


Aula 6 - Regressão - Aprendizado Supervisionado com Scykit-Learn

Introdução à Regressão

- Vamos explorar o outro tipo de aprendizado supervisionado: **regressão**.
- Em tarefas de regressão, a variável-alvo possui **valores contínuos**, como:
 - O PIB de um país
 - O preço de uma casa

Predizendo Níveis de Glicose no Sangue

- Para entender problemas de regressão, usaremos um conjunto de dados de saúde feminina.
- **Dataset:**
 diabetes_clean.csv 23.3KB
- **Objetivo:** prever níveis de glicose no sangue.

```
import pandas as pd
diabetes_df = pd.read_csv("diabetes_clean.csv")
print(diabetes_df.head())
```

	# pregnancies	# glucose	# diastolic	# triceps	# insulin	# bmi	# dpf	# age	# children
0	6	148	72	35	0	33.6	0.627	50	4
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

- O DataFrame contém:
 - Número de gestações
 - Espessura da dobra cutânea do tríceps
 - Níveis de insulina
 - Índice de massa corporal (IMC)
 - Idade
 - Diagnóstico de diabetes (1 = sim, 0 = não)

Criando Arrays de Features e Alvo

- O scikit-learn exige que as **features** e o **alvo (target)** estejam separados:
 - `X`: todas as colunas exceto a de glicose
 - `y`: coluna de glicose
- Ambos são convertidos em **arrays NumPy** usando `.values`

```
X = diabetes_df.drop("glucose", axis=1).values y =  
diabetes_df["glucose"].values print(type(X), type(y))
```

```
<class 'numpy.ndarray'> <class 'numpy.ndarray'>
```

Fazendo Previsões com uma Única Feature

- Vamos tentar prever a glicose a partir de uma única variável: o **IMC**
- Extraímos a coluna do IMC de `X` e salvamos como `X_bmi`
- `y` e `X_bmi` inicialmente são arrays 1D

```
X_bmi = X[:, 4] print(y.shape, X_bmi.shape)
```

```
(768,) (768,)
```

- Para o scikit-learn, `X_bmi` precisa ser 2D
- Usamos `.reshape(-1, 1)` para ajustar a forma

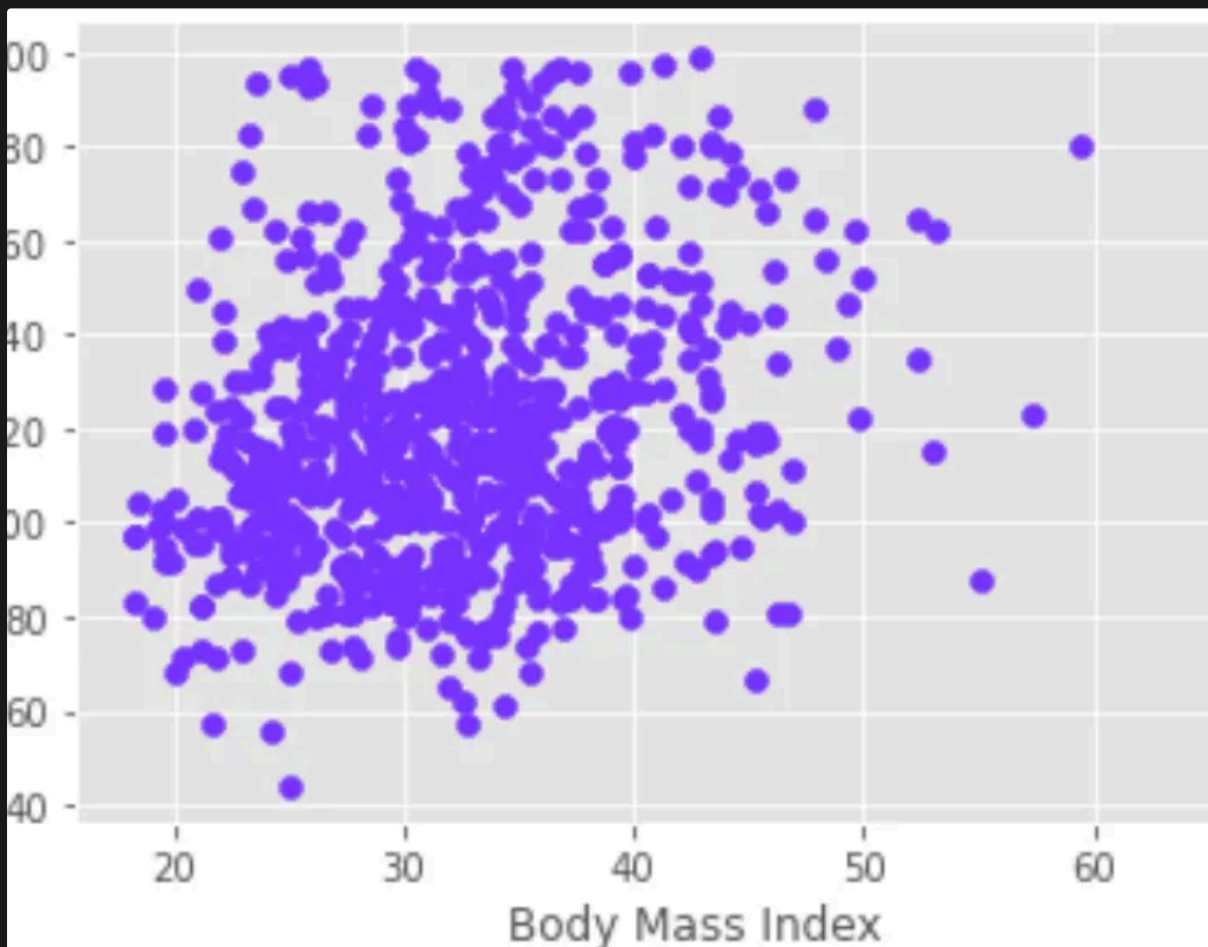
```
# -1 é equivalente a última linha, e 1 afirma que temos apenas uma única
coluna. X_bmi = X_bmi.reshape(-1, 1) # cria uma matriz 2D de uma única
coluna. print(y.shape, X_bmi.shape)
```

```
(768,) (768, 1)
```

Plotando Glicose vs. IMC

- Importamos `matplotlib.pyplot` como `plt`
- Usamos `plt.scatter(X_bmi, y)` para criar um gráfico de dispersão
- Adicionamos rótulos com `xlabel()` e `ylabel()`

```
import matplotlib.pyplot as plt plt.scatter(X_bmi, y) plt.ylabel("Blood
Glucose (mg/dl)") plt.xlabel("Body Mass Index") plt.show()
```



Análise Visual

- Em geral, quanto maior o IMC, maiores os níveis de glicose
- Há uma **correlação positiva** visível, ainda que moderada

Ajustando um Modelo de Regressão

- Usamos o modelo de **Regressão Linear**, que ajusta uma **linha reta** aos dados
- Importamos `LinearRegression` do `sklearn.linear_model`
- Instanciamos o modelo e o treinamos com `.fit(X_bmi, y)`
- Criamos as previsões com `.predict(X_bmi)`
- Reproduzimos o gráfico de dispersão e adicionamos a **linha de regressão** com `plt.plot(X_bmi, predictions)`

```
from sklearn.linear_model import LinearRegression reg = LinearRegression()
reg.fit(X_bmi, y) predictions = reg.predict(X_bmi) plt.scatter(X_bmi, y)
plt.plot(X_bmi, predictions) plt.ylabel("Blood Glucose (mg/dl)")
plt.xlabel("Body Mass Index") plt.show()
```



Resultado da Regressão

- A linha preta representa o modelo linear
- Indica uma **correlação positiva fraca a moderada** entre IMC e glicose

Vamos Praticar!

- Agora é sua vez de construir seu **próprio modelo de regressão**!


Exercício

Criando variáveis (features)

Neste capítulo, você irá trabalhar com um conjunto de dados chamado `sales_df`, que contém informações sobre os gastos com campanhas publicitárias em diferentes tipos de mídia, bem como o valor em dólares gerado em vendas pela respectiva campanha. O conjunto de dados precisa ser carregado por você, utilizando o Pandas. Aqui estão as duas primeiras linhas:

```
tv radio social_media sales 1 13000.0 9237.76 2409.57 46677.90 2 41000.0 1
5886.45 2913.41 150177.83
```

Conjunto de dados:

 advertising_and_sales_clean.csv 175.7KB

Você usará os gastos com publicidade como variáveis (features) para prever os valores de vendas, começando com a coluna `"radio"`. No entanto, antes de fazer qualquer previsão, será necessário criar os arrays de variáveis (features) e alvo (target), ajustando-os ao formato correto exigido pelo **scikit-learn**.

Instruções

1. Crie `x`, um array com os valores da coluna `"radio"` do DataFrame `sales_df`.
2. Crie `y`, um array com os valores da coluna `"sales"` do DataFrame `sales_df`.
3. Redimensione `x` para que seja um array NumPy bidimensional.
4. Imprima o formato (shape) de `x` e `y`.

```
import numpy as np # Create X from the radio column's values X = ____ #  
Create y from the sales column's values y = ____ # Reshape X X = ____ #  
Check the shape of the features and targets print(____)
```

Exercício

Construindo um modelo de regressão linear

Agora que você criou os arrays de variáveis (features) e alvo (target), você irá treinar um modelo de regressão linear com todos os valores de entrada.

Como o objetivo aqui é apenas avaliar a relação entre a variável preditora e o alvo, não é necessário dividir os dados em conjuntos de treino e teste.

`x` e `y` já foram carregados para você no exercício anterior.

Instruções

1. Importe `LinearRegression`.
2. Instancie um modelo de regressão linear.
3. Use `x` para prever os valores de vendas, armazenando as previsões em uma variável chamada `predictions`.

```
# Import LinearRegression from ____ import ____ # Create the model  
reg = ____() # Fit the model to the data ____ # Make predictions  
predictions = ____ print(predictions[:5])
```

Exercício

Visualizando um modelo de regressão linear

Agora que você construiu seu modelo de regressão linear e o treinou com todas as observações disponíveis, pode visualizá-lo para entender melhor como o modelo se ajusta aos dados. Isso permite interpretar a relação entre os gastos com publicidade no rádio e os valores de vendas.

As variáveis `x` (array com os valores de rádio), `y` (valores de vendas) e `predictions` (valores previstos pelo modelo para `y` dado `x`) já foram carregadas do exercício anterior.

Instruções

1. Importe `matplotlib.pyplot` como `plt`.
2. Crie um gráfico de dispersão (scatter plot) visualizando `y` em relação a `x`, com os pontos em azul.
3. Trace uma linha em **vermelho** representando `predictions` em relação a `x`.
4. Exiba o gráfico.

```
# Import matplotlib.pyplot import __.__ as __ # Create scatter plot  
plt.scatter(__, __, color="___") # Create line plot plt.plot(__,  
__, color="___") plt.xlabel("Radio Expenditure ($)") plt.ylabel("Sales  
($)") # Display the plot plt.__( )
```

