

# REGRESSÃO LOGÍSTICA BINÁRIA

Importando bibliotecas

```
In [85]: import numpy as np
import pandas as pd
import warnings

# Suprimir os warnings
warnings.simplefilter(action='ignore', category=pd.errors.SettingWithCopyWarning)
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
In [87]: doenca_pre = pd.read_csv('covid_doencas_preexistentes.csv',
                                sep=';', encoding='utf-8')
```

## ANÁLISE INICIAL

Verificando as variáveis carregadas no bando de dados

Base de dados tras a informação de pessoas que receberam o diagnóstico de COVID  
perfil do paciente e doenças pré existentes.

```
In [3]: doenca_pre.head()
```

```
Out[3]:
```

	nome_munic	codigo_ibge	idade	cs_sexo	diagnostico_covid19	data_inicio_sintom
0	Carapicuíba	3510609	36.0	FEMININO	CONFIRMADO	2020-07-
1	Jacareí	3524402	54.0	FEMININO	CONFIRMADO	2020-07-
2	Vargem Grande Paulista	3556453	33.0	FEMININO	CONFIRMADO	2020-07-
3	Paulínia	3536505	47.0	FEMININO	CONFIRMADO	2021-01-
4	Santo André	3547809	41.0	FEMININO	CONFIRMADO	2021-01-

Descrição de variáveis

- codigo\_ibge: Código do município no IBGE (7 dígitos) de residência do paciente
- nome\_munic: Nome do município de residência do paciente
- idade: Idade do paciente
- cs\_sexo: Sexo do paciente
- diagnostico\_covid19: Confirmação de COVID-19
- data\_inicio\_sintomas: Data de início dos sintomas
- obito: Indica se o paciente veio a óbito por COVID-19
- asma: Paciente apresenta esse fator de risco (asma)
- cardiopatia: Paciente apresenta esse fator de risco (cardiopatia)

- diabetes: Paciente apresenta esse fator de risco (diabetes)
- doenca\_hematologica: Paciente apresenta esse fator de risco (doença hematológica)
- doenca\_hepatica: Paciente apresenta esse fator de risco (doença hepática)
- doenca\_neurolgica: Paciente apresenta esse fator de risco (doença neurológica)
- doenca\_renal: Paciente apresenta esse fator de risco (doença renal)
- imunodepressao: Paciente apresenta esse fator de risco (imunodepressão)
- obesidade: Paciente apresenta esse fator de risco (obesidade)
- outros\_fatores\_de\_risco: Paciente apresenta outros fatores de risco
- pneumopatia: Paciente apresenta esse fator de risco (pneumopatia)
- puerpera: Paciente se encontra nesse estágio (puérpera)
- síndrome\_de\_down: Paciente apresenta esse fator de risco (síndrome de down)

```
In [4]: doenca_pre.shape
```

```
Out[4]: (1102362, 20)
```

**1ª Análise: Verificar se existe uma tendência de óbito entre pessoas do sexo feminino e masculino.**

```
In [5]: from collections import Counter
```

Fazer contagem por categoria das variáveis

```
In [6]: Counter(doenca_pre.cs_sexo)
```

```
Out[6]: Counter({'FEMININO': 589810,  
                'MASCULINO': 509667,  
                'INDEFINIDO': 2869,  
                'IGNORADO': 16})
```

```
In [7]: doenca_pre['cs_sexo'].value_counts()
```

```
Out[7]: cs_sexo  
FEMININO      589810  
MASCULINO     509667  
INDEFINIDO     2869  
IGNORADO       16  
Name: count, dtype: int64
```

Como queremos comparar o gênero feminino e masculino iremos desconsiderar as demais classes.

Valores Missing (NaN)

```
In [8]: doenca_pre.isnull().sum()
```

```
Out[8]: nome_munic          0
        codigo_ibge        0
        idade              1478
        cs_sexo            0
        diagnostico_covid19 0
        data_inicio_sintomas 18246
        obito              0
        asma               0
        cardiopatia        0
        diabetes           0
        doenca_hematologica 0
        doenca_hepatica    0
        doenca_neurologica  1
        doenca_renal        1
        imunodepressao     1
        obesidade          1
        outros_fatores_de_risco 1
        pneumopatia        1
        puerpera           1
        sindrome_de_down   1
        dtype: int64
```

Excluir valor NAN de cs\_sexo

```
In [9]: doenca_pre.dropna(subset=['cs_sexo'], inplace=True)
```

Excluir Ignorado

```
In [10]: relacao = doenca_pre.loc[doenca_pre.cs_sexo != 'IGNORADO']
```

Excluir Indefinido

```
In [11]: relacao = relacao.loc[relacao.cs_sexo != 'INDEFINIDO']
```

Verificando as variaveis que ficaram na base de dados

```
In [12]: relacao['cs_sexo'].value_counts()
```

```
Out[12]: cs_sexo
FEMININO    589810
MASCULINO   509667
Name: count, dtype: int64
```

Verificando os dados via análise gráfica

```
In [13]: import plotly.express as px

px.pie(relacao, names="cs_sexo")
```

### Analisando a quantidade de óbitos

```
In [14]: relacao.obito.value_counts()
```

```
Out[14]: obito  
0      1067102  
1        32375  
Name: count, dtype: int64
```

```
In [15]: px.pie(relacao, names="obito")
```

### Análise da classificação dos atributos

Verificando como o Python reconheceu as variáveis

```
In [16]: relacao.dtypes
```

```
Out[16]: nome_munic          object
          codigo_ibge        int64
          idade              float64
          cs_sexo            object
          diagnostico_covid19 object
          data_inicio_sintomas object
          obito              int64
          asma               object
          cardiopatia        object
          diabetes           object
          doenca_hematologica object
          doenca_hepatica    object
          doenca_neurológica object
          doenca_renal       object
          imunodepressao     object
          obesidade          object
          outros_fatores_de_risco object
          pneumopatia        object
          puerpera           object
          síndrome_de_down   object
          dtype: object
```

**Renomeando(sobrescrevendo) a variável obito**

```
In [17]: relacao["obito"] = relacao["obito"].replace({0:"nao", 1:"sim"})
```

```
In [18]: relacao.head()
```

```
Out[18]:
```

	nome_munic	codigo_ibge	idade	cs_sexo	diagnostico_covid19	data_inicio_sintom
0	Carapicuíba	3510609	36.0	FEMININO	CONFIRMADO	2020-07-
1	Jacareí	3524402	54.0	FEMININO	CONFIRMADO	2020-07-
2	Vargem Grande Paulista	3556453	33.0	FEMININO	CONFIRMADO	2020-07-
3	Paulínia	3536505	47.0	FEMININO	CONFIRMADO	2021-01-
4	Santo André	3547809	41.0	FEMININO	CONFIRMADO	2021-01-

```
In [19]: relacao.dtypes
```

```
Out[19]:
```

nome_munic	object
codigo_ibge	int64
idade	float64
cs_sexo	object
diagnostico_covid19	object
data_inicio_sintomas	object
obito	object
asma	object
cardiopatia	object
diabetes	object
doenca_hematologica	object
doenca_hepatica	object
doenca_neurológica	object
doenca_renal	object
imunodepressao	object
obesidade	object
outros_fatores_de_risco	object
pneumopatia	object
puerpera	object
sindrome_de_down	object
dtype:	object

```
In [20]: relacao.obito.value_counts()
```

```
Out[20]:
```

obito	
nao	1067102
sim	32375
Name: count, dtype: int64	

**Transformando em variáveis categóricas**

Transformando as variáveis que estão como objetos como categorias.

```
In [21]: relacao['cs_sexo'] = relacao['cs_sexo'].astype('category')
```

```
In [22]: relacao['obito'] = relacao['obito'].astype('category')
```

## Modelo 1: Uma variável independente

Queremos entender se a pessoa que foi diagnosticada com COVID, existe uma relação entre o gênero e o óbito.

Logo, teremos o primeiro modelo com uma única variável independente.

### Pressupostos :

- Variável dependente binária (dicotômica) -> Variável resposta (Y) dicotômica: Óbito - Sim ou Não.
- Categorias mutuamente exclusivas - a mesma pessoa não pode estar em duas situações.
- Independência das observações (sem medidas repetidas) -> a mesma pessoa é analisada uma única vez.

```
In [23]: import statsmodels.api as sm
import statsmodels.formula.api as smf
```

Criação do modelo de Regressão logística

```
In [24]: modelo1 = smf.glm(formula='obito ~ cs_sexo', data=relacao, family = sm.families.
print(modelo1.summary())
```

```

Generalized Linear Model Regression Results
=====
Dep. Variable:      ['obito[nao]', 'obito[sim]']    No. Observations:
1099477
Model:              GLM    Df Residuals:
1099475
Model Family:      Binomial    Df Model:
1
Link Function:      Logit    Scale:
1.0000
Method:              IRLS    Log-Likelihood:      -1.4
498e+05
Date:                Sat, 03 May 2025    Deviance:      2.8
995e+05
Time:                21:18:17    Pearson chi2:
1.10e+06
No. Iterations:      7    Pseudo R-squ. (CS):
0.001901
Covariance Type:      nonrobust
=====
=====
              coef      std err          z      P>|z|      [0.025
0.975]
-----
Intercept              3.7676      0.009     429.907      0.000      3.750
3.785
cs_sexo[T.MASCULINO]  -0.5192      0.011    -45.310      0.000     -0.542
-0.497
=====
=====

```

## Deviance Residuals

### Hipótese Testada:

- $H_0$ : O modelo ajusta bem os dados.
- $H_a$ : O modelo não ajusta bem os dados.

### Condição de Aceitação/Rejeição:

- A deviance total é comparada com uma distribuição qui-quadrado com  $n - p$  graus de liberdade, onde  $n$  é o número de observações e  $p$  é o número de parâmetros no modelo.
- **Aceitação de  $H_0$** : Se o valor  $p$  calculado a partir da deviance é maior que  $\alpha$ , aceitamos  $H_0$  e concluímos que o modelo ajusta bem os dados.
- **Rejeição de  $H_0$** : Se o valor  $p$  é menor que  $\alpha$ , rejeitamos  $H_0$  e concluímos que o modelo não ajusta bem os dados.

```

In [25]: import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
from scipy.stats import chi2, chisquare

```



```

from statsmodels.graphics.regressionplots import abline_plot
import scipy.stats as stats # Importando o módulo stats da biblioteca SciPy

# Teste de Deviance para os Resíduos
deviance_test_statistic = modelo1.deviance
deviance_df = modelo1.df_resid
deviance_p_value = 1 - stats.chi2.cdf(deviance_test_statistic, deviance_df)

print("Teste de Deviance para os Resíduos:")
print("Estatística de teste:", deviance_test_statistic)
print("Graus de liberdade:", deviance_df)
print("Valor p:", deviance_p_value)

```

Teste de Deviance para os Resíduos:  
 Estatística de teste: 289952.06753829453  
 Graus de liberdade: 1099475  
 Valor p: 1.0

## Testes de adequação do modelo

### 1. Pearson Chi-Square Test

#### Hipótese Testada:

- $H_0$ : O modelo ajusta bem os dados.
- $H_a$ : O modelo não ajusta bem os dados.

#### Condição de Aceitação/Rejeição:

- Calcula-se a estatística do teste qui-quadrado de Pearson, que segue uma distribuição qui-quadrado com  $n - p$  graus de liberdade.
- **Aceitação de  $H_0$** : Se o valor  $p$  é maior que  $\alpha$ , aceitamos  $H_0$  e concluímos que o modelo ajusta bem os dados.
- **Rejeição de  $H_0$** : Se o valor  $p$  é menor que  $\alpha$ , rejeitamos  $H_0$  e concluímos que o modelo não ajusta bem os dados.

```

In [26]: # Obtendo os resíduos de Pearson
residuos_pearson = modelo1.resid_pearson

# Teste de Pearson para os Resíduos
pearson_test_statistic = np.sum(residuos_pearson**2)
pearson_df = len(residuos_pearson) - modelo1.df_model - 1 # Graus de Liberdade
pearson_p_value = 1 - stats.chi2.cdf(pearson_test_statistic, pearson_df)

print("\nTeste de Pearson para os Resíduos:")
print("Estatística de teste:", pearson_test_statistic)
print("Graus de liberdade:", pearson_df)
print("Valor p:", pearson_p_value)

```

Teste de Pearson para os Resíduos:  
 Estatística de teste: 1099477.0000049898  
 Graus de liberdade: 1099475  
 Valor p: 0.49928258328631003

In [27]: `modelo1.params`

Out[27]: Intercept 3.767615  
 cs\_sexo[T.MASCULINO] -0.519236  
 dtype: float64

**Observação:** como vimos, a regressão logística retornará a probabilidade de sucesso (1), logo neste caso estamos analisando a probabilidade de ocorrência do óbito.

Porém iremos verificar se realmente é isso que ocorre:

In [28]: `modelo_prova = smf.glm(formula='cs_sexo ~ obito', data=relacao, family = sm.famibinomial, dataoffset=obito, weights=obito, robust=robust)`  
`print(modelo_prova.summary())`

```

                                Generalized Linear Model Regression Results
=====
Dep. Variable:                ['cs_sexo[FEMININO]', 'cs_sexo[MASCULINO]']    No. Observation
s:                            1099477
Model:                                GLM    Df Residuals:
1099475
Model Family:                    Binomial    Df Model:
1
Link Function:                    Logit    Scale:
1.0000
Method:                            IRLS    Log-Likelihood:
-7.5813e+05
Date:                                Sat, 03 May 2025    Deviance:
1.5163e+06
Time:                                21:18:34    Pearson chi2:
1.10e+06
No. Iterations:                    4    Pseudo R-squ. (C
S):                                0.001901
Covariance Type:                    nonrobust
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
Intercept      0.1613      0.002     83.042      0.000      0.157      0.165
obito[T.sim]   -0.5192      0.011    -45.310      0.000     -0.542     -0.497
=====

```

Invertamos a variável dependente com independente para verificar qual o Python está selecionando no modelo

In [29]: `print(modelo1.summary())`

```

=====
Generalized Linear Model Regression Results
=====
Dep. Variable:      ['obito[nao]', 'obito[sim]']    No. Observations:
1099477
Model:              GLM    Df Residuals:
1099475
Model Family:      Binomial    Df Model:
1
Link Function:      Logit    Scale:
1.0000
Method:              IRLS    Log-Likelihood:      -1.4
498e+05
Date:                Sat, 03 May 2025    Deviance:      2.8
995e+05
Time:                21:18:35    Pearson chi2:
1.10e+06
No. Iterations:      7    Pseudo R-squ. (CS):
0.001901
Covariance Type:      nonrobust
=====
=====
              coef      std err          z      P>|z|      [0.025
0.975]
-----
Intercept              3.7676      0.009     429.907      0.000      3.750
3.785
cs_sexo[T.MASCULINO]  -0.5192      0.011    -45.310      0.000     -0.542
-0.497
=====
=====

```

Como conseguir tirar alguma informação , interpretação do modelo?

Iremos utilizar o que chamamos como razão de chance com Intervalo de confiança de 95%:

- exponencial do coeficiente.

```
In [30]: razao = np.exp(modelo1.params[1])
         razao
```

```
Out[30]: 0.5949749678615635
```

Considerando o Odds Ratio, podemos dizer que a chance de um homem vir a falecer é 0.59 menor comparada a uma mulher.

## ROC Curve and AUC

Avalia a capacidade do modelo de discriminar entre as classes.

**Curva ROC:** Traça a taxa de verdadeiros positivos (sensibilidade) contra a taxa de falsos positivos (1 - especificidade) para vários limiares de classificação.

**AUC (Área Sob a Curva):** Um valor próximo de 1 indica excelente discriminação, enquanto um valor próximo de 0.5 indica discriminação aleatória. -

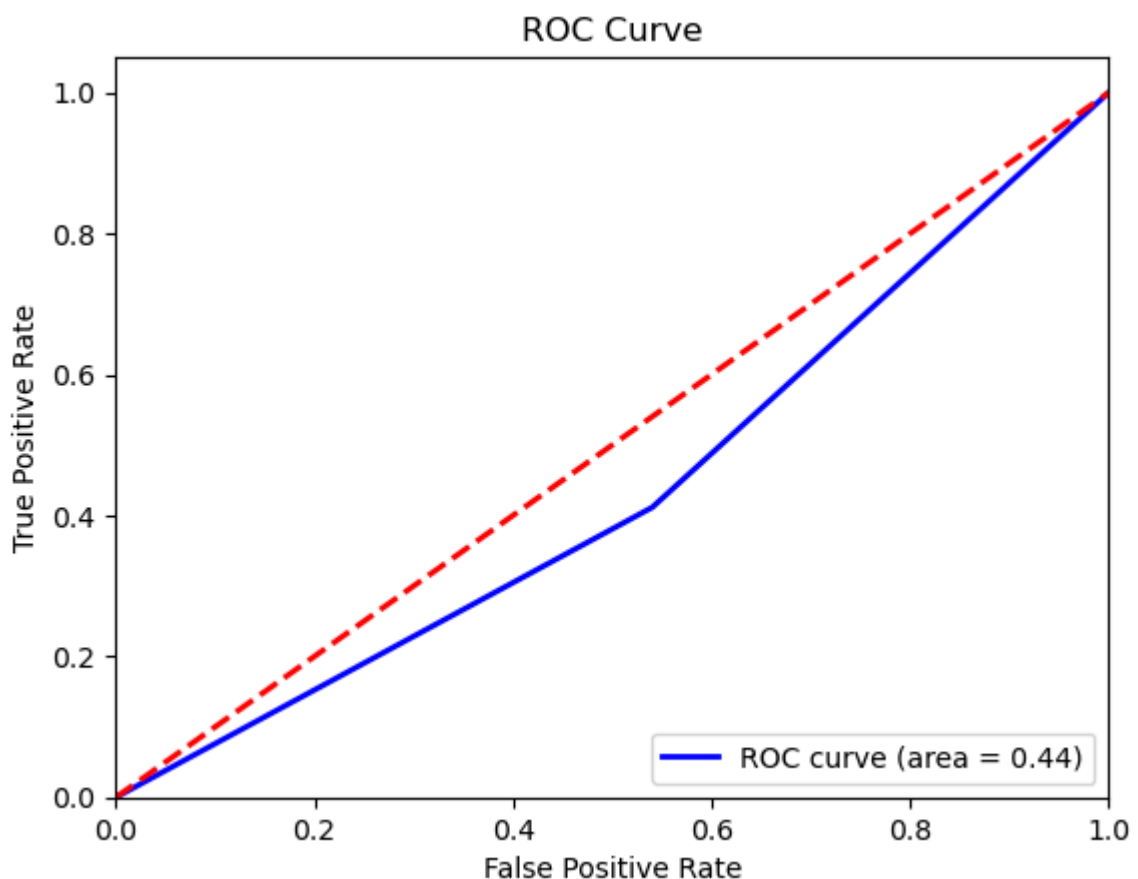
**Avaliação do Modelo:** Um modelo com AUC maior é considerado melhor em discriminar entre as classes.

```
In [31]: from sklearn.metrics import roc_curve, auc

y = relacao["obito"].replace({"nao":0, "sim":1})

# ROC Curve
fpr, tpr, _ = roc_curve(y, modelo1.fittedvalues)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()
```

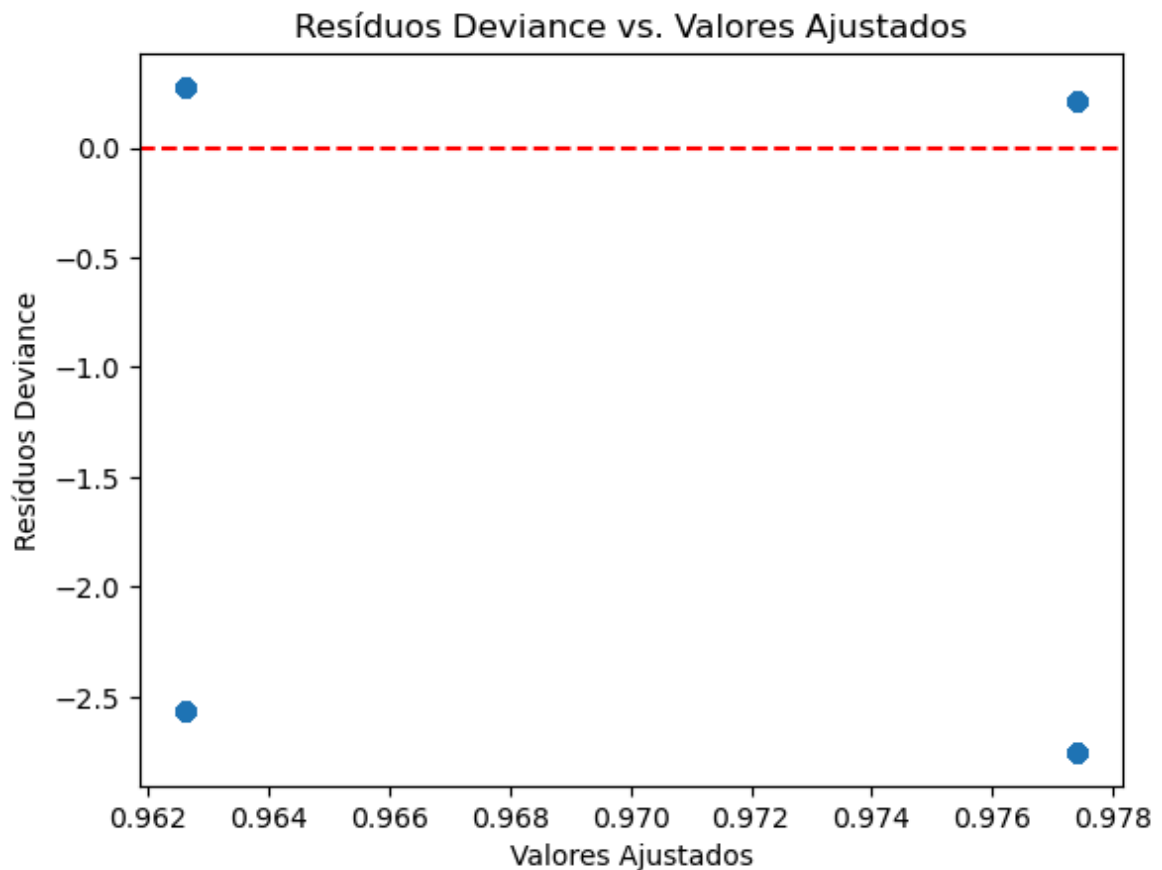


```
In [32]: # Resíduos deviance
residuos_deviance = modelo1.resid_deviance

# Resíduos de Pearson
residuos_pearson = modelo1.resid_pearson
```

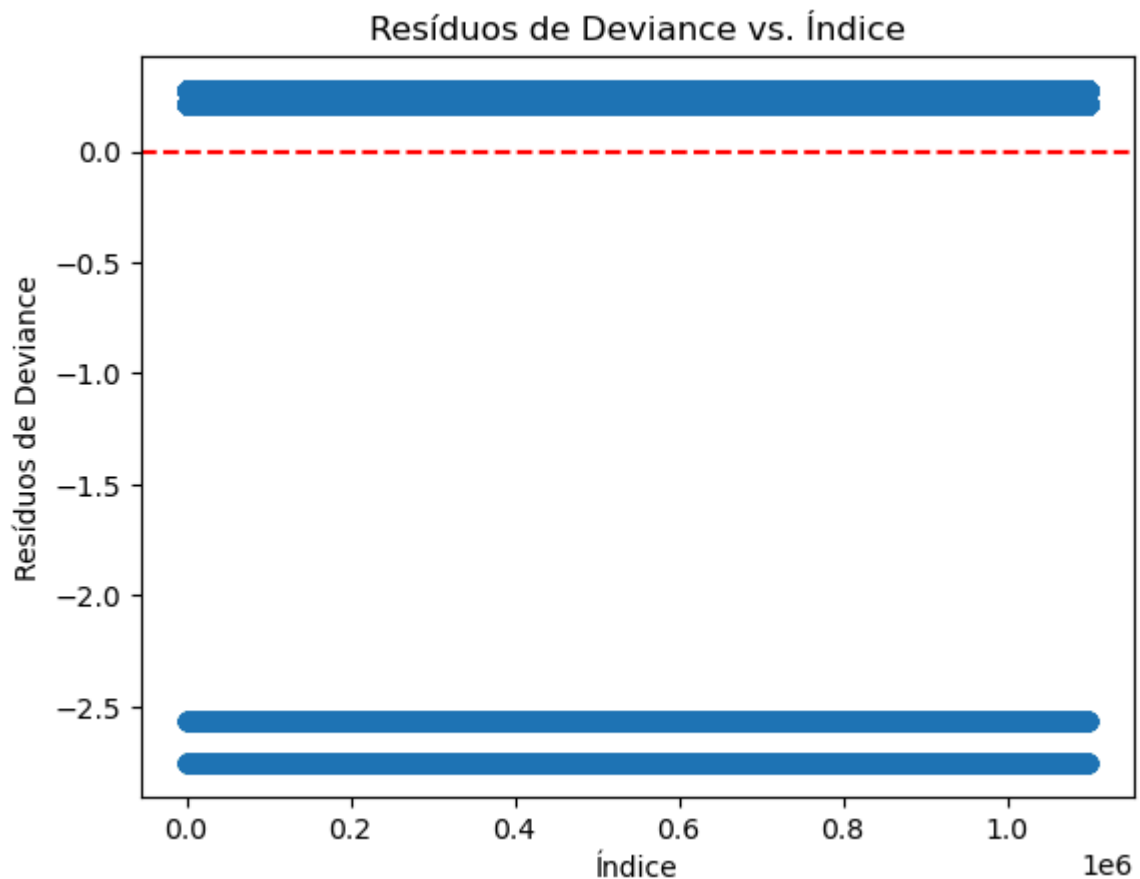
```
import matplotlib.pyplot as plt
import statsmodels.api as sm

# Resíduos vs. valores ajustados
valores_ajustados = modelo1.fittedvalues
plt.scatter(valores_ajustados, residuos_deviance, alpha=0.7)
plt.axhline(0, color='red', linestyle='--')
plt.title("Resíduos Deviance vs. Valores Ajustados")
plt.xlabel("Valores Ajustados")
plt.ylabel("Resíduos Deviance")
plt.show()
```



```
In [33]: plt.scatter(range(len(residuos_deviance)), residuos_deviance, alpha=0.7)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Índice')
plt.ylabel('Resíduos de Deviance')
plt.title('Resíduos de Deviance vs. Índice')
```

```
Out[33]: Text(0.5, 1.0, 'Resíduos de Deviance vs. Índice')
```



## Modelo 2: Mais de uma variável independente

Agora queremos modelar o óbito relacionado a pessoa ter diabetes e seu gênero.

```
In [34]: import statsmodels.api as sm  
import statsmodels.formula.api as smf
```

```
In [35]: relacao['diabetes'].value_counts()
```

```
Out[35]: diabetes  
IGNORADO    1003275  
SIM           67108  
NÃO          29094  
Name: count, dtype: int64
```

```
In [36]: import plotly.express as px  
  
px.pie(relacao, names="diabetes")
```

Observa-se um número alto de ignorado. Nestes casos devemos tomar muito cuidado, pois temos muitos dados faltantes.

Sempre é necessário avaliar se ao excluir esses dados massivos não iremos alterar a proporção do conjunto de dados.

```
In [37]: relacao2 = relacao.loc[relacao.diabetes != 'IGNORADO']
```

```
In [38]: px.pie(relacao2, names="diabetes")
```

Analizando a proporção do óbito **antes** da exclusão de ignorados em diabetes

```
In [39]: px.pie(relacao, names="obito")
```



Analizando a proporção do óbito **depois** da exclusão de ignorados em diabetes

```
In [40]: # Depois da exclusão de ignorados em diabetes  
px.pie(relacao2, names="obito")
```

```
In [41]: relacao2.dtypes
```

```
Out[41]: nome_munic          object
         codigo_ibge         int64
         idade              float64
         cs_sexo             category
         diagnostico_covid19 object
         data_inicio_sintomas object
         obito               category
         asma                object
         cardiopatia         object
         diabetes            object
         doenca_hematologica object
         doenca_hepatica     object
         doenca_neurológica  object
         doenca_renal        object
         imunodepressao      object
         obesidade           object
         outros_fatores_de_risco object
         pneumopatia         object
         puerpera            object
         sindrome_de_down    object
         dtype: object
```

```
In [42]: relacao2['diabetes']
```

```
Out[42]: 11      SIM
        13      NÃO
        18      SIM
        39      SIM
        106     SIM
        ...
        1102303  SIM
        1102313  NÃO
        1102317  NÃO
        1102318  SIM
        1102361  SIM
        Name: diabetes, Length: 96202, dtype: object
```

```
In [43]: relacao2['diabetes'] = relacao2['diabetes'].astype('category')

#relacao2.loc[:, 'diabetes'] = relacao2['diabetes'].astype('category').copy()
```

```
In [44]: relacao2.dtypes
```

```
Out[44]: nome_munic      object
         codigo_ibge     int64
         idade          float64
         cs_sexo         category
         diagnostico_covid19  object
         data_inicio_sintomas  object
         obito           category
         asma            object
         cardiopatia      object
         diabetes         category
         doenca_hematologica  object
         doenca_hepatica    object
         doenca_neurológica  object
         doenca_renal       object
         imunodepressao     object
         obesidade         object
         outros_fatores_de_risco  object
         pneumopatia       object
         puerpera          object
         sindrome_de_down   object
         dtype: object
```

## Criação do modelo 2

### Análise do modelo:

- Verificar a significancia dos coeficientes:
  - Estatisticamente significativo:  $p \leq 0,05$
  - Estatisticamente não é significativo:  $p > 0,05$
- Análise da Ausência de outliers e pontos de alavancagem
  - Deve estar entre -3 e 3
- Ausência de Multicolinearidade entre as variáveis independentes

```
In [45]: modelo2 = smf.glm(formula='obito ~ cs_sexo + diabetes', data=relacao2, family =
        print(modelo2.summary())
```

## Generalized Linear Model Regression Results

```

=====
=====
Dep. Variable:      ['obito[nao]', 'obito[sim]']    No. Observations:
96202
Model:                                GLM    Df Residuals:
96199
Model Family:                                Binomial    Df Model:
2
Link Function:                                Logit    Scale:
1.0000
Method:                                IRLS    Log-Likelihood:
-48162.
Date:                                Sat, 03 May 2025    Deviance:
96323.
Time:                                21:20:07    Pearson chi2:
9.60e+04
No. Iterations:                                5    Pseudo R-squ. (CS):
0.03069
Covariance Type:                                nonrobust
=====
=====

```

	coef	std err	z	P> z	[0.025
Intercept	0.9563	0.016	60.604	0.000	0.925
cs_sexo[T.MASCULINO]	-0.3245	0.016	-20.074	0.000	-0.356
diabetes[T.SIM]	0.8201	0.016	50.074	0.000	0.788

```

=====
=====

```

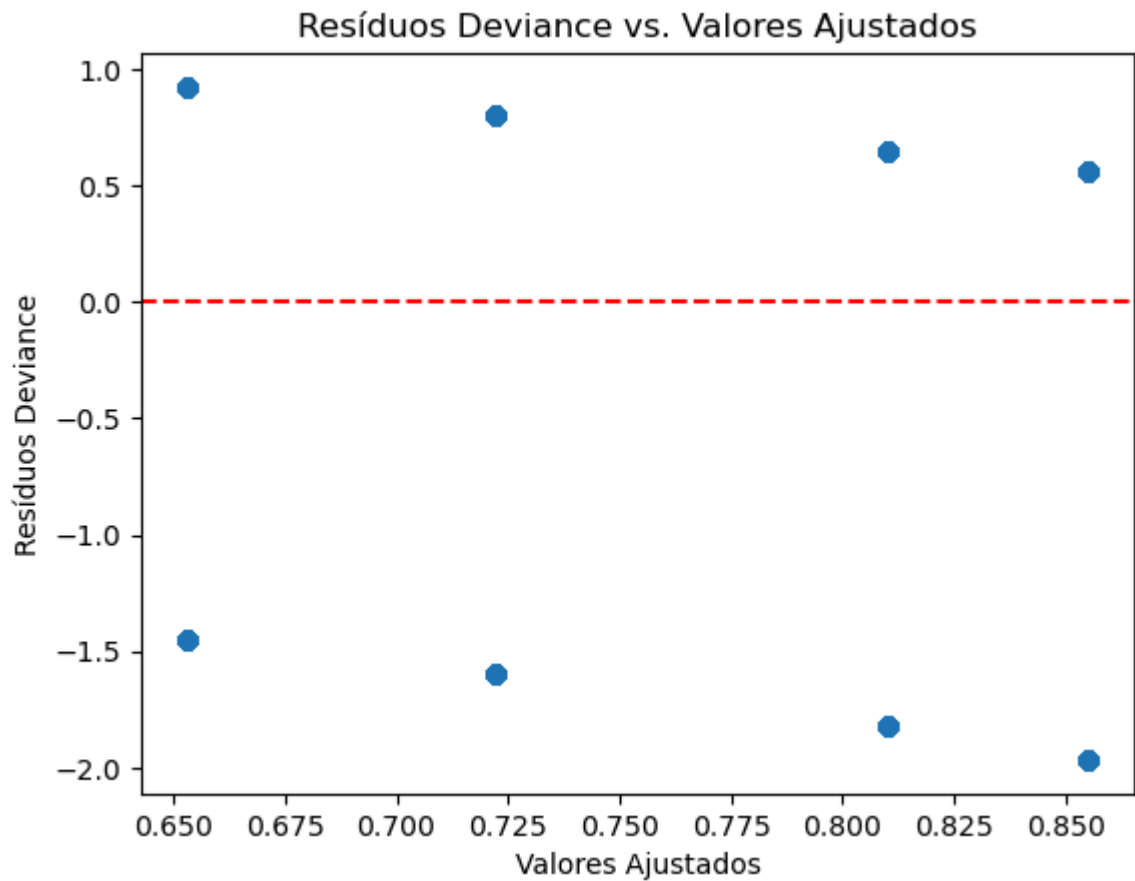
```

In [46]: # Resíduos deviance
residuos_deviance = modelo2.resid_deviance

# Resíduos de Pearson
residuos_pearson = modelo2.resid_pearson

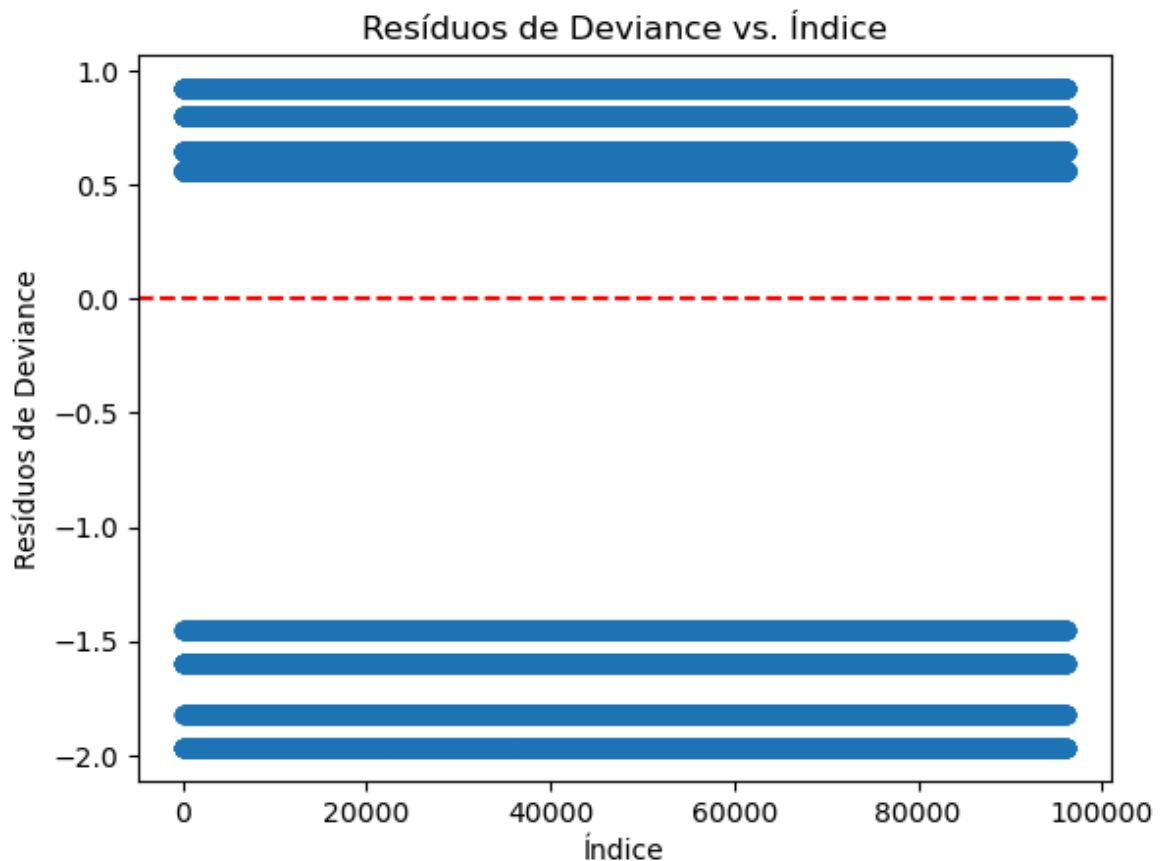
# Resíduos vs. valores ajustados
valores_ajustados = modelo2.fittedvalues
plt.scatter(valores_ajustados, residuos_deviance, alpha=0.7)
plt.axhline(0, color='red', linestyle='--')
plt.title("Resíduos Deviance vs. Valores Ajustados")
plt.xlabel("Valores Ajustados")
plt.ylabel("Resíduos Deviance")
plt.show()

```



```
In [47]: plt.scatter(range(len(residuos_deviance)), residuos_deviance, alpha=0.7)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Índice')
plt.ylabel('Resíduos de Deviance')
plt.title('Resíduos de Deviance vs. Índice')
```

```
Out[47]: Text(0.5, 1.0, 'Resíduos de Deviance vs. Índice')
```



## Deviance Residuals

### Hipótese Testada:

- $H_0$ : O modelo ajusta bem os dados.
- $H_a$ : O modelo não ajusta bem os dados.

### Condição de Aceitação/Rejeição:

- A deviance total é comparada com uma distribuição qui-quadrado com  $n - p$  graus de liberdade, onde  $n$  é o número de observações e  $p$  é o número de parâmetros no modelo.
- **Aceitação de  $H_0$** : Se o valor  $p$  calculado a partir da deviance é maior que  $\alpha$ , aceitamos  $H_0$  e concluímos que o modelo ajusta bem os dados.
- **Rejeição de  $H_0$** : Se o valor  $p$  é menor que  $\alpha$ , rejeitamos  $H_0$  e concluímos que o modelo não ajusta bem os dados.

```
In [48]: # Teste de Deviance para os Resíduos
deviance_test_statistic = modelo2.deviance
deviance_df = modelo2.df_resid
deviance_p_value = 1 - stats.chi2.cdf(deviance_test_statistic, deviance_df)

print("Teste de Deviance para os Resíduos:")
print("Estatística de teste:", deviance_test_statistic)
print("Graus de liberdade:", deviance_df)
print("Valor p:", deviance_p_value)
```

Teste de Deviance para os Resíduos:  
 Estatística de teste: 96323.38508958579  
 Graus de liberdade: 96199  
 Valor p: 0.387832545997736

## Pearson Chi-Square Test

### Hipótese Testada:

- $H_0$ : O modelo ajusta bem os dados.
- $H_a$ : O modelo não ajusta bem os dados.

### Condição de Aceitação/Rejeição:

- Calcula-se a estatística do teste qui-quadrado de Pearson, que segue uma distribuição qui-quadrado com  $n - p$  graus de liberdade.
- **Aceitação de  $H_0$** : Se o valor  $p$  é maior que  $\alpha$ , aceitamos  $H_0$  e concluímos que o modelo ajusta bem os dados.
- **Rejeição de  $H_0$** : Se o valor  $p$  é menor que  $\alpha$ , rejeitamos  $H_0$  e concluímos que o modelo não ajusta bem os dados.

```
In [49]: # Obtendo os resíduos de Pearson
residuos_pearson = modelo2.resid_pearson

# Teste de Pearson para os Resíduos
pearson_test_statistic = np.sum(residuos_pearson**2)
pearson_df = len(residuos_pearson) - modelo2.df_model - 1 # Graus de Liberdade
pearson_p_value = 1 - stats.chi2.cdf(pearson_test_statistic, pearson_df)

print("\nTeste de Pearson para os Resíduos:")
print("Estatística de teste:", pearson_test_statistic)
print("Graus de liberdade:", pearson_df)
print("Valor p:", pearson_p_value)
```

Teste de Pearson para os Resíduos:  
 Estatística de teste: 96035.52099281568  
 Graus de liberdade: 96199  
 Valor p: 0.6448276665303063

```
In [50]: modelo2.params
```

```
Out[50]: Intercept          0.956310
cs_sexo[T.MASCULINO]      -0.324520
diabetes[T.SIM]           0.820126
dtype: float64
```

```
In [51]: np.exp(modelo2.params[2])
```

```
Out[51]: 2.2707855496469618
```

### Comparação de modelos

```
In [52]: # Calcular a diferença nos Log-Likelihoods
diff_ll = modelo2.llf - modelo1.llf

# Calcular o número de graus de liberdade (número de parâmetros adicionais em mo
df = modelo2.df_model - modelo1.df_model

# Realizar o teste de razão de verossimilhança
p_value = 1 - stats.chi2.cdf(diff_ll, df)

# Imprimir o resultado do teste
print("Teste de Razão de Verossimilhança:")
print("Estatística de teste:", diff_ll)
print("Graus de liberdade:", df)
print("Valor p:", p_value)
```

Teste de Razão de Verossimilhança:  
 Estatística de teste: 96814.34122435428  
 Graus de liberdade: 1  
 Valor p: 0.0

## ROC Curve and AUC

Avalia a capacidade do modelo de discriminar entre as classes.

**Curva ROC:** Traça a taxa de verdadeiros positivos (sensibilidade) contra a taxa de falsos positivos (1 - especificidade) para vários limiares de classificação.

**AUC (Área Sob a Curva):** Um valor próximo de 1 indica excelente discriminação, enquanto um valor próximo de 0.5 indica discriminação aleatória. -

**Avaliação do Modelo:** Um modelo com AUC maior é considerado melhor em discriminar entre as classes.

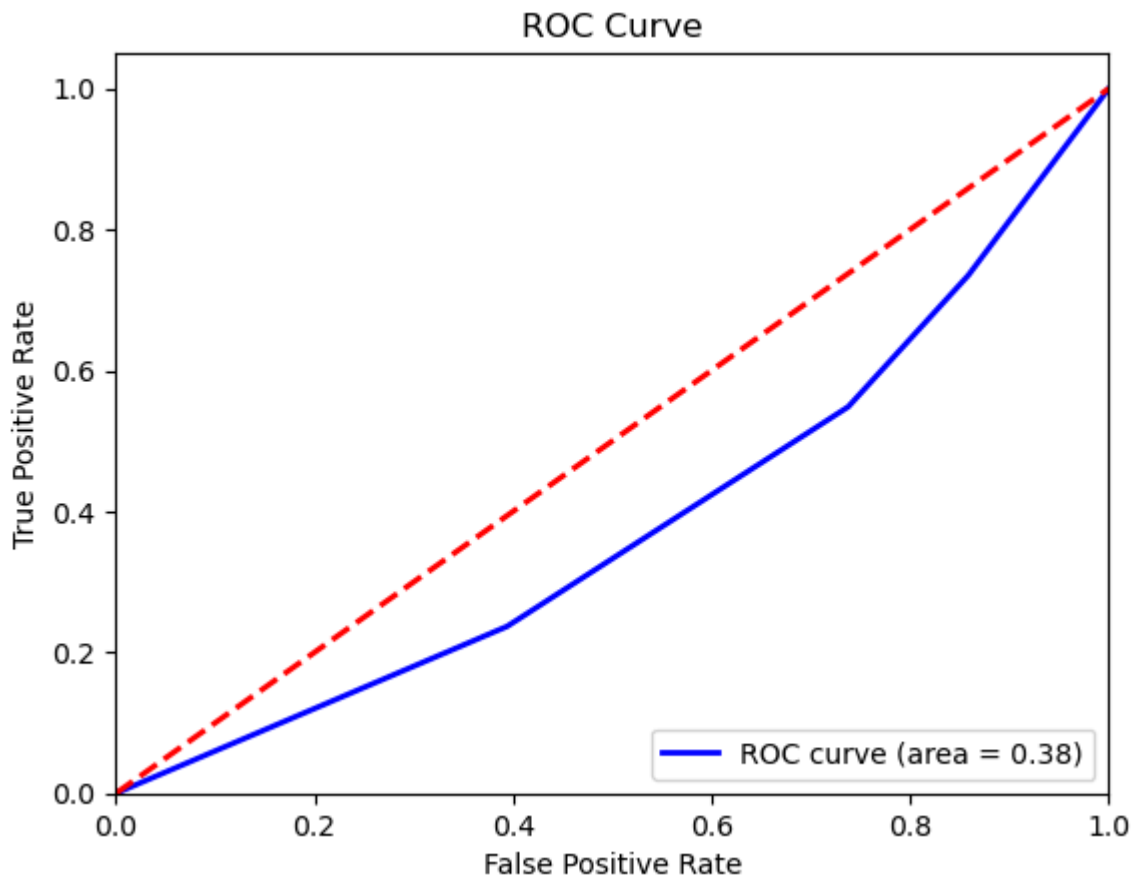
```
In [53]: from sklearn.metrics import roc_curve, auc

y = relacao2["obito"].replace({"nao":0, "sim":1})

# ROC Curve
fpr, tpr, _ = roc_curve(y, modelo2.fittedvalues)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()
```





## Modelo 3: Variável independente numérica

Relação da pessoa ir a óbito com sua idade Vamos fazer uma análise para a cidade de Santos e testar em outra cidade

```
In [54]: relacao3 = doenca_pre.loc[doenca_pre.nome_munic == 'Santos']
```

```
In [55]: relacao3.head()
```

```
Out[55]:
```

	nome_munic	codigo_ibge	idade	cs_sexo	diagnostico_covid19	data_inicio_sin
43	Santos	3548500	42.0	MASCULINO	CONFIRMADO	2020
47	Santos	3548500	43.0	MASCULINO	CONFIRMADO	2020
53	Santos	3548500	51.0	MASCULINO	CONFIRMADO	2020
206	Santos	3548500	23.0	FEMININO	CONFIRMADO	2020
267	Santos	3548500	46.0	MASCULINO	CONFIRMADO	2020

Verificando as dimensões da nossa tabela de dados

```
In [56]: relacao3.shape
```

```
Out[56]: (21728, 20)
```

Verificar como reconheceu a idade

```
In [57]: relacao3.dtypes
```

```
Out[57]: nome_munic          object
          codigo_ibge        int64
          idade              float64
          cs_sexo            object
          diagnostico_covid19 object
          data_inicio_sintomas object
          obito              int64
          asma               object
          cardiopatia        object
          diabetes           object
          doenca_hematologica object
          doenca_hepatica    object
          doenca_neurológica object
          doenca_renal        object
          imunodepressao     object
          obesidade          object
          outros_fatores_de_risco object
          pneumopatia        object
          puerpera           object
          sindrome_de_down    object
          dtype: object
```

Valores Missing (NaN)

```
In [58]: relacao3.isnull().sum()
```

```
Out[58]: nome_munic          0
          codigo_ibge        0
          idade              11
          cs_sexo            0
          diagnostico_covid19 0
          data_inicio_sintomas 145
          obito              0
          asma               0
          cardiopatia        0
          diabetes           0
          doenca_hematologica 0
          doenca_hepatica    0
          doenca_neurológica 0
          doenca_renal        0
          imunodepressao     0
          obesidade          0
          outros_fatores_de_risco 0
          pneumopatia        0
          puerpera           0
          sindrome_de_down    0
          dtype: int64
```

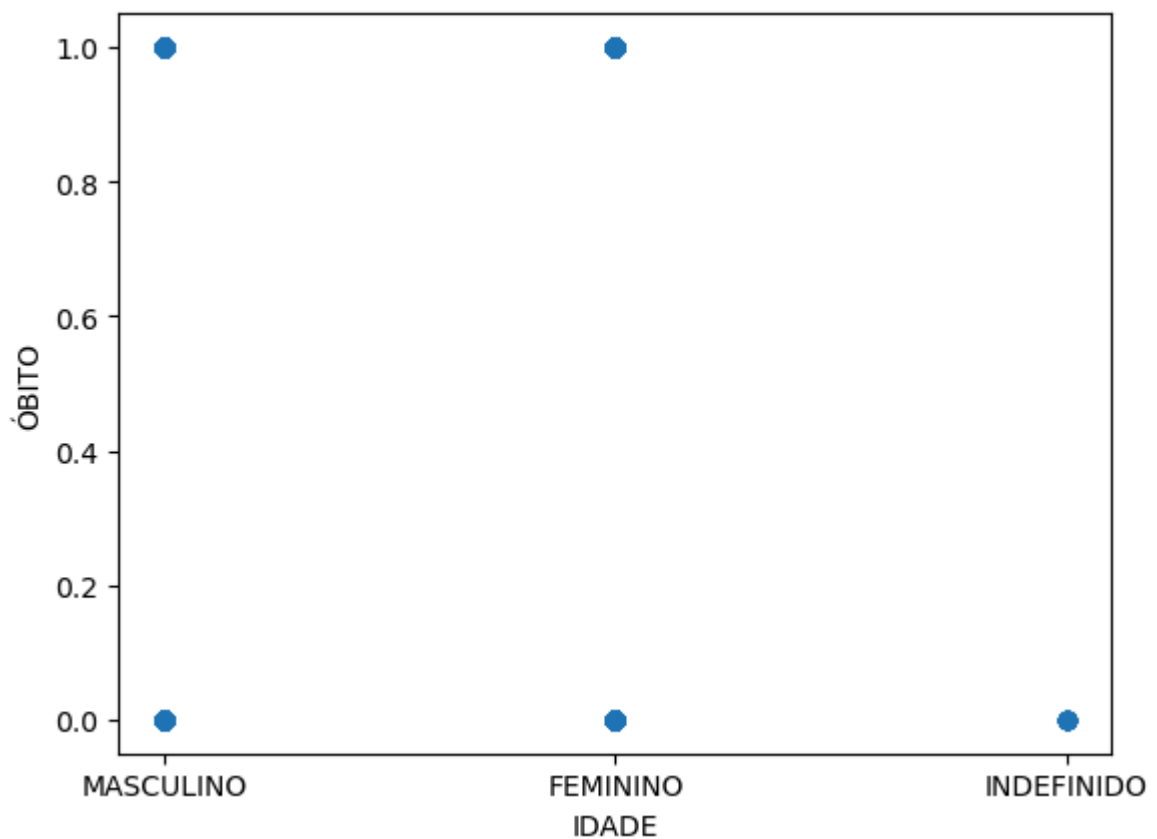
Excluir valores missing

```
In [59]: relacao3.dropna(subset=['idade'], inplace=True)
```

Verificando a relação entre idade e obito com um gráfico de dispersão

```
In [60]: import matplotlib.pyplot as plt
          plt.scatter(relacao3.cs_sexo, relacao3.obito)
```

```
plt.xlabel('IDADE')  
plt.ylabel('ÓBITO')  
plt.grid(False)  
plt.show()
```



### Correlação

```
In [61]: np.corrcoef(relacao3.obito, relacao3.idade)
```

```
Out[61]: array([[1.          , 0.28005415],  
                [0.28005415, 1.          ]])
```

## Criação do modelo 3 com StatsModels

```
In [62]: import statsmodels.api as sm  
import statsmodels.formula.api as smf
```

```
In [63]: modelo3 = smf.glm(formula='obito ~ idade', data=relacao3, family = sm.families.B  
print(modelo3.summary())
```

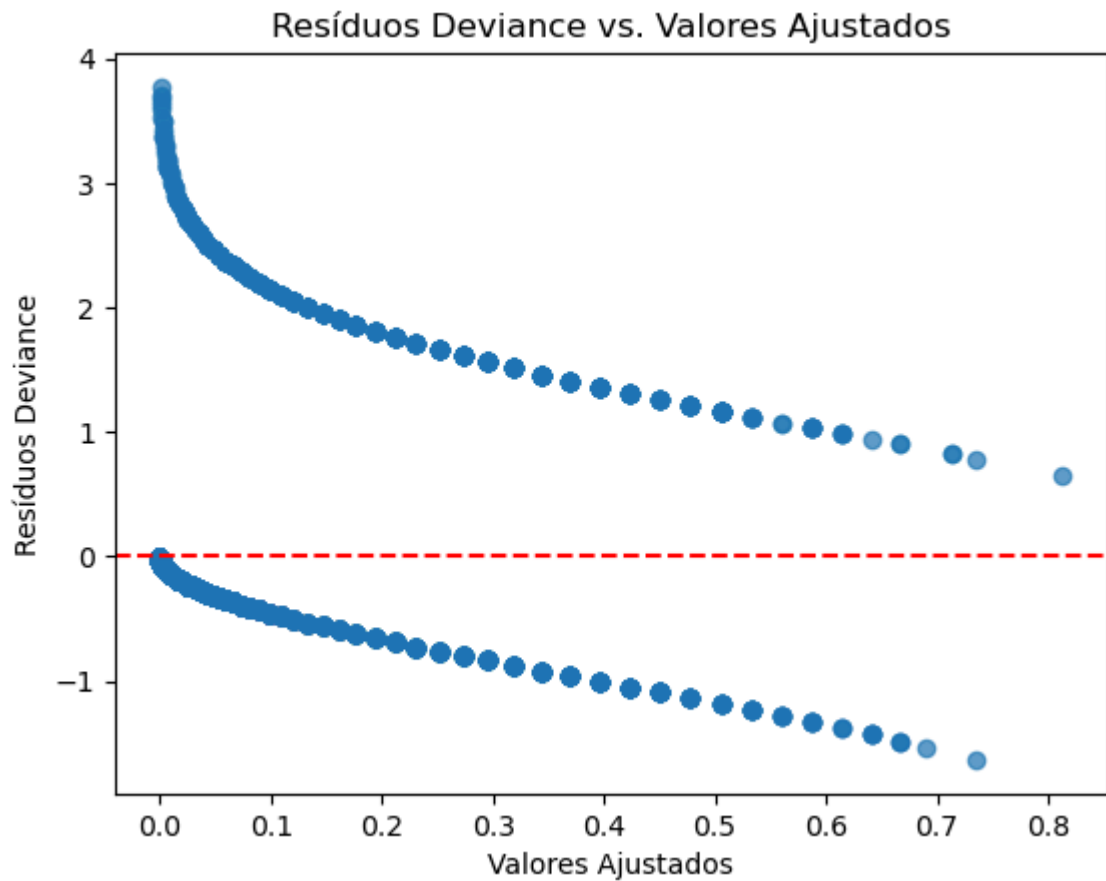
## Generalized Linear Model Regression Results

=====						
Dep. Variable:		obito	No. Observations:		21717	
Model:		GLM	Df Residuals:		21715	
Model Family:		Binomial	Df Model:		1	
Link Function:		Logit	Scale:		1.0000	
Method:		IRLS	Log-Likelihood:		-1799.5	
Date:		Sat, 03 May 2025	Deviance:		3599.0	
Time:		21:20:18	Pearson chi2:		1.81e+04	
No. Iterations:		8	Pseudo R-squ. (CS):		0.07611	
Covariance Type:		nonrobust				
=====						
	coef	std err	z	P> z	[0.025	0.975]
-----						
Intercept	-10.2259	0.243	-42.104	0.000	-10.702	-9.750
idade	0.1114	0.003	33.260	0.000	0.105	0.118
=====						

```
In [64]: # Resíduos deviance
residuos_deviance = modelo3.resid_deviance

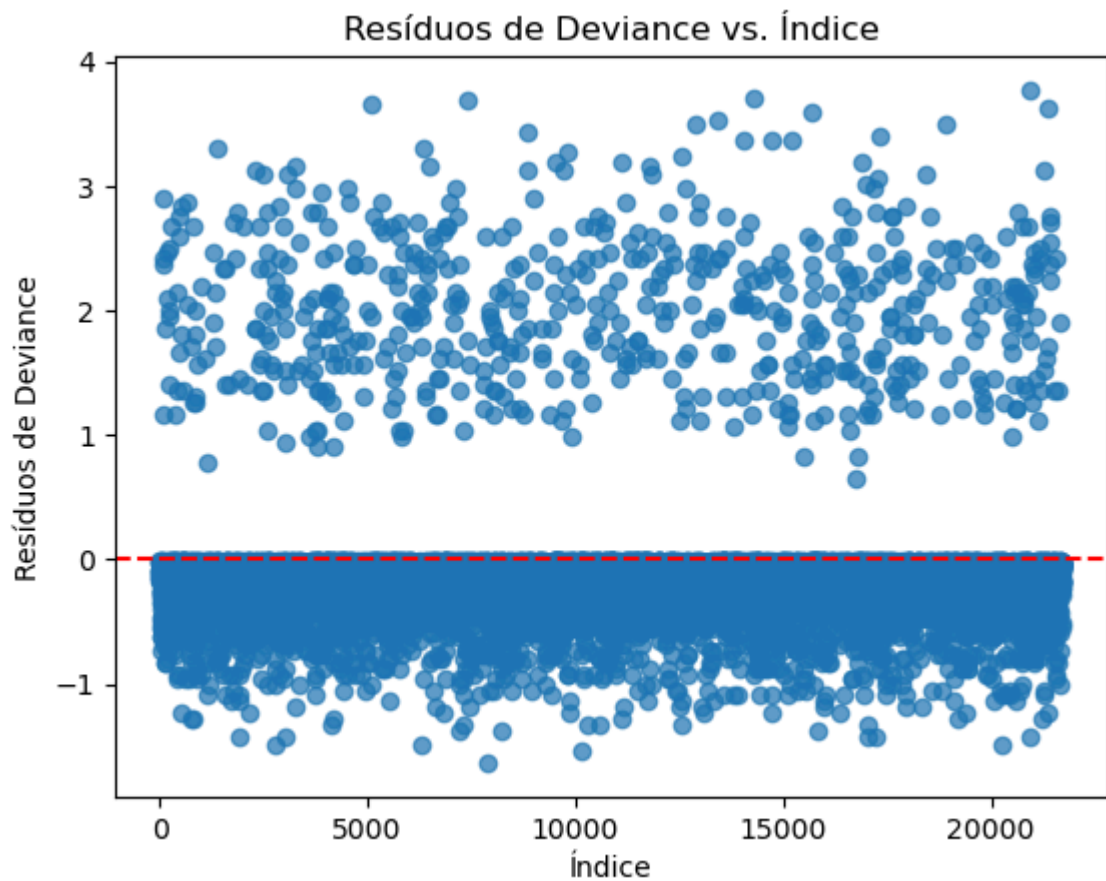
# Resíduos de Pearson
residuos_pearson = modelo3.resid_pearson

# Resíduos vs. valores ajustados
valores_ajustados = modelo3.fittedvalues
plt.scatter(valores_ajustados, residuos_deviance, alpha=0.7)
plt.axhline(0, color='red', linestyle='--')
plt.title("Resíduos Deviance vs. Valores Ajustados")
plt.xlabel("Valores Ajustados")
plt.ylabel("Resíduos Deviance")
plt.show()
```



```
In [65]: plt.scatter(range(len(residuos_deviance)), residuos_deviance, alpha=0.7)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Índice')
plt.ylabel('Resíduos de Deviance')
plt.title('Resíduos de Deviance vs. Índice')
```

```
Out[65]: Text(0.5, 1.0, 'Resíduos de Deviance vs. Índice')
```

**Análise do modelo:**

- Verificar a significancia dos coeficientes (Teste de Wald):
  - Estatisticamente significativo:  $p \leq 0,05$
  - Estatisticamente não é significativo:  $p > 0,05$
- Análise de Resíduos

## Deviance Residuals

### Hipótese Testada:

- $H_0$ : O modelo ajusta bem os dados.
- $H_a$ : O modelo não ajusta bem os dados.

### Condição de Aceitação/Rejeição:

- A deviance total é comparada com uma distribuição qui-quadrado com  $n - p$  graus de liberdade, onde  $n$  é o número de observações e  $p$  é o número de parâmetros no modelo.
- **Aceitação de  $H_0$** : Se o valor  $p$  calculado a partir da deviance é maior que  $\alpha$ , aceitamos  $H_0$  e concluímos que o modelo ajusta bem os dados.
- **Rejeição de  $H_0$** : Se o valor  $p$  é menor que  $\alpha$ , rejeitamos  $H_0$  e concluímos que o modelo não ajusta bem os dados.

```
In [66]: deviance_test_statistic = modelo3.deviance
deviance_df = modelo3.df_resid
deviance_p_value = 1 - stats.chi2.cdf(deviance_test_statistic, deviance_df)

print("Teste de Deviance para os Resíduos:")
print("Estatística de teste:", deviance_test_statistic)
print("Graus de liberdade:", deviance_df)
print("Valor p:", deviance_p_value)
```

Teste de Deviance para os Resíduos:  
Estatística de teste: 3598.977837680794  
Graus de liberdade: 21715  
Valor p: 1.0

## Pearson Chi-Square Test

### Hipótese Testada:

- $H_0$ : O modelo ajusta bem os dados.
- $H_a$ : O modelo não ajusta bem os dados.

### Condição de Aceitação/Rejeição:

- Calcula-se a estatística do teste qui-quadrado de Pearson, que segue uma distribuição qui-quadrado com  $n - p$  graus de liberdade.
- **Aceitação de  $H_0$** : Se o valor  $p$  é maior que  $\alpha$ , aceitamos  $H_0$  e concluímos que o modelo ajusta bem os dados.
- **Rejeição de  $H_0$** : Se o valor  $p$  é menor que  $\alpha$ , rejeitamos  $H_0$  e concluímos que o modelo não ajusta bem os dados.

```
In [67]: # Obtendo os resíduos de Pearson
residuos_pearson = modelo3.resid_pearson

# Teste de Pearson para os Resíduos
pearson_test_statistic = np.sum(residuos_pearson**2)
pearson_df = len(residuos_pearson) - modelo3.df_model - 1 # Graus de Liberdade
pearson_p_value = 1 - stats.chi2.cdf(pearson_test_statistic, pearson_df)

print("\nTeste de Pearson para os Resíduos:")
print("Estatística de teste:", pearson_test_statistic)
print("Graus de liberdade:", pearson_df)
print("Valor p:", pearson_p_value)
```

Teste de Pearson para os Resíduos:  
Estatística de teste: 18102.517494491673  
Graus de liberdade: 21715  
Valor p: 1.0

## Interpretação dos coeficientes

```
In [68]: # Razão de chance com Intervalo de confiança de 95%
np.exp(modelo3.params[1])
```

Out[68]: 1.1178251900757519

```
In [69]: modelo3.aic
```

Out[69]: 3602.977837680794

```
In [70]: modelo2.aic
```

Out[70]: 96329.38508958579

```
In [71]: modelo1.aic
```

Out[71]: 289956.06753829436

## ROC Curve and AUC

Avalia a capacidade do modelo de discriminar entre as classes.

**Curva ROC:** Traça a taxa de verdadeiros positivos (sensibilidade) contra a taxa de falsos positivos (1 - especificidade) para vários limiares de classificação.

**AUC (Área Sob a Curva):** Um valor próximo de 1 indica excelente discriminação, enquanto um valor próximo de 0.5 indica discriminação aleatória. -

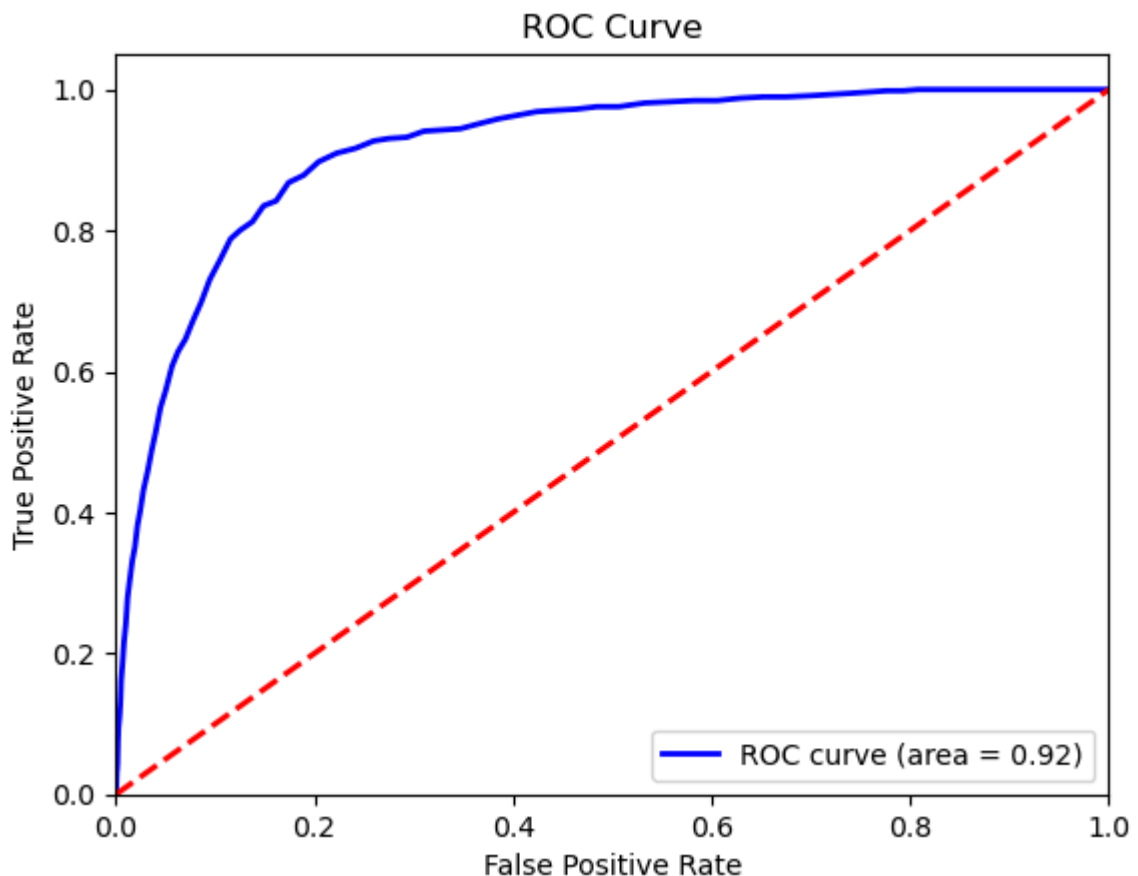
**Avaliação do Modelo:** Um modelo com AUC maior é considerado melhor em discriminar entre as classes.

```
In [72]: y = relacao3["obito"].replace({"nao":0, "sim":1})

# ROC Curve
fpr, tpr, _ = roc_curve(y, modelo3.fittedvalues)
roc_auc = auc(fpr, tpr)
```



```
plt.figure()
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()
```



## Criação do modelo 3 com Sklearn

In [73]: `from sklearn.linear_model import LogisticRegression`

In [74]: `relacao3.head()`

Out[74]:

	nome_munic	codigo_ibge	idade	cs_sexo	diagnostico_covid19	data_inicio_sin
43	Santos	3548500	42.0	MASCULINO	CONFIRMADO	2020
47	Santos	3548500	43.0	MASCULINO	CONFIRMADO	2020
53	Santos	3548500	51.0	MASCULINO	CONFIRMADO	2020
206	Santos	3548500	23.0	FEMININO	CONFIRMADO	2020
267	Santos	3548500	46.0	MASCULINO	CONFIRMADO	2020

Existe uma diferença na estruturação das informações nas bibliotecas Sklearn e Statsmodels.

No Statsmodels colocamos a formula :  $Y \sim X$  Já no Sklearn temos que criar as variáveis independentes e dependentes.

e neste caso ainda temos que realizar um transformação na variável X , para matriz.

```
In [75]: x = relacao3.iloc[:, 2].values  
y = relacao3.iloc[:, 6].values
```

```
In [76]: x
```

```
Out[76]: array([42., 43., 51., ..., 36., 75., 37.])
```

```
In [77]: y
```

```
Out[77]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

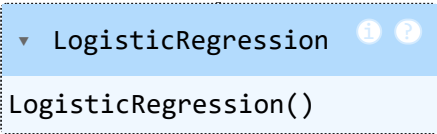
Transformando X para matriz :

```
In [78]: x = x.reshape(-1,1)  
x
```

```
Out[78]: array([[42.],  
                [43.],  
                [51.],  
                ...,  
                [36.],  
                [75.],  
                [37.]])
```

Ajustando o modelo

```
In [79]: modelo3s = LogisticRegression()  
modelo3s.fit(x, y)
```

```
Out[79]:  LogisticRegression()  
The tooltip shows the class name 'LogisticRegression' with an expand/collapse arrow, an information icon, and a help icon. Below the class name is the constructor signature 'LogisticRegression()'.
```

Verificando o coeficiente do modelo

```
In [80]: modelo3s.coef_
```

```
Out[80]: array([[0.11138135]])
```

Verificando o intercepto

```
In [81]: modelo3s.intercept_
```

```
Out[81]: array([-10.22562903])
```

Razão de chance com Intervalo de confiança de 95%

```
In [82]: np.exp(modelo3s.coef_)
```

```
Out[82]: array([[1.11782111]])
```

```
In [83]: modelo3s.aic
```

```
-----
AttributeError                                Traceback (most recent call last)
Cell In[83], line 1
----> 1 modelo3s.aic

AttributeError: 'LogisticRegression' object has no attribute 'aic'
```

CONCLUSÃO:

Para cada ano mais velho, o indivíduo aumenta em 1,12 a chance de ir a óbito

Fazendo o gráfico com a função sigmoide

```
In [84]: plt.scatter(x, y)
# Geração de novos dados para gerar a função sigmoide
x_teste = np.linspace(0, 120, 100)

def model(w): # função sigmoide
    return 1 / (1 + np.exp(-w))
# Geração de previsões (variável r) e visualização dos resultados
previsao = model(x_teste * modelo3s.coef_ + modelo3s.intercept_).ravel()
plt.plot(x_teste, previsao, color = 'red');
```

