A PENEIRA DE DADOS

Como o Python Garante que Só os Melhores Cheguem à Quadra

Tagor Murilo

Introdução

Preparando-se para o Jogo da Limpeza de Dados

Bem-vindo ao playbook da limpeza de dados! Assim como um técnico de basquete precisa das ferramentas certas para analisar o desempenho da equipe, nós precisamos de bibliotecas Python poderosas para transformar dados brutos e desorganizados em informações confiáveis e prontas para análise.

A principal jogadora em todos os tópicos de limpeza de dados que você verá é a biblioteca Pandas.

O Que é o Pandas?

Pandas é a biblioteca de código aberto mais popular em Python para manipulação e análise de dados. Ela fornece estruturas de dados de alto desempenho e fáceis de usar, sendo as duas principais:

- Series: Uma coluna de dados (como uma lista de jogadores).
- DataFrame: Uma tabela de dados completa (como a planilha de estatísticas de um time), sendo a estrutura que usamos em todos os exemplos.

O Pandas é essencial porque permite que você carregue, visualize, filtre, agrupe e, o mais importante, limpe seus dados de forma intuitiva e eficiente.

Todas as operações como preencher valores ausentes (.fillna()) ou remover duplicatas (.drop_duplicates()) são funções integradas do Pandas, fazendo dele o nosso canivete suíço na limpeza de dados.

Como Usamos o Pandas?

Para usar o Pandas, você geralmente o importa no início do seu código com um apelido padrão:

```
import pandas as pd
```

Isso nos permite acessar todas as suas funcionalidades usando o prefixo pd.

Ao dominar o Pandas, você ganha o controle total sobre a qualidade dos seus dados, garantindo que apenas as informações mais precisas cheguem à análise final, ou, no nosso trocadilho, que "só os melhores cheguem à quadra"!

01

A Coleta de Lixo

A Coleta de Lixo

Lidando com Valores Ausentes

Imagine seus dados como um time de basquete: se um jogador falta, a estatística fica incompleta! Valores ausentes (NaN) são como esses jogadores perdidos, usamos Python e a biblioteca Pandas para decidir se vamos preencher este este espaço ou remover a linha.

A técnica mais simples é preencher os valores ausentes com zero ou a média.

```
# Se a coluna 'Assistencias' tem valores ausentes, preenchemos com 0.
# Isso é comum se assumirmos que a ausência de registro significa 0 assistências.
dados['Assistencias'] = dados['Assistencias'].fillna(0)
```

02

Ajustando o Placar

Ajustando o Placar

Tratando Tipos de Dados Incorretos

Às vezes, as estatísticas de um jogo estão no formato errado; por exemplo os pontos podem estar como texto em vez de números, Python nos ajuda a garantir que cada coluna de dados tenha o tipo correto (número, texto, data), essencial para que as operações matemáticas funcionem.

É como garantir que o placar esteja sendo contado corretamente.

```
# Convertendo a coluna 'Salario_Anual' para o tipo numérico (float),
# ignorando erros onde a conversão não for possível (que serão tratados como NaN).
dados['Salario_Anual'] = pd.to_numeric(dados['Salario_Anual'], errors='coerce')
```

03

O Fim dos Clones

O Fim dos Clones

Removendo Duplicatas

Um erro comum é ter o mesmo jogador listado duas vezes nas estatísticas, deixando os números irreais ou incoerentes distorcendo a análise.

A remoção de duplicatas garante que cada observação ou "jogador" seja única, mantendo a integridade do seu conjunto de dados, com poucas linhas de código o processo torna-se rápido e simples.

```
# Identificando e removendo duplicatas
# Com base nas colunas 'Nome' e 'Data_Nascimento'.
# Se ambos forem iguais, a linha é uma duplicata e será removida
# (mantendo a primeira ocorrência).

dados.drop_duplicates(
subset=['Nome', 'Data_Nascimento'], keep='first', inplace=True
)
```

04

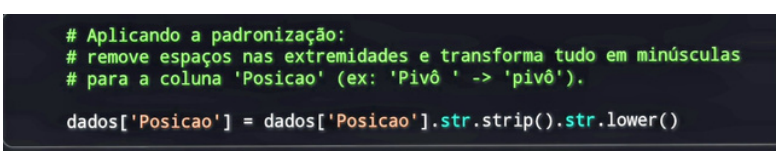
Medindo a Precisão do Arremesso

Medindo a Precisão do Arremesso

Padronizando Textos

Em uma lista de jogadores, "LeBron James" e "L. James" se referem à mesma pessoa, mas o computador os vê como diferentes.

Padronizar textos significa forçar tudo para minúsculas ou maiúsculas e remover espaços extras, sendo crucial para que os nomes e categorias sejam contados corretamente.



```
# Aplicando a padronização:
# remove espaços nas extremidades e transforma tudo em minúsculas
# para a coluna 'Posicao' (ex: 'Pivô ' -> 'pivô').

dados['Posicao'] = dados['Posicao'].str.strip().str.lower()
```

05

Bloqueando o Erro

Bloqueando o Erro

Identificando Outliers

Outliers são como um placar absurdamente alto ou baixo que não faz sentido, talvez um erro de digitação.

Eles são valores que se distanciam muito da maioria dos outros, ao identificarmos decidimos se devemos removê-los ou corrigi-los, garantindo que os resultados da sua análise sejam representações do "jogo" real.

```
# Calculando os limites para 'Idade' (Quartil 25 - 1.5*IQR e Quartil 75 + 1.5*IQR)
# e filtrando para manter apenas os dados dentro desses limites aceitáveis.
Q1 = dados['Idade'].quantile(0.25)
Q3 = dados['Idade'].quantile(0.75)
IQR = Q3 - Q1
dados = dados[
(dados['Idade'] >= Q1 - 1.5 * IQR) & (dados['Idade'] <= Q3 + 1.5 * IQR)
]
```

Encerramento

O Apito Final e a Vitória da Análise

Chegamos ao final deste playbook de limpeza de dados!

Espero que as analogias com o basquete tenham transformado a tarefa de limpar dados, que muitas vezes parece complexa, em um processo claro e estratégico.

Aqui você aprendeu a usar o Pandas, nosso MVP (Jogador Mais Valioso), para executar a "Coleta de Lixo", "Ajustar o Placar", acabar com os "Clones", e "Bloquear os Erros".

Com estas habilidades em mãos, seus dados estão agora higienizados, confiáveis e prontos para alimentar modelos de análise que trarão resultados realmente significativos.

Gostaria de expressar minha gratidão a você, leitor(a) por dedicar seu tempo e esforço para dominar esta fase crucial do Data Science.

Espero te ver em uma próxima partida, até mais!