

Trabajo_1_Laboratorio_aplicado

September 26, 2022

0.0.1 Analisis de datos y limpieza

```
[6]: # Tratamiento de datos
# =====
import pandas as pd
import numpy as np

# Gráficos
# =====
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
%matplotlib inline

# Preprocesado y análisis
# =====
import scipy
import sklearn
import statsmodels.api as sm
from scipy import stats
from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import plot_confusion_matrix
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Configuración matplotlib
# =====
plt.style.use('ggplot')

# Configuración warnings
# =====
import warnings
```

```
warnings.filterwarnings('ignore')

# Configuración matplotlib
# =====
plt.rcParams['image.cmap'] = "bwr"
#plt.rcParams['figure.dpi'] = "100"
plt.rcParams['savefig.bbox'] = "tight"
style.use('ggplot') or plt.style.use('ggplot')

#Libreria para analizar VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LinearRegression
```

```
[7]: # Cargando data
#junaeb = pd.read_csv('C:/Users/crist/Documents/GitHub/LAB-MAA/data/junaeb.csv')
junaeb = pd.read_csv('junaeb.csv')
junaeb
```

```
[7]:
```

	vive_padre	vive_madre	n_personas	n_habitaciones	cercania_juegos	\
0	0	1	3.0	4.0	1.0	
1	0	1	5.0	3.0	1.0	
2	1	1	5.0	3.0	1.0	
3	1	1	4.0	2.0	1.0	
4	1	1	5.0	3.0	2.0	
...	
6602	0	0	NaN	NaN	NaN	
6603	0	0	NaN	NaN	NaN	
6604	0	0	NaN	NaN	NaN	
6605	0	0	NaN	NaN	NaN	
6606	0	0	NaN	NaN	NaN	

	cercania_servicios	edad_primer_parto	area	educm	educp
0	1.0	25.0	1	0	0
1	1.0	23.0	1	13	13
2	1.0	19.0	1	12	17
3	1.0	27.0	1	6	13
4	1.0	20.0	1	13	16
...
6602	NaN	NaN	1	0	0
6603	NaN	NaN	1	0	0
6604	NaN	NaN	1	0	0
6605	NaN	NaN	1	0	0
6606	NaN	NaN	1	0	0

[6607 rows x 10 columns]

```
[3]: 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
```

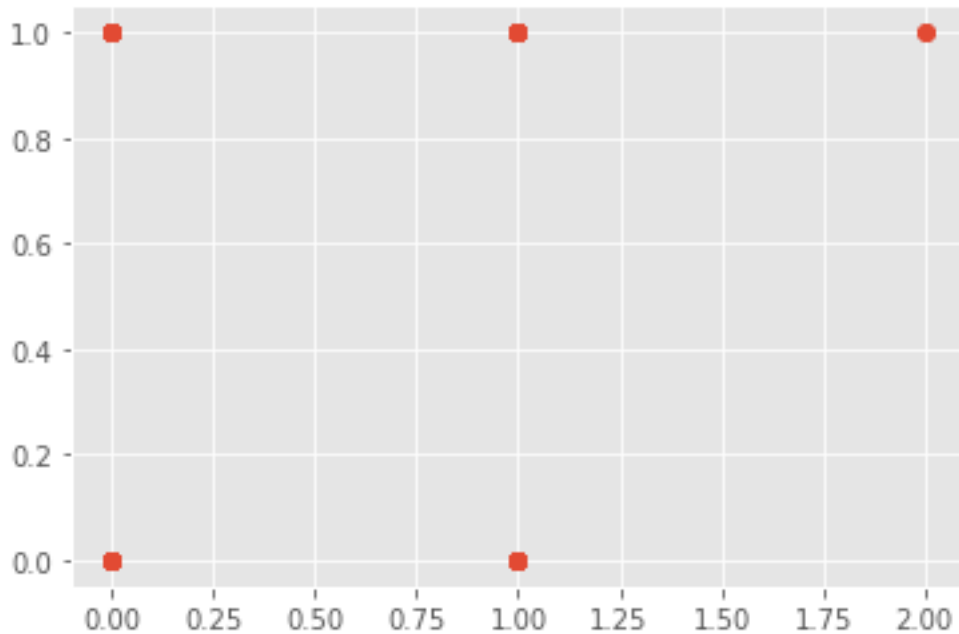
```
[4]: import qgrid
qgrid_widget = qgrid.show_grid(junaeb,show_toolbar=True)
qgrid_widget
```

```
QgridWidget(grid_options={'fullWidthRows': True, 'syncColumnCellResize': True,
↳ 'forceFitColumns': True, 'defau...
```

Outliers Vive_madre = 2 , no tiene sentido por lo que se procede a eliminar

```
[5]: plt.scatter(junaeb['vive_madre'],junaeb['vive_padre'])
```

```
[5]: <matplotlib.collections.PathCollection at 0x23b4d570b20>
```



```
[6]: junaeb.drop(junaeb[junaeb['vive_madre']==2].index,inplace =True)
```

```
[7]: print(junaeb["cercania_juegos"].value_counts(normalize=False))
junaeb.drop(junaeb[junaeb['cercania_juegos']==4].index,inplace =True)
junaeb.drop(junaeb[junaeb['cercania_servicios']==2].index,inplace =True)
```

```
1.0    5245
2.0    1173
4.0      52
Name: cercania_juegos, dtype: int64
```

Visualiacion de datos

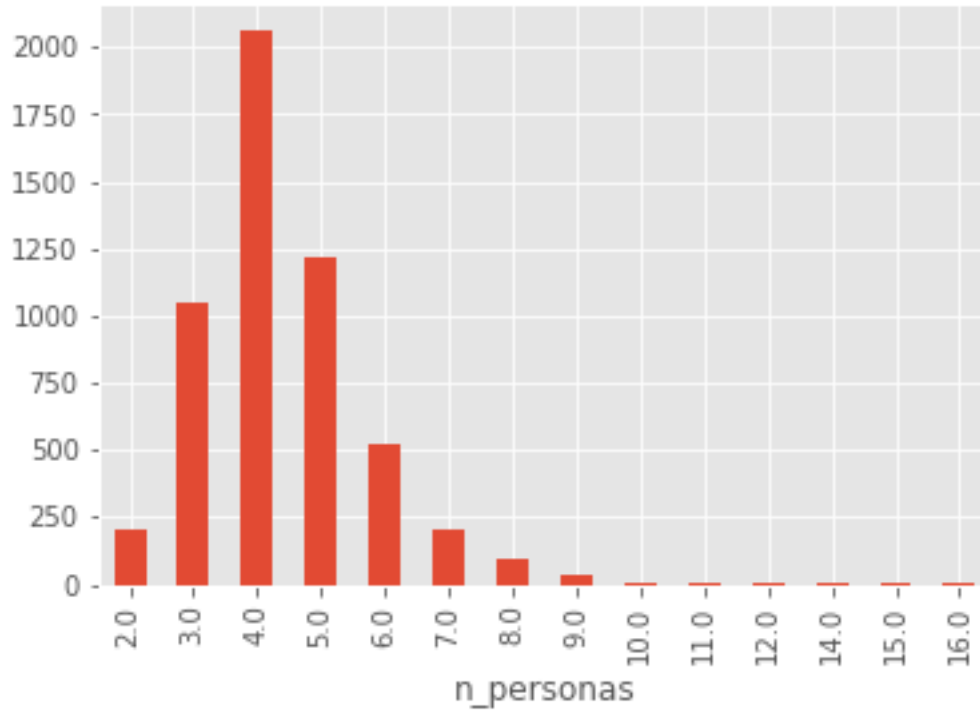
```
[8]: print(junaeb["vive_padre"].value_counts())
print("Porcentaje de padres que viven en la residencia vs los que no viven")
print(100*junaeb["vive_padre"].value_counts(normalize=True))
```

```
1    3826
0    1881
Name: vive_padre, dtype: int64
Porcentaje de padres que viven en la residencia vs los que no viven
1    67.040477
0    32.959523
Name: vive_padre, dtype: float64
```

```
[9]: temp_0 =junaeb.loc[junaeb.vive_madre==1].groupby(by = 'n_personas').
      ↪count()["vive_madre"].plot.bar()
```

```
temp_0
```

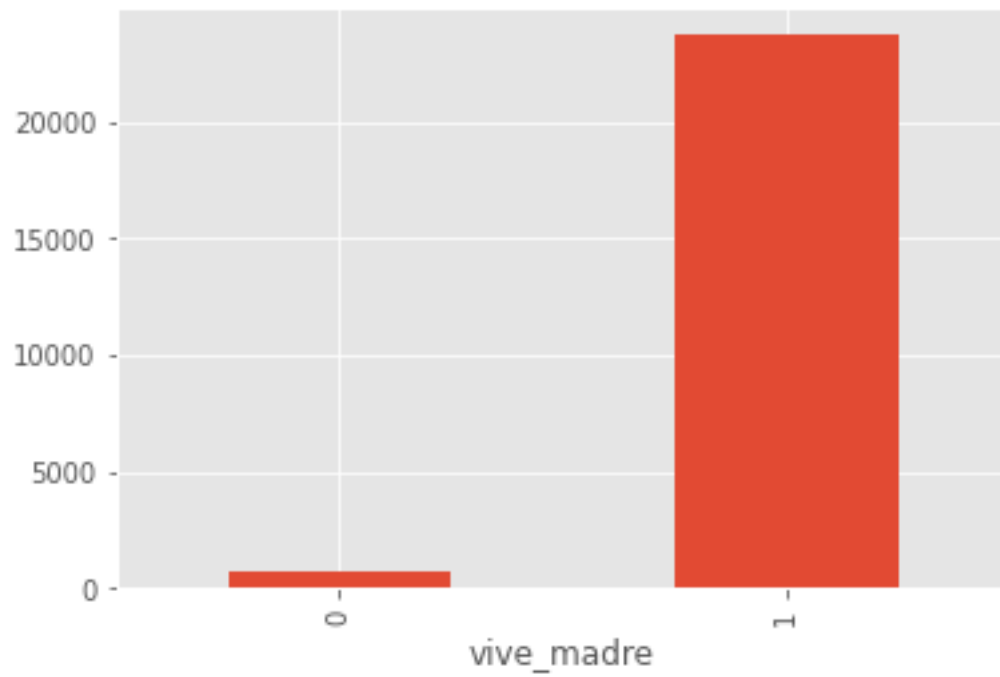
```
[9]: <AxesSubplot:xlabel='n_personas'>
```



```
[10]: temp = junaeb.groupby('n_personas').vive_padre.sum()  
temp.plot.bar();
```

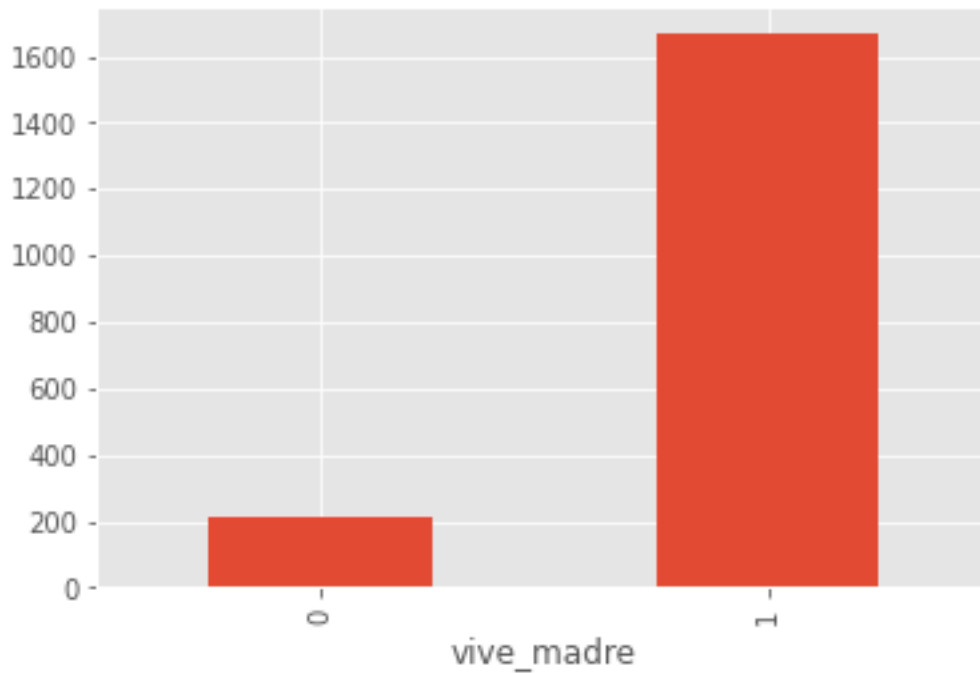


```
[11]: temp = junaeb.groupby('vive_madre').n_personas.sum()  
temp.plot.bar();
```



```
[12]: junaeb.loc[junaeb.vive_padre==0].groupby(by = 'vive_madre').  
      ↪count()["vive_padre"].plot.bar()
```

```
[12]: <AxesSubplot:xlabel='vive_madre'>
```



```
[13]: junaeb.groupby(by="vive_madre").count()["vive_padre"]  
      junaeb.loc[junaeb.vive_padre==0].groupby(by = 'vive_madre').  
      ↪count()["vive_padre"]
```

```
[13]: vive_madre  
      0      215  
      1     1666  
      Name: vive_padre, dtype: int64
```

```
[14]: junaeb.groupby(by = 'vive_madre').count()["vive_padre"]
```

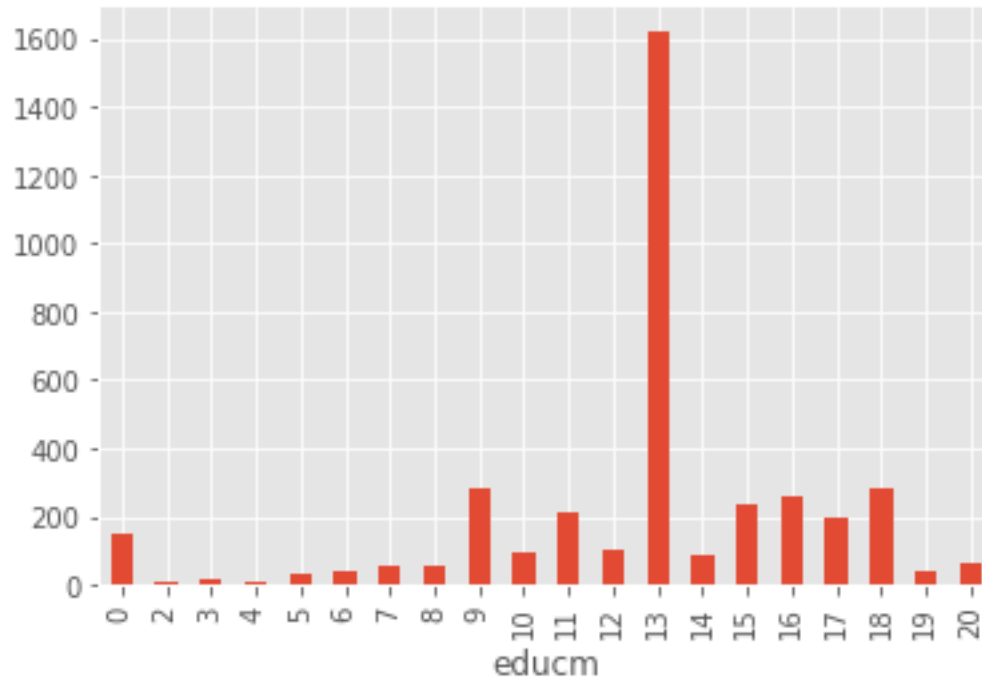
```
[14]: vive_madre  
      0      302  
      1     5405  
      Name: vive_padre, dtype: int64
```

```
[15]: junaeb.loc[junaeb.vive_padre==1].groupby(by = 'vive_madre').  
      ↪count()["vive_padre"]
```

```
[15]: vive_madre
      0      87
      1    3739
      Name: vive_padre, dtype: int64
```

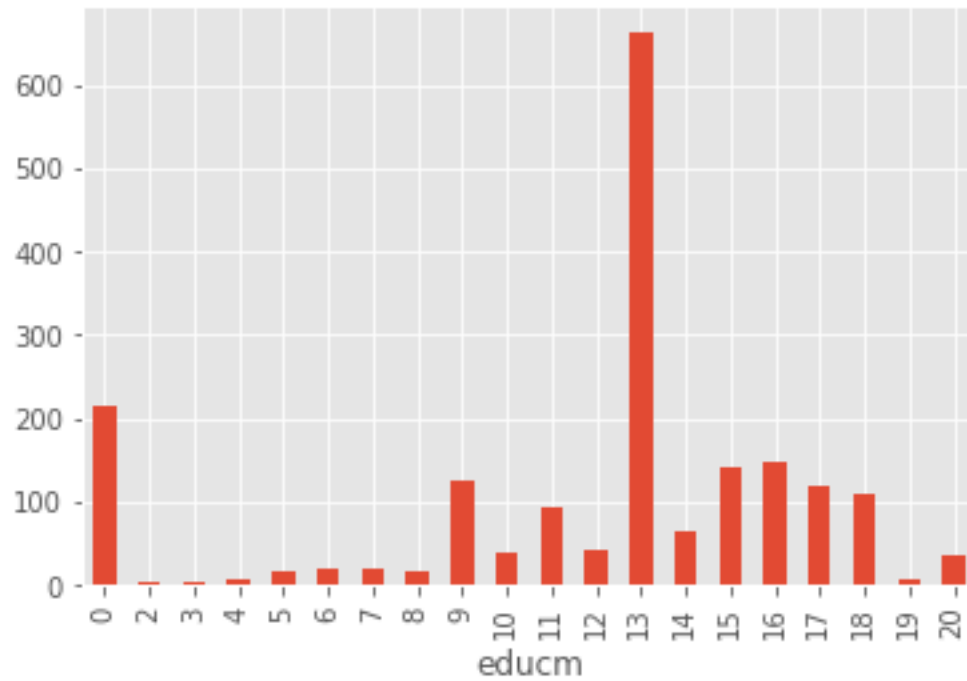
```
[16]: junaeb.loc[junaeb.vive_padre==1].groupby(by = 'educm').count()["vive_padre"].
      ↪plot.bar()
```

```
[16]: <AxesSubplot:xlabel='educm'>
```



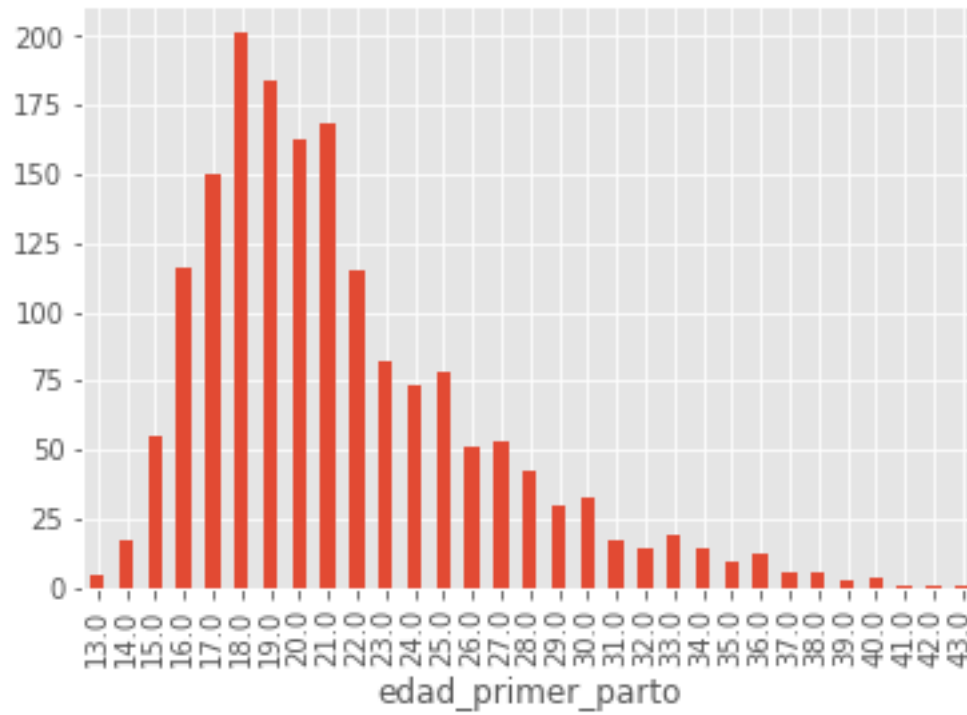
```
[17]: junaeb.loc[junaeb.vive_padre==0].groupby(by = 'educm').count()["vive_padre"].
      ↪plot.bar()
```

```
[17]: <AxesSubplot:xlabel='educm'>
```

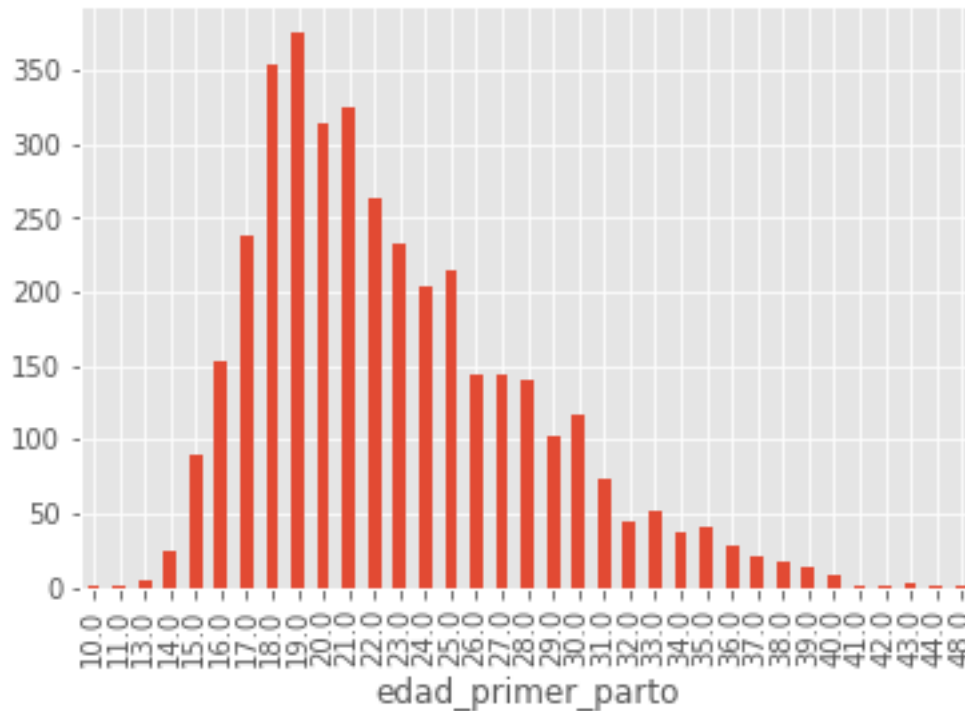
```
[18]: junaeb.loc[junaeb.vive_padre==0].groupby(by = 'edad_primer_parto').  
      ↪count()["vive_padre"].plot.bar()
```

```
[18]: <AxesSubplot:xlabel='edad_primer_parto'>
```



```
[19]: junaeb.loc[junaeb.vive_padre==1].groupby(by = 'edad_primer_parto').  
      ↪count()["vive_padre"].plot.bar()
```

```
[19]: <AxesSubplot:xlabel='edad_primer_parto'>
```



Análisis de nulos Se procede a analizar los nulos haciendo una suma de la cantidad que hay para ver si es que conviene eliminarlos o reemplazarlos por ejemplo por el promedio

```
[20]: junaeb.isna().sum().sort_values(ascending = False)
```

```
[20]: edad_primer_parto      200
      n_habitaciones        141
      n_personas            134
      cercania_juegos        132
      cercania_servicios     132
      vive_padre             0
      vive_madre             0
      area                   0
      educm                   0
      educp                   0
      dtype: int64
```

```
[21]: #Se procede a eliminar los nulos del data frame
      junaeb.dropna(inplace=True)
      junaeb.reset_index(drop=True, inplace=True)
      junaeb.head()
```

```
[21]:
```

	vive_padre	vive_madre	n_personas	n_habitaciones	cercania_juegos	\
0	0	1	3.0	4.0	1.0	
1	0	1	5.0	3.0	1.0	
2	1	1	5.0	3.0	1.0	
3	1	1	4.0	2.0	1.0	
4	1	1	5.0	3.0	2.0	

	cercania_servicios	edad_primer_parto	area	educm	educp
0	1.0	25.0	1	0	0
1	1.0	23.0	1	13	13
2	1.0	19.0	1	12	17
3	1.0	27.0	1	6	13
4	1.0	20.0	1	13	16

```
[22]: junaeb.edad_primer_parto.value_counts()
```

```
[22]:
```

19.0	558
18.0	556
21.0	494
20.0	477
17.0	388
22.0	379
23.0	314
25.0	292
24.0	275
16.0	270
27.0	197
26.0	195
28.0	183
30.0	150
15.0	144
29.0	132
31.0	90
33.0	70
32.0	58
34.0	51
35.0	50
14.0	42
36.0	41
37.0	26
38.0	22
39.0	15
40.0	12
13.0	9
43.0	3
41.0	2
42.0	2

```

11.0      1
44.0      1
10.0      1
48.0      1
Name: edad_primer_parto, dtype: int64

```

```

[23]: def estadisticos_cont(num):
        #Calculamos describe
        estadisticos = num.describe().T
        #Añadimos la mediana
        estadisticos['median'] = num.median()
        #Reordenamos para que la mediana esté al lado de la media
        estadisticos = estadisticos.iloc[:, [0,1,8,2,3,4,5,6,7]]
        #Lo devolvemos
        return(estadisticos)

```

```

[24]: estadisticos_cont(junaeb.select_dtypes('number'))

```

```

[24]:

```

	count	mean	median	std	min	25%	50%	\
vive_padre	5501.0	0.688057	1.0	0.463329	0.0	0.0	1.0	
vive_madre	5501.0	0.974187	1.0	0.158593	0.0	1.0	1.0	
n_personas	5501.0	4.385930	4.0	1.342025	1.0	4.0	4.0	
n_habitaciones	5501.0	2.606253	2.0	0.907659	0.0	2.0	2.0	
cercania_juegos	5501.0	1.128340	1.0	0.334499	1.0	1.0	1.0	
cercania_servicios	5501.0	1.013089	1.0	0.197741	1.0	1.0	1.0	
edad_primer_parto	5501.0	22.321941	21.0	5.217991	10.0	18.0	21.0	
area	5501.0	0.924196	1.0	0.264709	0.0	1.0	1.0	
educm	5501.0	12.795855	13.0	3.911051	0.0	12.0	13.0	
educp	5501.0	11.414652	13.0	5.244827	0.0	10.0	13.0	

	75%	max
vive_padre	1.0	1.0
vive_madre	1.0	1.0
n_personas	5.0	16.0
n_habitaciones	3.0	20.0
cercania_juegos	1.0	2.0
cercania_servicios	1.0	4.0
edad_primer_parto	25.0	48.0
area	1.0	1.0
educm	15.0	20.0
educp	13.0	20.0

```

[25]: junaeb.corr()

```

```

[25]:

```

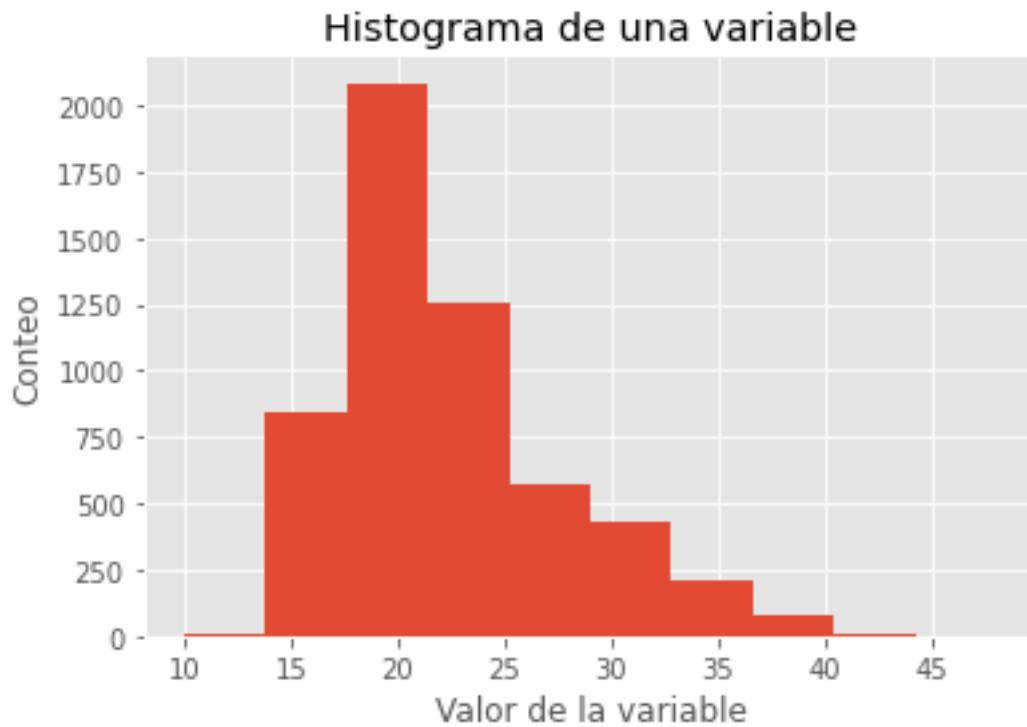
	vive_padre	vive_madre	n_personas	n_habitaciones	\
vive_padre	1.000000	0.056178	0.110020	-0.004613	
vive_madre	0.056178	1.000000	0.013499	-0.049149	

n_personas	0.110020	0.013499	1.000000	0.445391
n_habitaciones	-0.004613	-0.049149	0.445391	1.000000
cercania_juegos	-0.012633	-0.009513	0.019657	-0.022166
cercania_servicios	0.014804	0.010775	-0.006705	-0.016867
edad_primer_parto	0.113367	0.070025	-0.181605	-0.002905
area	-0.023838	0.035669	0.009178	0.012719
educm	-0.010567	0.163277	-0.045365	0.031183
educp	0.344436	0.066861	0.023163	0.054048

	cercania_juegos	cercania_servicios	edad_primer_parto	\
vive_padre	-0.012633	0.014804	0.113367	
vive_madre	-0.009513	0.010775	0.070025	
n_personas	0.019657	-0.006705	-0.181605	
n_habitaciones	-0.022166	-0.016867	-0.002905	
cercania_juegos	1.000000	0.032325	-0.054615	
cercania_servicios	0.032325	1.000000	-0.008842	
edad_primer_parto	-0.054615	-0.008842	1.000000	
area	-0.163209	0.008538	0.054002	
educm	-0.043761	-0.033220	0.130561	
educp	-0.075524	-0.015753	0.142151	

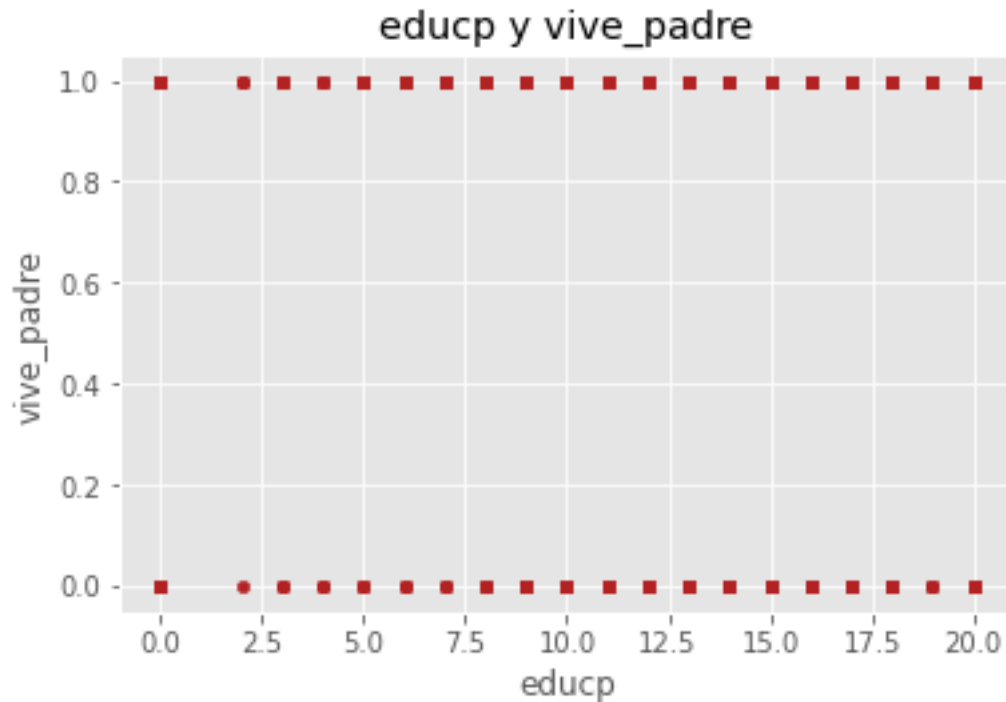
	area	educm	educp
vive_padre	-0.023838	-0.010567	0.344436
vive_madre	0.035669	0.163277	0.066861
n_personas	0.009178	-0.045365	0.023163
n_habitaciones	0.012719	0.031183	0.054048
cercania_juegos	-0.163209	-0.043761	-0.075524
cercania_servicios	0.008538	-0.033220	-0.015753
edad_primer_parto	0.054002	0.130561	0.142151
area	1.000000	0.082870	0.064682
educm	0.082870	1.000000	0.296504
educp	0.064682	0.296504	1.000000

```
[26]: plt.hist(junaeb["edad_primer_parto"])
plt.title('Histograma de una variable')
plt.xlabel('Valor de la variable')
plt.ylabel('Conteo')
plt.show()
```



```
[27]: ig, ax = plt.subplots(figsize=(6, 3.84))

junaeb.plot(
    x = "educp",
    y = 'vive_padre',
    c = 'firebrick',
    kind = "scatter",
    ax = ax
)
ax.set_title("educp y vive_padre");
```

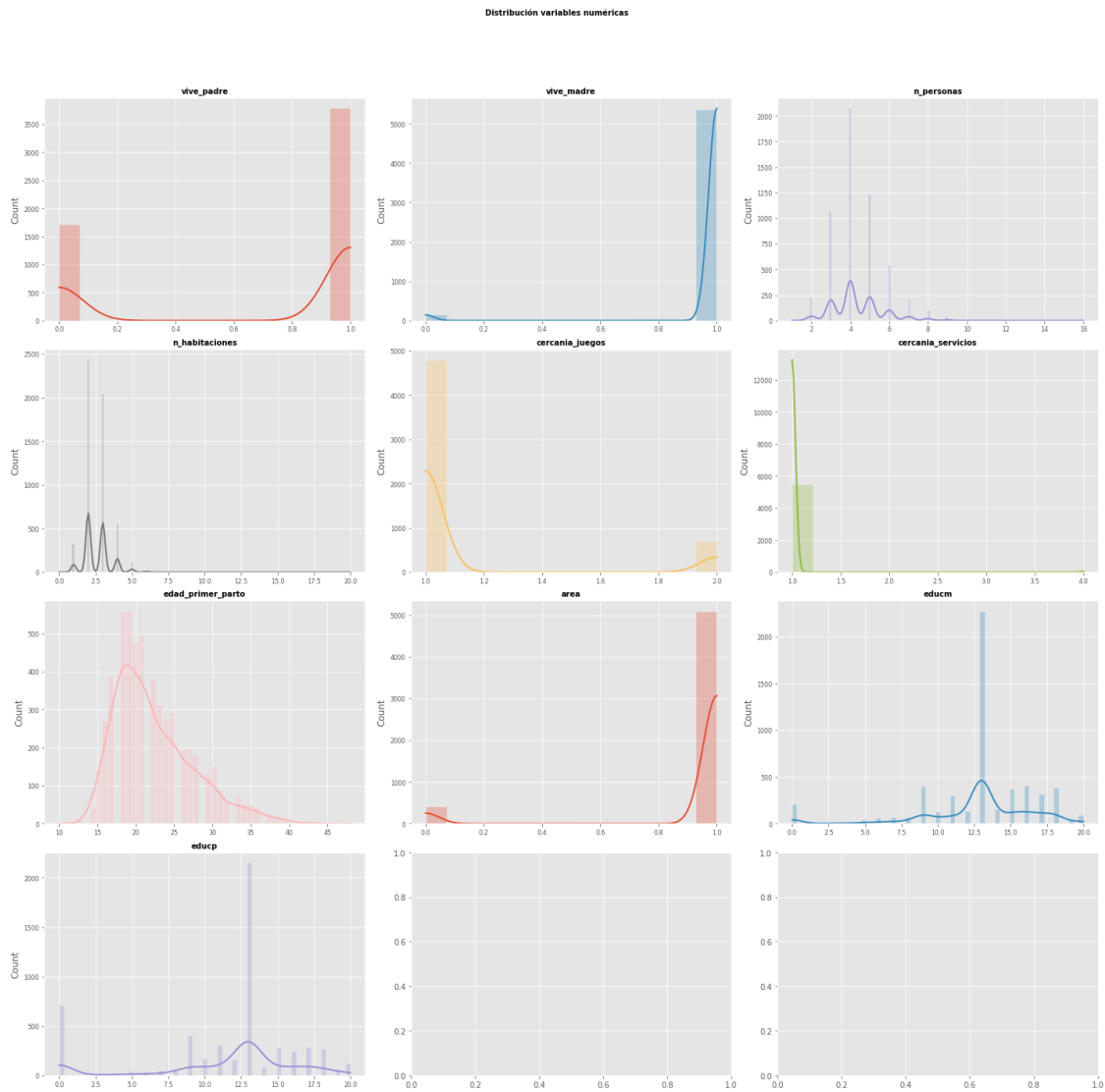


```
[28]: # Gráfico de distribución para cada variable numérica
# =====
# Ajustar número de subplots en función del número de columnas
fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(20, 20))
axes = axes.flat
columnas_numeric = junaeb.select_dtypes(include=['float64', 'int']).columns

for i, column in enumerate(columnas_numeric):
    sns.histplot(
        data = junaeb,
        x = column,
        stat = "count",
        kde = True,
        color = (list(plt.rcParams['axes.prop_cycle'])*2)[i]["color"],
        line_kws= {'linewidth': 2},
        alpha = 0.3,
        ax = axes[i]
    )
    axes[i].set_title(column, fontsize = 10, fontweight = "bold")
    axes[i].tick_params(labelsize = 8)
    axes[i].set_xlabel("")
```



```
fig.tight_layout()
plt.subplots_adjust(top = 0.9)
fig.suptitle('Distribución variables numéricas', fontsize = 10, fontweight = u
↪ "bold");
```



```
[29]: # Tratamiento de datos
# =====
import pandas as pd
import numpy as np

# Gráficos
# =====
import matplotlib.pyplot as plt
```

```

from matplotlib import style
import seaborn as sns

# Preprocesado y análisis
# =====

import statsmodels.api as sm
from scipy import stats

# Configuración matplotlib
# =====

plt.style.use('ggplot')

# Configuración warnings
# =====

import warnings
warnings.filterwarnings('ignore')
# Tratamiento de datos
# =====

import pandas as pd
import numpy as np

# Gráficos
# =====

import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

# Preprocesado y modelado
# =====

from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Configuración matplotlib
# =====

plt.rcParams['image.cmap'] = "bwr"
#plt.rcParams['figure.dpi'] = "100"
plt.rcParams['savefig.bbox'] = "tight"
style.use('ggplot') or plt.style.use('ggplot')

# Configuración warnings
# =====

import warnings

```

```
warnings.filterwarnings('ignore')
#Libreria para analizar VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LinearRegression
```

0.0.2 OLS

Análisis de variables independientes a través del estadístico T-student Podemos visualizar que las variables cercanía_juegos y cercanía_servicios son no significativas por lo que podemos sacarlas del modelo

```
[30]: y=junaeb['vive_padre']
X_0=junaeb[["vive_madre", "n_personas", "n_habitaciones", "edad_primer_parto", "area", "educm", "edu
X_0=sm.add_constant(X_0)
model = sm.OLS(y, X_0)
results = model.fit()
print(results.summary())
```

```

                                OLS Regression Results
=====
Dep. Variable:                vive_padre    R-squared:                0.161
Model:                        OLS          Adj. R-squared:         0.160
Method:                      Least Squares  F-statistic:              117.0
Date:                        Thu, 15 Sep 2022  Prob (F-statistic):    9.79e-202
Time:                        20:13:12      Log-Likelihood:          -3090.5
No. Observations:            5501          AIC:                   6201.
Df Residuals:                5491          BIC:                   6267.
Df Model:                    9
Covariance Type:              nonrobust
=====
=====
                                coef    std err          t      P>|t|      [0.025
-----
0.975]
-----
const                0.0952    0.065      1.475    0.140    -0.031
0.222
vive_madre           0.1167    0.037      3.172    0.002     0.045
0.189
n_personas           0.0527    0.005     10.795    0.000     0.043
0.062
n_habitaciones      -0.0436    0.007     -6.137    0.000    -0.058
-0.030
edad_primer_parto    0.0093    0.001      8.166    0.000     0.007
0.012
area                -0.0775    0.022     -3.517    0.000    -0.121
-0.034
educm               -0.0149    0.002     -9.557    0.000    -0.018

```

```

-0.012
educp          0.0326    0.001    28.193    0.000    0.030
0.035
cercania_juegos  0.0043    0.017    0.249    0.803    -0.030
0.039
cercania_servicios  0.0393    0.029    1.355    0.175    -0.018
0.096
=====
Omnibus:          684.852    Durbin-Watson:          1.987
Prob(Omnibus):    0.000    Jarque-Bera (JB):       721.910
Skew:            -0.832    Prob(JB):               1.73e-157
Kurtosis:        2.381    Cond. No.               353.
=====

```

Notes:

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

```

[31]: y=junaeb['vive_padre']
X=junaeb[["vive_madre","n_personas","n_habitaciones","edad_primer_parto","area","educm","educp"]
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:          0.161
Model:                  OLS          Adj. R-squared:       0.160
Method:                 Least Squares    F-statistic:       150.2
Date:                  Thu, 15 Sep 2022    Prob (F-statistic): 1.69e-203
Time:                  20:13:12          Log-Likelihood:    -3091.4
No. Observations:      5501             AIC:              6199.
Df Residuals:          5493             BIC:              6252.
Df Model:              7
Covariance Type:       nonrobust
=====
=====
                                coef    std err          t      P>|t|      [0.025
0.975]
-----
const          0.1416    0.052      2.706    0.007    0.039
0.244
vive_madre     0.1175    0.037      3.194    0.001    0.045
0.190
n_personas     0.0528    0.005     10.801    0.000    0.043
0.062

```

n_habitaciones	-0.0438	0.007	-6.162	0.000	-0.058
-0.030					
edad_primer_parto	0.0093	0.001	8.155	0.000	0.007
0.012					
area	-0.0780	0.022	-3.585	0.000	-0.121
-0.035					
educm	-0.0150	0.002	-9.607	0.000	-0.018
-0.012					
educp	0.0325	0.001	28.215	0.000	0.030
0.035					


Omnibus:	686.250	Durbin-Watson:	1.988
Prob(Omnibus):	0.000	Jarque-Bera (JB):	722.178
Skew:	-0.832	Prob(JB):	1.52e-157
Kurtosis:	2.380	Cond. No.	298.

Notes:

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

Interpretacion de los coeficientes Podemos notar a traves de los coeficientes del modelo OLS , que la proporcion de que el padre viva en la casa disminuye si el lugar donde habita es en zona urbana. Ademas la proporcion de que el padre viva en la casa disminuye a medida que la educacion de la madre se incrementa y tambien la proporcion de que el padre viva en la casa disminuye si se incrementa el numero de habitaciones. Estas tres variables hacen que la probabilidad de que el padre viva en la casa se vea disminuida.

Por otro lado las variables vive_madre , n_personas , edad_primer_parto , educp hacen que la probabilidad de que el padre viva en la casa aumente , por ejemplo si la madre vive en la casa la proporcion de que el padre viva en la casa aumenta.

Podemos notar que a medida si vive una madre en el hogar la probabilidad de que viva el padre aumenta en un 12.16% ¿Correcto? 

Analisis prediccion Se extrajo el 70% de los datos para entrenar al modelo con esos datos , para luego realizar la prediccion con el 30% restante

```
[32]: X_train, X_test, y_train, y_test = train_test_split(
        X,
        y,
        train_size = 0.7,
        random_state = 1,
        shuffle     = True
    )

# A la matriz de predictores se le tiene que añadir una columna de 1s para el
↪intercept del modelo
X_train = sm.add_constant(X_train, prepend=True)
```

```

modelo = sm.OLS(endog=y_train, exog=X_train,)
modelo = modelo.fit()
print(modelo.summary())

df_prediccion= pd.read_csv('C:/Users/crist/Documents/GitHub/LAB-MAA/data/
↳Prediccion_real.csv')
df_prediccion.rename(columns={'Sheet1.Prediccion':'vive_padre_prediccion'},
                      inplace=True)
print(df_prediccion.plot("vive_padre", "vive_padre_prediccion", kind="scatter"))

```

OLS Regression Results

```

=====
Dep. Variable:          vive_padre    R-squared:                0.159
Model:                  OLS           Adj. R-squared:           0.157
Method:                 Least Squares  F-statistic:              103.7
Date:                   Thu, 15 Sep 2022  Prob (F-statistic):      1.79e-139
Time:                   20:13:12       Log-Likelihood:          -2177.0
No. Observations:      3850           AIC:                     4370.
Df Residuals:          3842           BIC:                     4420.
Df Model:               7
Covariance Type:       nonrobust
=====

```

```

=====
coef      std err          t      P>|t|      [0.025
0.975]
-----
-----
const          0.1450      0.062      2.335      0.020      0.023
0.267
vive_madre     0.1333      0.043      3.082      0.002      0.048
0.218
n_personas     0.0514      0.006      8.853      0.000      0.040
0.063
n_habitaciones -0.0368      0.008     -4.380      0.000     -0.053
-0.020
edad_primer_parto 0.0089      0.001      6.443      0.000      0.006
0.012
area          -0.0826      0.026     -3.127      0.002     -0.134
-0.031
educm         -0.0163      0.002     -8.731      0.000     -0.020
-0.013
educp          0.0323      0.001     23.371      0.000      0.030
0.035
=====

```

```

Omnibus:         496.525    Durbin-Watson:           1.993
Prob(Omnibus):   0.000     Jarque-Bera (JB):        495.203
Skew:           -0.815     Prob(JB):                2.94e-108

```

Kurtosis:

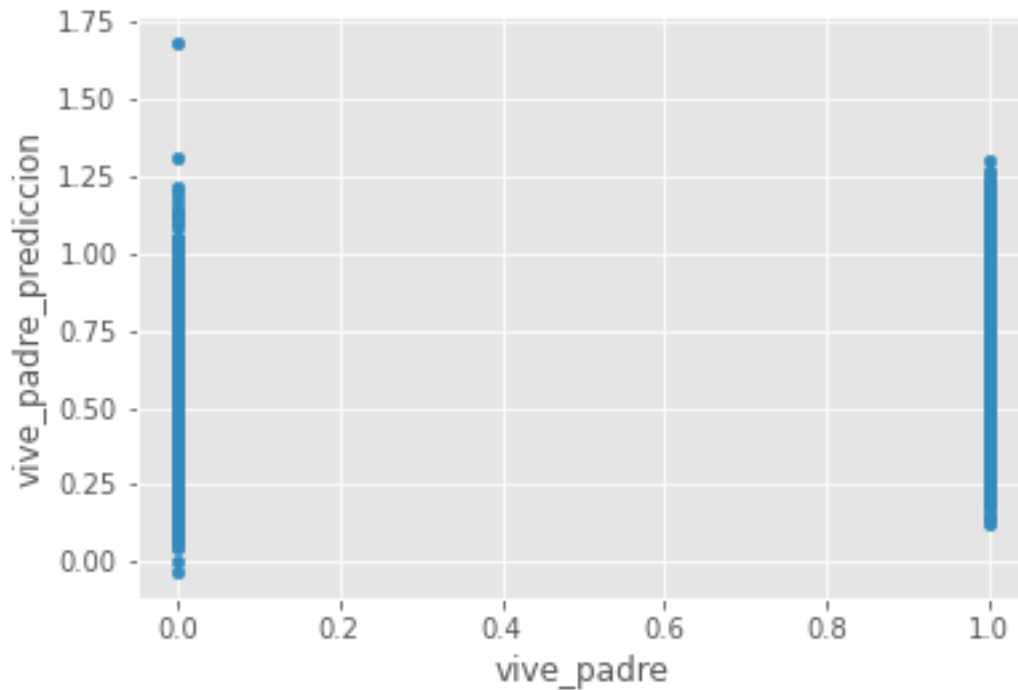
2.343 Cond. No.

293.

Notes:

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

AxesSubplot(0.125,0.125;0.775x0.755)



Resultados valor real vs predicho Podemos notar que hay valores que se escapan del rango [0-1]

```
[8]: df_prediccion= pd.read_csv('Predicciones_X_Test_Y_Test.csv')
      print(df_prediccion.plot("vive_padre", "Prediccion", kind="scatter"))
      df_prediccion
```

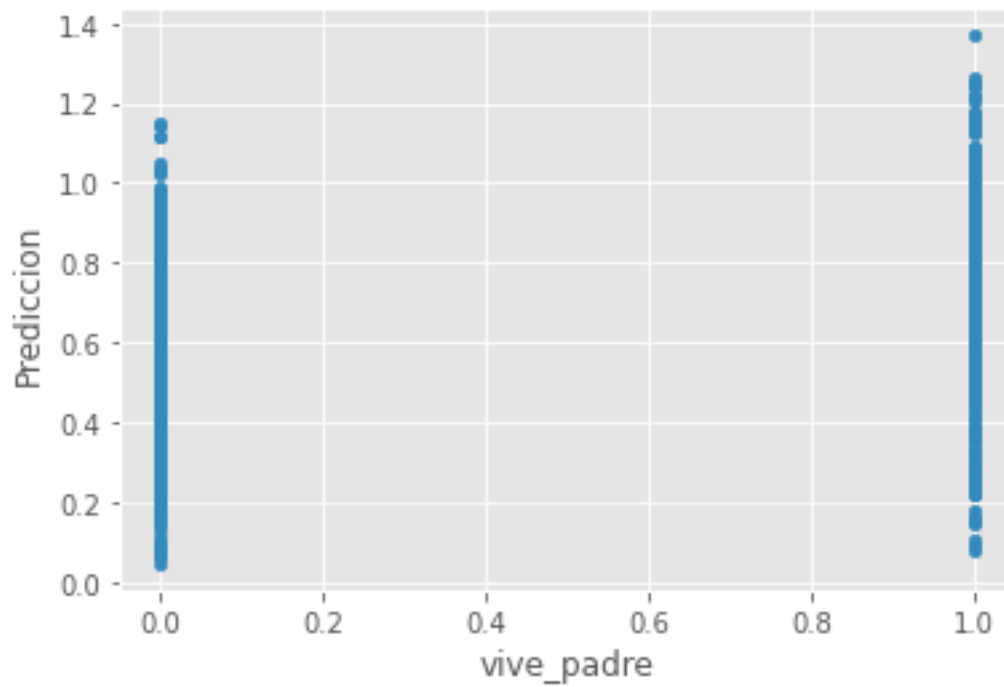
AxesSubplot(0.125,0.125;0.775x0.755)

```
[8]:
```

	ID	vive_padre	Prediccion
0	885	1	1.369920
1	5274	1	1.261178
2	840	1	1.249096
3	1098	1	1.246016
4	1946	1	1.244761
...

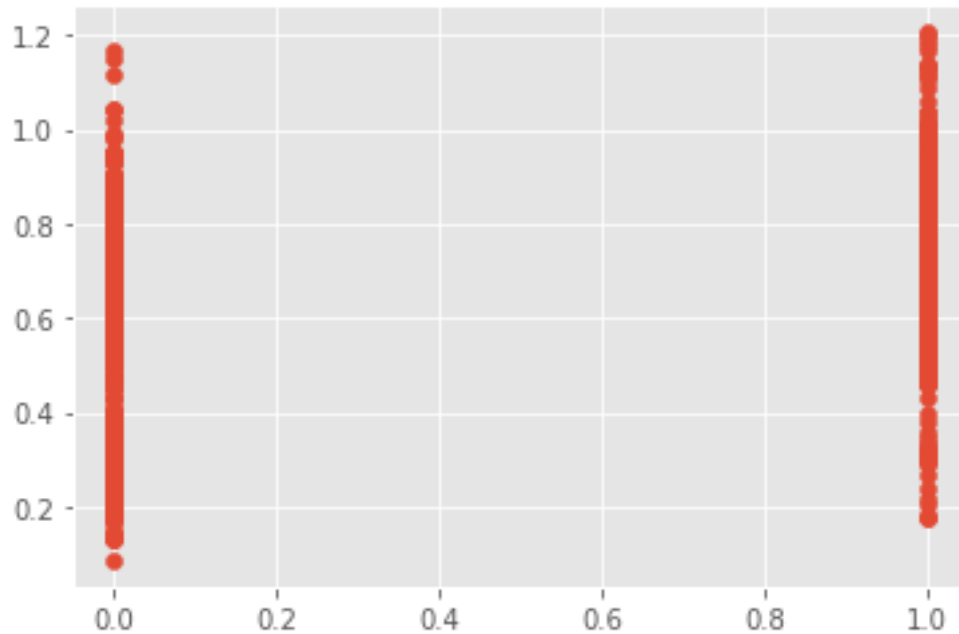
1908	4040	0	0.078419
1909	3059	0	0.074085
1910	3771	0	0.067901
1911	5104	0	0.066310
1912	1691	0	0.046000

[1913 rows x 3 columns]



```
[34]: prediccion_train = modelo.predict(exog = X_test)
      residuos_train  = prediccion_train - y_test
      plt.scatter(y_test,prediccion_train)
```

[34]: <matplotlib.collections.PathCollection at 0x23b51455910>



Valores maximos y minimos de la prediccion Aqui se visualiza los valores predichos por el modelo OLS que no tienen sentido

```
[35]: print(max(df_prediccion["Prediccion"]), "No tienen sentido estos valores_
      ↪predichos por el OLS que se pueden apreciar en el grafico y en el resultado")
```

1.369920345000249 No tienen sentido estos valores predichos por el OLS que se pueden apreciar en el grafico y en el resultado

analizar residuos graficos y shapiro_test A traves de los graficos y del test de shapiro podemos concluir que los errores no se distribuyen normalmente por lo que no se cumple uno de los supuestos del modelo ocasionando que los estimadores del modelo no sean MELI , ademas esto ya se podia concluir al saber que la variable dependiente vive_padre es una variable limitada binaria que admite valores 0 y 1 y como vimos en clases si la variable dependiente no se comporta de manera lineal entonces los residuos no siguen una dist normal

```
[36]: # Gráficos
# =====
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(20, 20))

sns.histplot(
    data    = residuos_train,
    stat    = "density",
    kde     = True,
    line_kws= {'linewidth': 1},
```

```

        color    = "firebrick",
        alpha    = 0.3,
        ax       = axes[0, 0]
    )

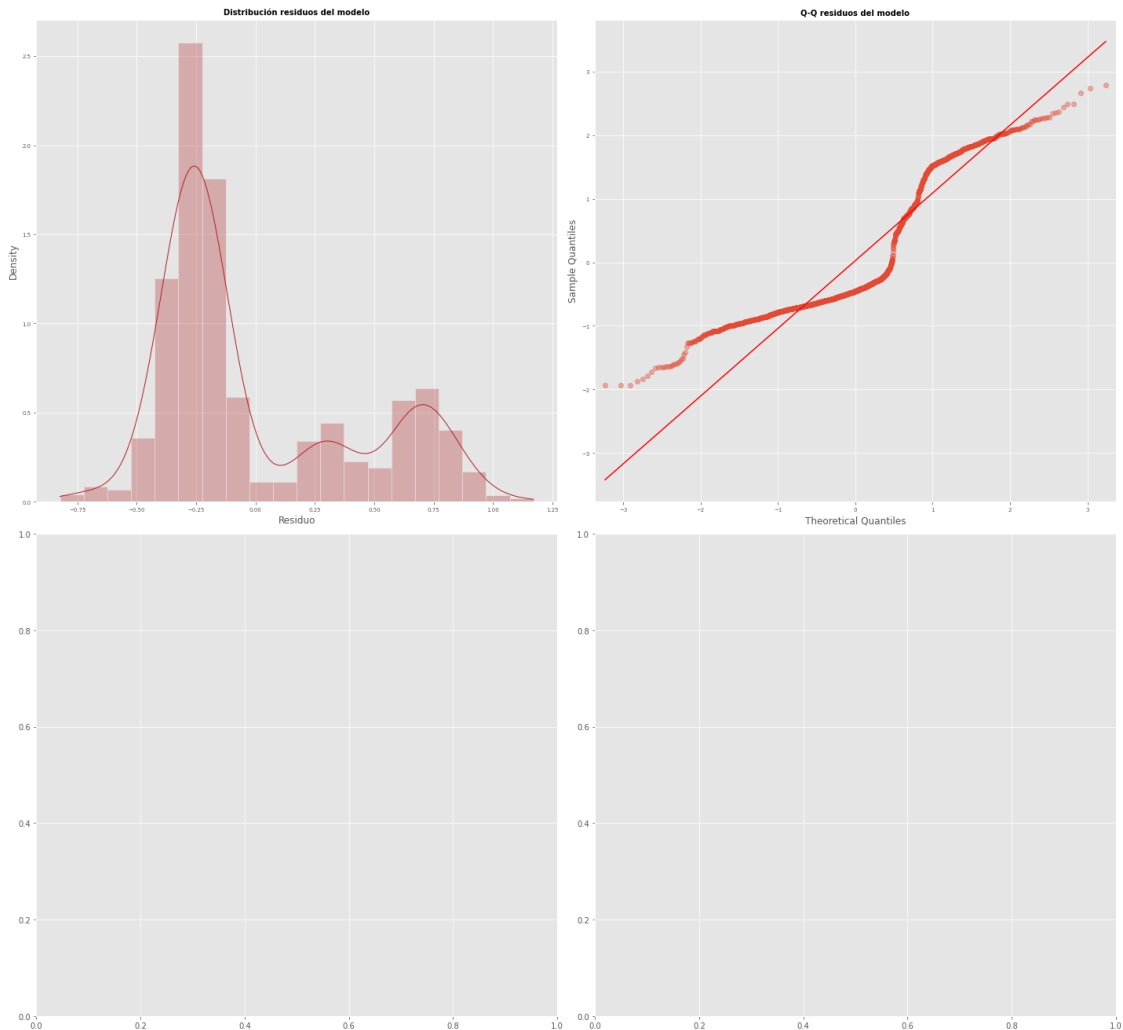
axes[0, 0].set_title('Distribución residuos del modelo', fontsize = 10,
                    fontweight = "bold")
axes[0, 0].set_xlabel("Residuo")
axes[0, 0].tick_params(labelsize = 7)

sm.qqplot(
    residuos_train,
    fit    = True,
    line   = 'q',
    ax     = axes[0, 1],
    color  = 'firebrick',
    alpha  = 0.4,
    lw     = 2
)
axes[0, 1].set_title('Q-Q residuos del modelo', fontsize = 10, fontweight = "bold")
axes[0, 1].tick_params(labelsize = 7)

fig.tight_layout()
plt.subplots_adjust(top=0.9)
fig.suptitle('Diagnóstico residuos', fontsize = 12, fontweight = "bold");

```

Diagnóstico residuos



```
[37]: shapiro_test = stats.shapiro(residuos_train)
      shapiro_test
```

```
[37]: ShapiroResult(statistic=0.8492597341537476, pvalue=5.073376203027274e-37)
```

análisis multicolinealidad y correlaciones Se puede ver que los VIF son inferiores a 10 por lo que no hay problemas de multicolinealidad

```
[38]: corr_matrix = junaeb.corr(method='pearson')
      corr_matrix
```

```
[38]:
```

	vive_padre	vive_madre	n_personas	n_habitaciones	\
vive_padre	1.000000	0.056178	0.110020	-0.004613	

vive_madre	0.056178	1.000000	0.013499	-0.049149
n_personas	0.110020	0.013499	1.000000	0.445391
n_habitaciones	-0.004613	-0.049149	0.445391	1.000000
cercania_juegos	-0.012633	-0.009513	0.019657	-0.022166
cercania_servicios	0.014804	0.010775	-0.006705	-0.016867
edad_primer_parto	0.113367	0.070025	-0.181605	-0.002905
area	-0.023838	0.035669	0.009178	0.012719
educm	-0.010567	0.163277	-0.045365	0.031183
educp	0.344436	0.066861	0.023163	0.054048

	cercania_juegos	cercania_servicios	edad_primer_parto	\
vive_padre	-0.012633	0.014804	0.113367	
vive_madre	-0.009513	0.010775	0.070025	
n_personas	0.019657	-0.006705	-0.181605	
n_habitaciones	-0.022166	-0.016867	-0.002905	
cercania_juegos	1.000000	0.032325	-0.054615	
cercania_servicios	0.032325	1.000000	-0.008842	
edad_primer_parto	-0.054615	-0.008842	1.000000	
area	-0.163209	0.008538	0.054002	
educm	-0.043761	-0.033220	0.130561	
educp	-0.075524	-0.015753	0.142151	

	area	educm	educp
vive_padre	-0.023838	-0.010567	0.344436
vive_madre	0.035669	0.163277	0.066861
n_personas	0.009178	-0.045365	0.023163
n_habitaciones	0.012719	0.031183	0.054048
cercania_juegos	-0.163209	-0.043761	-0.075524
cercania_servicios	0.008538	-0.033220	-0.015753
edad_primer_parto	0.054002	0.130561	0.142151
area	1.000000	0.082870	0.064682
educm	0.082870	1.000000	0.296504
educp	0.064682	0.296504	1.000000

```
[39]: def tidy_corr_matrix(corr_mat):
    '''
    Función para convertir una matriz de correlación de pandas en formato tidy.
    '''
    corr_mat = corr_mat.stack().reset_index()
    corr_mat.columns = ['variable_1', 'variable_2', 'r']
    corr_mat = corr_mat.loc[corr_mat['variable_1'] != corr_mat['variable_2'], :]
    corr_mat['abs_r'] = np.abs(corr_mat['r'])
    corr_mat = corr_mat.sort_values('abs_r', ascending=False)

    return(corr_mat)

tidy_corr_matrix(corr_matrix).head(10)
```

```
[39]:
```

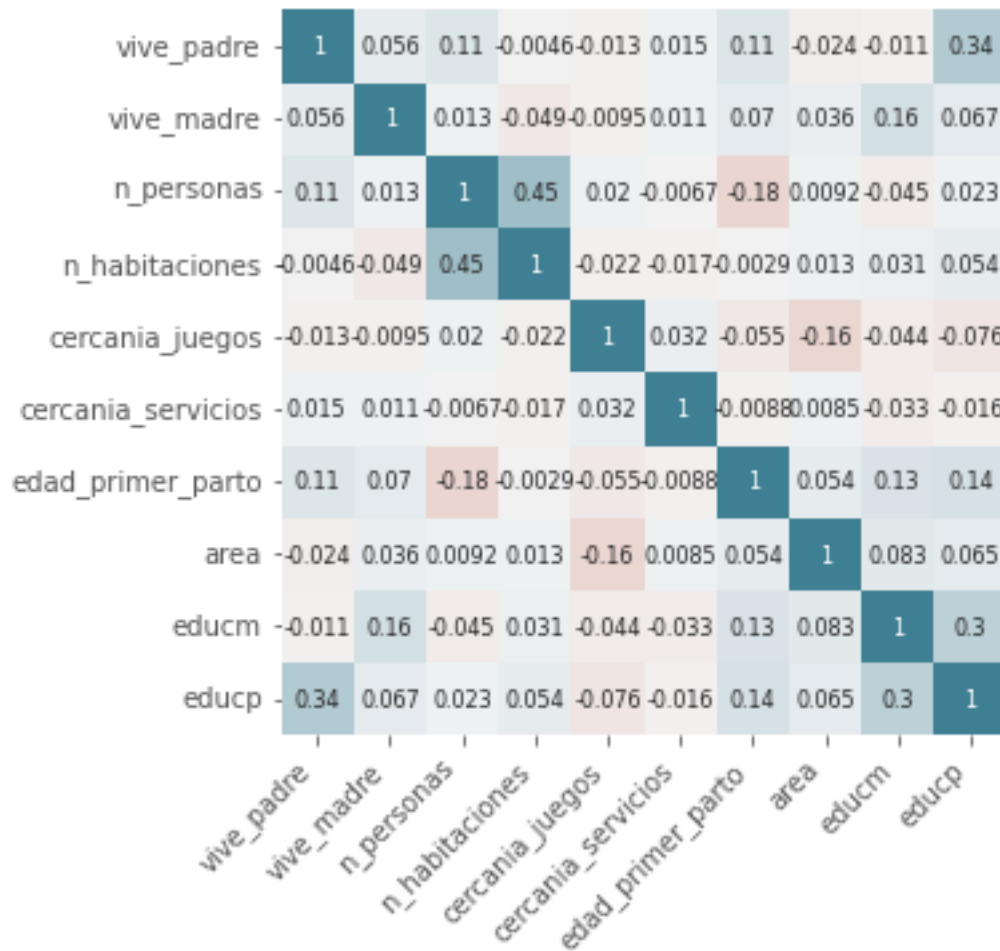
	variable_1	variable_2	r	abs_r
32	n_habitaciones	n_personas	0.445391	0.445391
23	n_personas	n_habitaciones	0.445391	0.445391
90	educp	vive_padre	0.344436	0.344436
9	vive_padre	educp	0.344436	0.344436
89	educm	educp	0.296504	0.296504
98	educp	educm	0.296504	0.296504
62	edad_primer_parto	n_personas	-0.181605	0.181605
26	n_personas	edad_primer_parto	-0.181605	0.181605
81	educm	vive_madre	0.163277	0.163277
18	vive_madre	educm	0.163277	0.163277

```
[40]: # Heatmap matriz de correlaciones
# =====
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(5, 5))

sns.heatmap(
    corr_matrix,
    annot      = True,
    cbar       = False,
    annot_kws  = {"size": 8},
    vmin       = -1,
    vmax       = 1,
    center     = 0,
    cmap       = sns.diverging_palette(20, 220, n=200),
    square     = True,
    ax         = ax
)

ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation = 45,
    horizontalalignment = 'right',
)

ax.tick_params(labelsize = 10)
```



```
[41]: def calculateVIF(var_predictoras_df):
    var_pred_labels = list(var_predictoras_df.columns)
    num_var_pred = len(var_pred_labels)

    lr_model = LinearRegression()

    result = pd.DataFrame(index = ['VIF'], columns = var_pred_labels)
    result = result.fillna(0)

    for ite in range(num_var_pred):
        x_features = var_pred_labels[:]
        y_feature = var_pred_labels[ite]
        x_features.remove(y_feature)

        x = var_predictoras_df[x_features]
        y = var_predictoras_df[y_feature]
```

```

        lr_model.fit(var_predictoras_df[x_features],
↪var_predictoras_df[y_feature])

        result[y_feature] = 1/(1 - lr_model.
↪score(var_predictoras_df[x_features], var_predictoras_df[y_feature]))

    return result
calculateVIF(X.copy(deep = True)).T

```

```

[41]:
          VIF
const      inf
vive_madre  1.037656
n_personas  1.309967
n_habitaciones  1.268409
edad_primer_parto  1.077390
area        1.011015
educm       1.138913
educp       1.115835

```

0.0.3 Probit

Aqui se emplea el modelo probit en donde los residuos se distribuyen de manera normal

```

[42]: model = sm.Probit(y, X)
probit_model = model.fit()
print(probit_model.summary())
mfx = probit_model.get_margeff(at='overall')
print(mfx.summary())

```

Optimization terminated successfully.

Current function value: 0.540627

Iterations 5

```

          Probit Regression Results
=====
Dep. Variable:          vive_padre  No. Observations:          5501
Model:                  Probit      Df Residuals:              5493
Method:                  MLE        Df Model:                  7
Date:                   Thu, 15 Sep 2022  Pseudo R-squ.:          0.1289
Time:                   20:13:17      Log-Likelihood:          -2974.0
converged:              True         LL-Null:                -3414.2
Covariance Type:        nonrobust     LLR p-value:             8.283e-186
=====
=====
          coef      std err          z      P>|z|      [0.025
0.975]
-----
-----

```

const	-1.1454	0.170	-6.747	0.000	-1.478
-0.813					
vive_madre	0.3280	0.116	2.829	0.005	0.101
0.555					
n_personas	0.1644	0.016	10.269	0.000	0.133
0.196					
n_habitaciones	-0.1321	0.023	-5.868	0.000	-0.176
-0.088					
edad_primer_parto	0.0329	0.004	8.563	0.000	0.025
0.040					
area	-0.2421	0.072	-3.356	0.001	-0.383
-0.101					
educm	-0.0477	0.005	-9.024	0.000	-0.058
-0.037					
educp	0.0947	0.004	24.510	0.000	0.087
0.102					

=====

=====

Probit Marginal Effects

Dep. Variable: vive_padre

Method: dydx

At: overall

=====

=====

	dy/dx	std err	z	P> z	[0.025
--	-------	---------	---	------	--------

0.975]

vive_madre	0.1009	0.036	2.834	0.005	0.031
0.171					
n_personas	0.0506	0.005	10.500	0.000	0.041
0.060					
n_habitaciones	-0.0406	0.007	-5.913	0.000	-0.054
-0.027					
edad_primer_parto	0.0101	0.001	8.700	0.000	0.008
0.012					
area	-0.0744	0.022	-3.366	0.001	-0.118
-0.031					
educm	-0.0147	0.002	-9.169	0.000	-0.018
-0.012					
educp	0.0291	0.001	29.147	0.000	0.027
0.031					

=====

=====

analisis prediccion


```
[43]: print(junaeb["vive_padre"].value_counts())
print("Porcentaje de padres que viven en la residencia vs los que no viven")
print(100*junaeb["vive_padre"].value_counts(normalize=True))
```

```
1    3785
0    1716
Name: vive_padre, dtype: int64
Porcentaje de padres que viven en la residencia vs los que no viven
1    68.805672
0    31.194328
Name: vive_padre, dtype: float64
```

```
[44]: X_train = sm.add_constant(X_train, prepend=True)
modelo = sm.Probit(endog=y_train, exog=X_train,)
modelo = modelo.fit()
print(modelo.summary())
```

```
Optimization terminated successfully.
Current function value: 0.543263
Iterations 5
```

Probit Regression Results

```
=====
Dep. Variable:          vive_padre  No. Observations:          3850
Model:                  Probit      Df Residuals:          3842
Method:                  MLE        Df Model:              7
Date:                   Thu, 15 Sep 2022  Pseudo R-squ.:          0.1278
Time:                   20:13:18      Log-Likelihood:         -2091.6
converged:               True        LL-Null:              -2398.1
Covariance Type:         nonrobust    LLR p-value:           3.601e-128
=====
=====
```

	coef	std err	z	P> z	[0.025
0.975]					

const	-1.1121	0.200	-5.561	0.000	-1.504
-0.720					
vive_madre	0.3761	0.135	2.787	0.005	0.112
0.641					
n_personas	0.1583	0.019	8.396	0.000	0.121
0.195					
n_habitaciones	-0.1078	0.026	-4.096	0.000	-0.159
-0.056					
edad_primer_parto	0.0316	0.005	6.839	0.000	0.023
0.041					
area	-0.2623	0.088	-2.993	0.003	-0.434
-0.091					
educm	-0.0529	0.006	-8.309	0.000	-0.065

```
-0.040
educp          0.0943      0.005      20.388      0.000      0.085
0.103
```

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

```
[45]: # Accuracy de test del modelo
# =====
X_test = sm.add_constant(X_test, prepend=True)
predicciones = modelo.predict(exog = X_test)
clasificacion = np.where(predicciones<0.5, 0, 1)
accuracy = accuracy_score(
    y_true = y_test,
    y_pred = clasificacion,
    normalize = True
)
print("")
print(f"El accuracy de test es: {100*accuracy}%")
```

El accuracy de test es: 77.83161720169595%

```
[46]: # Matriz de confusión de las predicciones de test
# =====
confusion_matrix = pd.crosstab(
    y_test.ravel(),
    clasificacion,
    rownames=['Real'],
    colnames=['Predicción']
)
confusion_matrix
```

```
[46]: Predicción    0    1
Real
0          181   323
1           43  1104
```

```
[47]: # Accuracy de test del modelo utilizando un umbral de 0.68 equivalente al
      ↪ Porcentaje de padres que viven en la residencia vs los que no viven
# =====
X_test = sm.add_constant(X_test, prepend=True)
predicciones = modelo.predict(exog = X_test)
clasificacion = np.where(predicciones<0.68, 0, 1)
accuracy = accuracy_score(
    y_true = y_test,
    y_pred = clasificacion,
    normalize = True
)
```

```
)
print("")
print(f"El accuracy de test es: {100*accuracy}%")
```

El accuracy de test es: 70.07874015748031%

```
[48]: # Matriz de confusión de las predicciones de test
```

```
# =====
confusion_matrix = pd.crosstab(
    y_test.ravel(),
    clasificacion,
    rownames=['Real'],
    colnames=['Predicción']
)
confusion_matrix
```

```
[48]: Predicción    0    1
Real
0           293  211
1           283  864
```

0.0.4 Logit

```
[49]: 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: 0.538392

Iterations 6

Logit Regression Results

```
=====
Dep. Variable:    vive_padre    No. Observations:    5501
Model:            Logit        Df Residuals:           5493
Method:           MLE         Df Model:              7
Date:            Thu, 15 Sep 2022    Pseudo R-squ.:        0.1325
Time:            20:13:18          Log-Likelihood:       -2961.7
converged:        True           LL-Null:            -3414.2
Covariance Type:  nonrobust       LLR p-value:         4.058e-191
=====
```

```
=====
coef    std err          z      P>|z|      [0.025
0.975]
```

```

-----
-----
const          -2.0050      0.291      -6.886      0.000      -2.576
-1.434
vive_madre      0.5851      0.193       3.025      0.002       0.206
0.964
n_personas      0.2993      0.030     10.075      0.000       0.241
0.358
n_habitaciones -0.2355      0.041     -5.700      0.000     -0.316
-0.155
edad_primer_parto 0.0580      0.007       8.617      0.000       0.045
0.071
area           -0.4043      0.124     -3.263      0.001     -0.647
-0.161
educm          -0.0884      0.010     -9.226      0.000     -0.107
-0.070
educp          0.1617      0.007     23.789      0.000       0.148
0.175
=====
=====
Logit Marginal Effects
=====
Dep. Variable:      vive_padre
Method:             dydx
At:                 overall
=====
=====
              dy/dx      std err          z      P>|z|      [0.025
0.975]
-----
-----
vive_madre      0.1048      0.035       3.034      0.002       0.037
0.172
n_personas      0.0536      0.005     10.368      0.000       0.043
0.064
n_habitaciones -0.0422      0.007     -5.753      0.000     -0.057
-0.028
edad_primer_parto 0.0104      0.001       8.784      0.000       0.008
0.013
area           -0.0724      0.022     -3.274      0.001     -0.116
-0.029
educm          -0.0158      0.002     -9.464      0.000     -0.019
-0.013
educp          0.0290      0.001     29.561      0.000       0.027
0.031
=====
=====

```

```
[50]: # Creación del modelo utilizando matrices como en scikitlearn
# =====
# A la matriz de predictores se le tiene que añadir una columna de 1s para el
# ↪ intercept del modelo
X_train = sm.add_constant(X_train, prepend=True)
modelo = sm.Logit(endog=y_train, exog=X_train,)
modelo = modelo.fit()
print(modelo.summary())
```

Optimization terminated successfully.

Current function value: 0.540975

Iterations 6

Logit Regression Results

```
=====
Dep. Variable:          vive_padre    No. Observations:          3850
Model:                  Logit         Df Residuals:              3842
Method:                  MLE          Df Model:                  7
Date:                   Thu, 15 Sep 2022    Pseudo R-squ.:            0.1315
Time:                   20:13:18           Log-Likelihood:           -2082.8
converged:               True             LL-Null:                  -2398.1
Covariance Type:         nonrobust         LLR p-value:              5.783e-132
=====
=====
              coef      std err          z      P>|z|      [0.025
0.975]
-----
-----
const          -1.9632      0.344      -5.700      0.000      -2.638
-1.288
vive_madre       0.6674      0.228       2.933      0.003       0.221
1.113
n_personas       0.2918      0.035       8.288      0.000       0.223
0.361
n_habitaciones  -0.1939      0.048      -4.014      0.000      -0.289
-0.099
edad_primer_parto  0.0553      0.008       6.863      0.000       0.039
0.071
area            -0.4324      0.150      -2.878      0.004      -0.727
-0.138
educm           -0.0972      0.012      -8.395      0.000      -0.120
-0.074
educp            0.1610      0.008     19.750      0.000       0.145
0.177
=====
=====
```

```
[51]: predicciones = modelo.predict(exog = X_train)

# Clasificación predicha
# =====
clasificacion = np.where(predicciones<0.5, 0, 1)
clasificacion
```

```
[51]: array([1, 1, 1, ..., 1, 1, 1])
```

```
[52]: # Accuracy de test del modelo
X_test = sm.add_constant(X_test, prepend=True)
predicciones = modelo.predict(exog = X_test)
clasificacion = np.where(predicciones<0.5, 0, 1)
accuracy = accuracy_score(
    y_true = y_test,
    y_pred = clasificacion,
    normalize = True
)
print("")
print(f"El accuracy de test es: {100*accuracy}%")
```

El accuracy de test es: 77.71047849788008%

```
[53]: X_test = sm.add_constant(X_test, prepend=True)
predicciones = modelo.predict(exog = X_test)
clasificacion = np.where(predicciones<0.68, 0, 1)
accuracy = accuracy_score(
    y_true = y_test,
    y_pred = clasificacion,
    normalize = True
)
print("")
print(f"El accuracy de test es: {100*accuracy}%")
```

El accuracy de test es: 70.44215626892793%

```
[54]: # Matriz de confusión de las predicciones de test
confusion_matrix = pd.crosstab(
    y_test.ravel(),
    clasificacion,
    rownames=['Real'],
    colnames=['Predicción']
)
confusion_matrix
```

```
[54]: Predicción    0    1
      Real
      0          290  214
      1          274  873
```

0.0.5 Poisson

```
[55]: y=junaeb['n_personas']
      X_0=junaeb[["vive_padre","vive_madre","n_habitaciones","edad_primer_parto","area","educm","educp"]]
      poisson=sm.GLM(y,X_0,family=sm.families.Poisson()).fit()
      print(poisson.summary())
```

```

                                Generalized Linear Model Regression Results
=====
Dep. Variable:                  n_personas    No. Observations:                  5501
Model:                            GLM        Df Residuals:                      5492
Model Family:                     Poisson     Df Model:                          8
Link Function:                     Log        Scale:                          1.0000
Method:                           IRLS       Log-Likelihood:                     -10081.
Date:                            Thu, 15 Sep 2022    Deviance:                          1934.4
Time:                            20:13:18         Pearson chi2:                       2.09e+03
No. Iterations:                    5             Pseudo R-squ. (CS):                  0.03630
Covariance Type:                   nonrobust
=====
=====
                                coef      std err          z      P>|z|      [0.025
0.975]
-----
-----
vive_padre          0.1079      0.015      7.038      0.000      0.078
0.138
vive_madre          0.5643      0.041     13.814      0.000      0.484
0.644
n_habitaciones      0.1460      0.005     28.829      0.000      0.136
0.156
edad_primer_parto   -0.0044      0.001     -3.610      0.000     -0.007
-0.002
area                0.1756      0.025      6.974      0.000      0.126
0.225
educm               0.0018      0.002      1.011      0.312     -0.002
0.005
educp              0.0007      0.001      0.484      0.628     -0.002
0.003
cercania_juegos     0.1438      0.018      8.149      0.000      0.109
0.178
cercania_servicios  0.1866      0.023      7.979      0.000      0.141
0.232

```

```

=====
[56]: y=junaeb['n_personas']
      X=junaeb[["vive_padre","vive_madre","n_habitaciones","edad_primer_parto","area","cercania_jueg
      poisson=sm.GLM(y,X,family=sm.families.Poisson()).fit()
      print(poisson.summary())

```

Generalized Linear Model Regression Results

```

=====
Dep. Variable:          n_personas    No. Observations:          5501
Model:                  GLM           Df Residuals:              5494
Model Family:           Poisson       Df Model:                  6
Link Function:          Log           Scale:                    1.0000
Method:                 IRLS          Log-Likelihood:          -10082.
Date:                   Thu, 15 Sep 2022    Deviance:                1936.1
Time:                   20:13:18           Pearson chi2:            2.10e+03
No. Iterations:         5                Pseudo R-squ. (CS):      0.03600
Covariance Type:        nonrobust
=====

```

```

=====
              coef      std err          z      P>|z|      [0.025
0.975]
-----
-----
vive_padre      0.1101      0.014      7.679      0.000      0.082
0.138
vive_madre      0.5800      0.039     14.869      0.000      0.504
0.657
n_habitaciones  0.1466      0.005     29.113      0.000      0.137
0.156
edad_primer_parto -0.0041      0.001     -3.420      0.001     -0.006
-0.002
area            0.1804      0.025      7.233      0.000      0.131
0.229
cercania_juegos  0.1442      0.018      8.183      0.000      0.110
0.179
cercania_servicios 0.1869      0.023      8.006      0.000      0.141
0.233
=====
=====

```

```

[57]: import math
      Coef=[]
      for i in range (len(poisson.params)):
          math.exp(poisson.params[i])
          Coef.append((math.exp(poisson.params[i])-1))

```



```
[58]: Coef
coef_df = pd.DataFrame(Coef,
                        columns=['Betas_Poisson'],
                        index=['vive_padre', 'vive_madre', 'n_habitaciones', 'edad_primer_parto', "area", "cercania_jueg",
coef_df
```

```
[58]:
```

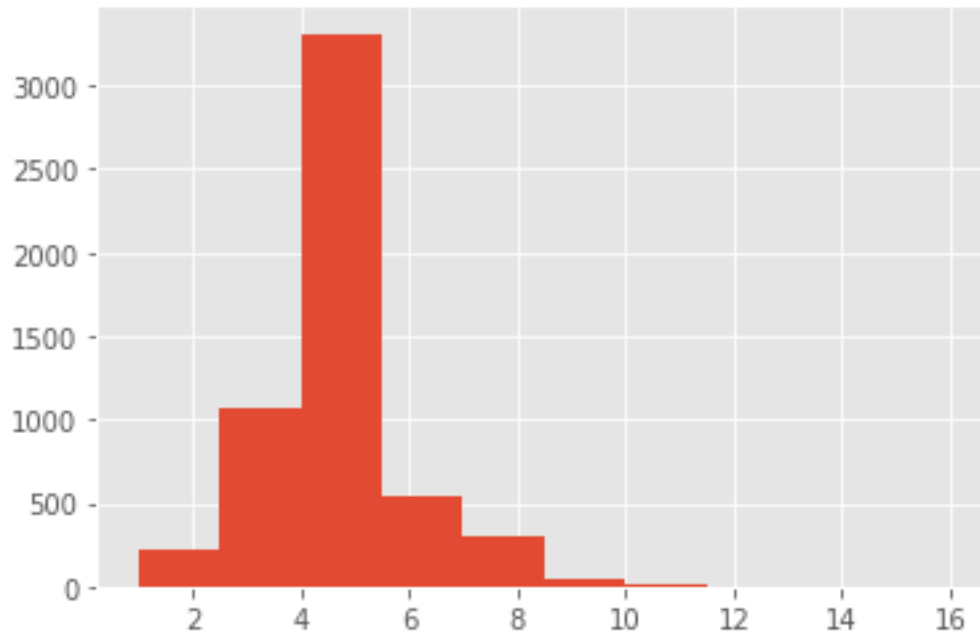
	Betas_Poisson
vive_padre	0.116406
vive_madre	0.786125
n_habitaciones	0.157864
edad_primer_parto	-0.004069
area	0.197671
cercania_juegos	0.155063
cercania_servicios	0.205447

```
[59]: junaeb.n_personas.value_counts()
```

```
[59]: 4.0    2075
5.0    1237
3.0    1067
6.0     541
2.0     223
7.0     207
8.0      94
9.0      38
10.0      7
11.0      6
12.0      2
16.0      1
14.0      1
15.0      1
1.0       1
Name: n_personas, dtype: int64
```

```
[60]: plt.hist(junaeb.n_personas)
```

```
[60]: (array([2.240e+02, 1.067e+03, 3.312e+03, 5.410e+02, 3.010e+02, 3.800e+01,
1.300e+01, 2.000e+00, 1.000e+00, 2.000e+00]),
array([ 1. ,  2.5,  4. ,  5.5,  7. ,  8.5, 10. , 11.5, 13. , 14.5, 16. ]),
<BarContainer object of 10 artists>)
```



```
[61]: poisson=sm.GLM(y,X,family=sm.families.Poisson()).fit()
      print(poisson.summary())
```

Generalized Linear Model Regression Results

```
=====
Dep. Variable:          n_personas    No. Observations:          5501
Model:                  GLM           Df Residuals:                5494
Model Family:           Poisson       Df Model:                   6
Link Function:           Log           Scale:                   1.0000
Method:                 IRLS          Log-Likelihood:          -10082.
Date:                   Thu, 15 Sep 2022    Deviance:                1936.1
Time:                   20:13:19           Pearson chi2:            2.10e+03
No. Iterations:         5                Pseudo R-squ. (CS):      0.03600
Covariance Type:        nonrobust
=====
```

```
=====
              coef    std err          z      P>|z|      [0.025
0.975]
-----
-----
vive_padre      0.1101     0.014     7.679     0.000     0.082
0.138
vive_madre      0.5800     0.039    14.869     0.000     0.504
0.657
n_habitaciones  0.1466     0.005    29.113     0.000     0.137
0.156
```

edad_primer_parto	-0.0041	0.001	-3.420	0.001	-0.006
-0.002					
area	0.1804	0.025	7.233	0.000	0.131
0.229					
cercania_juegos	0.1442	0.018	8.183	0.000	0.110
0.179					
cercania_servicios	0.1869	0.023	8.006	0.000	0.141
0.233					

=====

=====

Analisis Prediccion

```
[62]: X_train, X_test, y_train, y_test = train_test_split(
        X,
        y,
        train_size = 0.7,
        random_state = 1,
        shuffle = True
    )

# A la matriz de predictores se le tiene que añadir una columna de 1s para el
↪intercept del modelo
X_train = sm.add_constant(X_train, prepend=True)
poisson=sm.GLM(endog = y, exog = X,family=sm.families.Poisson()).fit()
print(poisson.summary())
```

Generalized Linear Model Regression Results

```
=====
Dep. Variable:          n_personas    No. Observations:          5501
Model:                  GLM           Df Residuals:              5494
Model Family:           Poisson       Df Model:                  6
Link Function:          Log           Scale:                    1.0000
Method:                 IRLS          Log-Likelihood:          -10082.
Date:                   Thu, 15 Sep 2022    Deviance:                 1936.1
Time:                   20:13:19           Pearson chi2:             2.10e+03
No. Iterations:         5               Pseudo R-squ. (CS):       0.03600
Covariance Type:        nonrobust
=====
```

```
=====
               coef      std err          z      P>|z|      [0.025
0.975]
-----
vive_padre      0.1101      0.014      7.679      0.000      0.082
0.138
vive_madre      0.5800      0.039     14.869      0.000      0.504
0.657
n_habitaciones  0.1466      0.005     29.113      0.000      0.137
```

```

0.156
edad_primer_parto      -0.0041      0.001      -3.420      0.001      -0.006
-0.002
area                   0.1804      0.025       7.233      0.000      0.131
0.229
cerkania_juegos        0.1442      0.018       8.183      0.000      0.110
0.179
cerkania_servicios     0.1869      0.023       8.006      0.000      0.141
0.233

```

```

=====
=====

```

```

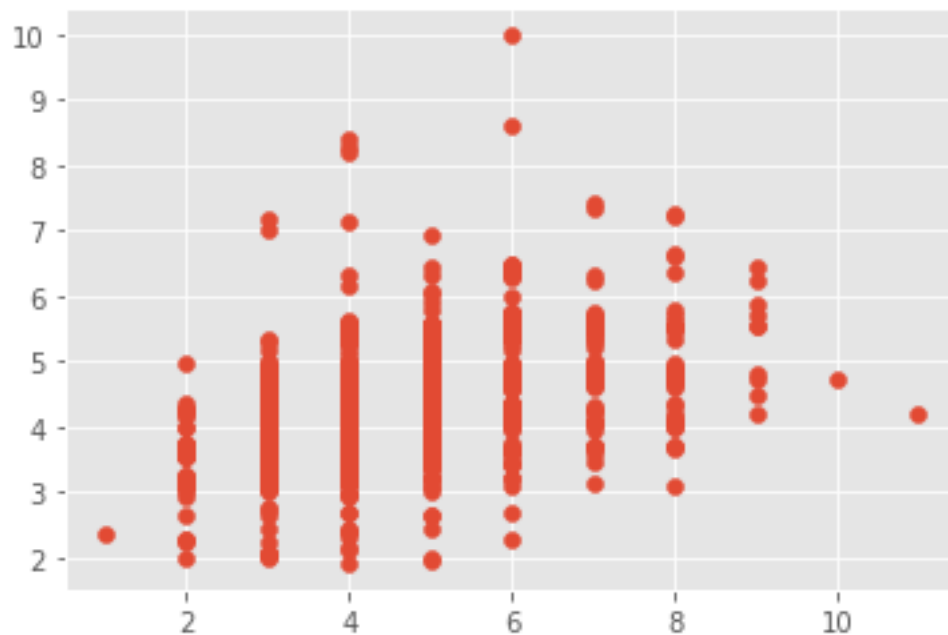
[63]: prediccion_train = poisson.predict(exog = X_test)
      residuos_train   = prediccion_train - y_test
      plt.scatter(y_test,prediccion_train)

```

```

[63]: <matplotlib.collections.PathCollection at 0x23b518b5100>

```



0.0.6 test sobre-dispersion

```

[64]: media = junaeb["n_personas"].mean()
      std = junaeb["n_personas"].std(ddof=0)
      var = junaeb["n_personas"].var(ddof=0)
      print("Media : ",media,"Desviacion estandar :",std, "Varianza : ",var)
      poisson.mu.mean()

```

Media : 4.385929830939829 Desviacion estandar : 1.3419030086363548 Varianza : 1.800703684587301

[64]: 4.340626008842084

```
[65]: print("Ratio pearson_chi2/df_resid =", poisson.pearson_chi2/poisson.df_resid)
```

Ratio pearson_chi2/df_resid = 0.3815161686244171

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

x1 -0.113728
dtype: float64

Binomial negativa

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

*# este comando asume un alpha = 1 sin embargo el alpha es negativo por lo que
↪no se interpretaran sus coeficientes.*

Generalized Linear Model Regression Results

```
=====
Dep. Variable:          n_personas    No. Observations:          5501
Model:                  GLM          Df Residuals:              5494
Model Family:          NegativeBinomial    Df Model:              6
Link Function:          Log            Scale:              1.0000
Method:                 IRLS          Log-Likelihood:        -14189.
Date:                  Thu, 15 Sep 2022    Deviance:              336.70
Time:                  20:13:19          Pearson chi2:          374.
No. Iterations:         13          Pseudo R-squ. (CS):      0.01031
Covariance Type:        nonrobust
=====
```

```
=====
              coef      std err          z      P>|z|      [0.025
0.975]
-----
vive_padre      0.1164      0.033      3.564      0.000      0.052
0.180
vive_madre      0.3956      0.082      4.853      0.000      0.236
0.555
n_habitaciones  0.1897      0.015     12.249      0.000      0.159
0.220
edad_primer_parto -0.0050      0.003     -1.828      0.068     -0.010
0.000
```

area	0.1448	0.055	2.651	0.008	0.038
0.252					
cercania_juegos	0.1361	0.042	3.256	0.001	0.054
0.218					
cercania_servicios	0.3198	0.064	4.972	0.000	0.194
0.446					

=====

=====

```
[68]: print("fitted lambda")
      print(negbin.mu)
```

```
fitted lambda
[5.1083008  4.26769778 4.89076875 ... 3.24362766 3.84901098 4.12153482]
```

OLS Podemos notar a través de los coeficientes del modelo OLS, que la proporción de que el padre viva en la casa disminuye si el lugar donde habita es en zona urbana. Además, la proporción de que el padre viva en la casa disminuye a medida que la educación de la madre se incrementa y también la proporción de que el padre viva en la casa disminuye si se incrementa el número de habitaciones. Estas tres variables hacen que la probabilidad de que el padre viva en la casa se vea disminuida.

Por otro lado las variables vive_madre, n_personas, edad_primer parto, educp hacen que la probabilidad de que el padre viva en la casa aumente, por ejemplo si la madre vive en la casa la proporción de que el padre viva en la casa aumenta.

Todas las variables son significativas a excepción de carcania_juegos y cercania_servicios por lo que fueron eliminadas del modelo al tener un valor $p > 0.05$

Como resultado tenemos un $R^2 = 0.173$ el cual no es un valor deseable ya que esperamos que sea cercano a 1.

Test F es Significativo ya que su valor $p < 0.05$, así, podemos notar que las variables explicativas en su conjunto son capaces de explicar el modelo ya que el valor p del test F es inferior al 5%.

Probit Para la interpretación de los coeficientes del modelo probit se utilizó el método de efecto parcial promedio (EPP) a través de la función `get_margeff(at='overall')`.

-Si la variable vive_madre recibe el valor 1 vs 0 la probabilidad de ocurrencia de la variable vive_padre aumenta en 0.1009. Así se puede decir ante un aumento en una unidad de la variable n_personas la probabilidad de ocurrencia de la variable vive_padre aumenta en 0.0506.

-Ante un aumento en una unidad de la variable n_habitaciones la probabilidad de ocurrencia de la variable vive_padre disminuye en 0.0406

-Ante un aumento en una unidad de la variable edad_primer_parto la probabilidad de ocurrencia de la variable vive_padre aumenta en 0.0101

-Si la variable área recibe el 1 vs 0 es decir que pertenece a zona urbana la probabilidad de ocurrencia de la variable vive_padre disminuye en 0.0744

-Si la variable educm aumenta en unidad la probabilidad de ocurrencia de la variable vive_padre aumenta en 0.0147

-Si la variable educp aumenta en unidad la probabilidad de ocurrencia de la variable vive_padre disminuye en 0.0291 Entiéndase como vive_padre = 1 es la ocurrencia es decir de que el padre viva en la casa.

El pseudo R-cuadrado utilizado es el de Mcfadden, la formula para calcularlo es $1 - (\text{Log-Likelihood}/\text{LL-Null})$ y para este caso es igual a 0.1278 al ser este valor mayor a 0 se puede decir que el modelo propuesto mejora al modelo nulo.

Logit Para interpretar los coeficientes betas del modelo logit se utilizó el método de efecto parcial promedio (EPP) a través de la función `get_margeff(at='overall')`.

-Así se puede decir ante un aumento en una unidad de la variable n_personas la probabilidad de ocurrencia de la variable vive_padre aumenta en 0.0536.

-Si la variable vive_madre recibe el valor 1 vs 0 la probabilidad de ocurrencia de la variable vive_padre aumenta en 0.1048

-Ante un aumento en una unidad de la variable n_habitaciones la probabilidad de ocurrencia de la variable vive_padre disminuye en 0.0422

-Un aumento de la edad_primer_parto la probabilidad de ocurrencia de vive_padre aumenta en 0.0104

-Si la variable área recibe el 1 vs 0 es decir que pertenece a zona urbana la probabilidad de ocurrencia de la variable vive_padre disminuye en 0.0724

-Si la variable educm aumenta en unidad la probabilidad de ocurrencia de la variable vive_padre disminuye en 0.0291

-Si la variable educp aumenta en unidad la probabilidad de ocurrencia de la variable vive_padre aumenta en 0.0290

El pseudo R-cuadrado utilizado es el de Mcfadden, la formula para calcularlo es $1 - (\text{Log-Likelihood}/\text{LL-Null})$ y para este caso es igual a 0.1325 al ser este valor mayor a 0 se puede decir que el modelo propuesto mejora al modelo nulo.

Comparacion modelo OLS PROBIT y LOGIT El modelo OLS no es el más adecuado para estudiar variables dependientes limitadas ya que presenta ciertas desventajas, como vimos en clases cuando la variable dependiente es binaria es decir no continua como lo es en este caso de estudio, solo puede tomar valores 0 o 1, esto implica que los errores no se distribuyan de manera normal lo que afecta a uno de los supuestos de la OLS, lo dicho anteriormente también se puede confirmar con las graficas Q-Q o el test de shapiro wilk el cual entrega un valor $P < 0.05$, causando que los estimadores no sean MELI. También las predicciones que entrega el modelo OLS son valores continuos y muchos se escapan del rango [0-1] por lo que no tienen sentido ciertos valores para explicar la variable dependiente limitada. Aunque el modelo OLS se puede utilizar para para identificar las variables que si son significativas para el modelo aun cuando no se cumpla el supuesto de normalidad.

Los modelos Probit y Logit son utilizados para este tipo de casos donde se tiene una variable dependiente limitada que puede tomar valores 0 o 1. Estos modelos presentan una ventaja sobre

la OLS ya que limitan los resultados de las predicciones en un intervalo [0-1] , además son modelos que tienen como propósito estimar el efecto de la variable independiente sobre la probabilidad de éxito $P(y=1 | x)$.

Se hizo un análisis de predicción utilizando un umbral de 0.5 y un umbral de 0.68 equivalente al porcentaje de padres que viven en la residencia eso hace referencia a que si el valor predicho es inferior al umbral la variable predicha vive_padre toma el valor de 0 y 1 en caso contrario. Como podemos ver en el código el modelo Probit con un umbral de 0.5 entrega una probabilidad de predecir correctamente de 77.83% y con un umbral de 0.68 entrega una probabilidad de predecir correctamente de 70.07%. Para el modelo Logit se utilizó la misma metodología donde para un umbral de 0.5 el modelo entrega una probabilidad de predecir correctamente de 77.7% con las variables explicativas seleccionadas y con un umbral de 0.68 el modelo entrega una probabilidad de predecir correctamente de 70.4%.

Podemos notar que no hay una gran diferencia entre ambos modelos a la hora de predecir correctamente, esto utilizando un umbral igual a 0.5 lo que significa de que si la variable es predicha es inferior a 0.5 entrega el valor de 0 y 1 en caso contrario por lo que esta forma de predecir puede estar sujeta a errores según la literatura.

No hay una razón para preferir el modelo probit por sobre el modelo logit ya que presentan resultados similares, sin embargo ambos modelos presentan ventajas para este tipo de análisis sobre el OLS ya que la variable dependiente es una variable dependiente limitada (Binaria)

Modelo poisson Utilizando $\alpha = 0.05$ y con los resultados obtenidos tenemos que todas las variables son significativas a excepción de las variables educm y educp ya que sus valores p son mayores a 0.05



El n_personas esperadas que viven en el hogar para cuando el padre esta en la casa se estima que es 11.64% mayor que cuando el padre no se encuentra en la casa.

El n_personas esperados que viven en el hogar para cuando la madre vive en la casa se estima que es 78,61% mayor que cuando la madre no vive en la casa.

El n_personas esperadas que viven en el hogar cuando se vive en zona urbana se estima que es 19.76% mayor que cuando se vive en zona rural

El n_personas esperadas que viven en el hogar cuando no hay servicios de salud cerca de este se estima que es 20.54% mayor que cuando si hay servicios de salud

El n_personas esperadas que viven en el hogar cuando no hay juegos infantiles cerca se estima que es 15.50% mayor que cuando se vive cerca de juegos infantiles

Test sobre dispersion No existe sobre dispersión dado que el valor de $\alpha = -0.108976$ y además

ratio pearson_chi2/df_resid = 0.38 es menor a 1.

Modelo binomial negativo Dado que α es un valor estimado desde la data como aproximación por ende no es un valor arbitrario que podemos asignar, al ser este valor de α negativo para los datos utilizados, se concluye que, no se puede ejecutar un modelo binomial negativo para explicar el número de personas que hay dentro de un hogar

Comparacion de modelos Poisson y Binomial Negativo Las diferencias en los resultados se presentan debido a que el modelo binomial negativo es un metodo para relajar el modelo poisson y esto se utiliza cuando el modelo poisson presenta sobredispersión es decir la varianza condicional es mayor a la media condicional y como argumentaremos mas adelante esto no se cumple.

Preeliminarmente podemos observar que la variable `n_personas` no presenta problemas de sobredispersión mas bien están subdispersos o pocos dispersos esto se visualiza a traves del histograma, donde los datos se concentran en los valores 3 , 4 ,5 ,6. Los datos estarían sobredispersos si la varianza condicional supera la media condicional.

Podemos calcular la media de la variable de conteo y la varianza de la variable de conteo en donde los resultados arrojan que la media es mucho mayor que la varianza si bien no son las medias y varianzas condicionales esta comparación es útil ya que, “La media de la media condicional no cambiará, porque la media de las medias ajustadas es igual a la media de la muestra y la regresión de Poisson posterior disminuye un poco la varianza condicional de la variable dependiente.” (Regression análisis of count second edition, A. Colin Cameron , Pravin K. Trivedi Pag 89.).Esto indicaria que la diferencia entre la media y la varianza condicionales es aun mayor.

Luego de este análisis preliminar podemos utilizar el Test de sobredispersión para validar las observaciones preliminares, asi, a través del ratio Pearson χ^2 / Df Residuals el cual nos brinda un valor de 0.38 menor a 1 ademas de un valor $\text{Alpha} < 0$ podriamos concluir que los datos estan pocos dispersos por lo que poisson es el modelo aconsejable para manejar la variable de conteo `n_personas`.

[]: