Aprendizado de Máquina - Treinar/Testar

‹ Anterior Próximo ›

Avalie seu modelo

Em Machine Learning criamos modelos para prever o resultado de determinados eventos, como no capítulo anterior onde previmos a emissão de CO2 de um carro quando sabíamos o peso e o tamanho do motor.

Para medir se o modelo é bom o suficiente, podemos usar um método chamado Train/Test.

O que é Treinar/Teste

Train/Test é um método para medir a precisão do seu modelo.

Ele é chamado de Treinar/Teste porque você divide o conjunto de dados em dois conjuntos: um conjunto de treinamento e um conjunto de teste.

80% para treinamento e 20% para teste.

Você *treina* o modelo usando o conjunto de treinamento.

Você *testa* o modelo usando o conjunto de testes.

Treinar o modelo significa *criar* o modelo.

Testar o modelo significa testar a precisão do modelo.

Comece com um conjunto de dados

Comece com um conjunto de dados que você deseja testar.

Nosso conjunto de dados ilustra 100 clientes em uma loja e seus hábitos de compra.

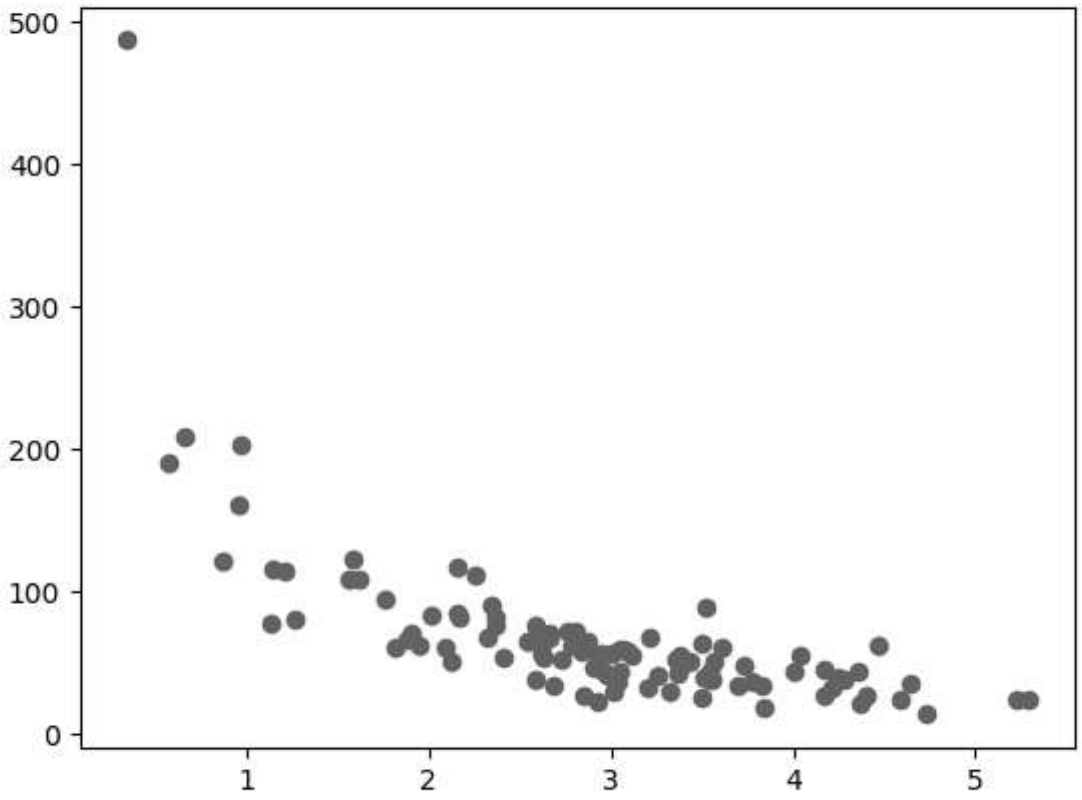
Exemplo

```
import numpy
import matplotlib.pyplot as plt
numpy.random.seed(2)

x = numpy.random.normal(3, 1, 100)
y = numpy.random.normal(150, 40, 100) / x

plt.scatter(x, y)
plt.show()
```

Resultado:

O eixo x representa o número de minutos antes de fazer uma compra.

O eixo y representa a quantidade de dinheiro gasto na compra.



Executar exemplo »

Dividir em Trem/Teste

O conjunto de *treinamento* deve ser uma seleção aleatória de 80% dos dados originais.

O conjunto de *teste* deve ser os 20% restantes.

```
train_x = x[:80]
train_y = y[:80]

test_x = x[80:]
test_y = y[80:]
```

Exibir o conjunto de treinamento

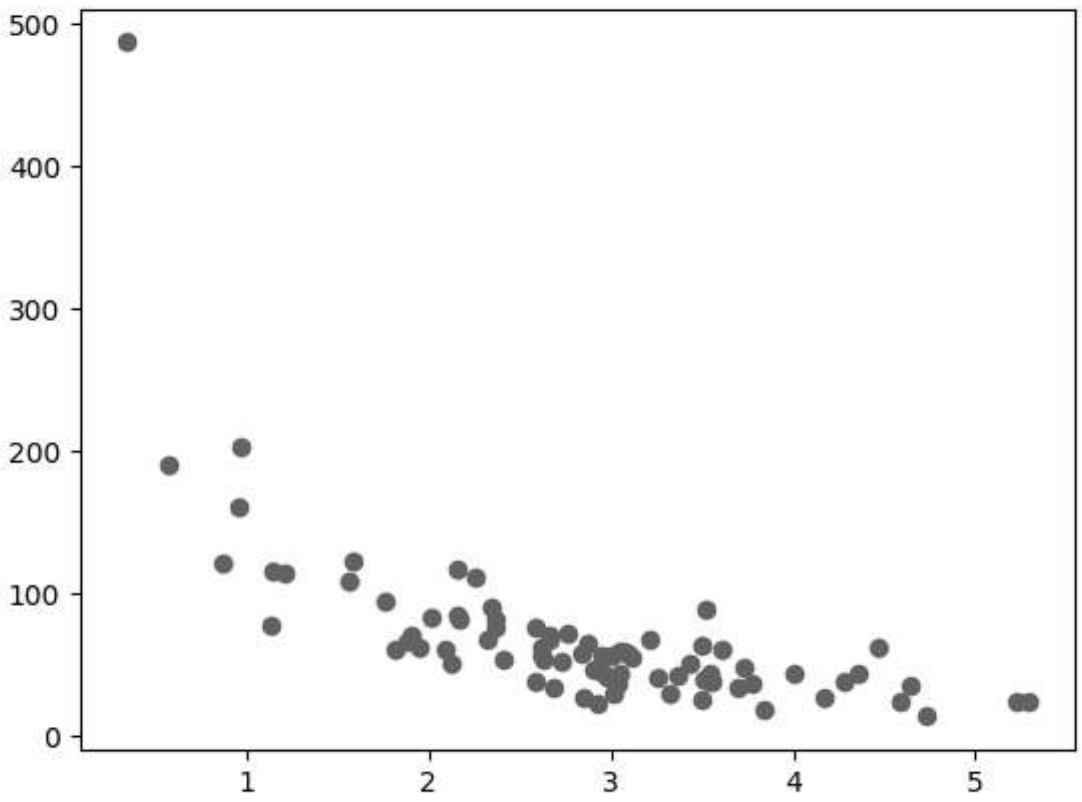
Display the same scatter plot with the training set:

Example

```
plt.scatter(train_x, train_y)
plt.show()
```

Result:

It looks like the original data set, so it seems to be a fair selection:



Run example »

Display the Testing Set

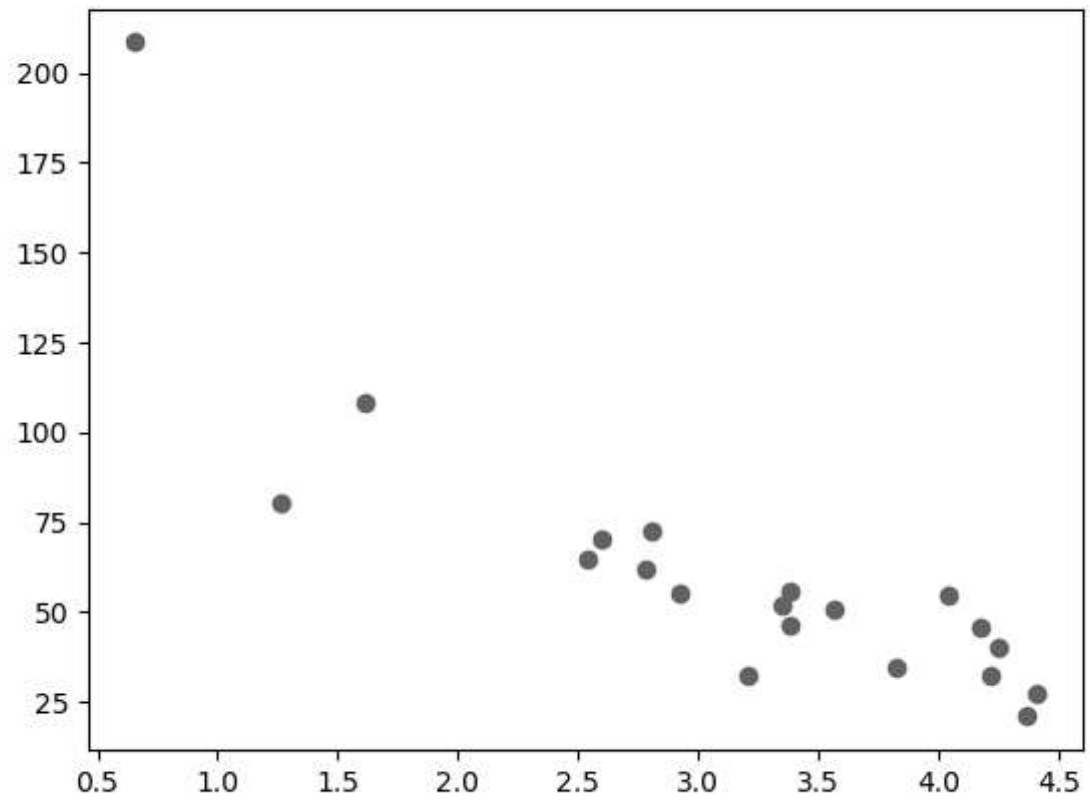
To make sure the testing set is not completely different, we will take a look at the testing set as well.

Example

```
plt.scatter(test_x, test_y)
plt.show()
```

Result:

The testing set also looks like the original data set:



Run example »

Fit the Data Set

What does the data set look like? In my opinion I think the best fit would be a polynomial regression, so let us draw a line of polynomial regression.

To draw a line through the data points, we use the `plot()` method of the matplotlib module:

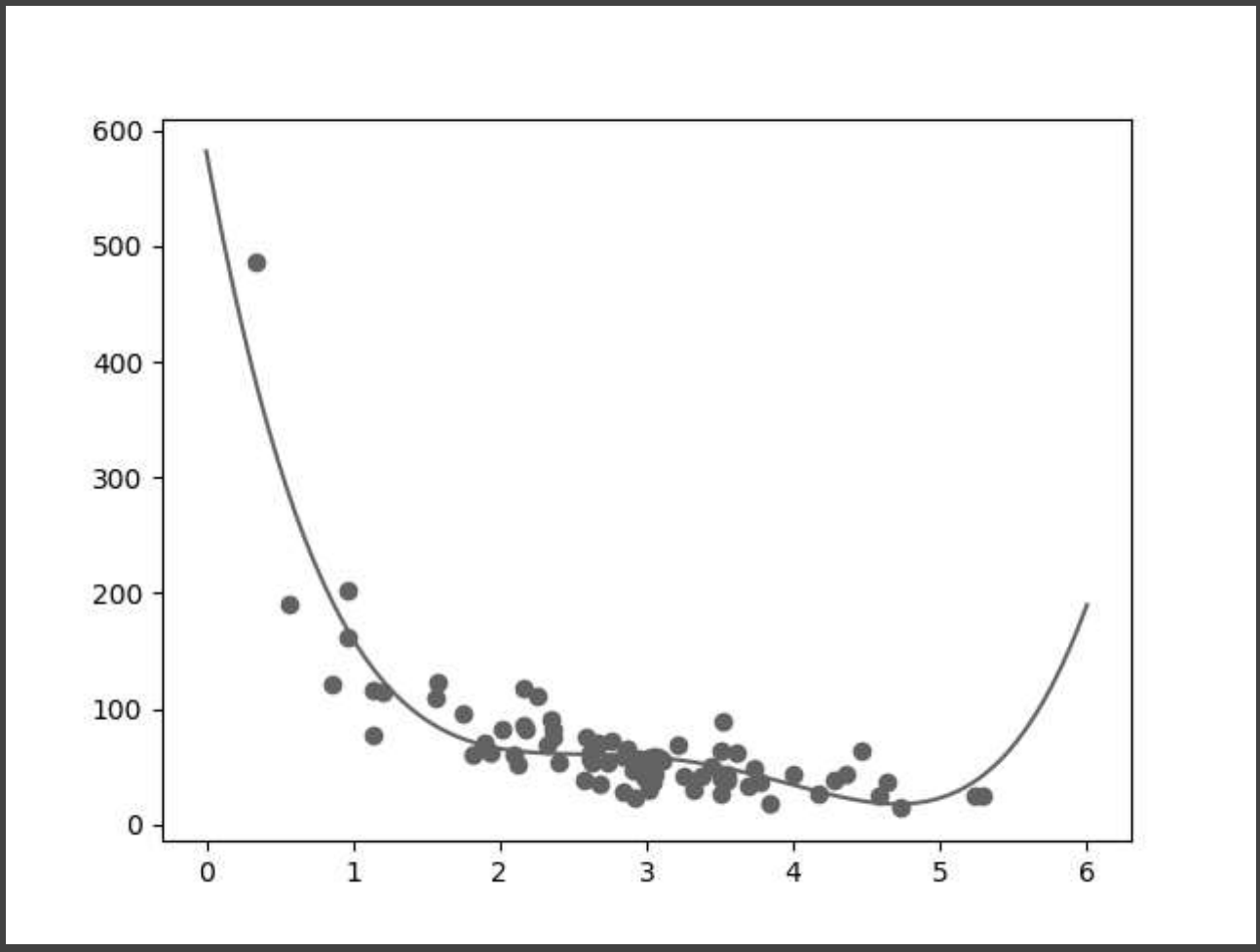
Example

Draw a polynomial regression line through the data points:

```
import numpy
import matplotlib.pyplot as plt
numpy.random.seed(2)

x = numpy.random.normal(3, 1, 100)
```

```
y = numpy.random.normal(150, 40, 100) / x

train_x = x[:80]
train_y = y[:80]

test_x = x[80:]
test_y = y[80:]

mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))

myline = numpy.linspace(0, 6, 100)

plt.scatter(train_x, train_y)
plt.plot(myline, mymodel(myline))
plt.show()
```

Result:



Run example »

The result can back my suggestion of the data set fitting a polynomial regression, even though it would give us some weird results if we try to predict values outside of

the data set. Example: the line indicates that a customer spending 6 minutes in the shop would make a purchase worth 200. That is probably a sign of overfitting.

But what about the R-squared score? The R-squared score is a good indicator of how well my data set is fitting the model.

R2

Remember R2, also known as R-squared?

It measures the relationship between the x axis and the y axis, and the value ranges from 0 to 1, where 0 means no relationship, and 1 means totally related.

The sklearn module has a method called `r2_score()` that will help us find this relationship.

In this case we would like to measure the relationship between the minutes a customer stays in the shop and how much money they spend.

Example

How well does my training data fit in a polynomial regression?

```
import numpy
from sklearn.metrics import r2_score
numpy.random.seed(2)

x = numpy.random.normal(3, 1, 100)
y = numpy.random.normal(150, 40, 100) / x

train_x = x[:80]
train_y = y[:80]

test_x = x[80:]
test_y = y[80:]

mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))

r2 = r2_score(train_y, mymodel(train_x))

print(r2)
```

Try it Yourself »

Note: The result 0.799 shows that there is a OK relationship.

Bring in the Testing Set

Now we have made a model that is OK, at least when it comes to training data.

Now we want to test the model with the testing data as well, to see if gives us the same result.

Example

Let us find the R2 score when using testing data:

```
import numpy
from sklearn.metrics import r2_score
numpy.random.seed(2)

x = numpy.random.normal(3, 1, 100)
y = numpy.random.normal(150, 40, 100) / x

train_x = x[:80]
train_y = y[:80]

test_x = x[80:]
test_y = y[80:]

mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))

r2 = r2_score(test_y, mymodel(test_x))

print(r2)
```

Try it Yourself »

Note: The result 0.809 shows that the model fits the testing set as well, and we are confident that we can use the model to predict future values.

Predict Values

Now that we have established that our model is OK, we can start predicting new values.

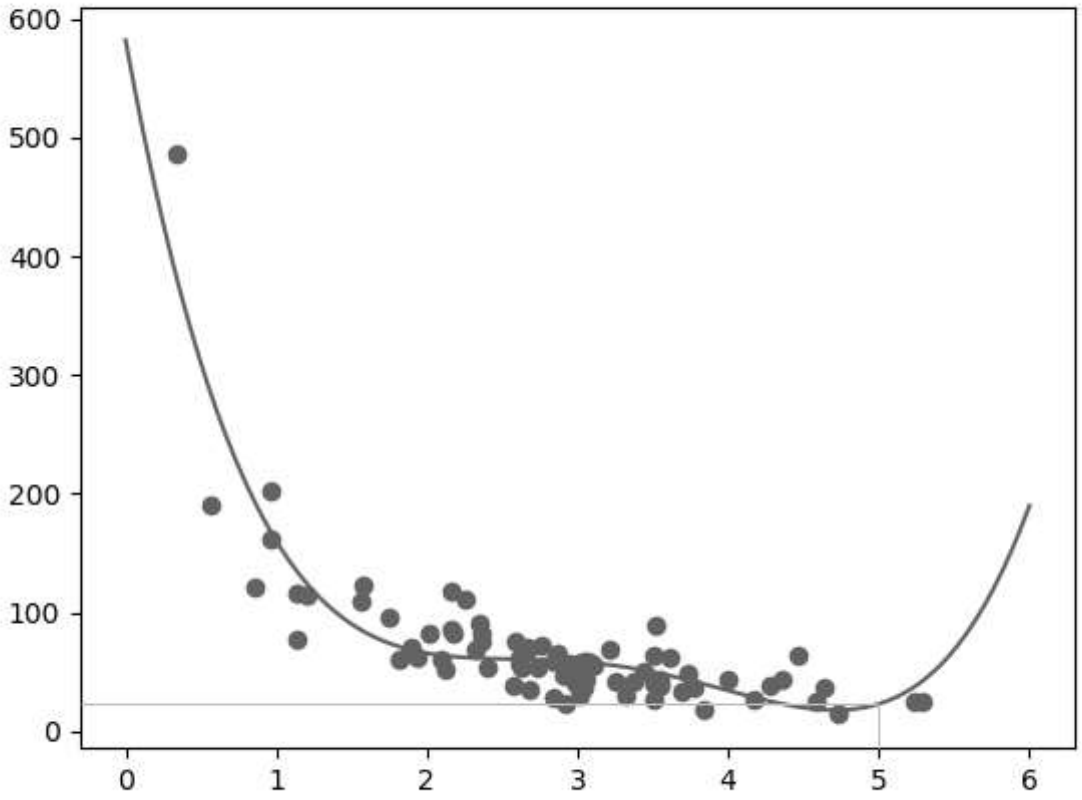
Example

Quanto dinheiro um cliente comprador gastará se ficar na loja por 5 minutos?

```
print(mymodel(5))
```

Executar exemplo »

O exemplo previu que o cliente gastaria 22,88 dólares, como parece corresponder ao diagrama:

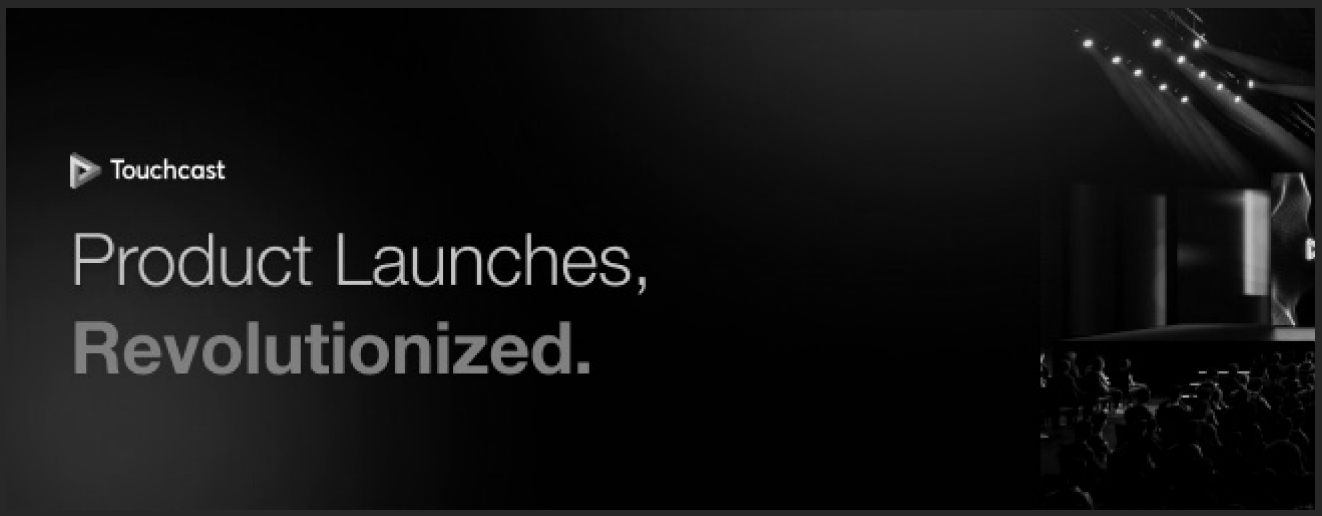


‹ Anterior

Próximo ›

SELETOR DE CORES

PROPAGANDA



PROPAGANDA

Reportar erro

Fórum

Sobre

Comprar

Principais tutoriais

Tutorial HTML Tutorial
CSS Tutorial
JavaScript
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery

Principais referências

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference