📌 **Exercício de Código**

Chegou a hora da mão no código! Vamos treinar o nosso modelo?

Se precisar, vamos deixar aqui o passo a passo seguido pela instrutora Carol!

Parte 1:

Nós vamos utilizar o modelo LinearRegression da biblioteca sklearn e fazemos o import assim:

```
from sklearn.linear_model import LinearRegression
```

Criamos o objeto:

```
model = LinearRegression()
```

E então enviando os parâmetros de treino na função fit

```
model.fit(X_train_scaled, y_train)
```

Vamos fazer as predições do nosso conjunto de teste utilizando a função predict:

```
model.predict(X_test_scaled)
```

Vamos atribuir o resultado para uma variável que vamos chamar de y_pred:

```
y_pred = model.predict(X_test_scaled)
```

Após treinar um modelo de machine learning, vamos avaliar seu desempenho para entender o quão bem ele está fazendo previsões.

Fazemos o import das funções da biblioteca do sklearn assim:

```
from sklearn.metrics import mean_squared_error, r2_score
```

Vamos calcular o erro quadrático médio, o MSE.

```
mse = mean_squared_error(y_test, y_pred)
```

Vamos calcular mais uma métrica o MAE. Primeiro, precisamos importar a função. Podemos adicionar à importação anterior e executar em seguida.

```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

Agora sim, fazemos o cálculo do MAE:

```
mae = mean_absolute_error(y_test, y_pred)
```

Vamos agora calcular o R^2

```
r2 = r2_score(y_test, y_pred)
```

Vale analisar com as 3 métricas em conjunto!

Parte 2:

Vamos plotar um gráfico dos valores reais versus os valores preditos pelo modelo.

Para isso, vamos importar a biblioteca matplotlib:

```
import matplotlib.pyplot as plt
```

Agora sim, vamos plotar o gráfico:

Na aula, usamos a figura com um tamanho específico (10 polegadas de largura por 6 polegadas de altura). Posteriormente diminuímos para 8,4.

Plotamos um gráfico de dispersão dos valores reais contra os valores previstos e definimos a transparência dos pontos por 0.5.

Definimos os rótulos e o título do gráfico.

Definimos os pontos de início e fim da linha nos eixos X e Y.

E definimos a cor da linha como red (vermelho) e a espessura da linha como 2.

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Valores Real')
plt.ylabel('Valores Previsto')
plt.title('Dispersão dos dados')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linewidth=2)
plt.show()
```

Vamos ver quais atributos tiveram mais peso positivo ou negativo para o resultado do modelo.

Vamos pegar os nomes das colunas de atributos e atribuir a uma variável

```
nomes_atributos =  X_train.columns
```

Vamos criar um dataframe com o pandas, adicionamos os coeficientes do modelo, indicamos qual o nome dessa coluna que acabamos de criar e qual vai ser o nome das linhas, ou seja, dos index. E vamos armazenar o resultado em uma variável.

```
coefs = pd.DataFrame( model.coef_, columns=["coeficientes"], index=nomes_atributos)
```

Vamos ordenar essa coluna de coeficientes pra gente conseguir ver melhor o que ta influenciando mais ou menos. Fazemos isso usando a função de sort_values:

```
coefs = coefs.sort_values(by ='Coefficients', ascending=False)
```

Vamos plotar um gráfico de barras horizontal para visualizar melhor e adicionamos uma linha na vertical no ponto 0 para termos como referência

```
coefs.plot.barh(figsize=(8,6))
plt.axvline(x=0, color'red')
```

Com o gráfico, vamos analisar e refletir sobre os resultados?

Módulo 08 - Para ir além 0/9

COMPLETE E CONTINUE SEU CURSO! →