Implementación del método de la máxima verosimilitud para la regresión logística.

Definir la función de entorno L(b)

$L(\beta) = \sum i = 1^n P_i^{y_i} (1 - Pi)^{1-y_i}$

In [1]:
```
def likelihood (y, pi):
    import numpy as np
    total_sum = 1
    sum_in = list(range(1, len(y)+1))
    for i in range(len(y)):
        sum_in[i] = np.where(y[i]==1, pi[i], 1-pi[i])
        total_sum = total_sum * sum_in[i]
    return total_sum
```

Calcular las probabilidades para cada observación

$L(\beta) = P_i = P(x_i) = \frac{1}{1+e^{-\sum_{j=0}^{k}(\beta_j \cdot X_{ij})}}$

In [2]:
```
def logitprobs(X, beta):
    import numpy as np
    n_rows = np.shape(X)[0]
    n_cols = np.shape(X)[1]
    pi = list(range (1, n_rows+1))
    expon = list(range(1, n_rows+1))
    for i in range (n_rows): # Este for es para las i, o sea para las probabilidades
        expon[i] = 0
        for j in range(n_cols):  # Calculo el sumatorio de las k columnas, y saco el exponente
            ex = X[i][j] * beta[j] #Producto de los exponentes Beta_j y X_ij
            expon[i] = ex + expon[i] #Suma total de los exponentes
        with np.errstate(divide="ignore", invalid="ignore"):
            pi[i]=1/(1+np.exp(-expon[i]))
    return pi
```

Calcular la matriz diagonal W

$\omega_{(\beta)} = diag(P_i \cdot (1 - P_i))_{i=1}^{n}$

In [3]:
```
def findW(pi):
    import numpy as np
    n = len(pi)
    W = np.zeros(n*n).reshape(n,n) #Creo un vector de 0s  y luego una matriz de n x n de ceros con resha
    for i in range (n):
        print(i)
        W[i, i] = pi[i]*(1-pi[i])
        W[i, i].astype(float)
    return W
```

Definir la función logística

$\Delta\beta = \frac{f(\beta)}{f'(\beta)}$

$f(\beta) = \frac{\delta l}{\delta \beta} = X \cdot (y - P(\beta))$

$f'(\beta) = \frac{\delta^2 l}{\delta \beta^2} = X \cdot \omega_(\beta) \cdot X^t$

In [34]:
```
def logistics (X, Y, limit): #El límite del cambio de un paso al siguiente hasta que converja
    from numpy import linalg
    nrow = np.shape(X)[0]
    bias = np.ones(nrow).reshape(nrow, 1)
    X_new = np.append (X, bias, axis = 1) #Agrego una columna de unos a la derecha del los datos, aux.
```

```
    ncol = np.shape(X_new)[1] #Número de columnas en la posición 1 porque tendrá una más que antes.
    beta = np.zeros(ncol).reshape(ncol, 1) #Toda una col de 0 de la misma lon que e ncol, apiladas
    root_dif = np.array(range(1,ncol+1)).reshape(ncol,1) #Diferencias entre las raíces / los betas.
    iter_i = 10000
    counter = 0
    while (iter_i>limit):
        print("El producto de iteraciones es iter_i: " + str(iter_i) + ", El límite es: " + str(limit))
        pi = logitprobs(X_new, beta)
        print ("El valor de las probabilidades parciales Pi es: " + str(pi))
        W = findW(pi)
        print("La matriz es W: " + str(W))
        #num y den, que son numerador y denominador se transponen para poder ser multiplicadas.
        num = (np.transpose(np.matrix(X_new))*np.matrix(Y-np.transpose(pi)).transpose())
        den = (np.matrix(np.transpose(X_new))*np.matrix(W)*np.matrix(X_new))
        root_dif = np.array(linalg.inv(den)*num) #Acá está el delta beta calculado
        beta = beta + root_dif
        print("El valor de beta es: " + str(beta))
        iter_i = np.sum(root_dif*root_dif)
        ll = likelihood(Y, pi) #Factor de verosimilitud
        counter += 1
        print("El n° iteraciones a la que converge es :" + str(counter))
    return beta
```

Comprobación experimental

In [35]:
```
import numpy as np
```

In [36]:
```
X = np.array(range(10)).reshape(10,1)
X
```

Out[36]:
```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5],
       [6],
       [7],
       [8],
       [9]])
```

In [37]:
```
Y = [0,0,0,0,1,0,1,0,1,1]
```

In [38]:
```
bias = np.ones(10).reshape(10,1)
X_new = np.append(X, bias, axis = 1)
```

In [39]:
```
X_new
```

Out[39]:
```
array([[0., 1.],
       [1., 1.],
       [2., 1.],
       [3., 1.],
       [4., 1.],
       [5., 1.],
       [6., 1.],
       [7., 1.],
       [8., 1.],
       [9., 1.]])
```

In [41]:
```
a = logistics(X, Y, 0.00001)
```

```
El producto de iteraciones es iter_i: 10000, El límite es: 1e-05
El valor de las probabilidades parciales Pi es: [array([0.5]), array([0.5]), array([0.5]), array([0.5]),
array([0.5]), array([0.5]), array([0.5]), array([0.5]), array([0.5]), array([0.5])]
0
1
2
3
```

```
4
5
6
7
8
9
La matriz es W: [[0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.25 0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25]]
El valor de beta es: [[ 0.43636364]
 [-2.36363636]]
El nº iteraciones a la que converge es :1
El producto de iteraciones es iter_i: 5.777190082644626, El límite es: 1e-05
El valor de las probabilidades parciales Pi es: [array([0.08598797]), array([0.12705276]), array([0.1837
8532]), array([0.2583532]), array([0.35019508]), array([0.45467026]), array([0.56329497]), array([0.6661
6913]), array([0.75533524]), array([0.82687453])]
0
1
2
3
4
5
6
7
8
9
La matriz es W: [[0.07859404 0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.11091035 0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.15000827 0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.19160683 0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.22755849 0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.24794521
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.24599375 0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.22238782 0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.18480392 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.14315304]]
El valor de beta es: [[ 0.60426056]
 [-3.34641372]]
El nº iteraciones a la que converge es :2
El producto de iteraciones es iter_i: 0.9940407075349111, El límite es: 1e-05
El valor de las probabilidades parciales Pi es: [array([0.0340128]), array([0.06053134]), array([0.10546
805]), array([0.1774629]), array([0.28305225]), array([0.41943069]), array([0.56933774]), array([0.70752
84]), array([0.81572841]), array([0.89011647])]
0
1
2
3
4
5
6
7
8
9
La matriz es W: [[0.03285593 0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.0568673  0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.09434454 0.         0.         0.
```

```
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.14596982 0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.20293367 0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.24350859
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.24519228 0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.20693196 0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.15031557 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.09780914]]
El valor de beta es: [[ 0.65761412]
 [-3.66759924]]
El nº iteraciones a la que converge es :3
El producto de iteraciones es iter_i: 0.10600674406801981, El límite es: 1e-05
El valor de las probabilidades parciales Pi es: [array([0.02490177]), array([0.04697681]), array([0.0868
775]), array([0.15515129]), array([0.26170168]), array([0.40624059]), array([0.56907679]), array([0.7182
3018]), array([0.83108181]), array([0.90473054])]
0
1
2
3
4
5
6
7
8
9
La matriz es W: [[0.02428167 0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.04476999 0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.0793298  0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.13107937 0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.19321391 0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.24120917
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.2452284  0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.20237559 0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.14038483 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.08619319]]
El valor de beta es: [[ 0.66217766]
 [-3.6953843 ]]
El nº iteraciones a la que converge es :4
El producto de iteraciones es iter_i: 0.0007928351246008185, El límite es: 1e-05
El valor de las probabilidades parciales Pi es: [array([0.02423594]), array([0.04594805]), array([0.0854
0873]), array([0.15331276]), array([0.25986436]), array([0.40504298]), array([0.56897776]), array([0.719
07124]), array([0.83230289]), array([0.90586963])]
0
1
2
3
4
5
6
7
8
9
La matriz es W: [[0.02364856 0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.04383683 0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.07811408 0.         0.         0.
  0.         0.         0.         0.        ]
```

```
 [0.         0.         0.         0.12980796 0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.19233487 0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.24098316
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.24524207 0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.20200779 0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.13957479 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.08526985]]
El valor de beta es: [[ 0.66220827]
 [-3.69557172]]
El nº iteraciones a la que converge es :5
```

La ecuación conchasumadre es: Y = 0.66220827 -3.69557172X esto es lo que vendría siendo el exponente de la función logística:

$$P = \frac{1}{1+e^{-(0.66220827-3.69557172 \cdot X}}$$

In [11]:
```
ll = likelihood (Y, logitprobs(X,a))
```

In [12]:
```
ll #La suma total
```

Out[12]:
```
array([1.32622426e-06])
```

Con el paquete statsmodel de python

In [13]:
```
import statsmodels.api as sm
```

In [14]:
```
logit_model = sm.Logit(Y, X_new)
```

In [15]:
```
result = logit_model.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.431012
         Iterations 6
```

In [43]:
```
print(result.summary2())
```

```
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.360
Dependent Variable: y                AIC:              12.6202
Date:               2022-03-24 15:30 BIC:              13.2254
No. Observations:   10               Log-Likelihood:   -4.3101
Df Model:           1                LL-Null:          -6.7301
Df Residuals:       8                LLR p-value:      0.027807
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
-----------------------------------------------------------------
          Coef.    Std.Err.      z      P>|z|     [0.025   0.975]
-----------------------------------------------------------------
x1        0.6622     0.4001    1.6551   0.0979   -0.1220   1.4464
const    -3.6956     2.2889   -1.6145   0.1064   -8.1818   0.7906
=================================================================
```

Como podemos ver en el summary x1 es el eje de coordenadas y const el factor de multiplicación y estos coinciden exactamente con nuestro modelo, por lo que podemos decir es que nuestro porgrama está bien hecho y la aproximación de Newton - Rapson están bien calculados.

In [ ]: