

Tarea1_Cuevas_Reyes

September 26, 2022

1 Integrantes:

- Matías Cuevas Torres
- Pablo Reyes Polanco

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf
import sklearn
import scipy
from scipy.stats import nbinom
import seaborn as sns

%matplotlib inline
```

2 1. Carga de datos, inspección y filtro

```
[ ]: try:
    junaeb = pd.read_csv("../data/junaeb.csv")
except:
    junaeb = pd.read_csv("https://raw.githubusercontent.com/juancaros/LAB-MAA/
    ↪main/data/junaeb.csv")
```

Los datos se cargan directamente desde github por lo que para correr el código es necesario contar con conexión a internet

```
[ ]: junaeb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6607 entries, 0 to 6606
Data columns (total 10 columns):
#   Column              Non-Null Count  Dtype
---  -
0   vive_padre          6607 non-null   int64
1   vive_madre          6607 non-null   int64
2   n_personas          6472 non-null   float64
```

```

3  n_habitaciones      6457 non-null   float64
4  cercania_juegos     6475 non-null   float64
5  cercania_servicios  6475 non-null   float64
6  edad_primer_parto   6386 non-null   float64
7  area                6607 non-null   int64
8  educm               6607 non-null   int64
9  educp               6607 non-null   int64
dtypes: float64(5), int64(5)
memory usage: 516.3 KB

```

Para determinar los datos nulos de la base de datos se utiliza el comando `.info` describiendo la cantidad de no nulos y el tipo de dato que es para cada variable (entero y float).

```
[ ]: junaeb.describe()
```

```

[ ]:
      vive_padre  vive_madre  n_personas  n_habitaciones  cercania_juegos  \
count  6607.000000  6607.000000  6472.000000      6457.000000      6475.000000
mean    0.669593    0.949902    4.387361      2.583862      1.205405
std     0.470395    0.221607    1.344752      0.900471      0.459817
min     0.000000    0.000000    1.000000      0.000000      1.000000
25%     0.000000    1.000000    4.000000      2.000000      1.000000
50%     1.000000    1.000000    4.000000      2.000000      1.000000
75%     1.000000    1.000000    5.000000      3.000000      1.000000
max     1.000000    2.000000   16.000000     20.000000      4.000000

      cercania_servicios  edad_primer_parto      area      educm  \
count      6475.000000      6386.000000  6607.000000  6607.000000
mean         1.154286         22.220169    0.900863    12.314364
std          0.421625          5.193370    0.298869     4.329315
min          1.000000         10.000000    0.000000     0.000000
25%          1.000000         18.000000    1.000000    11.000000
50%          1.000000         21.000000    1.000000    13.000000
75%          1.000000         25.000000    1.000000    15.000000
max          4.000000         48.000000    1.000000    20.000000

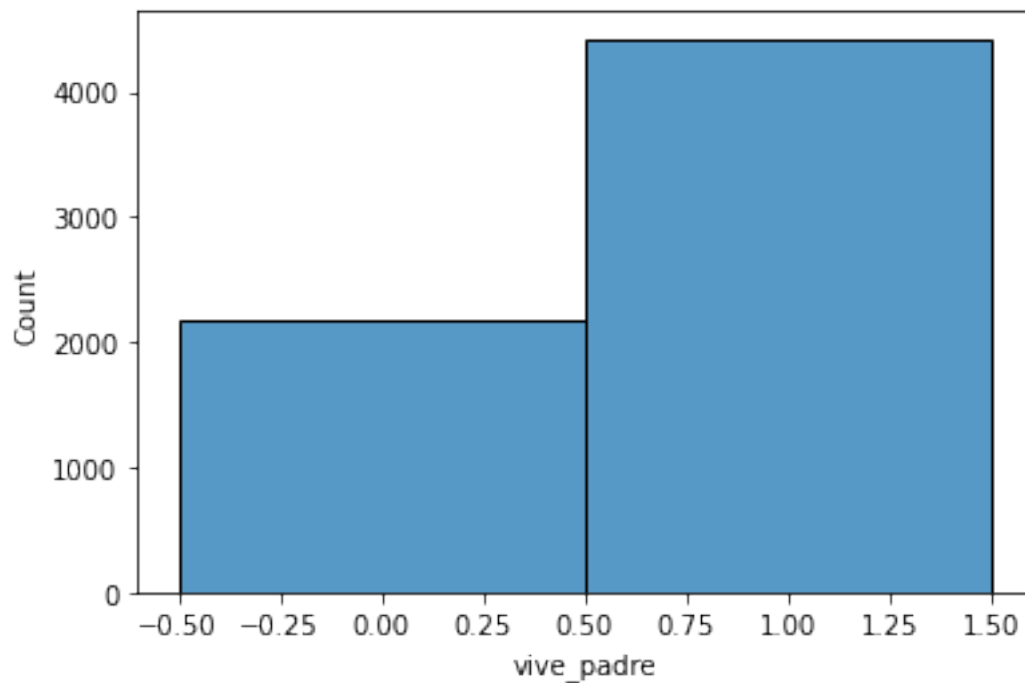
      educp
count  6607.000000
mean   10.917209
std     5.481642
min     0.000000
25%     9.000000
50%    13.000000
75%    13.000000
max    20.000000

```

Luego se realizó el calculo de estadísticas descriptivas para cada una de las variables considerando el tamaño de la muestra (count), promedio (mean), desviación estándar (STD), el valor mínimo (min), máximo (max), y los intervalos de confianza en 25%, 50% y 75%.

```
[ ]: sns.histplot(junaeb, x = "vive_padre", discrete=True)
junaeb.value_counts("vive_padre")
```

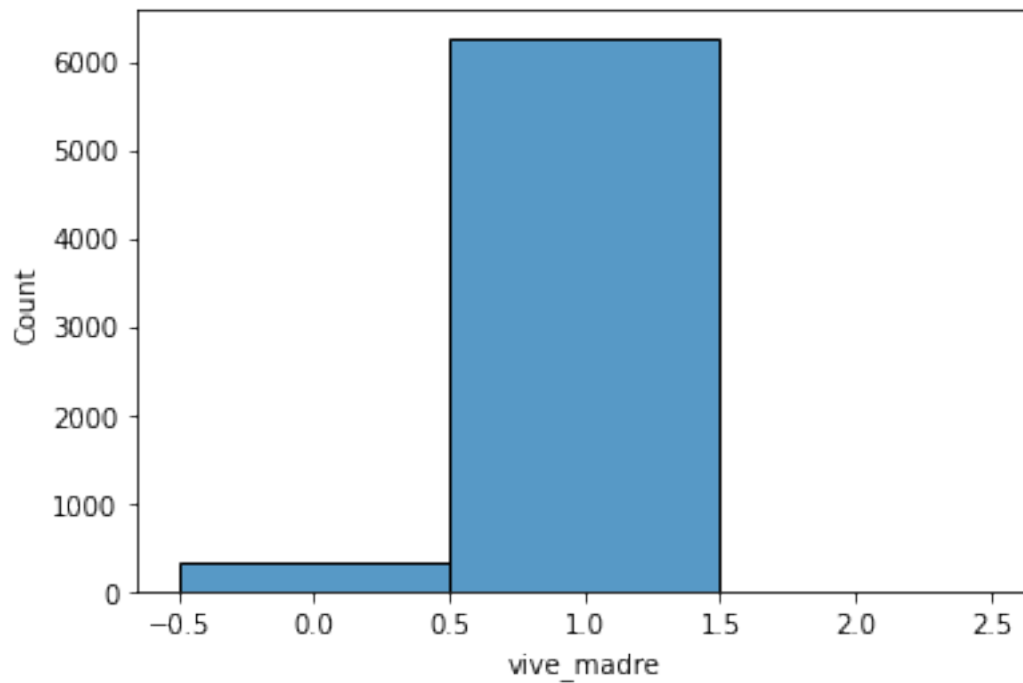
```
[ ]: vive_padre
1    4424
0    2183
dtype: int64
```



Vive_padre es una variable dummy que no tiene outliers, solo valores de 0 y 1

```
[ ]: var = "vive_madre"
sns.histplot(junaeb, x = var, discrete=True)
junaeb.value_counts(var)
```

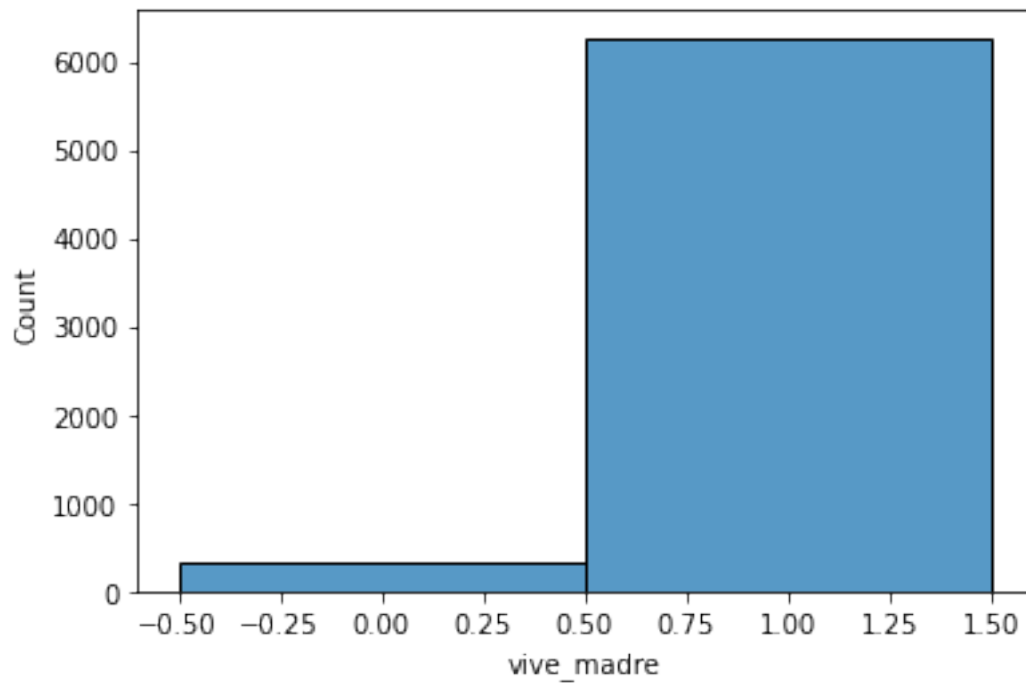
```
[ ]: vive_madre
1    6266
0    336
2      5
dtype: int64
```



en la variable vive_madre existen 5 datos con valor 2, lo que no tiene sentido ya que esa variable solo puede valer 0 o 1 por lo que se eliminan esos datos

```
[ ]: junaeb = junaeb[junaeb.vive_madre < 2]
      var = "vive_madre"
      sns.histplot(junaeb, x = var, discrete=True)
      junaeb.value_counts(var)
```

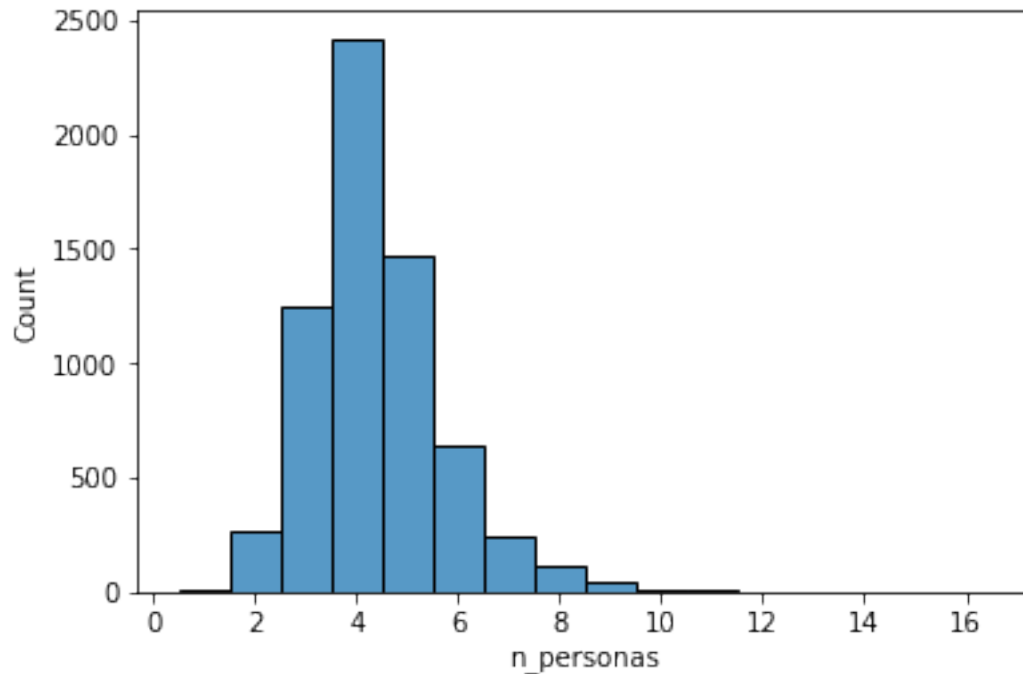
```
[ ]: vive_madre
      1    6266
      0     336
      dtype: int64
```



Así queda la variable vive_madre luego de la eliminación de dichos datos

```
[ ]: var = "n_personas"
sns.histplot(junaeb, x = var, discrete=True)
junaeb.value_counts(var)
```

```
[ ]: n_personas
4.0    2418
5.0    1466
3.0    1249
6.0     641
2.0     265
7.0     244
8.0     112
9.0      45
10.0      8
1.0       7
11.0      6
12.0      3
14.0      1
15.0      1
16.0      1
dtype: int64
```



para la variable n_personas que indica la cantidad de personas que componen los integrantes del hogar, existe datos con muy poca frecuencia por lo que se decidió quedarse con los datos que estuvieran dentro de un intervalo de confianza de 3 sigmas

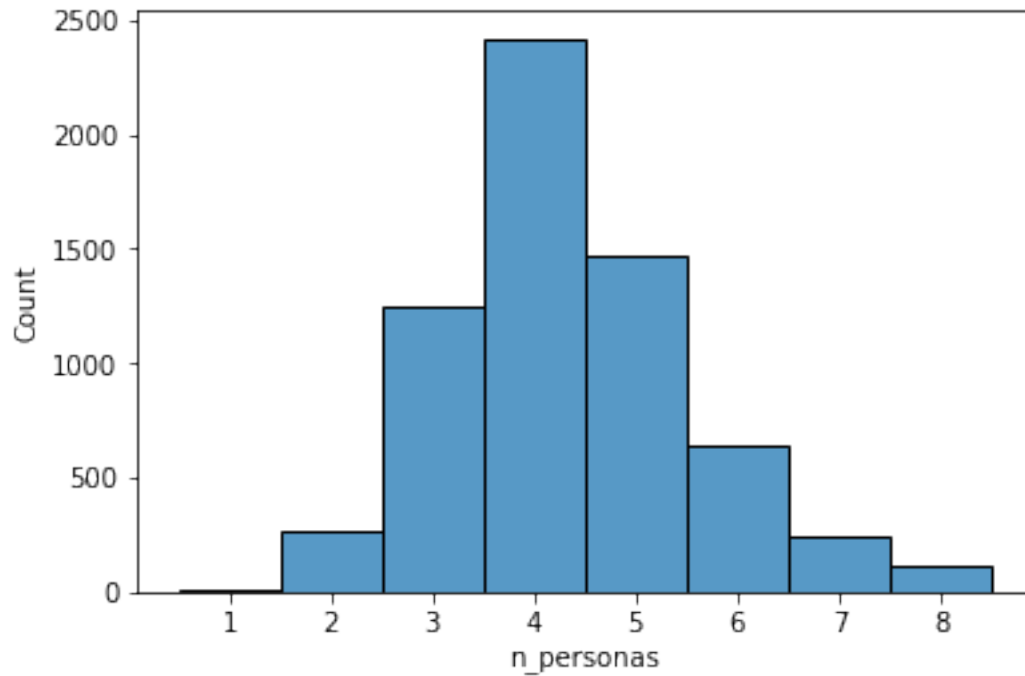
```
[ ]: var = "n_personas"

media = junaeb[var].mean()
std = junaeb[var].std()
mask = (junaeb[var] > media - 3*std) & (junaeb[var] < media + 3*std)
junaeb = junaeb[mask]

sns.histplot(junaeb, x = var, discrete=True)
junaeb.value_counts(var)

print(media-3*std, media, media+3*std)
```

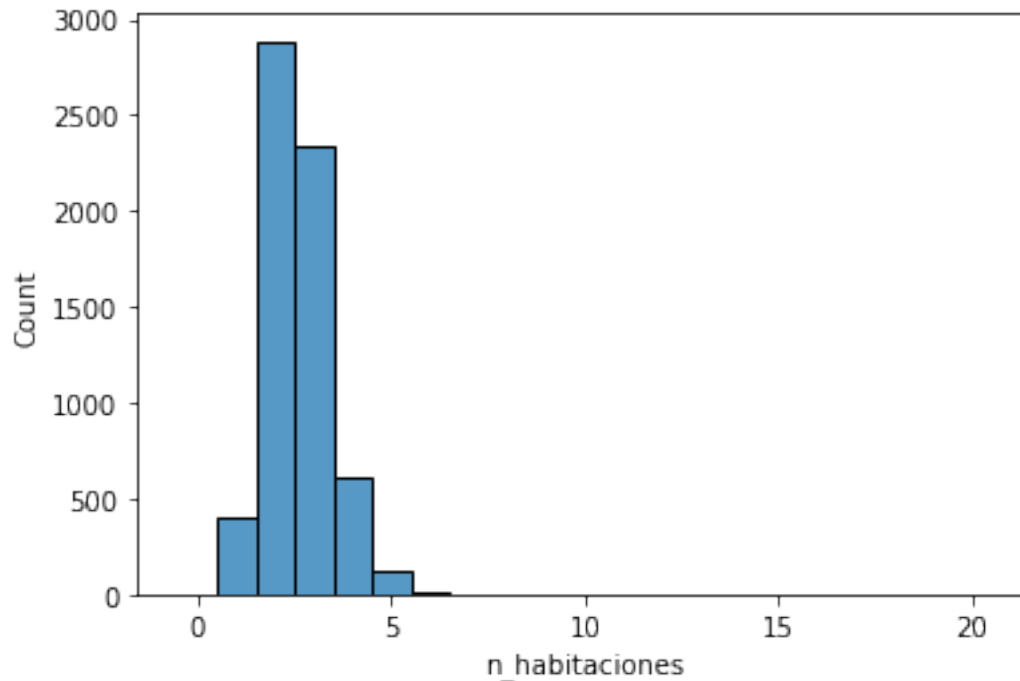
```
0.3532342555546739 4.386578011442709 8.419921767330745
```



las posibles valores que estan dentro del intervalo de confianza son: 1,2,3,4,5,6,7,8

```
[ ]: var = "n_habitaciones"
sns.histplot(junaeb, x = var, discrete=True)
junaeb.value_counts(var)
```

```
[ ]: n_habitaciones
2.0    2885
3.0    2341
4.0     618
1.0     398
5.0     119
6.0       19
7.0        2
0.0         1
8.0         1
9.0         1
10.0        1
20.0        1
dtype: int64
```



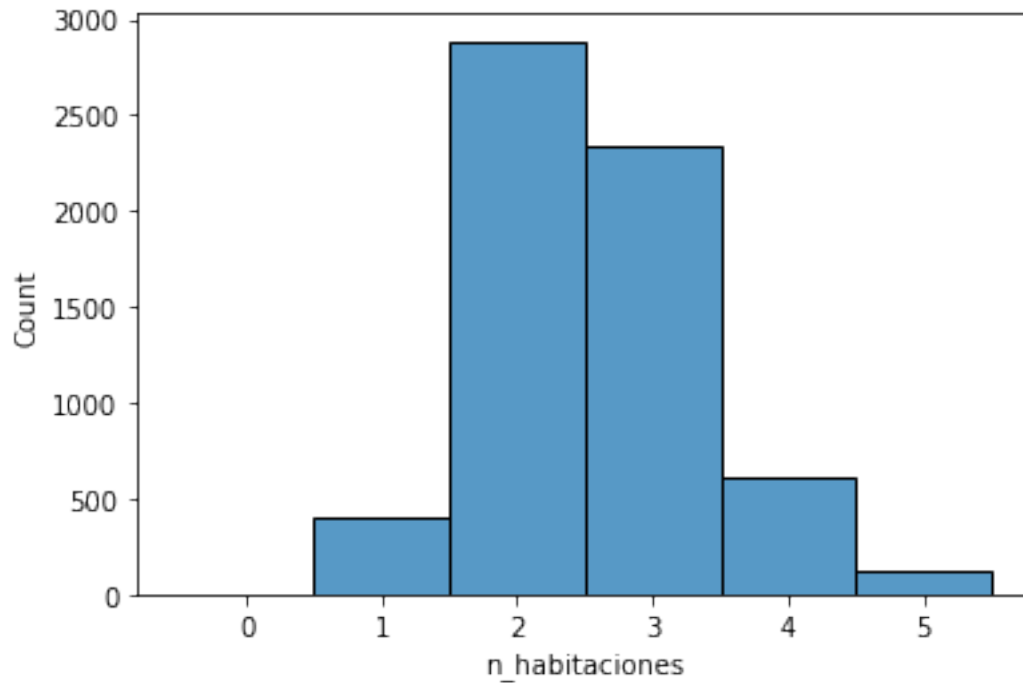
Para la variable n_habitaciones, se encuentra que puede tener outliers por lo que se eliminan aquellas filas que estén a más de 3 desviaciones estandar

```
[ ]: var = "n_habitaciones"

media = junaeb[var].mean()
std = junaeb[var].std()
mask = (junaeb[var] > media - 3*std) & (junaeb[var] < media + 3*std)
junaeb = junaeb[mask]

sns.histplot(junaeb, x = var, discrete=True)
junaeb.value_counts(var)
```

```
[ ]: n_habitaciones
2.0    2885
3.0    2341
4.0     618
1.0     398
5.0     119
0.0        1
dtype: int64
```

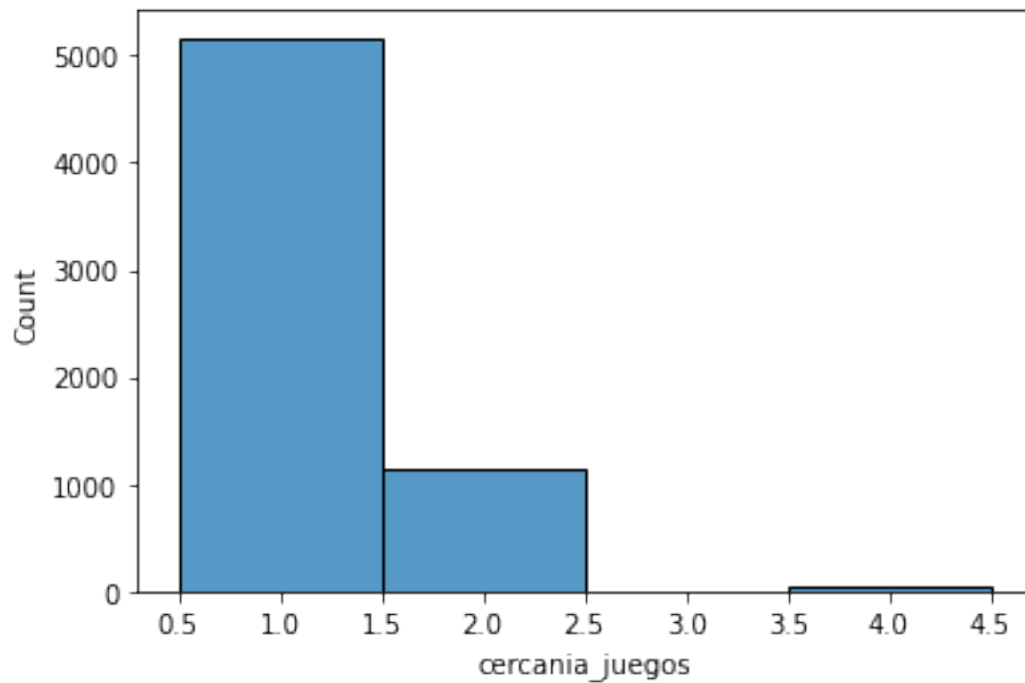



finalmente los posibles valores que puede tomar n_habitaciones son: 0,1,2,3,4,5

Cercanía_juegos tiene un código numérico para quienes respondieron no sabe/ no responde (toma valor 4 la variable), una primera aproximación será eliminar dichos valores

```
[ ]: var = "cercania_juegos"
sns.histplot(junaeb, x = var, discrete=True)
junaeb.value_counts(var)
```

```
[ ]: cercania_juegos
1.0    5163
2.0    1156
4.0      43
dtype: int64
```



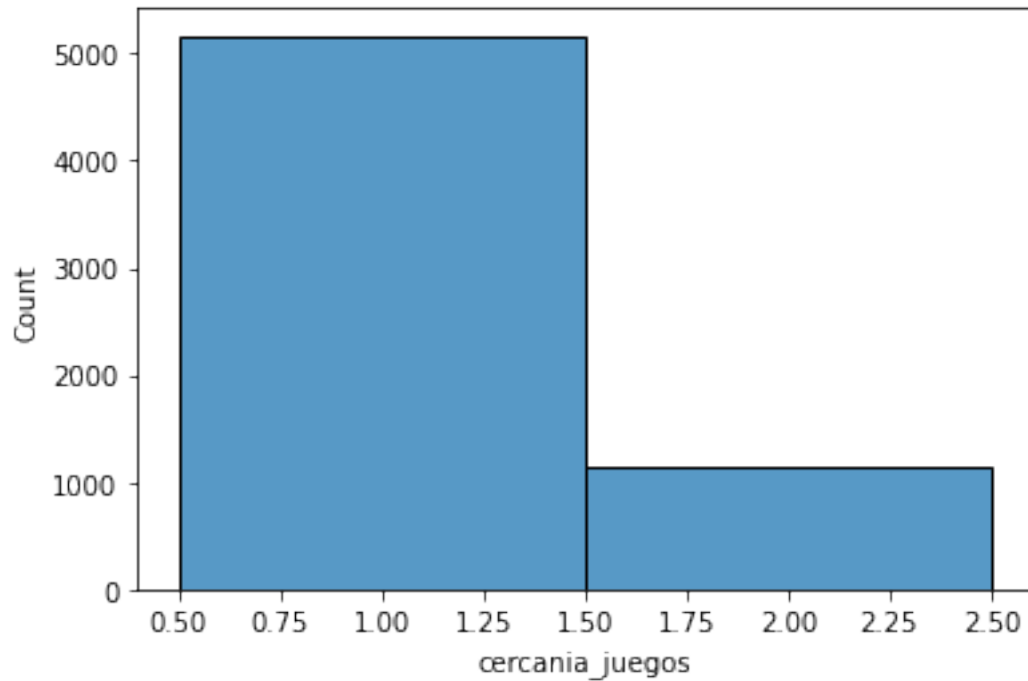
Se decide eliminar las respuestas no sabe/no responde de la muestra

```
[ ]: var = "cercania_juegos"

mask = (junaeb[var] < 3)
junaeb = junaeb[mask]

sns.histplot(junaeb, x = var, discrete=True)
junaeb.value_counts(var)
```

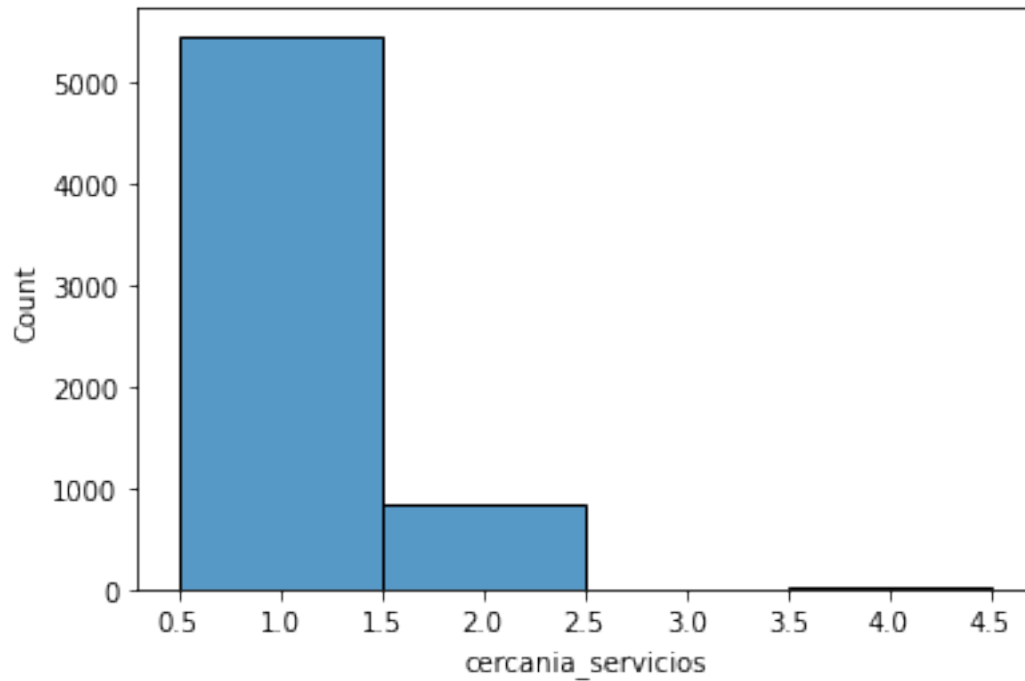
```
[ ]: cercania_juegos
1.0    5163
2.0    1156
dtype: int64
```



cercania_juegos queda como una variable binaria con 1 y 2 como posibles valores, como no tiene outliers, no se realiza ninguna modificacion en la variable

```
[ ]: var = "cercania_servicios"  
sns.histplot(junaeb, x = var, discrete=True)  
junaeb.value_counts(var)
```

```
[ ]: cercania_servicios  
1.0    5463  
2.0     833  
4.0      23  
dtype: int64
```



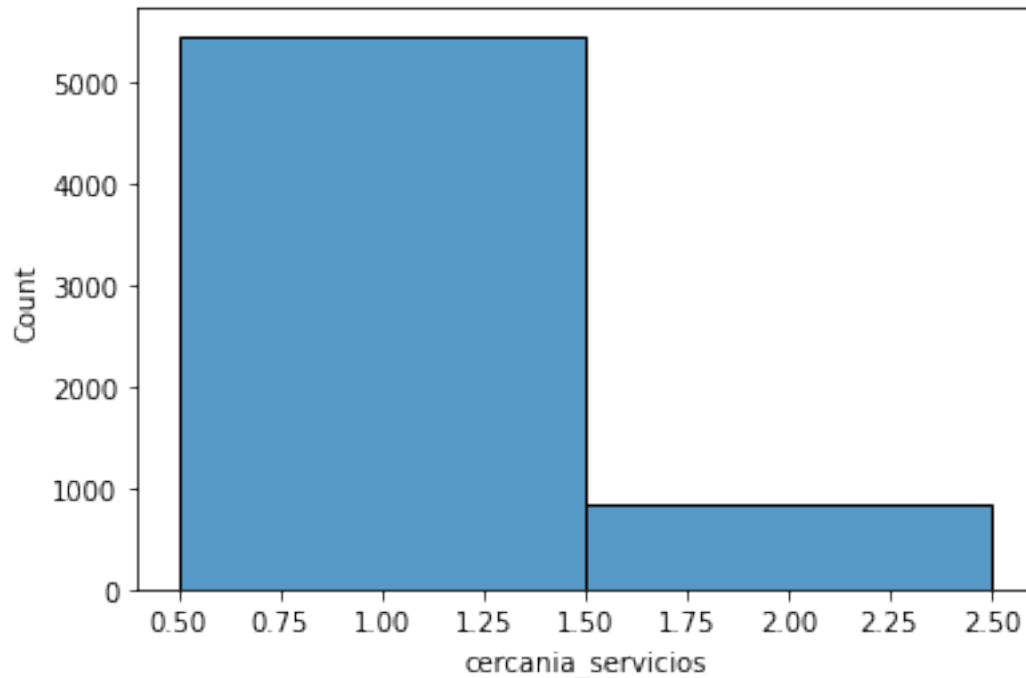
Cercanía_servicios tiene un código numérico para quienes respondieron no sabe/ no responde (toma valor 4 la variable), una primera aproximación será eliminar dichos valores

```
[ ]: var = "cercania_servicios"

mask = (junaeb[var] < 3)
junaeb = junaeb[mask]

sns.histplot(junaeb, x = var, discrete=True)
junaeb.value_counts(var)
```

```
[ ]: cercania_servicios
1.0    5463
2.0     833
dtype: int64
```



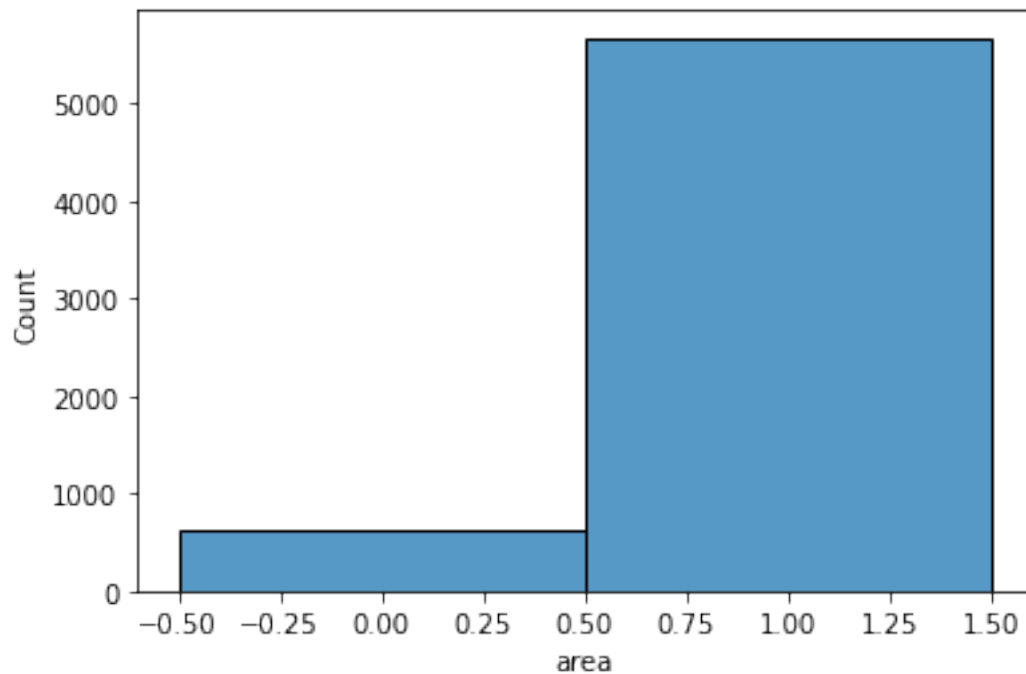
Se decide eliminar las respuestas no sabe/no responde de la muestra

Se decide eliminar la variable edad_primer_parto, ya que aquellos valores que son nulos podrían ser originados por casos en que no se sepa sobre el paradero de la madre, o adopciones, por lo que se cree podría sesgar los datos.

```
[ ]: junaeb = junaeb.drop("edad_primer_parto", axis = 1).dropna() # si se elimina la
    ↪columna respecto al primer parto y luego los na
```

```
[ ]: var = "area"
sns.histplot(junaeb, x = var, discrete=True)
junaeb.value_counts(var)
```

```
[ ]: area
1    5669
0     627
dtype: int64
```



la variable area es una variable binaria sin outlier por lo que no se modifica

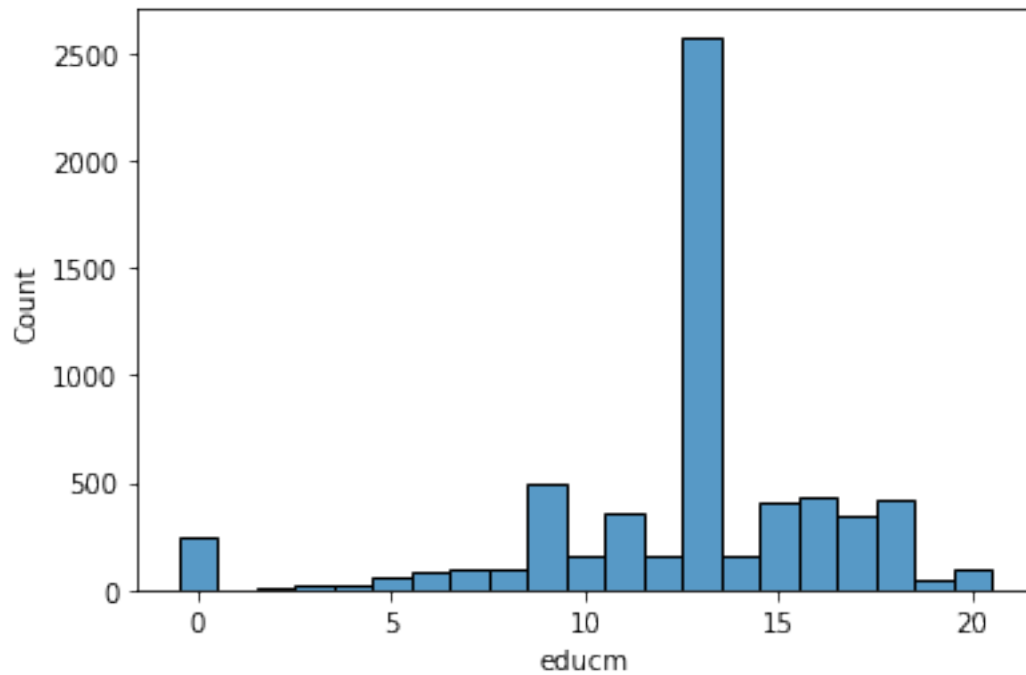
```
[ ]: var = "educm"
sns.histplot(junaeb, x = var, discrete=True)
junaeb.value_counts(var)
```

```
[ ]: educm
13    2578
9      498
16     436
18     419
15     405
11     363
17     342
0      249
12     165
10     160
14     158
20      98
7       97
8       92
6       79
5       63
19      47
3       20
```

```

4      19
2       8
dtype: int64

```



```

[ ]: var = "educp"
     sns.histplot(junaeb, x = var, discrete=True)
     junaeb.value_counts(var)

```

```

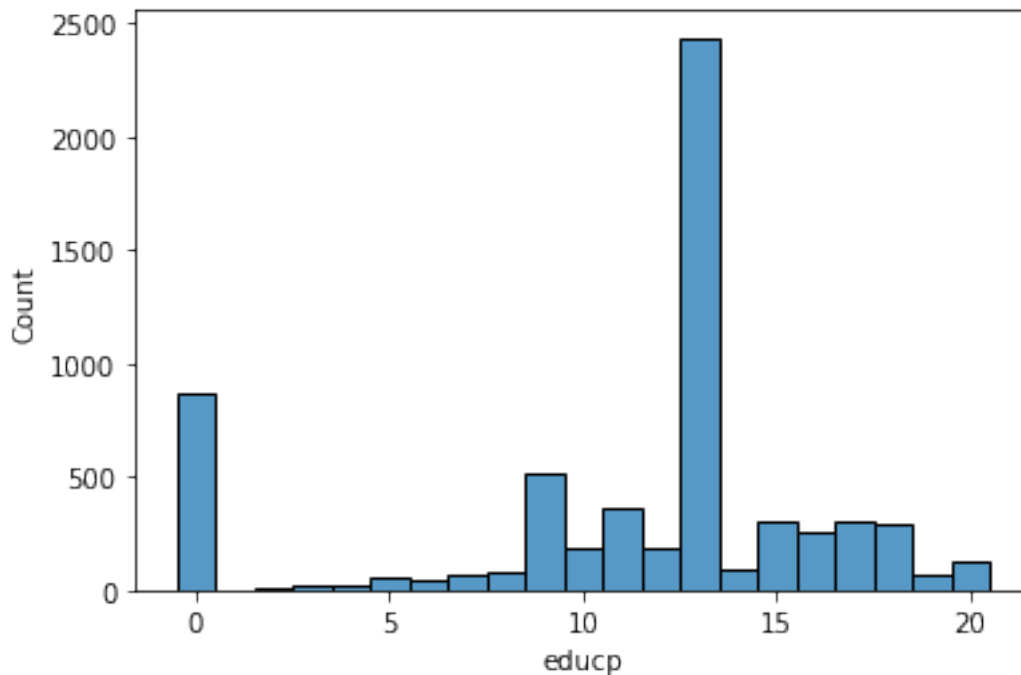
[ ]: educp
13    2436
0      865
9      519
11     359
15     303
17     303
18     290
16     262
10     188
12     180
20     123
14      92
8       79
7       73
19      67
5       56

```

```

6      49
3      25
4      21
2       6
dtype: int64

```




```
[ ]: junaeb.info()
```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 6296 entries, 0 to 6474
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   vive_padre            6296 non-null   int64
1   vive_madre            6296 non-null   int64
2   n_personas            6296 non-null   float64
3   n_habitaciones        6296 non-null   float64
4   cercania_juegos       6296 non-null   float64
5   cercania_servicios    6296 non-null   float64
6   area                  6296 non-null   int64
7   educm                 6296 non-null   int64
8   educp                 6296 non-null   int64
dtypes: float64(4), int64(5)
memory usage: 620.9 KB

```


finalmente las variables a utilizar en el modelo son  vive_padre como variable dependiente y las variables independientes serán vive_madre, n_personas, n_habitaciones, cercania_juegos, cercania_servicios, area, educm y educp

Además luego de la limpieza la cantidad de datos no nulos se corresponde con la cantidad de entradas en la base de datos

3 2. Modelo Lineal: Mínimos Cuadrados Ordinarios (MCO)

```
[ ]: y=junaeb['vive_padre']
X=junaeb.drop('vive_padre', axis = 1)
X=sm.add_constant(X)
model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```

                                OLS Regression Results
=====
Dep. Variable:                vive_padre    R-squared:                0.167
Model:                        OLS          Adj. R-squared:         0.166
Method:                      Least Squares  F-statistic:              158.0
Date:                        Wed, 14 Sep 2022  Prob (F-statistic):    1.94e-243
Time:                        03:56:59       Log-Likelihood:           -3530.9
No. Observations:            6296          AIC:                    7080.
Df Residuals:                6287          BIC:                    7141.
Df Model:                    8
Covariance Type:             nonrobust
=====
=====
                                coef    std err          t      P>|t|      [0.025
-----
0.975]
-----
const                0.2656      0.050      5.304      0.000      0.167
0.364
vive_madre           0.1358      0.032      4.245      0.000      0.073
0.198
n_personas           0.0582      0.005     11.718      0.000      0.048
0.068
n_habitaciones       -0.0411      0.007     -5.581      0.000     -0.056
-0.027
cercania_juegos      -0.0020      0.015     -0.134      0.894     -0.032
0.028
cercania_servicios    0.0042      0.017      0.245      0.807     -0.029
0.038
area                 -0.0723      0.019     -3.875      0.000     -0.109
-0.036

```

educm	-0.0148	0.001	-10.115	0.000	-0.018
-0.012					
educp	0.0348	0.001	32.529	0.000	0.033
0.037					

```
=====
```

Omnibus:	739.509	Durbin-Watson:	1.992
Prob(Omnibus):	0.000	Jarque-Bera (JB):	824.270
Skew:	-0.840	Prob(JB):	1.03e-179
Kurtosis:	2.436	Cond. No.	189.

```
=====
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142:

FutureWarning: In a future version of pandas all arguments of concat except for the argument 'objs' will be keyword-only

```
x = pd.concat(x[:, :order], 1)
```

Corriendo el OLS por primera vez, se puede apreciar que segun la prueba t de significancia individual las variables cercania_juegos y cercania_servicios, segun su valor p, no se puede asegurar que sean distintas de 0

Por otro lado las variables que influyen positivamente la variable dependiente Y son: vive_madre, n_personas, Cercania_servicios, Educop

mientras que las que la afectan negativamente son n_habitaciones, cercania_uegos, area y educm

```
[ ]: y = junaeb['vive_padre']
X = junaeb.drop(['vive_padre', "cercania_juegos", "cercania_servicios"], axis = 1)
X = sm.add_constant(X)
model = sm.OLS(y, X)
OLS_model = model.fit()
print(OLS_model.summary())
```

OLS Regression Results

```
=====
```

Dep. Variable:	vive_padre	R-squared:	0.167
Model:	OLS	Adj. R-squared:	0.167
Method:	Least Squares	F-statistic:	210.7
Date:	Wed, 14 Sep 2022	Prob (F-statistic):	9.47e-246
Time:	03:56:59	Log-Likelihood:	-3531.0
No. Observations:	6296	AIC:	7076.
Df Residuals:	6289	BIC:	7123.
Df Model:	6		
Covariance Type:	nonrobust		

```
=====
```

	coef	std err	t	P> t	[0.025

--					
const	0.2684	0.042	6.439	0.000	0.187
0.350					
vive_madre	0.1359	0.032	4.250	0.000	0.073
0.199					
n_personas	0.0583	0.005	11.742	0.000	0.049
0.068					
n_habitaciones	-0.0412	0.007	-5.603	0.000	-0.056
-0.027					
area	-0.0726	0.018	-4.036	0.000	-0.108
-0.037					
educm	-0.0148	0.001	-10.139	0.000	-0.018
-0.012					
educp	0.0348	0.001	32.647	0.000	0.033
0.037					
=====					
Omnibus:		739.793	Durbin-Watson:		1.992
Prob(Omnibus):		0.000	Jarque-Bera (JB):		824.422
Skew:		-0.840	Prob(JB):		9.53e-180
Kurtosis:		2.435	Cond. No.		167.
=====					

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142:

FutureWarning: In a future version of pandas all arguments of concat except for the argument 'objs' will be keyword-only

```
x = pd.concat(x[:, :order], 1)
```

```
[ ]: comp = junaeb['vive_padre'].to_frame()
comp["estimado"] = OLS_model.predict(X).to_frame()
comp["binario_estimado"] = pd.cut(comp["estimado"], bins = [-1000, 0.5, 1000],
↳ labels = [0,1])

prop_correctas = len(comp[comp["binario_estimado"] == comp["vive_padre"]]) /
↳ len(comp)
print(f"porentaje de clasificaciones correcttas {prop_correctas * 100 :.2f}%")
```

porentaje de clasificaciones correcttas 77.00%

4 3. Modelo Probit

```
[ ]: y = junaeb['vive_padre']
X = junaeb.drop(['vive_padre'], axis = 1)
X = sm.add_constant(X)
model = sm.Probit(y, X)
probit_model = model.fit()
print(probit_model.summary())

mfx = probit_model.get_margeff()
print(mfx.summary())
```

Optimization terminated successfully.

Current function value: 0.541123

Iterations 5

/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142:

FutureWarning: In a future version of pandas all arguments of concat except for the argument 'objs' will be keyword-only

x = pd.concat(x[:, :order], 1)

Probit Regression Results

```
=====
Dep. Variable:          vive_padre  No. Observations:          6296
Model:                  Probit      Df Residuals:          6287
Method:                  MLE        Df Model:              8
Date:                   Wed, 14 Sep 2022  Pseudo R-squ.:          0.1318
Time:                   03:56:59          Log-Likelihood:         -3406.9
converged:               True          LL-Null:              -3924.0
Covariance Type:         nonrobust      LLR p-value:           5.968e-218
=====
=====
              coef      std err          z      P>|z|      [0.025
0.975]
-----
const          -0.6627      0.162     -4.101      0.000     -0.979
-0.346
vive_madre       0.3871      0.102      3.809      0.000       0.188
0.586
n_personas       0.1790      0.016     11.010      0.000       0.147
0.211
n_habitaciones  -0.1243      0.024     -5.184      0.000     -0.171
-0.077
cercania_juegos -0.0053      0.049     -0.108      0.914     -0.102
0.092
cercania_servicios  0.0049      0.056      0.087      0.930     -0.105
0.115
area           -0.2153      0.062     -3.481      0.001     -0.337
```

```

-0.094
educm          -0.0461      0.005      -9.454      0.000      -0.056
-0.037
educp          0.1006      0.004      27.832      0.000      0.094
0.108
=====
=====
                Probit Marginal Effects
=====
Dep. Variable:          vive_padre
Method:                dydx
At:                    overall
=====
=====
                dy/dx      std err      z      P>|z|      [0.025
0.975]
-----
-----
vive_madre          0.1191      0.031      3.819      0.000      0.058
0.180
n_personas          0.0551      0.005      11.267      0.000      0.045
0.065
n_habitaciones     -0.0382      0.007      -5.213      0.000      -0.053
-0.024
cercania_juegos    -0.0016      0.015      -0.108      0.914      -0.031
0.028
cercania_servicios  0.0015      0.017      0.087      0.930      -0.032
0.035
area               -0.0662      0.019      -3.489      0.000      -0.103
-0.029
educm               -0.0142      0.001      -9.602      0.000      -0.017
-0.011
educp               0.0309      0.001      33.922      0.000      0.029
0.033
=====
=====

```

```

[ ]: y = junaeb['vive_padre']
X = junaeb.drop(['vive_padre', "cercania_juegos", "cercania_servicios"], axis = 1)
X = sm.add_constant(X)
model = sm.Probit(y, X)
probit_model = model.fit()
print(probit_model.summary())

mfx = probit_model.get_margeff()
print(mfx.summary())

```

Optimization terminated successfully.

Current function value: 0.541124

Iterations 5

/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142:

FutureWarning: In a future version of pandas all arguments of concat except for the argument 'objs' will be keyword-only

x = pd.concat(x[:, :order], 1)

Probit Regression Results

```
=====
Dep. Variable:          vive_padre    No. Observations:          6296
Model:                  Probit        Df Residuals:              6289
Method:                 MLE          Df Model:                  6
Date:                  Wed, 14 Sep 2022    Pseudo R-squ.:          0.1318
Time:                  03:57:00          Log-Likelihood:         -3406.9
converged:              True            LL-Null:              -3924.0
Covariance Type:        nonrobust        LLR p-value:            3.481e-220
=====
```

==

	coef	std err	z	P> z	[0.025
0.975]					

--					
const	-0.6640	0.134	-4.944	0.000	-0.927
-0.401					
vive_madre	0.3873	0.102	3.811	0.000	0.188
0.586					
n_personas	0.1790	0.016	11.021	0.000	0.147
0.211					
n_habitaciones	-0.1243	0.024	-5.196	0.000	-0.171
-0.077					
area	-0.2147	0.060	-3.597	0.000	-0.332
-0.098					
educm	-0.0461	0.005	-9.468	0.000	-0.056
-0.037					
educp	0.1006	0.004	27.915	0.000	0.094
0.108					

==

Probit Marginal Effects

```
=====
Dep. Variable:          vive_padre
Method:                 dydx
At:                     overall
=====
```

==

	dy/dx	std err	z	P> z	[0.025
0.975]					

```

-----
--
vive_padre      0.1191      0.031      3.821      0.000      0.058
0.180
n_personas      0.0550      0.005     11.280      0.000      0.045
0.065
n_habitaciones -0.0382      0.007     -5.225      0.000     -0.053
-0.024
area            -0.0660      0.018     -3.606      0.000     -0.102
-0.030
educm           -0.0142      0.001     -9.616      0.000     -0.017
-0.011
educp           0.0309      0.001     34.067      0.000      0.029
0.033
=====
==

```

```

[ ]: comp = junaeb['vive_padre'].to_frame()
comp["estimado"] = probit_model.predict(X).to_frame()
comp["binario_estimado"] = pd.cut(comp["estimado"], bins = [-1000, 0.5, 1000],
    ↪ labels = [0,1])

prop_correctas = len(comp[comp["binario_estimado"] == comp["vive_padre"]]) /
    ↪ len(comp)
print(f"porentaje de clasificaciones correcttas {prop_correctas * 100 :.2f}%")

```

porentaje de clasificaciones correcttas 77.02%

5 4. Modelo Logit

```

[ ]: y = junaeb['vive_padre']
X = junaeb.drop(['vive_padre'], axis = 1)
X = sm.add_constant(X)

model = sm.Logit(y, X)
logit_model = model.fit()
print(logit_model.summary())

mfx = logit_model.get_margeff()
print(mfx.summary())

```

```

/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
    x = pd.concat(x[:,::order], 1)

Optimization terminated successfully.

```

Current function value: 0.539055

Iterations 6

Logit Regression Results

```
=====
Dep. Variable:      vive_padre    No. Observations:      6296
Model:              Logit         Df Residuals:              6287
Method:             MLE          Df Model:                  8
Date:               Wed, 14 Sep 2022    Pseudo R-squ.:          0.1351
Time:               03:57:00          Log-Likelihood:          -3393.9
converged:          True            LL-Null:              -3924.0
Covariance Type:    nonrobust        LLR p-value:            1.425e-223
=====
```

```
=====
              coef      std err          z      P>|z|      [0.025
0.975]
-----
const          -1.1446      0.275      -4.168      0.000      -1.683
-0.606
vive_madre       0.6922      0.169       4.098      0.000       0.361
1.023
n_personas       0.3168      0.029     10.884      0.000       0.260
0.374
n_habitaciones  -0.2117      0.041     -5.108      0.000      -0.293
-0.130
cercania_juegos -0.0155      0.084     -0.185      0.853      -0.180
0.149
cercania_servicios 0.0086      0.095       0.091      0.928      -0.177
0.194
area            -0.3693      0.106     -3.473      0.001      -0.578
-0.161
educm           -0.0842      0.009     -9.580      0.000      -0.101
-0.067
educp            0.1715      0.006     26.735      0.000       0.159
0.184
=====
```

Logit Marginal Effects

```
=====
Dep. Variable:      vive_padre
Method:             dydx
At:                 overall
=====
```

```
=====
              dy/dx      std err          z      P>|z|      [0.025
0.975]
-----
```


vive_madre	0.1239	0.030	4.116	0.000	0.065
0.183					
n_personas	0.0567	0.005	11.201	0.000	0.047
0.067					
n_habitaciones	-0.0379	0.007	-5.141	0.000	-0.052
-0.023					
cercania_juegos	-0.0028	0.015	-0.185	0.853	-0.032
0.027					
cercania_servicios	0.0015	0.017	0.091	0.928	-0.032
0.035					
area	-0.0661	0.019	-3.483	0.000	-0.103
-0.029					
educm	-0.0151	0.002	-9.802	0.000	-0.018
-0.012					
educp	0.0307	0.001	34.057	0.000	0.029
0.032					

=====

=====

```
[ ]: y = junaeb['vive_padre']
X = junaeb.drop(['vive_padre', "cercania_juegos", "cercania_servicios"], axis = 1)
X = sm.add_constant(X)

model = sm.Logit(y, X)
logit_model = model.fit()
print(logit_model.summary())

mfx = logit_model.get_margeff()
print(mfx.summary())
```

Optimization terminated successfully.

Current function value: 0.539058

Iterations 6

/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142:

FutureWarning: In a future version of pandas all arguments of concat except for the argument 'objs' will be keyword-only

```
x = pd.concat(x[:, :order], 1)
```

Logit Regression Results

Dep. Variable:	vive_padre	No. Observations:	6296
Model:	Logit	Df Residuals:	6289
Method:	MLE	Df Model:	6
Date:	Wed, 14 Sep 2022	Pseudo R-squ.:	0.1351
Time:	03:57:00	Log-Likelihood:	-3393.9
converged:	True	LL-Null:	-3924.0

```

Covariance Type:          nonrobust    LLR p-value:          8.194e-226
=====
==
                                coef      std err          z      P>|z|      [0.025
0.975]
-----
--
const          -1.1564      0.228      -5.065      0.000      -1.604
-0.709
vive_madre      0.6925      0.169       4.101      0.000       0.361
1.023
n_personas      0.3167      0.029     10.894      0.000       0.260
0.374
n_habitaciones -0.2115      0.041     -5.114      0.000      -0.293
-0.130
area           -0.3663      0.103     -3.567      0.000      -0.568
-0.165
educm          -0.0842      0.009     -9.592      0.000      -0.101
-0.067
educp          0.1715      0.006     26.817      0.000       0.159
0.184
=====
==
                Logit Marginal Effects
=====
Dep. Variable:          vive_padre
Method:                  dydx
At:                      overall
=====
==
                                dy/dx      std err          z      P>|z|      [0.025
0.975]
-----
--
vive_madre      0.1240      0.030       4.119      0.000       0.065
0.183
n_personas      0.0567      0.005     11.213      0.000       0.047
0.067
n_habitaciones -0.0379      0.007     -5.148      0.000      -0.052
-0.023
area           -0.0656      0.018     -3.579      0.000      -0.101
-0.030
educm          -0.0151      0.002     -9.815      0.000      -0.018
-0.012
educp          0.0307      0.001     34.224      0.000       0.029
0.032
=====
==

```

```
[ ]: comp = junaeb['vive_padre'].to_frame()
comp["estimado"] = logit_model.predict(X).to_frame()
comp["binario_estimado"] = pd.cut(comp["estimado"], bins = [-1000, 0.5, 1000],
↳labels = [0,1])

prop_correctas = len(comp[comp["binario_estimado"] == comp["vive_padre"]]) /
↳len(comp)
print(f"porcentaje de clasificaciones correctas {prop_correctas * 100 :.2f}%")
```

porcentaje de clasificaciones correctas 77.13%


6 5. Comentarios

2. Corriendo el OLS por primera vez, se puede apreciar que segun la prueba t de significancia individual las variables cercania_juegos y cercania_servicios, segun su valor p, no se puede asegurar que sean distintas de 0, por lo que se decidieron sacar del modelo. al correrlo por segunda vez sin las variables antes mencionadas resultó con que todas son significativas y que aquellas variables que influyen positivamente la variable dependiente son:

vive_madre: indica que en vive_padre cambia en 0.1359 cuando aumenta en una unidad vive madre

n_personas: indica que en vive_padre cambia en 0.0583 cuando aumenta en una unidad n_personas

Educp: indica que en vive_padre cambia en 0.0348 cuando aumenta en una unidad educp

mientras que las que la afectan negativamente son: n_habitaciones: indica que en vive_padre cambia en 0.1359 cuando aumenta en una  unidad vive madre

cercania_juegos: indica que en vive_padre cambia en 0.1359 cuando aumenta en una unidad vive madre

area: indica que en vive_padre cambia en 0.1359 cuando aumenta en una unidad vive madre

educm: indica que en vive_padre cambia en 0.1359 cuando aumenta en una unidad vive madre

Sin embargo al tener la regresión lineal cuyo resultado no esta restringido entre 0 y 1, podría no ser el modelo mas adecuado para este tipo de estimación y los metodo no lineales podrían resultar mejor.

Respecto a como interpretar los parámetros de la regresión, dado que ninguno está en un logaritmo, simplemente indica que ante un cambio de una unidad en la variable explicativa la variable explicada cambia en la medida que indica el parametro, con todo el resto constante.

Al correr los modelos de probit y logit los valores obtenidos de log-likelihood, para el probit (-3406.9) son menores a los de logit (-3393.9), y debido que a mayor log-likelihood implica un mejor ajuste al modelo, el logit funciona mejor para este caso. Cabe destacar que tanto el modelo Probit como el modelo Logit tienen una mejor especificación que el modelo estimado con OLS, dado que tienen un likelihood mayor a este (-3531.0). Ademas mencionar que para realizar este test los modelos deben tener la misma cantidad de variables independientes (se cumple)

En síntesis para este caso el modelo logit es el que funciona mejor ya que está mejor especificado según lo descrito en la prueba de log-likelihood. (no existe mucha diferencia con el probit).

En el caso de la estimación probit y logit, los parámetros no pueden ser interpretados en forma tan simple en comparación al OLS, por lo que se calculan los efectos marginales, estos indican en qué medida aumenta la probabilidad de que la variable explicada sea 1, ante un cambio en la variable explicativa correspondiente, con el resto constante.

[]:

7 6. Poisson

```
[ ]: y=junaeb['n_personas']
X = junaeb.drop(['n_personas'], axis = 1)

poisson=sm.GLM(y,X,family=sm.families.Poisson()).fit()
print(poisson.summary())

print("fitted lambda")
print(poisson.mu)
```

Generalized Linear Model Regression Results

```
=====
Dep. Variable:          n_personas    No. Observations:          6296
Model:                  GLM          Df Residuals:              6288
Model Family:           Poisson      Df Model:                  7
Link Function:          log          Scale:                    1.0000
Method:                 IRLS         Log-Likelihood:          -11345.
Date:                   Wed, 14 Sep 2022    Deviance:                1881.7
Time:                   03:57:00           Pearson chi2:            2.01e+03
No. Iterations:         4
Covariance Type:        nonrobust
=====
```

	coef	std err	z	P> z	[0.025
0.975]					
vive_padre	0.1114	0.014	7.740	0.000	0.083
0.140					
vive_madre	0.4628	0.034	13.751	0.000	0.397
0.529					
n_habitaciones	0.1954	0.007	29.145	0.000	0.182
0.209					
cercania_juegos	0.1011	0.016	6.267	0.000	0.069
0.133					
cercania_servicios	0.1231	0.018	6.862	0.000	0.088

0.158					
area	0.1551	0.020	7.614	0.000	0.115
0.195					
educm	0.0009	0.002	0.545	0.586	-0.002
0.004					
educp	0.0002	0.001	0.119	0.905	-0.002
0.003					

=====

=====

fitted lambda

[5.07173229 4.22906281 4.7259875 ... 3.47756338 3.88822163 4.2413198]

```
[ ]: y=junaeb['n_personas']
X = junaeb.drop(['n_personas', "educm", "educp"], axis = 1)

poisson=sm.GLM(y,X,family=sm.families.Poisson()).fit()
print(poisson.summary())

print("fitted lambda")
print(poisson.mu)
```

Generalized Linear Model Regression Results

Dep. Variable:	n_personas	No. Observations:	6296
Model:	GLM	Df Residuals:	6290
Model Family:	Poisson	Df Model:	5
Link Function:	log	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-11345.
Date:	Wed, 14 Sep 2022	Deviance:	1882.1
Time:	03:57:00	Pearson chi2:	2.01e+03
No. Iterations:	4		
Covariance Type:	nonrobust		

=====

	coef	std err	z	P> z	[0.025
0.975]					

vive_padre	0.1121	0.013	8.423	0.000	0.086
0.138					
vive_madre	0.4713	0.031	15.252	0.000	0.411
0.532					
n_habitaciones	0.1959	0.007	29.505	0.000	0.183
0.209					
cercania_juegos	0.1015	0.016	6.303	0.000	0.070
0.133					
cercania_servicios	0.1230	0.018	6.859	0.000	0.088
0.158					

```
area          0.1577      0.020      7.912      0.000      0.119
0.197
```

```
=====
=====
```

```
fitted lambda
[5.14120256 4.22643736 4.72759468 ... 3.47443473 3.88642201 4.22643736]
```

8 7. Dispersión

8.0.1 no tiene sentido decir que hay sobredispersión ni utilizar un $\alpha < 0$ como valor para la binomial negativa

```
[ ]: aux=((y-poisson.mu)**2-poisson.mu)/poisson.mu
      auxr = sm.OLS(aux,poisson.mu).fit()
      print(auxr.params)
```

```
x1      -0.156321
dtype: float64
```

Como el α estimado es menor a 1 no se utiliza para el modelo binomial negativo, además este es indicio de que no hay sobredispersión en los datos

9 8. Binomial Negativa

```
[ ]: negbin=sm.GLM(y,X,family=sm.families.NegativeBinomial()).fit()
      print(negbin.summary())
```

```

              Generalized Linear Model Regression Results
=====
Dep. Variable:          n_personas      No. Observations:          6296
Model:                  GLM            Df Residuals:              6290
Model Family:          NegativeBinomial  Df Model:                  5
Link Function:          log             Scale:                    1.0000
Method:                 IRLS            Log-Likelihood:          -16178.
Date:                  Wed, 14 Sep 2022  Deviance:                 362.53
Time:                  03:57:00          Pearson chi2:              398.
No. Iterations:         7
Covariance Type:        nonrobust
=====
=====
              coef      std err          z      P>|z|      [0.025
0.975]
-----
vive_padre      0.1216      0.030      4.040      0.000      0.063
0.181
vive_madre      0.3998      0.065      6.147      0.000      0.272
```

0.527					
n_habitaciones	0.2054	0.016	13.166	0.000	0.175
0.236					
cercania_juegos	0.1151	0.037	3.075	0.002	0.042
0.189					
cercania_servicios	0.1436	0.042	3.426	0.001	0.061
0.226					
area	0.1685	0.044	3.845	0.000	0.083
0.254					

=====

=====

```
[ ]: y=junaeb['n_personas']
X = junaeb.drop(['n_personas', "educm", "educp"], axis= 1)

negbin=sm.GLM(y,X,family=sm.families.NegativeBinomial()).fit()
print(negbin.summary())
```

Generalized Linear Model Regression Results

Dep. Variable:	n_personas	No. Observations:	6296
Model:	GLM	Df Residuals:	6290
Model Family:	NegativeBinomial	Df Model:	5
Link Function:	log	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-16178.
Date:	Wed, 14 Sep 2022	Deviance:	362.53
Time:	03:57:00	Pearson chi2:	398.
No. Iterations:	7		
Covariance Type:	nonrobust		

=====

=====

	coef	std err	z	P> z	[0.025
0.975]					

vive_padre	0.1216	0.030	4.040	0.000	0.063
0.181					
vive_madre	0.3998	0.065	6.147	0.000	0.272
0.527					
n_habitaciones	0.2054	0.016	13.166	0.000	0.175
0.236					
cercania_juegos	0.1151	0.037	3.075	0.002	0.042
0.189					
cercania_servicios	0.1436	0.042	3.426	0.001	0.061
0.226					
area	0.1685	0.044	3.845	0.000	0.083
0.254					

=====

=====

10 9. Comentarios

Para explicar el numero de personas que viven en un hogar utilizando poisson, resulto que las variables correspondiente a la educación de los padres (educm, edup) resultaron ser no significativas por lo que fueron quitadas del modelo.

Luego al realizar nuevamente el modelo, resultó que las variables que si fueron significativa dentro del modelo (se puede asegurarse que son distintas de 0) son: vive_padre, vive_madre, n_habitaciones, cercania_juego, cercania_servicios y el area, las cuales afectan positivamente a la variable dependiente, lo que indica que un aumento en una de estas variables creará un aumento en la variable dependiente n_personas.

Una característica de la distribución de poisson es que $\text{varianza} = \text{media} = \text{lambda}$, por lo que si la dispersión de los datos es distinta a la media, se recomienda utilizar otro tipo de modelo como la regresión binomial negativa, la cual es útil cuando se encuentra una sobredispersión en los datos. Para esto se estimó α como la medida de la dispersión de los datos.

Dado que este es menor a 1, no se encuentra sobredispersión en los datos, es más, como este es un valor negativo tampoco se puede utilizar como valor inicial para el modelo de regresión binomial negativa ya que este requiere que tenga valor positivo, por lo que se concluye que es mejor quedarse con el modelo Poisson.

Por las razones anteriormente expuestas se espera que el modelo Poisson responda de mejor forma la pregunta de investigación, al haber poca dispersión se debería ajustar relativamente bien, mientras que el modelo binomial negativo al estar diseñado para cuando hay sobredispersión no logra ajustarse tan bien a los datos, además se encuentra que el parametro de log-likelihood del modelo Poisson es mucho más cercano a cero.

Finalmente las variables robustas para el modelo son:

- vive_padre
- vive_madre
- n_habitaciones
- cercania_juegos
- cercania_servicios
- area

[]: