Sistemas Inteligentes – Aula 6

Hoje iremos explorar os conceitos fundamentais de Big Data na prática. Vamos abordar os 5Vs do Big Data (volume, variedade, velocidade, veracidade e valor), aplicando técnicas básicas de análise de dados.

1- Analisando o volume

Vamos carregar um dataset público diretamente do Kaggle (https://www.kaggle.com/). Você precisará de uma conta no Kaggle e de uma chave API (arquivo kaggle.json) para autenticação.

Passos para configurar a chave API:

1. Acesse o Kaggle e faça login.
2. Vá na sua conta, no menu superior direito e clique em Settings.
3. Role até a seção "API" e clique em "Create New Token".
4. Um arquivo kaggle.json será baixado.
5. Adicione o código abaixo em um novo notebook do Google Colab e execute-o
6. Quando o botão Escolher Arquivo surgir, escolha o arquivo .json baixado

```
# Fazer upload da chave API do Kaggle (arquivo kaggle.json)
from google.colab import files
files.upload()

# Mover o arquivo para o local correto
!mkdir -p ~/.kaggle
!mv kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json

# Baixar um dataset do Kaggle
!kaggle datasets download -d zynicide/wine-reviews
!unzip wine-reviews.zip

import pandas as pd
file_path = '/content/winemag-data_first150k.csv'
df = pd.read_csv(file_path)

df.head()
```

Aqui podemos ver as primeiras linhas dos dados presentes no conjunto de dados sendo exibidos em uma tabela.

2- Analisando a Variedade

Os dados podem ser categóricos, numéricos, textuais, entre outros. Nesta seção, vamos explorar a variedade no dataset.

Copie e execute o código a seguir:

```
# Identificar os tipos de dados no dataset
df.info()
coluna_exemplo = 'country'
valores_unicos = df[coluna_exemplo].unique()

# Exibir os valores únicos
print(f"Valores únicos na coluna '{coluna_exemplo}':")
print(valores_unicos)
```

Ao executar esse código, somos apresentados com uma contagem dos dados no conjunto carregado: Primeiro, temos o número da coluna, seguido por seu nome. Depois, podemos ver quantas entradas não-nulas estão presentes para aquela coluna em todo o conjunto e, por fim, vemos qual o tipo de dado é salvo naquela coluna.

Abaixo desses dados, temos uma contagem de quantas colunas usam cada tipo de dado, quanta memória é usada e os possíveis valores presentes em uma coluna que, nesse caso, foi a coluna “country”.

3- Avaliando a Veracidade

Nesta etapa, verificaremos se os dados são confiáveis, identificando valores ausentes e outliers.

Execute o código a seguir:

```
# Verificar valores ausentes em cada coluna
print("Valores ausentes por coluna:")
print(df.isnull().sum())

# Trabalhando com a coluna 'price'
coluna_numerica = 'price'

# Filtrar valores maiores que 100
df_limpado = df.dropna(subset=['price'])
outliers = df_limpado[df_limpado['price'] > 100]
outliers = df[df['points'] > 100]

# Remover linhas com valores ausentes
df_limpo = df.dropna()
print(f"Tamanho do dataset após remoção de valores ausentes: {df_limpo.shape}")
```

Nesse código, estamos verificando e limpando os dados do DataFrame para identificar problemas como valores ausentes e outliers. Aqui está o que cada parte faz:

1. Identificar valores ausentes
    a. Usamos a função `isnull()` para localizar células com valores ausentes (NaN) no DataFrame.
    b. A função `sum()` soma quantos valores ausentes existem em cada coluna. Isso nos dá uma visão geral das "falhas" nos dados.
2. Remover valores ausentes em uma coluna específica

    a. O comando `dropna(subset=['price'])` remove todas as linhas onde a coluna `price` tem valores ausentes (NaN). Isso garante que só trabalharemos com linhas que possuem informações completas nessa coluna.
3. Identificar valores extremos (outliers)
    a. Após limpar os valores ausentes em `price`, o comando `df_limpado[df_limpado['price'] > 100]` filtra as linhas onde `price` é maior que 100. Esses valores são considerados "outliers" (valores extremos que podem distorcer a análise).
4. Remover todas as linhas com valores ausentes
    a. O comando `dropna()` elimina qualquer linha que tenha pelo menos um valor ausente, independentemente da coluna. Isso deixa o dataset "completo", mas pode reduzir bastante o tamanho do DataFrame.
5. Ver o tamanho do DataFrame limpo
    a. A função `shape` mostra quantas linhas e colunas ainda existem após a limpeza. Assim, sabemos quantos dados "bons" ainda temos para trabalhar.

4- Explorando o Valor

Transformar dados em informações valiosas é uma das principais características do Big Data. Vamos criar algumas visualizações simples:

```
import matplotlib.pyplot as plt

# Gráfico de barras usando a coluna 'country'
df['country'].value_counts().plot(kind='bar', figsize=(10, 6))
plt.title('Distribuição de Países')
plt.xlabel('País')
plt.ylabel('Contagem')
plt.show()

# Gráfico de linha usando a coluna 'points'
df['points'].plot(kind='line', figsize=(10, 6))
plt.title('Tendência de Pontuação')
plt.xlabel('Índice')
plt.ylabel('Pontuação')
plt.show()
```

Esse trecho de código nos ajuda a visualizar os dados presentes no conjunto usado, criando gráficos a partir da bilbioteca Matlab, um potente software de análise matemática. O primeiro gráfico nos mostra a quantidade de vinhos cadastrados por país e o segundo mostra a pontuação que cada vinho recebeu, de acordo com seu índice (posição na lista).

5- Avaliando a Velocidade

Vamos medir o tempo de execução de operações em datasets grandes:

```
import time

# Usando a coluna 'points' como exemplo
```

```
coluna_numerica = 'points'

# Medir o tempo de execução de uma operação
start_time = time.time()
df['nova_coluna'] = df[coluna_numerica] * 2  # Multiplicando os valores por 2
execution_time = time.time() - start_time

print(f"Tempo de execução: {execution_time:.4f} segundos")
```

Esse trecho de código avalia a velocidade da operação, se baseando na coluna “points” e fazendo o cálculo do tempo gasto entre o início e o fim do processo.

Agora é com você:

1. Volume: Modifique o código para carregar um dataset diferente do Kaggle. Escolha um dataset relacionado ao clima ou à saúde pública, baixe e exiba as cinco primeiras linhas (use head()).
2. Variedade: Identifique os tipos de dados de outra coluna do dataset e exiba os valores únicos. Em seguida, crie um código para contar quantas vezes cada valor aparece nessa coluna.
3. Veracidade: Verifique valores ausentes em uma coluna à sua escolha. Depois, substitua esses valores por um valor padrão (ex.: "Desconhecido" para texto ou 0 para números) e exiba a tabela corrigida.
4. Valor: Altere a criação dos gráficos, usando o gráfico de “pizza” no lugar do primeiro e o scatter plot no lugar do segundo.
5. Velocidade: Adicione uma operação mais complexa para medir o tempo de execução, como calcular a média de uma coluna para cada país. Registre o tempo e exiba os resultados.
   Dica: Combine métodos como groupby() e mean() para calcular os valores.