Regressão Logística

Esse documento tem como principal objetivo mostrar a teoria da regressão logística, sua relação com a álgebra linear e um exemplo de implementação realizado a partir da programação. Assim, esse documento foi dividido nos seguintes subtópicos:

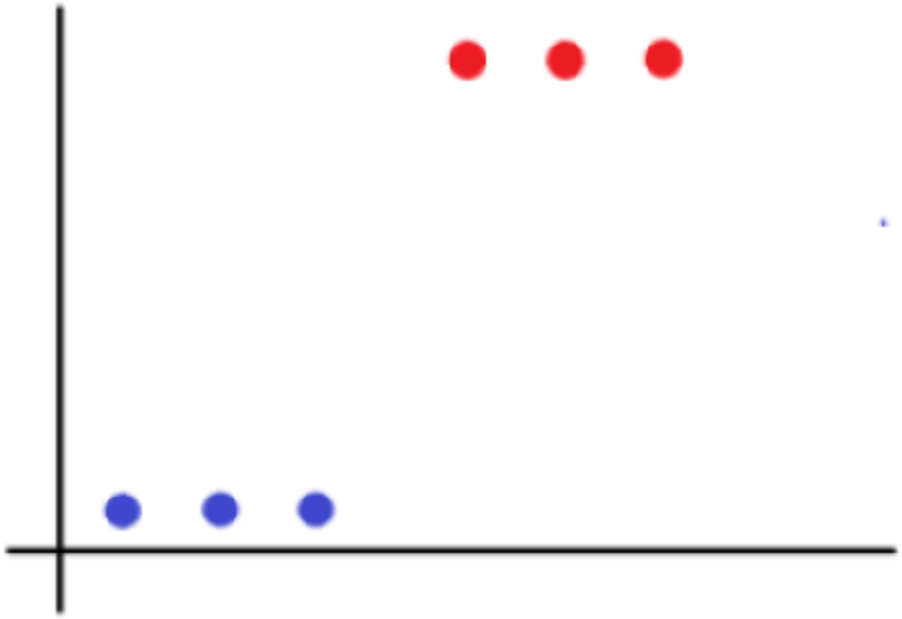
- Teoria
- Regressão Linear
- Implementação
- Conclusão

Teoria

A regressão logística é uma técnica de análise de dados com funcionalidades muito importantes voltadas, por exemplo, para a probabilidade e encontrar a relação entre duas variáveis distintas. Ela é muito utilizada em modelos de Machine Learning e Inteligência artificial, de acordo com a AWS, e possui diversos benefícios, como sua simplicidade de implementação, velocidade de processamento, flexibilidade e visibilidade.

Assim, é possível realizar uma análise preditiva a partir do resultado obtido da probabilidade de determinado cenário acontecer.

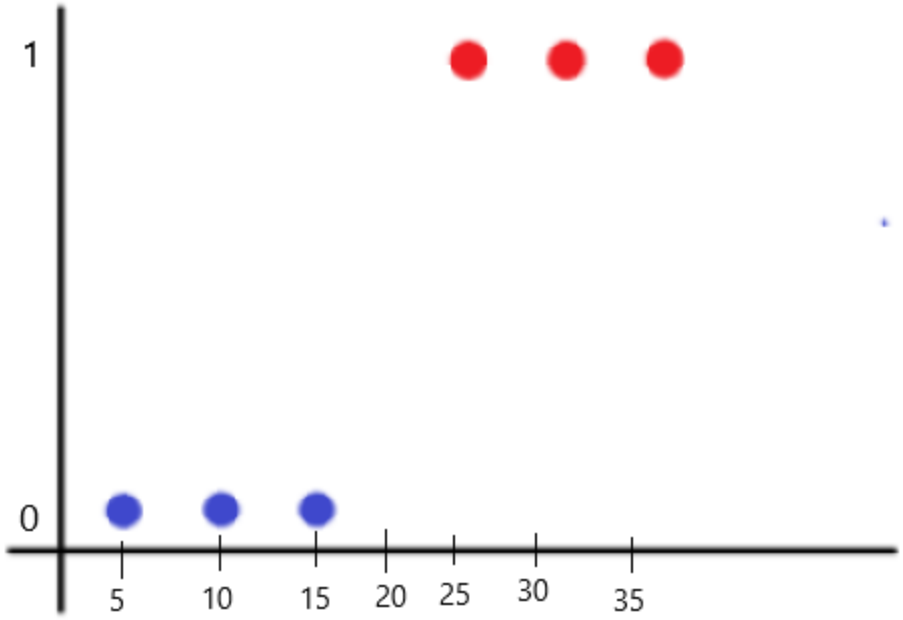
Vamos supor, por exemplo que somos uma agência de seguros e gostaríamos de saber a probabilidade de uma pessoa sofrer um acidente com base no tempo que ela passou na autoescola. Teríamos um gráfico da seguinte forma:



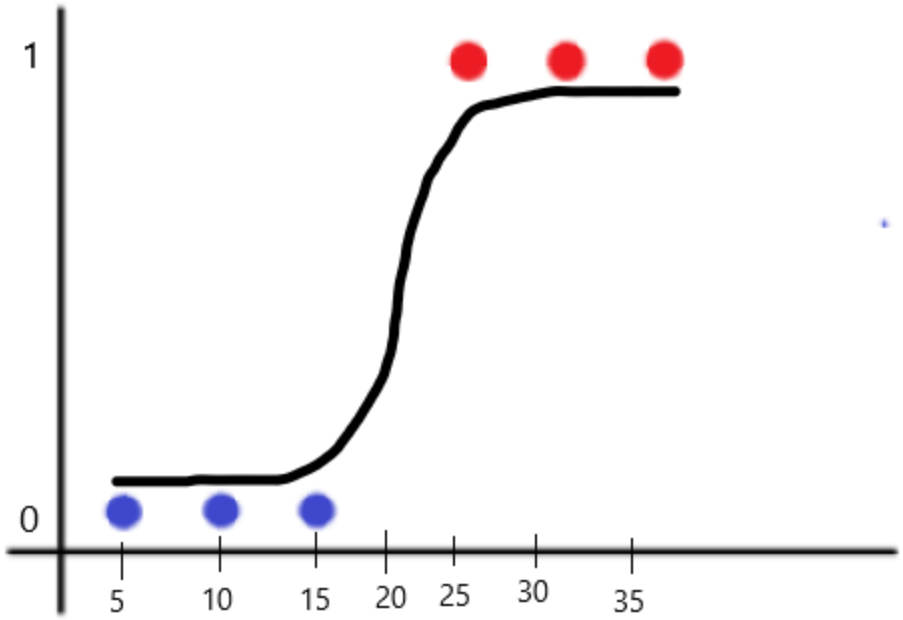
Os círculos vermelhos representam as pessoas que nunca sofreram acidente, enquanto os círculos azuis representam os que já sofreram, enquanto isso, o eixo x representa o número de dias que passaram na autoescola.

Se utilizássemos a regressão linear, ela não iria conseguir definir exatamente a probabilidade por causa da falta de linearidade em relação aos dados e seu limite em relação à variável binária de sofrer acidente. Isso ocorre, porque as pessoas que já sofreram acidente são classificadas em "Sim" ou "Não". Dessa forma, poderíamos representar numericamente esse gráfico da seguinte forma, sendo o eixo y se essas pessoas já sofreram acidente, e o eixo x o número de dias que eles permaneceram na autoescola.

Assim, a regressão logística geralmente é utilizada para realizar modelos de análise de dados de probabilidade de dados categóricos, muitas vezes binários como nesse exemplo.



Desse modo, ao utilizar a regressão logística, estaríamos desenhando uma sigmoide para entender a probabilidade de um dos resultados dessa variável binária acontecer, com isso, podemos perceber um gráfico da seguinte maneira:



Essa sigmoide apresenta a essa equação:

$$f(x) = \frac{1}{1 + e^{-x}}$$

A partir da equação, nota-se que todos os valores para f(x) estão entre 1 e 0, pois, caso x tenha valor igual a infinito, seu limite tenderá a 0, com a equação se igualando a 1/1. Enquanto isso, se seu expoente for igual a menos infinito, seu valor crescerá infinitamente e, consequentemente, tenderá a 0.

Nota-se que, quando os dados apresentados não possuem como resultado exatamente 1 e 0, eles são classificados a partir de sua proximidade, então todo número x que estiver classificado como 0.5 < x, será

considerado como 1.

Para calcular essa probabilidade de ser um ou outro, podemos representar pela equação dada a seguir:

$$P\left(Y\ =\ 1\mid X\ =\ x_i\right)\ =\ p_i$$

$$P\left(Y\ =\ 0\mid X\ =\ x_i\right)\ =1-\ p_i$$

A primeira equação representa as chances de darem sucesso (1) , enquanto a segunda representa as chances de darem fracasso (0).

Nesse caso não é utilizada a regressão linear, pois ela acaba superando o valor 1 do gráfico, podendo fornecer valores maiores que 1 e menor do que 0, violando o critério de probabilidade, cujos valores precisamo estar entre 0 e 1. Assim, a partir da regressão logística, temos essa condição de pé, além de uma possibilidade de ajuste de acordo com os dados apresentados no conjunto.

Regressão Linear

Além dessa sua importância e aplicações, podemos perceber que a equação da regressão logística apresenta uma certa semelhança com a equação da regressão linear, outra técnica muito utilizada para análise de dados e análise preditiva.

$$y\ =\ \beta_0+\beta_1x$$

Essa equação da regressão linear é composta pelos seguintes elementos:

y = variável resposta ou dependente

β_0 = intercepto

β_1 = coeficiente angular

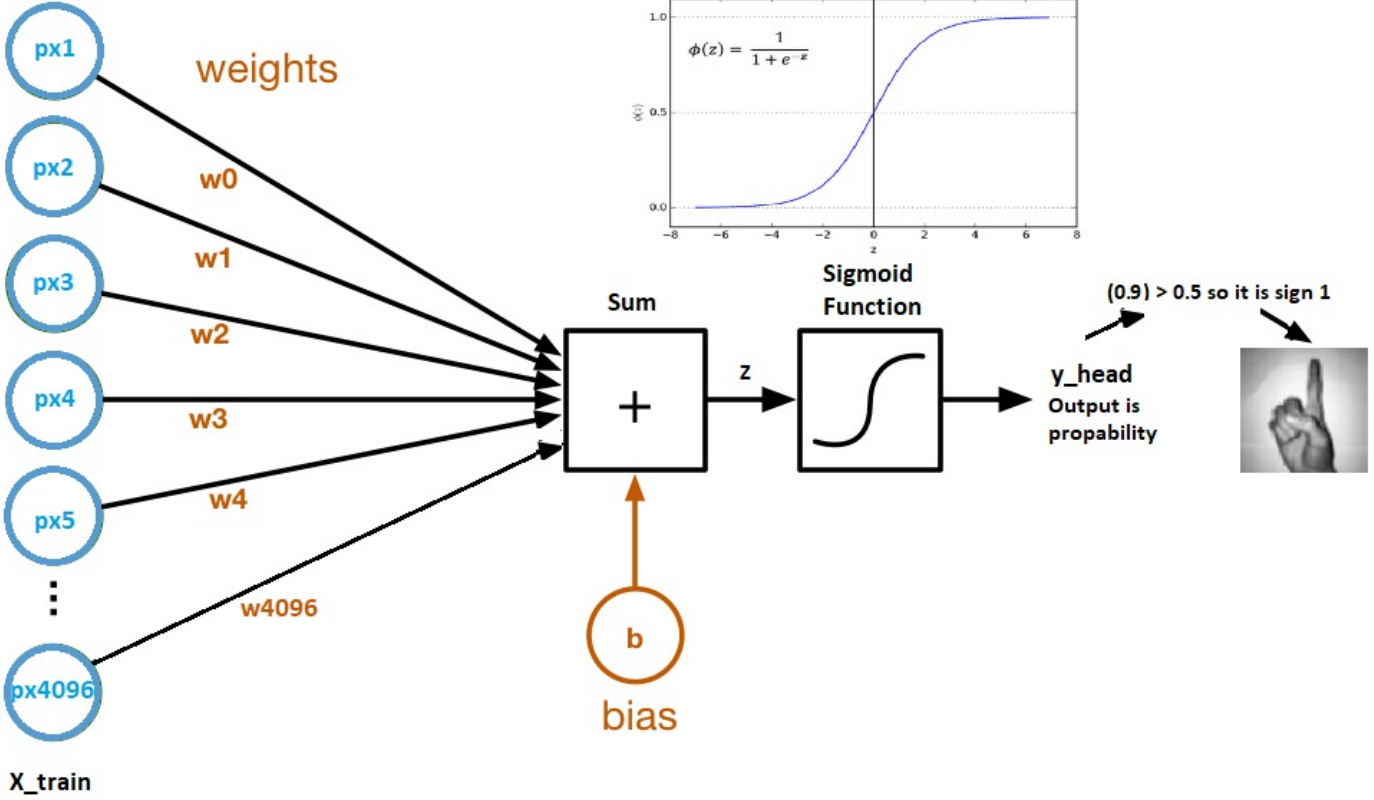
x = variável explicativa ou independente

Acaba que a regressão linear pode ser representada em forma de matriz a partir da seguinte multiplicação:

$$\begin{bmatrix}y_1\\y_2\\y_3\\\vdots\\y_n\end{bmatrix}=\begin{bmatrix}1&x_1&x_1^2&\dots&x_1^m\\1&x_2&x_2^2&\dots&x_2^m\\1&x_3&x_3^2&\dots&x_3^m\\\vdots&\vdots&\vdots&\ddots&\vdots\\1&x_n&x_n^2&\dots&x_n^m\end{bmatrix}\begin{bmatrix}\beta_0\\\beta_1\\\beta_2\\\vdots\\\beta_m\end{bmatrix}$$

A equação mostrada anteriormente não coincide totalmente com a multiplicação mostrada, pois ela representa uma regressão linear simples, enquanto a matriz representa as variações de regressão linear a partir do conjunto de resíduos apresentados.

Nota-se que nessa multiplicação de matrizes, a matriz composto pelas diferentes variações de x é composto por equações lineares. A partir disso, é possível notar a relação que o sistema apresenta com a álgebra linear, seus conceitos e aplicações.



Fonte:https://www.kaggle.com/code/tanavbajaj/logistic-regression-math-behind-without-sklearn

In [5]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

CAR PRICE (https://www.kaggle.com/datasets/shaistashaikh/carprice-assignment?resource=download)

Importando Base

In [6]:
```
#importando a base de dados
df = pd.read_csv("CarPrice_Assignment.csv")
df
```

Out[6]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	whe
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	
...	...	...	...	...	...	...	...	...	...	...
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	whe
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	

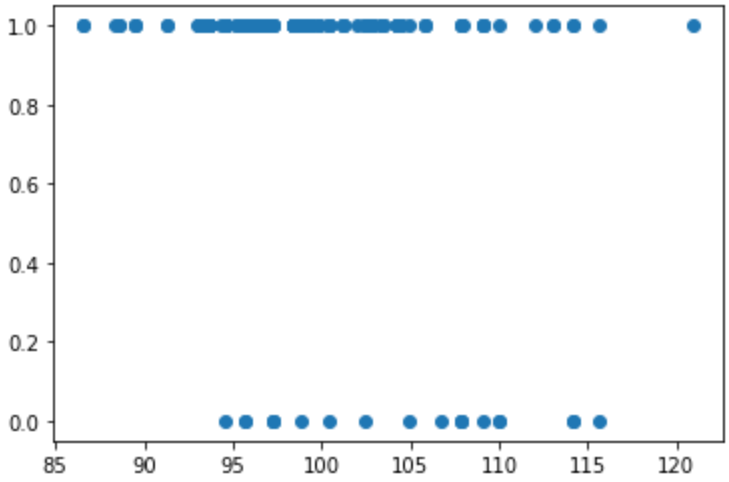
205 rows × 26 columns

In [7]:
```
#substituindo gas e diesel por 1 e 0
df2 = df.replace("diesel",0)
df3 = df2.replace("gas",1)
```

In [8]:
```
#definindo as variáveis independentes e a variável dependente sendo observada
y = df3["fueltype"]
x = df3.drop(df3.columns, axis=1)
```

Gráfico de Dispersão dos Dados

In [9]:
```
plt.figure()
plt.scatter(df3["wheelbase"], df3["fueltype"])
plt.show()
```



Definição de amostras para treinar o modelo e para testar o modelo

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(df3[["wheelbase"]],df3["fueltype"],tra
```

Dados de Teste

In [11]:
```
X_test
```

Out[11]:

	wheelbase
163	94.5
44	94.5
80	96.3

	wheelbase
175	102.4
85	96.3
67	110.0
115	107.9
132	99.1
172	98.4
89	94.5
173	102.4
110	114.2
34	93.7
81	96.3
40	96.5
45	94.5
108	107.9
25	93.7
200	109.1
187	97.3
127	89.5
73	120.9
75	102.7
29	95.9
131	96.1
33	93.7
0	88.6
149	96.9
38	96.5
146	97.0
9	99.5
104	91.3
5	99.8
135	99.1
100	97.2
154	95.7
118	93.7
133	99.1
14	103.5

	wheelbase
201	109.1
60	98.8

Treinamento do modelo

In [12]:
```
model = LogisticRegression()
model.fit(X_train, y_train)
print("Modelo treinado")
```

```
Modelo treinado
```

Previsão dos dados de teste

In [13]:
```
y_predicted = model.predict(X_test)
print("Previsão realizada")
```

```
Previsão realizada
```

Acurácia do modelo

In [14]:
```
acuracia = model.score(X_test,y_test)
print(f"A acurácia é de :{acuracia}")
```

```
A acurácia é de :0.8780487804878049
```

Comparação da previsão com a realidade dos dados de teste

1 - Gás

0 - Diesel

In [15]:
```
df3_real = df3["fueltype"].filter(items=X_test.index)
previsao_realidade = pd.DataFrame((X_test))
previsao_realidade["previsao"] = y_predicted
previsao_realidade["realidade"] = df3_real
previsao_realidade
```

Out[15]:

	wheelbase	previsao	realidade
163	94.5	1	1
44	94.5	1	1
80	96.3	1	1
175	102.4	1	1
85	96.3	1	1
67	110.0	1	0
115	107.9	1	1
132	99.1	1	1
172	98.4	1	1
89	94.5	1	1

	wheelbase	previsao	realidade
173	102.4	1	1
110	114.2	1	0
34	93.7	1	1
81	96.3	1	1
40	96.5	1	1
45	94.5	1	1
108	107.9	1	0
25	93.7	1	1
200	109.1	1	1
187	97.3	1	0
127	89.5	1	1
73	120.9	0	1
75	102.7	1	1
29	95.9	1	1
131	96.1	1	1
33	93.7	1	1
0	88.6	1	1
149	96.9	1	1
38	96.5	1	1
146	97.0	1	1
9	99.5	1	1
104	91.3	1	1
5	99.8	1	1
135	99.1	1	1
100	97.2	1	1
154	95.7	1	1
118	93.7	1	1
133	99.1	1	1
14	103.5	1	1
201	109.1	1	1
60	98.8	1	1

DIABETES (https://www.kaggle.com/datasets/mathchi/diabetes-data-set)

- Nº de Gravidezes X Diabetes

In [16]:

```
diabetes = pd.read_csv("diabetes.csv")
diabetes
```

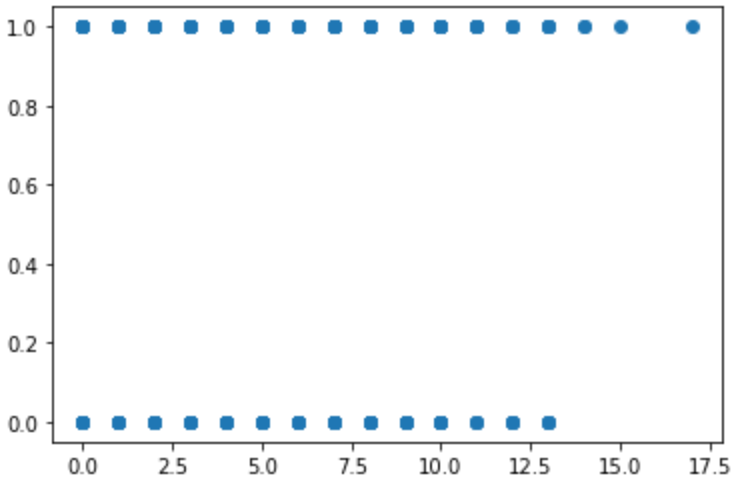
Out[16]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [17]:

```
plt.figure()
plt.scatter(diabetes["Pregnancies"], diabetes["Outcome"])
plt.show()
```



Definição de amostras para treinar o modelo e para testar o modelo

In [18]:

```
X_train, X_test, y_train, y_test = train_test_split(diabetes[["Pregnancies"]],diabetes["Ou
```

Dados de Teste

In [19]:

```
X_test
```

Out[19]:

	Pregnancies
317	3
705	6

	Pregnancies
642	6
589	0
349	5
...	...
442	4
570	3
457	5
103	1
544	1

154 rows × 1 columns

Treinamento do modelo

In [20]:
```
model = LogisticRegression()
model.fit(X_train, y_train)
print("Modelo treinado")
```

```
Modelo treinado
```

Previsão dos dados de teste

In [21]:
```
y_predicted = model.predict(X_test)
print("Previsão realizada")
```

```
Previsão realizada
```

Acurácia do modelo

In [22]:
```
acuracia = model.score(X_test,y_test)
print(f"A acurácia é de :{acuracia}")
```

```
A acurácia é de :0.6623376623376623
```

Comparação da previsão com a realidade dos dados de teste

1 - Diabética

0 - Não Diabética

In [24]:
```
diabetes_real = diabetes["Outcome"].filter(items=X_test.index)
previsao_realidade = pd.DataFrame((X_test))
previsao_realidade["previsao"] = y_predicted
previsao_realidade["realidade"] = diabetes_real
previsao_realidade
```

Out[24]:

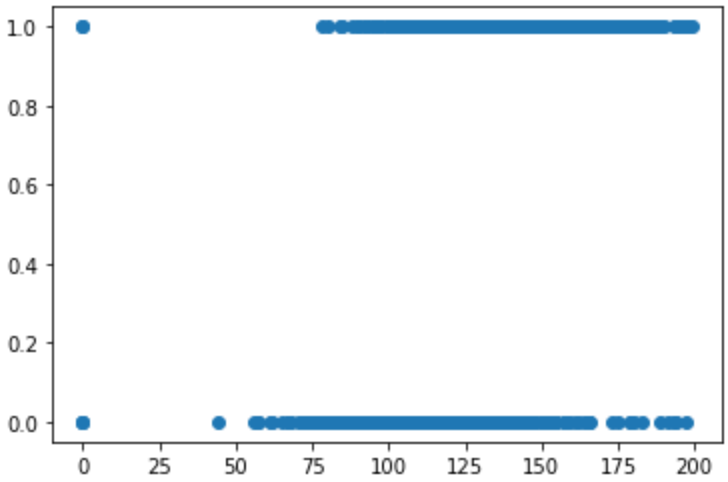
	Pregnancies	previsao	realidade
317	3	0	1
705	6	0	0

	Pregnancies	previsao	realidade
642	6	0	1
589	0	0	0
349	5	0	1
...	...	...	...
442	4	0	0
570	3	0	0
457	5	0	0
103	1	0	0
544	1	0	0

154 rows × 3 columns

- Glicose X Diabetes

In [33]:
```
plt.figure()
plt.scatter(diabetes["Glucose"], diabetes["Outcome"])
plt.show()
```



Definição de amostras para treinar o modelo e para testar o modelo

In [41]:
```
X_train, X_test, y_train, y_test = train_test_split(diabetes[["Glucose"]],diabetes["Outcor
```

Dados de Teste

In [42]:
```
X_test
```

Out[42]:

	Glucose
550	116
397	131

	Glucose
36	138
382	109
38	90
...	...
218	85
78	131
58	146
381	105
254	92

154 rows × 1 columns

Treinamento do modelo

In [43]:
```
model = LogisticRegression()
model.fit(X_train, y_train)
print("Modelo treinado")
```

```
Modelo treinado
```

Previsão dos dados de teste

In [44]:
```
y_predicted = model.predict(X_test)
print("Previsão realizada")
```

```
Previsão realizada
```

Acurácia do modelo

In [45]:
```
acuracia = model.score(X_test,y_test)
print(f"A acurácia é de :{acuracia}")
```

```
A acurácia é de :0.7142857142857143
```

Comparação da previsão com a realidade dos dados de teste

1 - Diabética

0 - Não Diabética

In [46]:
```
diabetes_real = diabetes["Outcome"].filter(items=X_test.index)
previsao_realidade = pd.DataFrame((X_test))
previsao_realidade["previsao"] = y_predicted
previsao_realidade["realidade"] = diabetes_real
previsao_realidade
```

Out[46]:

	Glucose	previsao	realidade
550	116	0	0
397	131	0	1

	Glucose	previsao	realidade
36	138	0	0
382	109	0	0
38	90	0	1
...	...	...	...
218	85	0	1
78	131	0	1
58	146	1	0
381	105	0	0
254	92	0	1

154 rows × 3 columns