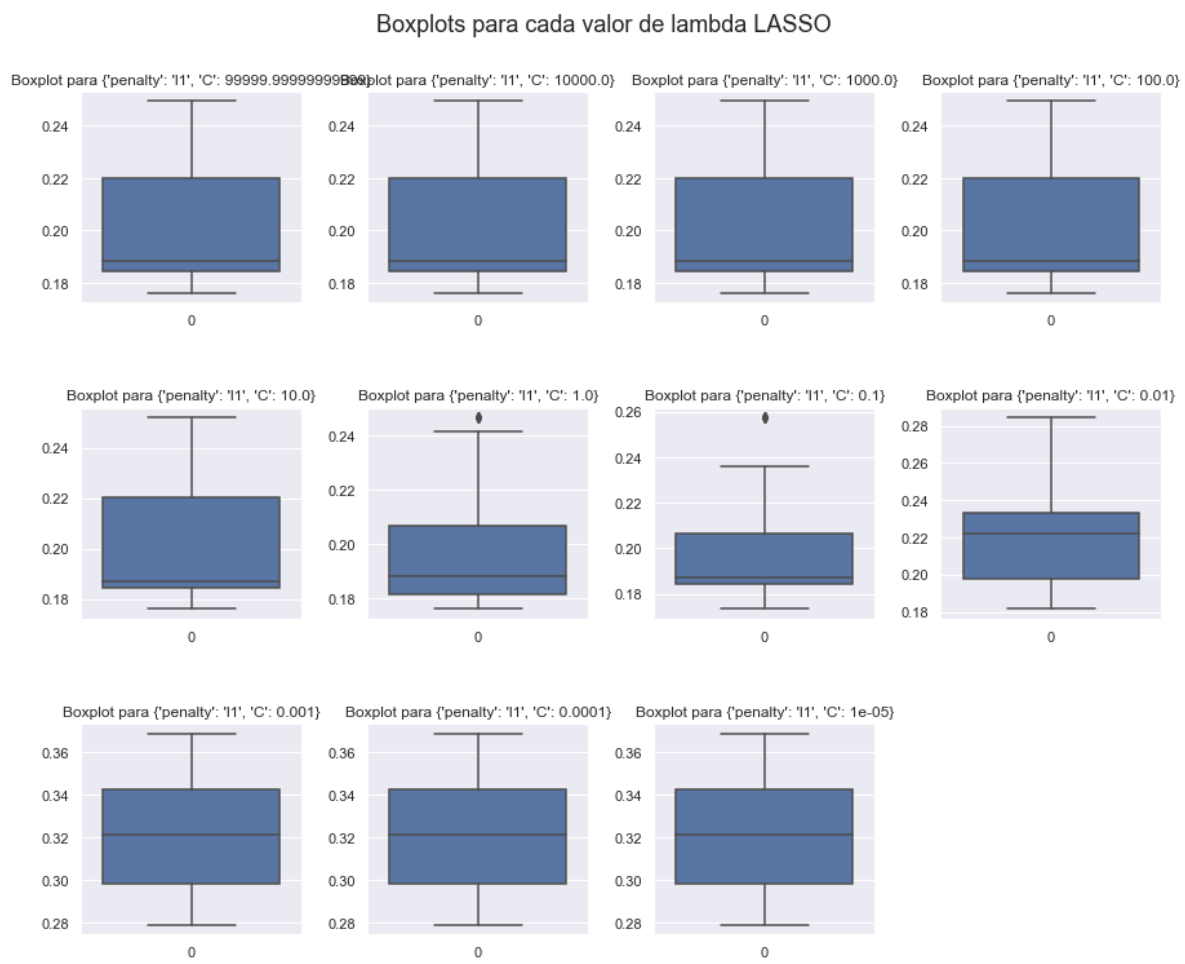
. . .

. . .

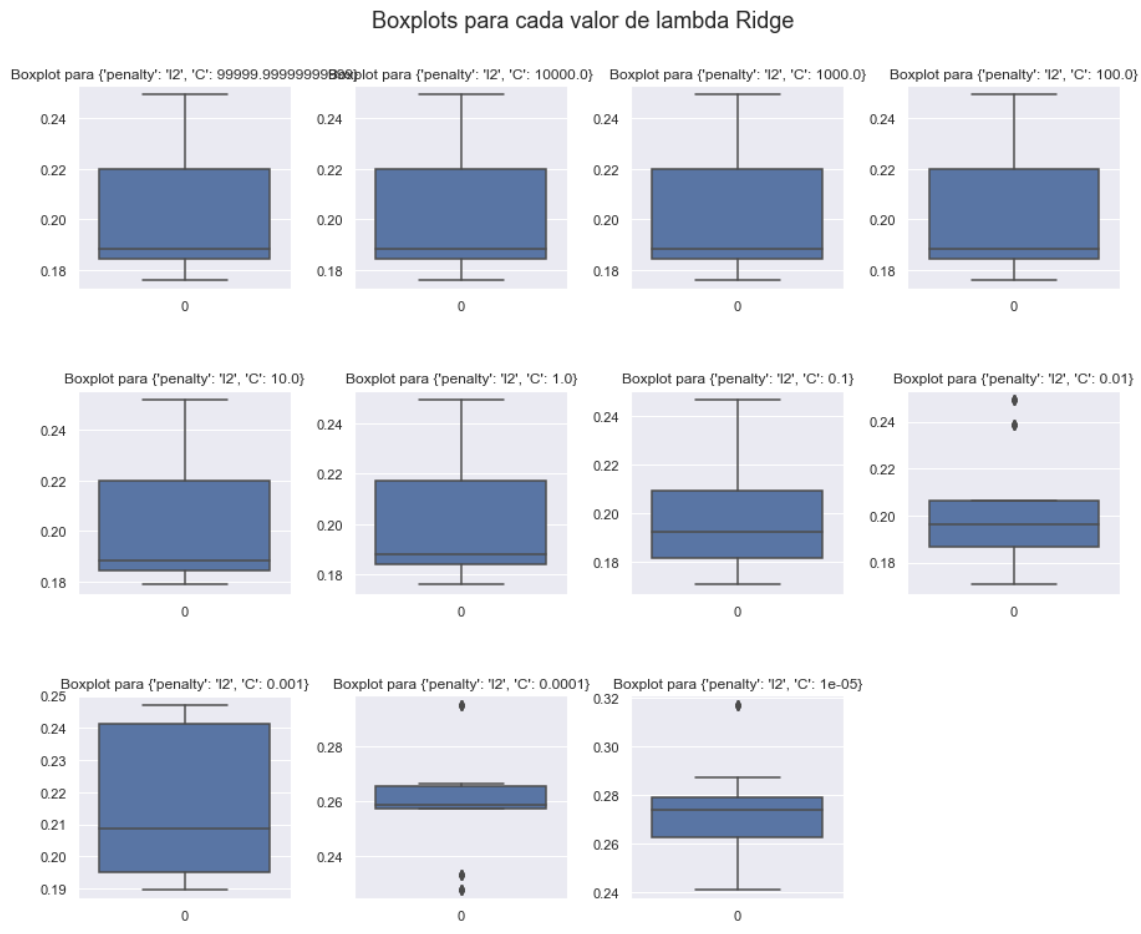
Tanto con Ridge como con Lasso el Lamda óptimo escogido es el lamda = 1. Sin embargo, es importante notar que en el caso de Lasso el ECM al utilizar este lamda es de 0.1980234, mientras que con Ridge, es de 0.199516.

Boxplots con distribución del error de predicción para cada Lamda

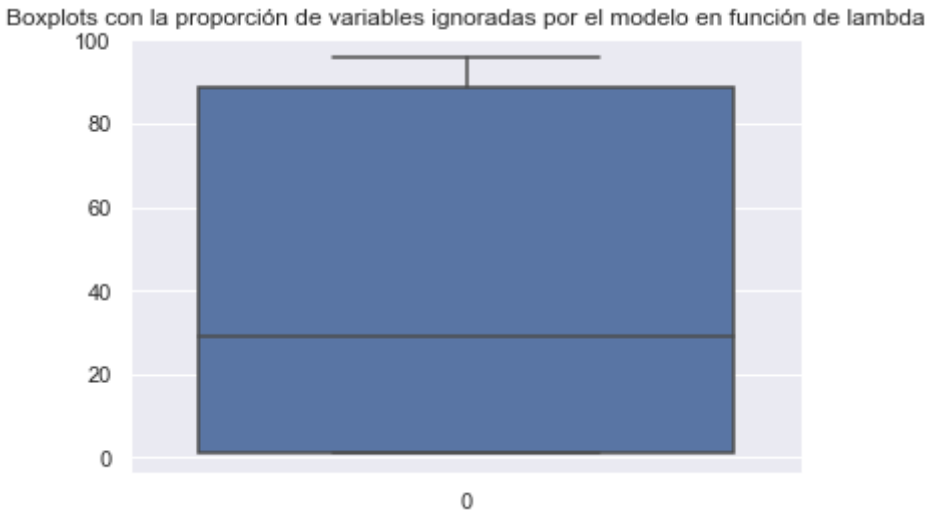
```
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\3111193304.py:53: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\3111193304.py:53: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\3111193304.py:53: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\3111193304.py:53: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\3111193304.py:53: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
```



```
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\3111193304.py:53: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\3111193304.py:53: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\3111193304.py:53: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\3111193304.py:53: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\3111193304.py:53: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\4104559683.py:20: UserWa
rning: Matplotlib is currently using module://matplotlib_inline.backend_in
line, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



Boxplot con la proporción de variables ignoradas por el modelo en función de lamda



. . .

. . .

6)

```
Coeficientes del modelo con lamda óptimo:
```

Out[123]:

	0
AGLOMERADO_x	0.468799
ANO4_x	-0.007932
CAT_INAC	0.247002
CAT_OCUP	-0.017567
CH03	-0.039821
...	...
VII2_2	-0.012490
VII2_3	-0.182876
VII2_4	0.000000
Varon	0.000000
cte	0.000000

97 rows × 1 columns

En el caso del lamda óptimo hallado en el inciso anterior, las variables que son descartadas son:

- CH16: Dónde vivía hace 5 años
- H15: Entrevista individual realizada (sí/no)
- II3: Utiliza alguna habitación del hogar exclusivamente como lugar de trabajo (consultorio, estudio, taller, negocio, etc.)
- II4_1: Cuarto de cocina (sí/no)
- II6: De los lugares planteados en la pregunta 4 utiliza alguno exclusivamente como lugar de trabajo
- II6_1: No figura en el diccionario
- IV11: Cómo es el desague del baño
- IV12_1: La vivienda está ubicada cerca de un basural
- IV12_3: La vivienda está ubicada en villa de emergencia
- IX_MEN10: Cantidad de miembros del hogar menores de 10 años
- PP02C1: Hizo contactos, entrevistas (búsqueda laboral)
- PP02C3: Se presentó en establecimientos (búsqueda laboral)
- PP02C4: Hizo algo para ponerse por su cuenta (búsqueda laboral)
- PP02C5: Puso carteles en negocios, preguntó en el barrio (búsqueda laboral)
- PP02C8: De otra forma activa (búsqueda laboral)
- PP02H: En los últimos 12 meses, ¿buscó trabajo en algún momento?
- REALIZADA: Entrevista realizada (hogar respondió o no)
- REGION_x
- TRIMESTRE_x
- V15: En los últimos tres meses, las personas del hogar han vivido pedir préstamos a bancos, financieras, etc.
- V18: Tuvieron otros ingresos en efectivo (limosnas, juegos de azar, etc.)
- V19_A: ¿Menores de 10 años ayudan con algún dinero trabajando?
- V19_B: ¿Menores de 10 años ayudan con algún menores pidiendo?
- V3: En los últimos tres meses, las personas del hogar han vivido de indemnización por despido

- V4: En los últimos tres meses, las personas del hogar han vivido de seguro de desempleo
- V9: En los últimos tres meses, las personas del hogar han vivido ganancias de algún negocio en el que no trabajan
- VII2_4: Otras personas que ayudan en las tareas de la casa
- Varon: Variable armada por nosotras para poder adjuntar los valores de la tabla de equivalencias
- cte

En general, podemos observar que todas las variables que habíamos considerado como relevantes para evaluar los niveles de pobreza han sido conservadas por el modelo. Las que se han descartado tienen que ver con detalles más específicos acerca de las viviendas o los individuos, que no habíamos considerado como relevantes. Sin embargo, hay cuatro variables que nos llama la atención que no hayan sumado poder predictivo al modelo y hayan sido descartadas como, por ejemplo, si los menores de edad trabajan o ayudan pidiendo, si el hogar está cerca de un basural o si el hogar está en en un barrio de emergencia.

7)

Para responder este punto nos referiremos a todos los modelos de regresión logística que evaluamos en el inciso 5 de esta parte III del trabajo. En particular, aprovecharemos el hecho de que ambos métodos de regularización (Ridge y Lasso) obtuvieron el mismo lamda (lamda=1) como hiperparámetro óptimo. En este sentido, si observamos el ECM producido por cada una de los modelos de predicción, veremos que Lasso es el método de regularización que mejor funcionó: su ECM fue de 0.1980234, mientras que el de Ridge, fue de 0.199516.

8)

```
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\2125924564.py:50: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\2125924564.py:50: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\2125924564.py:50: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
C:\Users\pilih\AppData\Local\Temp\ipykernel_12256\2125924564.py:50: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  ecms = ecms.append({"grado": grado, "particion": i, "ECM": metricas.get
("ECM")}, ignore_index=True)
```

Para determinar cuál de todos los métodos es el que predice mejor, recurrimos a la función "evalua_multiples_metodos". Mediante esta, evaluamos diferentes métodos de predicción, tomando como configuración los hiperparámetros que en los incisos anteriores habíamos encontrado como los óptimos para el modelo de regresión logística.

Así, podemos observar que tanto el modelo de "Regresión Logística" (con Lamda igual a 1 y el método de Lasso) como el de "Análisis Discrimante Lineal" están muy cercanos con los valores de las medidas de precisión. Incluso, se puede señalar que coinciden en el valor del "Accuracy score". Sin embargo, si nos

guiáramos estrictamente por la medida del ECM, el modelo que convendría elegir dado que es el que mejor predice es el de Análisis Discriminante Lineal.

9)

```
. . .
```

```
. . .
```

```
Hogares pobres predichos: 1542.0
Hogares que no reportaron ingreso: 1542
Proporción de hogares pobres: 100.0 %
```

Vemos que la proporción de hogares pobres obtenida de nuestra predicción con análisis discriminante lineal es del 100%. Nos resulta extraño por lo que creemos que puede haber algún error previo en la estimación. Por lo tanto chequeamos el modelo con las x de entrenamiento (sabemos que está bien porque el modelo se construyó en base a eso), lo que nos da una proporción de pobres del 42,47% , más similar a la declarada por el INDEC.