

ANA RAQUEL

POSTECH

IA PARA DEVS

MACHINE LEARNING

AULA 04

SUMÁRIO

O QUE VEM POR AÍ?	3
HANDS ON	4
SAIBA MAIS	5
O QUE VOCÊ VIU NESTA AULA?	12
REFERÊNCIAS	13

EMSE

O QUE VEM POR AÍ?

Quando trabalhamos com diversos tipos de dados, podemos nos deparar com tipos de **escalas de dados diferentes**. Será que a diferença entre essas escalas pode afetar a performance de um algoritmo de aprendizado de máquina? Bem, nesta aula, você vai aprender sobre a **técnica de “feature scaling”**, e como podemos utilizá-la a favor do aprendizado de máquina. Vamos lá?!

EXEMPLO

HANDS ON

Utilizando o Sci-Learn, você vai aprender na prática a como aplicar as técnicas de normalização e padronização em bases de dados, e qual é o impacto do escalonamento no aprendizado de máquina.

EMEND

SAIBA MAIS

Afinal, por que as escalas dos dados são importantes?

Para definirmos o que seriam as escalas dos dados, podemos dizer que as escalas se referem à **amplitude ou intervalo dos valores de um conjunto numérico de dados**. Imagine que você vai criar um algoritmo preditivo para previsão de crédito com o objetivo de avaliar o risco de inadimplência de um cliente, onde temos o seguinte cenário:

idade	salario	historico_credito	score	numero_emprestimos	alvo
25	48769	0	419	1	0
38	109431	0	500	4	0
34	128730	0	358	4	0
38	60998	0	718	4	0
61	98757	0	685	2	0

Figura 1 - Exemplo de previsão de crédito
Fonte: elaborado pela autora (2024)

Observe as colunas dessa base de dados. Aqui podemos analisar diferentes escalas em uma mesma base. Para você entender a importância das escalas, observe a **variável salário** e a **variável idade**. Se analisarmos os mínimos e máximos dessas variáveis, fica bem claro que essas variáveis possuem amplitudes diferentes, pois quando falamos de dinheiro, as escalas podem ser muito maiores do que as de idade.

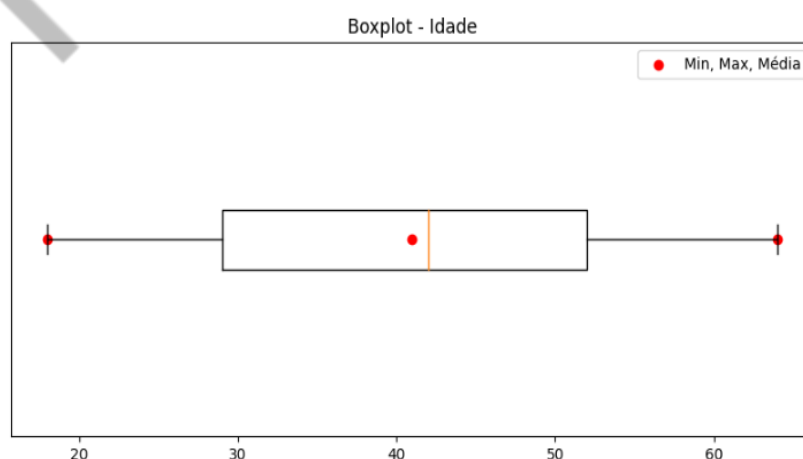


Figura 2 - Boxplot idade

Fonte: elaborado pela autora (2024)

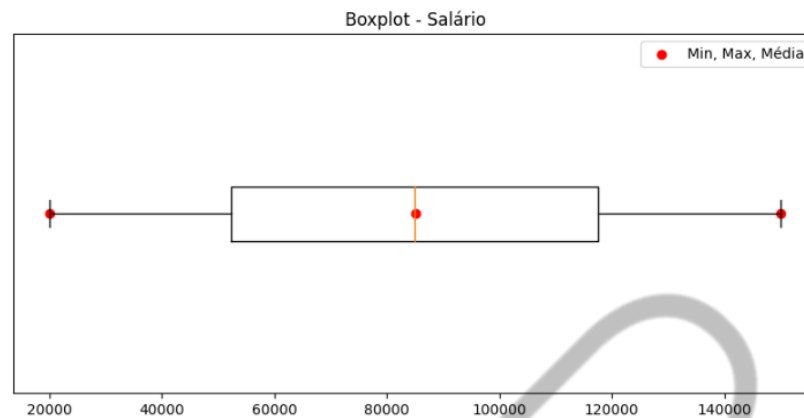


Figura 3 - Boxplot salário
Fonte: elaborado pela autora (2024)

Comparar a importância dessas duas features em questão de amplitude parece injusto quando analisamos as escalas delas. A técnica "**feature scaling**" é o processo de **normalizar as escalas das features**, colocando-as em uma **escala comum**. Existem duas abordagens comuns para fazer isso: a normalização (também conhecida como **min-max scaling**) e a padronização (também conhecida como **z-score** normalization). No aprendizado de máquina, muitos algoritmos podem ser sensíveis às escalas dos dados, podendo se confundir e achar que as escalas maiores, por exemplo, são mais relevantes do que as variáveis de escalas menores.

Esse problema pode afetar principalmente algoritmos de redes neurais, onde a normalização dessas variáveis é obrigatória para a representação da dimensão ficar em uma mesma escala e contribuir com a convergência mais rápida dos dados. Essa técnica pode ajudar não somente em Deep Learning, mas também com algoritmos que lidam com distâncias, como o k-means, KNN, PCA, SVM e regressão logística por exemplo.

Neste exemplo de previsão de crédito, se aplicarmos a padronização com z-score, teríamos o seguinte resultado:

```
X_train_scaled
array([[ -1.07390401,  0.88521066,  0.95836592, -0.30241262,  1.43257423],
       [ -0.0506162 , -0.76703153, -1.04344279,  1.61565123,  1.43257423],
       [ -0.5622601 ,  0.76428799, -1.04344279,  1.64740725, -0.68844869],
       ...,
       [  0.16865976,  0.03134694,  0.95836592, -0.43578792, -1.39545633],
       [  0.89957962,  0.3834538 , -1.04344279, -1.26779574,  0.72556659],
       [  0.68030366, -1.68470462, -1.04344279,  1.07579882,  0.72556659]])
```

Figura 4 - Aplicação da padronização nos dados
Fonte: elaborado pela autora (2024)

Tipos de escalonamento nos dados

Como dito antes, dentro do mundo de data science, existem duas principais maneiras de realizar esse escalonamento dos dados: a normalização e a padronização. Mas, como funciona na prática essa transformação dos dados?

O escalonamento min-max (normalização) é muito simples. Basicamente, os valores são deslocados e redimensionados para que acabem variando de 0 a 1. Observe a fórmula a seguir: esse cálculo subtrai o valor mínimo e divide pelo máximo, menos o mínimo:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Figura 5 - Fórmula Normalização
Fonte: O'Reilly home [s.d]

Neste exemplo da fórmula, se substituirmos o valor de X pelo dado da variável, temos a transformação realizada. Na biblioteca do Scikit-Learn temos o [MinMaxScaler](#) para realizar essa tarefa. Ele possui um hiperparâmetro **feature_range** que permite alterar o intervalo caso você não queira o padrão de 0 e 1. Normalmente utilizamos o padrão da classe que é 0 e 1.

Já a padronização funciona diferente. Ela consiste em subtrair o valor médio (assim os valores padronizados sempre têm média zero), e em seguida ela divide pela variância, de modo que a distribuição tenha uma variância unitária.

$$z = \frac{x - \mu}{\sigma}$$

μ = Mean
 σ = Standard Deviation

Figura 6 - Fórmula Padronização
Fonte: Courtney Taylor (2020)

A padronização não vincula valores específicos nos mínimos e máximos, o que pode ser um ponto de atenção em alguns algoritmos (por exemplo, redes neurais). No entanto, a padronização é muito menos afetada por outliers. Na biblioteca do Scikit-Learn temos a padronização em [StandardScaler](#).

Aplicar a padronização no Python é muito simples com a biblioteca do Scikit-Learn. Imagine que vamos criar uma base de dados escalonada padronizada utilizando o **z-score**. Para este exemplo, vamos utilizar uma base de dados fictícia de previsão de crédito:

```
# Instalando bibliotecas necessárias
! pip install pandas
! pip install numpy
! pip install scikit-learn

# Importando as bibliotecas necessárias
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Criando um conjunto de dados fictício de previsão de crédito
np.random.seed(42)

# Gerando dados aleatórios
n_samples = 1000
idade = np.random.randint(18, 65, size=n_samples)
salario = np.random.randint(20000, 150000, size=n_samples)
historico_credito = np.random.choice(['bom', 'ruim'], size=n_samples)
score = np.random.randint(300, 850, size=n_samples)
numero_emprestimos = np.random.randint(0, 5, size=n_samples)

# Criando a variável alvo (1 para aprovado, 0 para não aprovado)
```



```
alvo = np.where((idade < 30) & (salario > 70000) & (historico_credito
== 'bom'), 1, 0)

# Criando o DataFrame
data = pd.DataFrame({
    'idade': idade,
    'salario': salario,
    'historico_credito': historico_credito,
    'score': score,
    'numero_emprestimos': numero_emprestimos,
    'alvo': alvo
})

# Usando LabelEncoder para transformar variáveis categóricas em
numéricas
label_encoder = LabelEncoder()
data['historico_credito'] =
label_encoder.fit_transform(data['historico_credito'])

# Embaralhando o conjunto de dados
data_shuffled = data.sample(frac=1, random_state=42)

# Salvando o conjunto de dados embaralhado em um arquivo CSV
data_shuffled.to_csv('dataset_credito_embaralhado_com_features.csv',
index=False)

# Carregando o conjunto de dados fictício de previsão de crédito
data = data_shuffled

# Pré-processamento dos dados
# Vamos supor que o conjunto de dados possui as seguintes colunas:
'idade', 'salario', 'historico_credito', 'alvo' (0 para não aprovado, 1
para aprovado)

# Separando features (X) e target (y)
X = data.drop('alvo', axis=1)
y = data['alvo']

# Dividindo o conjunto de dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
from sklearn.preprocessing import StandardScaler

# Inicializar o scaler usando apenas o conjunto de treino

scaler = StandardScaler()

scaler.fit(X_train)

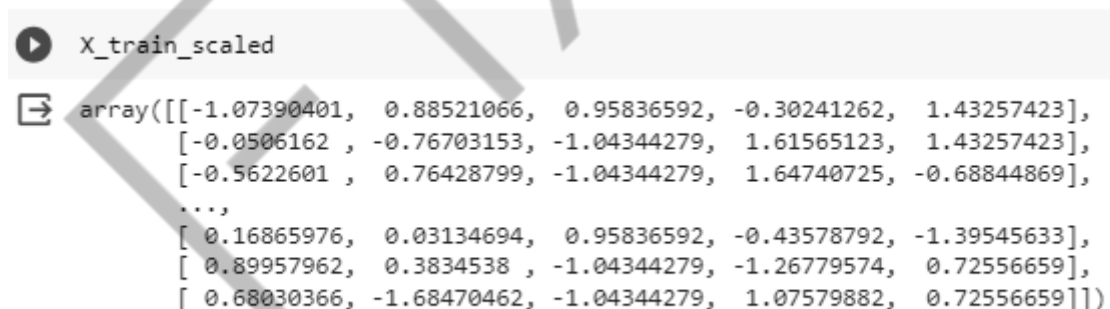
# Aplicar o Z-score nas features de treino

X_train_scaled = scaler.transform(X_train)

# Aplicar o Z-score nas features de teste usando as estatísticas do
conjunto de treino

X_test_scaled = scaler.transform(X_test)
```

Depois de executar todo o código, você pode visualizar a base de dados após escalonamento com o z-score:



```
X_train_scaled
array([[ -1.07390401,  0.88521066,  0.95836592, -0.30241262,  1.43257423],
       [-0.0506162 , -0.76703153, -1.04344279,  1.61565123,  1.43257423],
       [-0.5622601 ,  0.76428799, -1.04344279,  1.64740725, -0.68844869],
       ...,
       [ 0.16865976,  0.03134694,  0.95836592, -0.43578792, -1.39545633],
       [ 0.89957962,  0.3834538 , -1.04344279, -1.26779574,  0.72556659],
       [ 0.68030366, -1.68470462, -1.04344279,  1.07579882,  0.72556659]])
```

Figura 7 - Base de dados após escalonamento
Fonte: elaborado pela autora (2024)

Lembrando que aqui, estamos realizando essa transformação na base de treinamento do algoritmo. Depois de transformada, ela deve ser aplicada para a base de teste também. Não se preocupe, pois, essas questões de base de treino e teste você aprenderá com mais detalhes na aula de Machine Learning Avançado. Mas você

deve estar se perguntando: “por que é realizado o escalonamento (fit) na base treino e não na base de teste?”. Bem, realizamos a transformação do escalonamento na base de treino para evitar que a base de teste fique exatamente igual às estatísticas da base de treino, o que evita “vazamento” desses dados. A base de teste em geral deve representar uma base de dados nunca vista antes pelo algoritmo, justamente para testar se o algoritmo consegue generalizar os dados.

EMAP

O QUE VOCÊ VIU NESTA AULA?

Nessa aula você aprendeu quais são os tipos de escalonamento mais utilizados na área de ciência de dados e qual é a importância de deixar os dados em uma mesma escala para os algoritmos de aprendizado de máquina.

EMAP

REFERÊNCIAS

GÉRON, Aurélien. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. 2nd Edition. O'Reilly Media, Inc. 2019.

O'REILLY HOME. **Min-max normalization**. [s.d]. Disponível em: <<https://www.oreilly.com/library/view/hands-on-machine-learning/9781788393485/fd5b8a44-e9d3-4c19-bebb-c2fa5a5ebfee.xhtml>> Acesso em: 15 fev. 2024.

SCIKILT-LEARN. **sklearn.preprocessing.MinMaxScaler**. [s.d]. Disponível em: <<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>> Acesso em: 15 fev. 2024.

SCIKILT-LEARN. **sklearn.preprocessing.StandardScaler**. [s.d]. Disponível em: <<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>> Acesso em: 15 fev. 2024.

TAYLOR, C. **Z-Score Formula**. 2020. Disponível em: <<https://www.thoughtco.com/z-score-formula-3126281>> Acesso em: 15 fev. 2024.

PALAVRAS-CHAVE

MinMaxScaler. StandardScaler. Escalonamento de Dados.

EMAP



PDS TECH