MODELAGEM E INFERÊNCIA ESTATÍSTICA

Modelo de regressão múltipla com variáveis transformadas e com modelo logístico - Exercícios

O QUE VOU ESTUDAR HOJE?

Exemplo aplicando transformação de variáveis

Exemplo aplicando modelo logístico

TRANSFORMAÇÃO DE VARIÁVEIS

O artigo "*The influence of honing process parameterson surface quality, productivity, cuttingangle, andcoefficientoffriction" (Industrial LubricationandTribology, 2012: 77-83*) incluiu os seguintes dados sobre x_1 = velocidade de corte (m/s), x_2 = pressão específica do processo de pré-afilamento (N/mm²), x_3 = pressão específica do processo de conclusão do afilamento, e y = produtividade no processo de afilamento (mm³/s para uma determinada ferramenta; a produtividade é o volume do material cortado em um segundo).

O artigo propôs um modelo de potência multivariado

$$Y = \alpha x_1^{\beta_1} x_2^{\beta_2} x_3^{\beta_3} \varepsilon$$

O modelo de regressão linear implicado envolve regressar ln(y) em relação aos três preditores ln(x1), ln(x2) e ln(x3).

A função de regressão de potência estimada correspondente apareceu no artigo mencionado.

TRANSFORMAÇÃO DE VARIÁVEIS

Conjunto de dados

x_1	x_2	x_3	y	x_1	x_2	x_3	y
0.93	1.00	0.20	32.95	0.93	1.40	0.50	33.67
1.11	1.00	0.20	38.72	1.11	1.40	0.50	38.72
0.93	1.00	0.50	35.20	1.02	1.18	0.31	35.20
1.11	1.00	0.50	38.72	1.02	1.18	0.31	33.67
0.93	1.40	0.20	32.27	1.02	1.18	0.31	36.02
1.11	1.40	0.20	39.71	1.02	1.18	0.31	32.27

TRANSFORMAÇÃO DE VARIÁVEIS

a) Variáveis originais $Y = ax_1^{\beta_1} x_2^{\beta_2} x_3^{\beta_3} \varepsilon$

```
#Conjunto de dados original
dx1 = (0.93, 1.11, 0.93, 1.11, 0.93, 1.11, 0.93, 1.11,
       1.02, 1.02, 1.02, 1.02)
dx2 = (1.00, 1.00, 1.00, 1.00, 1.40, 1.40, 1.40, 1.40,
       1.18, 1.18, 1.18, 1.18)
dx3= (0.20, 0.20, 0.50, 0.50, 0.20, 0.20, 0.50, 0.50,
      0.31, 0.31, 0.31, 0.31)
dy= (32.95, 38.72, 35.20, 38.72, 32.27, 39.71, 33.67,
     38.72, 35.20, 33.67, 36.02, 32.27)
# Construir o DataFrame e nomear as colunas
dfo = pd.DataFrame(list(zip(dx1, dx2, dx3, dy)),
                   columns =["x1","x2","x3","y"])
dfo.head(20)
```

	x1	x2	x3	y
0	0.93	1.00	0.20	32.95
1	1.11	1.00	0.20	38.72
2	0.93	1.00	0.50	35.20
3	1.11	1.00	0.50	38.72
4	0.93	1.40	0.20	32.27
5	1.11	1.40	0.20	39.71
6	0.93	1.40	0.50	33.67
7	1.11	1.40	0.50	38.72
8	1.02	1.18	0.31	35.20
9	1.02	1.18	0.31	33.67
10	1.02	1.18	0.31	36.02
11	1.02	1.18	0.31	32.27

TRANSFORMAÇÃO DE VARIÁVEIS

a) Variáveis transformadas

$$Y = ax_1^{\beta_1} x_2^{\beta_2} x_3^{\beta_3} \varepsilon$$

$$\ln(y) = \ln(\alpha) + \beta_1 \ln(x_1) + \beta_2 \ln(x_2) + \beta_3 \ln(x_3) + \ln(\varepsilon$$

$$y' = \beta_0{}' + \beta_1 x_1{}' + \beta_2 x_2{}' + \beta_3 x_3{}' + \varepsilon'$$

```
1 # Conjunto de dados transformado
2 lnx1 = np.log(dx1)
3 lnx2 = np.log(dx2)
4 lnx3= np.log(dx3)
5 lny= np.log(dy)
6 # Construir o DataFrame e nomear as colunas
7 dft = pd.DataFrame(list(zip(lnx1, lnx2, lnx3, lny)),
8                     columns =["lnx1","lnx2","lnx3","lny"])
9 dft.head(20)
```

	lnx1	lnx2	lnx3	lny
0	-0.072571	0.000000	-1.609438	3.494991
1	0.104360	0.000000	-1.609438	3.656356
2	-0.072571	0.000000	-0.693147	3.561046
3	0.104360	0.000000	-0.693147	3.656356
4	-0.072571	0.336472	-1.609438	3.474138
5	0.104360	0.336472	-1.609438	3.681603
6	-0.072571	0.336472	-0.693147	3.516607
7	0.104360	0.336472	-0.693147	3.656356
8	0.019803	0.165514	-1.171183	3.561046
9	0.019803	0.165514	-1.171183	3.516607
10	0.019803	0.165514	-1.171183	3.584074
11	0.019803	0.165514	-1.171183	3.474138

TRANSFORMAÇÃO DE VARIÁVEIS

b) Realizando o ajuste

```
#regressão com a fórmula import statsmodels.formula.api as smf
regmul = smf.ols('lny ~ lnx1 + lnx2 + lnx3', data = dft)
#Realizar o processo de modelagem
res = regmul.fit()
```

```
#Resultado detalhado
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    lny   R-squared:                       0.706
Model:                            OLS   Adj. R-squared:                  0.595
Method:                 Least Squares   F-statistic:                     6.398
Date:                Tue, 05 Apr 2022   Prob (F-statistic):             0.0161
Time:                        17:02:54   Log-Likelihood:                 21.634
No. Observations:                  12   AIC:                            -35.27
Df Residuals:                       8   BIC:                            -33.33
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      3.5880      0.049     73.095      0.000       3.475       3.701
lnx1           0.8439      0.195      4.324      0.003       0.394       1.294
lnx2          -0.0280      0.103     -0.272      0.792      -0.265       0.209
lnx3           0.0245      0.038      0.650      0.534      -0.062       0.111
==============================================================================
```

TRANSFORMAÇÃO DE VARIÁVEIS

b) Realizando o ajuste

$$\ln(y) = \ln(\alpha) + \beta_1 \ln(x_1) + \beta_2 ln(x_2) + \beta_3\, ln(x_3) + \ln(\varepsilon)$$

$$y' = 3,5880 + 0,8439x_1' - 0,0280{x_2}' + 0,0245{x_3}' + \varepsilon'$$

OLS Regression Results

Dep. Variable:	lny	R-squared:	0.706
Model:	OLS	Adj. R-squared:	0.595
Method:	Least Squares	F-statistic:	6.398
Date:	Tue, 05 Apr 2022	Prob (F-statistic):	0.0161
Time:	17:02:54	Log-Likelihood:	21.634
No. Observations:	12	AIC:	-35.27
Df Residuals:	8	BIC:	-33.33
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.5880	0.049	73.095	0.000	3.475	3.701
lnx1	0.8439	0.195	4.324	0.003	0.394	1.294
lnx2	-0.0280	0.103	-0.272	0.792	-0.265	0.209
lnx3	0.0245	0.038	0.650	0.534	-0.062	0.111

TRANSFORMAÇÃO DE VARIÁVEIS

b) Testar utilidade do modelo

$$f = \frac{R^2/k}{(1-R^2)/[n-(k+1)]}$$

Com n=12, k=3 e $R^2 = 0,706$ → $f = 6,403$

OLS Regression Results

Dep. Variable:	lny	R-squared:	0.706
Model:	OLS	Adj. R-squared:	0.595
Method:	Least Squares	F-statistic:	6.398
Date:	Tue, 05 Apr 2022	Prob (F-statistic):	0.0161
Time:	17:02:54	Log-Likelihood:	21.634
No. Observations:	12	AIC:	-35.27
Df Residuals:	8	BIC:	-33.33
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.5880	0.049	73.095	0.000	3.475	3.701
lnx1	0.8439	0.195	4.324	0.003	0.394	1.294
lnx2	-0.0280	0.103	-0.272	0.792	-0.265	0.209
lnx3	0.0245	0.038	0.650	0.534	-0.062	0.111

TRANSFORMAÇÃO DE VARIÁVEIS

b) Testar utilidade do modelo

$$f = \frac{R^2/k}{(1-R^2)/[n-(k+1)]}$$

Com n=12, k=3 e $R^2 = 0{,}706 \rightarrow f = 6{,}403$

OLS Regression Results

Dep. Variable:	lny	R-squared:	0.706
Model:	OLS	Adj. R-squared:	0.595
Method:	Least Squares	F-statistic:	6.398
Date:	Tue, 05 Apr 2022	Prob (F-statistic):	0.0161
Time:	17:02:54	Log-Likelihood:	21.634

```
1 import scipy.stats
2 F=res.fvalue
3 k=res.df_model # grau do modelo
4 n=res.nobs # num. amostras
5 dfn=k
6 dfd=n-(k+1)
7 alpha = 0.05 #nível de confiança.
8 F_critico=scipy.stats.f.ppf(1-alpha, dfn, dfd)
9 print("F_crit-",F_critico) #tabela F-dist
```

```
F_crit- 4.06618055135116
```

$F_{k,n-(k+1)} \rightarrow F_{3,8}$
$\rightarrow F_{crit}$

$f \geq F_{crit}$ rejeitar H_0
6,403 ≥ 4,066 ?? SIM,
portanto, rejeitar H_0

TRANSFORMAÇÃO DE VARIÁVEIS

c) O grande valor-p correspondente à proporção t, para $\ln(x_2)$ e $\ln(x_3)$, sugere que estes preditores podem ser eliminados do modelo?

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.5880	0.049	73.095	0.000	3.475	3.701
lnx1	0.8439	0.195	4.324	0.003	0.394	1.294
lnx2	-0.0280	0.103	-0.272	0.792	-0.265	0.209
lnx3	0.0245	0.038	0.650	0.534	-0.062	0.111

- Hipótese nula para os parâmetros β_2 *e* β_3

$$H_0: \beta_2 = \beta_3 = 0$$

- Hipótese alternativa H_a pelo menos um $\beta_i \neq 0$
- Valor-p >α, portanto, NÃO REJEITAR H_0, isto é $\beta_2 = \beta_3 = 0$
- O modelo pode ser reduzido a um modelo linear simples

$$Y = ax_1^{\beta_1} \text{ ou } \ln(y) = \beta_0 + \beta_1 \ln(x_1)$$

TRANSFORMAÇÃO DE VARIÁVEIS

d) Analisar resíduos e gráficos de adequabilidade

```
#Valores previstos e residuos
y_pred=list(reslin.predict())
resi=reslin.resid
#crear instancia influence
influence = reslin.get_influence()
#obter residuos standardizados
stdresid = list(influence.resid_studentized_internal)
prop=np.divide(resi,stdresid) #e/e*

```

```
dftab = pd.DataFrame(list(zip(lnx1, lny, y_pred, resi, stdresid, prop)),
                     columns =["lnx1","lny","lnyc","e","e*","e/e*"])
dftab
```

TRANSFORMAÇÃO DE VARIÁVEIS

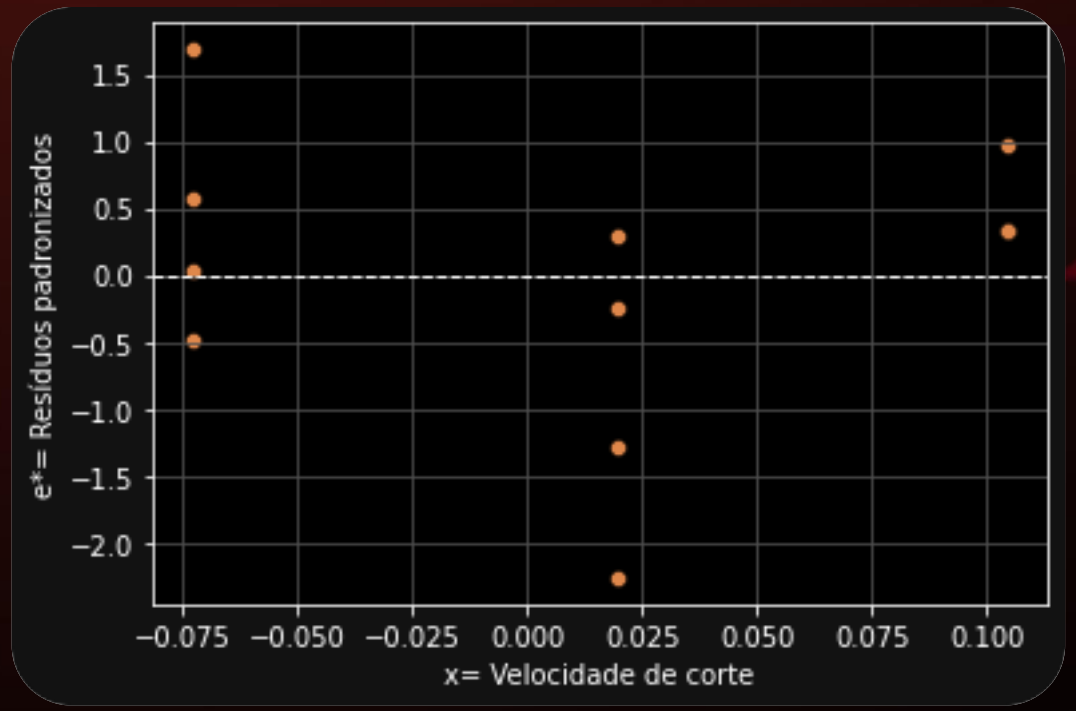
d) Analisar resíduos e gráficos de adequabilidade

	lnx1	lny	lnyc	e	e*	e/e*
0	-0.072571	3.494991	3.493690	0.001301	0.032553	0.039971
1	0.104360	3.656356	3.642998	0.013358	0.332646	0.040157
2	-0.072571	3.561046	3.493690	0.067356	1.685111	0.039971
3	0.104360	3.656356	3.642998	0.013358	0.332646	0.040157
4	-0.072571	3.474138	3.493690	-0.019552	-0.489153	0.039971
5	0.104360	3.681603	3.642998	0.038605	0.961341	0.040157
6	-0.072571	3.516607	3.493690	0.022917	0.573341	0.039971
7	0.104360	3.656356	3.642998	0.013358	0.332646	0.040157
8	0.019803	3.561046	3.571642	-0.010596	-0.245800	0.043108
9	0.019803	3.516607	3.571642	-0.055035	-1.276683	0.043108
10	0.019803	3.584074	3.571642	0.012432	0.288405	0.043108
11	0.019803	3.474138	3.571642	-0.097504	-2.261875	0.043108

TRANSFORMAÇÃO DE VARIÁVEIS

d) Analisar resíduos e gráficos de adequabilidade

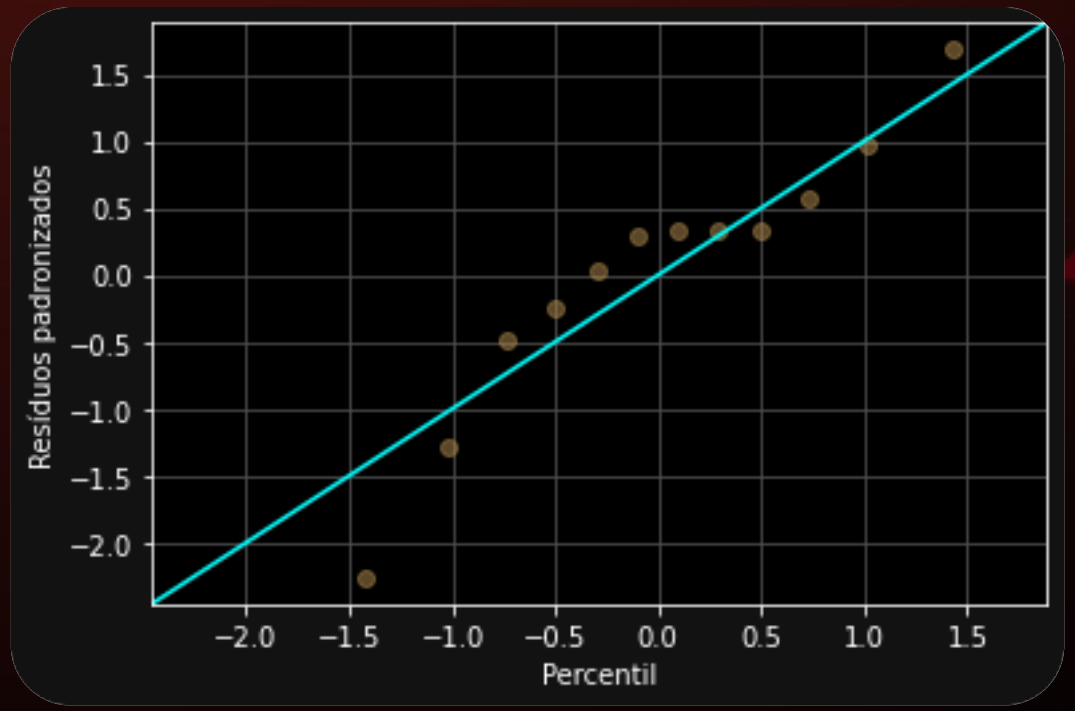
```
1 sns.scatterplot(x='lnx1', y='e*', data=dftab);plt.grid(True)
2 plt.xlabel('x= Velocidade de corte')
3 plt.ylabel('e*= Resíduos padronizados')
4 plt.axhline(y=0, color='black', linestyle='--', linewidth=1)
5 plt.show()
```



TRANSFORMAÇÃO DE VARIÁVEIS

d) Analisar resíduos e gráficos de adequabilidade

```
#qqpolot vs. normal distribution
QQ = ProbPlot(influence.resid_studentized_internal)
plot_lm_2 = QQ.qqplot(line='45', alpha=0.5, color='#4C72B0', lw=1)
plot_lm_2.axes[0].set_xlabel('Percentil')
plot_lm_2.axes[0].set_ylabel('Resíduos padronizados')
plt.grid(True)
```



MODELO LOGÍSTICO

A estabilidade dos pilares é um fator muito importante para garantir condições seguras nas minas subterrâneas. Os autores do estudo "*Developing coal pillar stability chart using logistic regression" (Intl. J. of Rock Mechanics & Mining Sci., 2013: 55-60)* utilizaram um modelo de regressão logística para prever a estabilidade.

O artigo relatou os dados a seguir:

x_1 = proporção entre altura e largura dos pilares,
x_2 = proporção da resistência ao estresse dos pilares,
y = situação de estabilidade para 29 pilares de carvão.

a) Obtenha o modelo no python.
b) Utilize o resultado com α = 0,1 para determinar se as duas variáveis do preditor parecem ter um impacto significativo na estabilidade dos pilares.
c) Forneça interpretações para e2,774 e e5,668.

MODELO LOGÍSTICO

ID	x_1	x_2	Stable?	ID	x_1	x_2	Stable?
1	1.80	2.40	Y	16	0.80	1.37	N
2	1.65	2.54	Y	17	0.60	1.27	N
3	2.70	0.84	Y	18	1.30	0.87	N
4	3.67	1.68	Y	19	0.83	0.97	N
5	1.41	2.41	Y	20	0.57	0.94	N
6	1.76	1.93	Y	21	1.44	1.00	N
7	2.10	1.77	Y	22	2.08	0.78	N
8	2.10	1.50	Y	23	1.50	1.03	N
9	4.57	2.43	Y	24	1.38	0.82	N
10	3.59	5.55	Y	25	0.94	1.30	N
11	8.33	2.58	Y	26	1.58	0.83	N
12	2.86	2.00	Y	27	1.67	1.05	N
13	2.58	3.68	Y	28	3.00	1.19	N
14	2.90	1.13	Y	29	2.21	0.86	N
15	3.89	2.49	Y				

x_1 = proporção entre altura e largura dos pilares,
x_2 = proporção da resistência ao estresse dos pilares,
y = situação de estabilidade para 29 pilares de carvão.

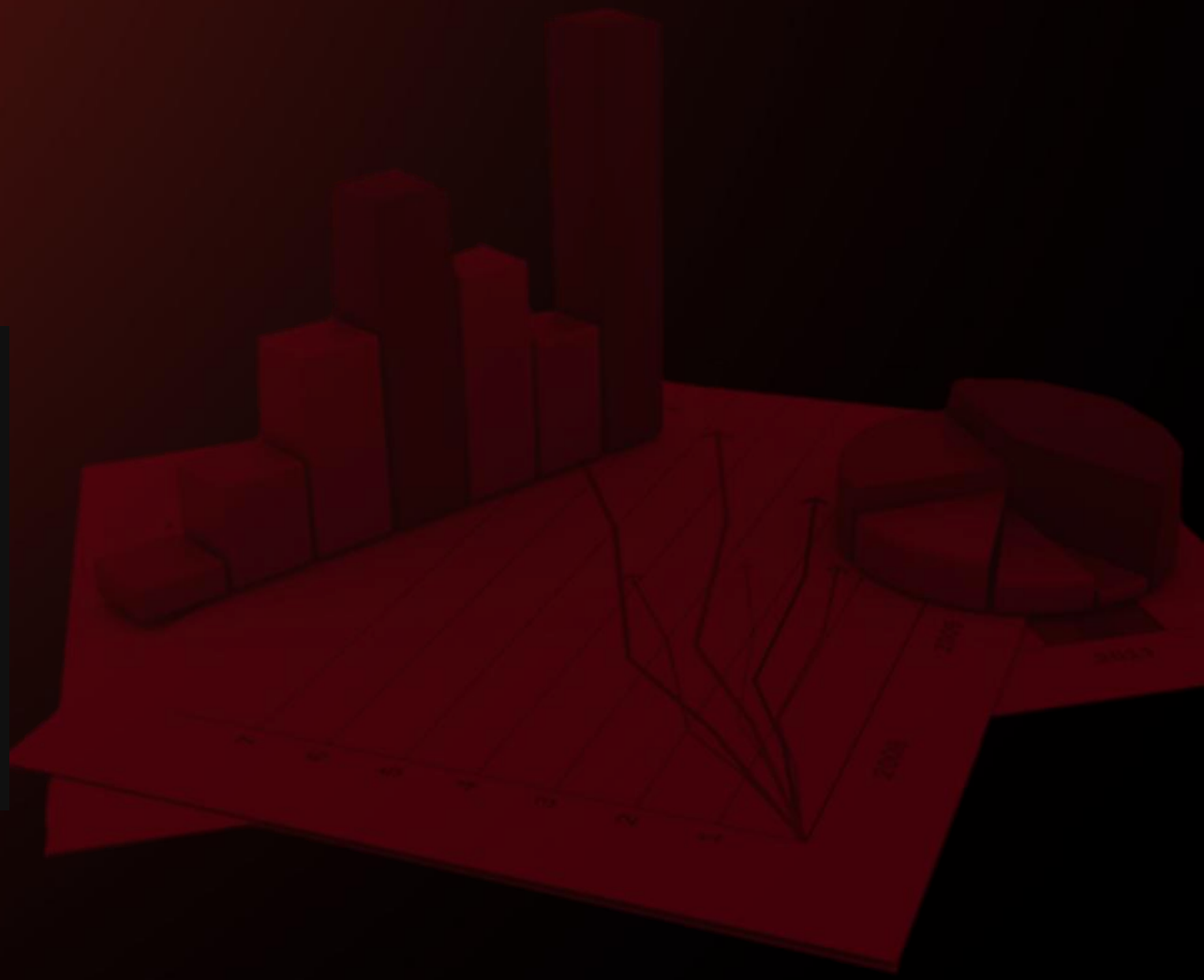
Y= 1 estável
Y= 0 não estável

MODELO LOGÍSTICO

a) Obtenha o modelo no python.

```
lstx1 = (1.80, 1.65, 2.70, 3.67, 1.41, 1.76, 2.10, 2.10, 4.57, 3.59,
         8.33, 2.86, 2.58, 2.90, 3.89, 0.80, 0.60, 1.30, 0.83, 0.57,
         1.44, 2.08, 1.50, 1.38, 0.94, 1.58, 1.67, 3.00, 2.21)
lstx2 - (2.40, 2.54, 0.84, 1.68, 2.41, 1.93, 1.77, 1.50, 2.43, 5.55,
         5.58, 2.00, 3.68, 1.13, 2.49, 1.37, 1.27, 0.87, 0.97, 0.94,
         1.00, 0.78, 1.03, 0.82, 1.30, 0.83, 1.05, 1.19, 0.86)
lsty- (1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
# Construir o DataFrame e nomear as colunas
df = pd.DataFrame(list(zip(lstx1, lstx2,lsty)),
                  columns -["x1","x2","y"])
x-df[['x1','x2']]
y=df['y']
df.head(30)
```

```
#adicionar uma constante preditora
x = sm.add_constant(x)
# Construir o modelo e ajustar os dados
model = sm.Logit(y, x).fit()
print(model.summary())
```



MODELO LOGÍSTICO

a) **Obtenha o modelo no python.**

$$\frac{p(x_1, \dots, x_k)}{1 - p(x_1, \dots, x_k)} = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}$$

```
Optimization terminated successfully.
         Current function value: 0.151888
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   29
Model:                          Logit   Df Residuals:                       26
Method:                           MLE   Df Model:                            2
Date:                Tue, 05 Apr 2022   Pseudo R-squ.:                  0.7807
Time:                        22:04:50   Log-Likelihood:                -4.4048
converged:                       True   LL-Null:                       -20.084
Covariance Type:            nonrobust   LLR p-value:                 1.551e-07
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -13.1457      5.184     -2.536      0.011     -23.306      -2.985
x1             2.7740      1.477      1.878      0.060      -0.122       5.670
x2             5.6682      2.642      2.145      0.032       0.490      10.847
==============================================================================
```

MODELO LOGÍSTICO

b) Teste de hipótese dos parâmetros com um nível de confiança de 90%.

	coef	std err	z	P>|z|	[0.025	0.975]
const	-13.1457	5.184	-2.536	0.011	-23.306	-2.985
x1	2.7740	1.477	1.878	0.060	-0.122	5.670
x2	5.6682	2.642	2.145	0.032	0.490	10.847

Para 90% → α=0,1

Todos os valores-p são menores do que 0,1, portanto, a hipótese nula se rejeita e o modelo mantém todos os parâmetros.

$$\frac{p(x_1, \dots, x_k)}{1 - p(x_1, \dots, x_k)} = e^{-13{,}1457 + 2{,}7740x_1 + 5.6682x_2}$$

MODELO LOGÍSTICO

$$\frac{p(x_1, \dots, x_k)}{1 - p(x_1, \dots, x_k)} = e^{-13{,}1457 + 2{,}7740x_1 + 5.6682x_2}$$

c) Razão das chances.

```
# ... Define and fit model
odds_ratios = pd.DataFrame(
    {
        "OR": model.params,
        "Lower CI": model.conf_int()[0],
        "Upper CI": model.conf_int()[1],
    }
)
print(odds_ratios)
odds_ratios = np.exp(odds_ratios)
print(odds_ratios)
```

```
              OR   Lower CI   Upper CI
const -13.145657 -23.306390  -2.984923
x1      2.774021  -0.121639   5.669680
x2      5.668211   0.489678  10.846743
                OR      Lower CI      Upper CI
const     0.000002  7.553761e-11      0.050543
x1       16.022926  8.854677e-01    289.941871
x2      289.516064  1.631791e+00  51366.594509
```

$x_1 \rightarrow e^{2{,}7740} = 16{,}0226$

$x_2 \rightarrow e^{5{,}6682} = 289{,}512$

x_1 = proporção entre altura e largura dos pilares,

x_2 = proporção da resistência ao estresse dos pilares,

Y= 1 estável

Y= 0 não estável

MODELO LOGÍSTICO

d) Comparação dos valores observados e previstos.

```
# performing predictions on the test datdaset
yhat = model.predict()
prediction = list(map(round, yhat))

# comparing original and predicted values of y
print('Valores observados:', list(y))
print('Valores previstos:', prediction)
df2 = pd.DataFrame(list(zip(y, prediction)),
                   columns =["x","y"])
#df2.to_csv(index=False)
```

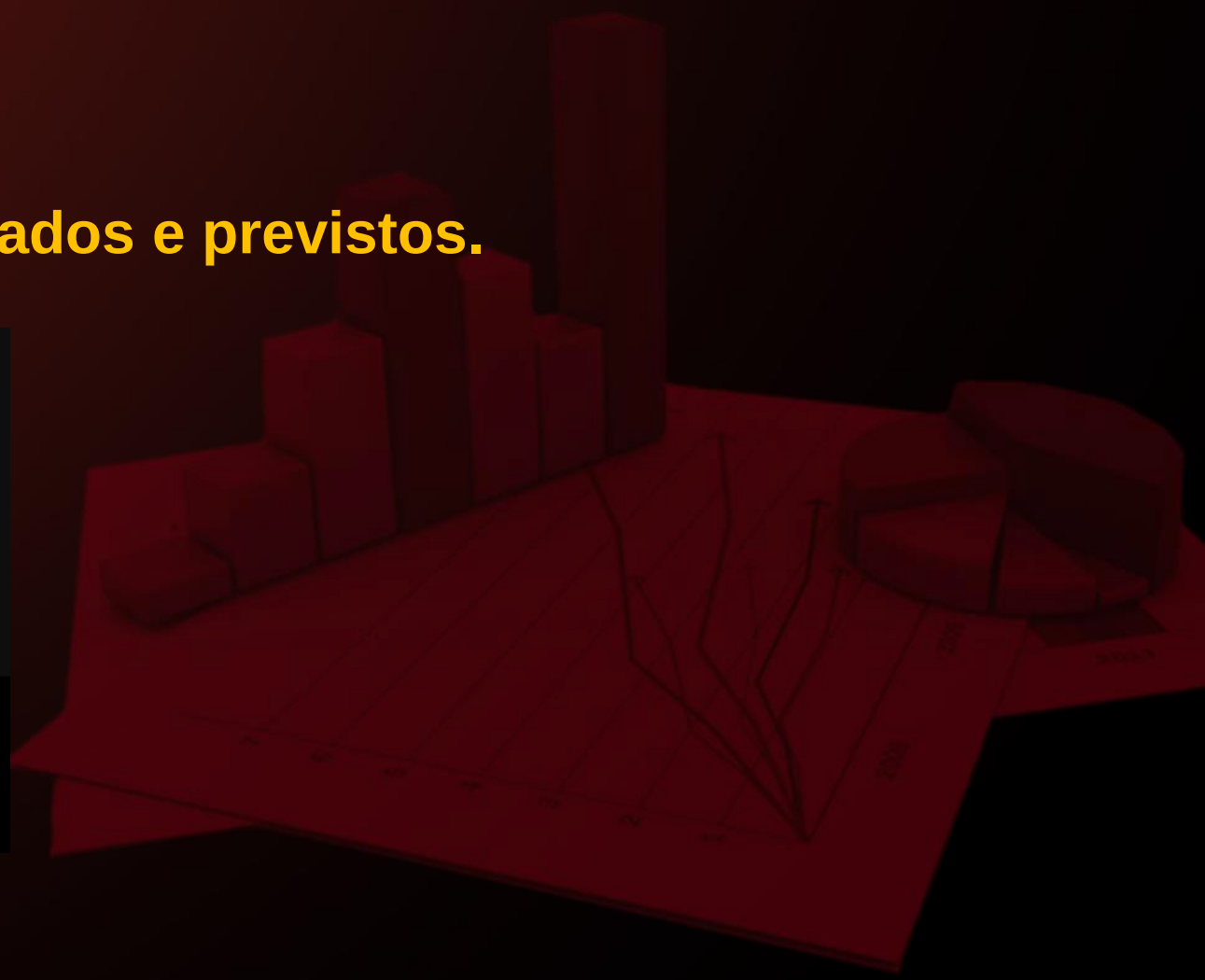
```
Valores observados: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Valores previstos: [1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
```

MODELO LOGÍSTICO

d) Comparação dos valores observados e previstos.

```
1 from sklearn.metrics import (confusion_matrix,accuracy_score)
2
3 # confusion matrix
4 cm = confusion_matrix(y, prediction)
5 print ("Confusion Matrix : \n", cm)
6
7 # accuracy score of the model
8 print('Test accuracy = ', accuracy_score(y, prediction))
```

```
Confusion Matrix :
 [[13  1]
 [ 1 14]]
Test accuracy =  0.9310344827586207
```



MODELAGEM E INFERÊNCIA ESTATÍSTICA

Modelo de regressão múltipla com variáveis transformadas e com modelo logístico - Exercícios