Projeto ML (Machine Learning)

Nesta aula, vamos aprender na prática como treinar um modelo de aprendizado de máquina, fazendo uso de algumas bibliotecas como scikit-learn, numpy e matplotlib. O primeiro passo é criar um novo ambiente virtual, vamos utilizar o venv que vem instalado por padrão com o interpretador python:

```
python -m venv mlreg
```

Em seguida, vamos ativar este ambiente, e instalar as dependencias desse projeto:

```
Scripts\activate.bat
pip install scikit-learn matplotlib numpy
```

Pronto, agora podemos realizar a inclusão destes no nosso script

```
import numpy as np
import matplotlib.pyplot as pl
from sklearn.linear_model import LinearRegression
```

Com o nosso setup inicial finalizado, vamos atacar o problema em questão. Para este exercício vamos utilizar um data set completamente fictício, e diferentemente do que você pode encontrar no mundo real, este data set já está completamente tratado e pronto para uso. Em uma situação de pesquisa científica, é muito comum que os dados passem por um tratamento antes de sua utilização no aprendizado de máquina, no entanto, não vamos abordar este tipo de manutenção neste momento.

O nosso data set vai ser constituído de um input, que vamos chamar de x, e um output, que vamos chamar de y. Para que possamos trabalhar com a biblioteca de regressões lineares do scikit-learn, precisamos que nossos inputs estejam organizados em uma matriz de duas dimensões, para isso, vamos aproveitar de um método disponível na biblioteca numpy, chamado reshape, que com os argumentos (-1, 1) realiza exatamente essa tarefa.

```
x = np.array([4, 10, 14, 19, 25, 31]).reshape(-1, 1)
y = np.array([6, 19, 14, 29, 25, 41])
```

Com o nosso data set pronto, podemos utilizar a classe LinearRegression para criar um modelo computacional, para isso, vamos utilizar o método fit(), que recebe como parametros, os nossos inputs e os outputs. Esse processo também é muitas vezes conhecido como a etapa de treinamento.

```
model = LinearRegression().fit(x, y)
```

Agora podemos utilizar o nosso modelo para prever os valores de y com base em x, vamos utilizar a função predict(x) do nosso modelo para isso.

```
y_prev = model.predict(x)
print(y_prev)
# [ 7.40067912 14.20543294 18.74193548 24.41256367 31.21731749 38.02207131]
```

Observe que os valores obtidos diferem bastante dos valores de y informados lá no começo, mas é importante lembrar que os modelos de regressão linear são utilizados para obter uma expressão estatisticamente significativa sobre as variáveis do estudo, ao contrário de uma expressão exata. Podemos verificar o coeficiente de determinação do nosso modelo utilizando o método score()

```
r_2 = model.score(x, y)
```

```
print(r_2)
# 0.8447769015355864
```

Um score de 0.84 é bastante satisfatório, apesar de valores próximos de 1 retratarem com maior acurácia a realidade dos dados, é importante não utilizar modelos que acabem por representar um overfitting ao data set.

Para ficar mais interessante, vamos aprender como podemos visualizar essas informações em um gráfico, vamos utilizar o matplotlib para isso. No nosso gráfico, vamos representar com pontos espalhados, utilizando o método scatter(), os valores de x e y em um sistemas de coordenadas cartesiana, vamos adicionar a essa representação, a linha traçada (usando o método plot()) pelo nosso modelo de regressão.

```
plot.scatter(x, y, color="red")
plot.plot(x, y_prev, color="black", linewidth=2)
plot.show()
```

Com isso finalizamos o nosso pequeno exercício sobre regressões lineares com python, lembre-se que essa é só a pontinha do iceberg, e que existe muito conteúdo para se aprender sobre a área de machine learning, não deixe de estudar e colocar seus conhecimentos a prova com exercícios e projetos!

```
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plot

x = np.array([4, 10, 14, 19, 25, 31]).reshape(-1, 1)
y = np.array([6, 19, 14, 29, 25, 41])

model = LinearRegression().fit(x, y)

r_2 = model.score(x, y)
print(r_2)

y_prev = model.predict(x)
print(y_prev)

plot.scatter(x, y, color="red")
plot.plot(x, y_prev, color="black", linewidth=2)
plot.show()
```