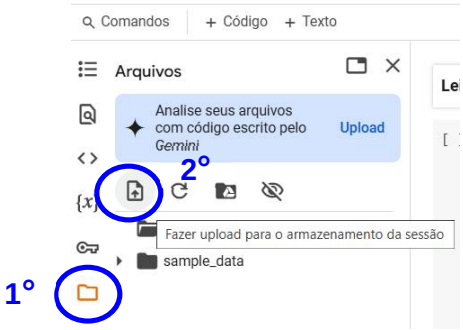
PÓS-GRADUAÇÃO LATO SENSU ESPECIALIZAÇÃO EM MINERAÇÃO DE DADOS EDUCACIONAIS

Disciplina: Visualização de Dados
Professor: Dr. Richard Godinez Tello
E-mail: richard@ifes.edu.br

Atividade N° 3: Som. Princípios de percepção visual e cognição. Resultados de classificadores.

Procedimento para fazer upload dos arquivos de teste:



1. Visualização de áudio no domínio do tempo, frequência e espectograma: (Transformação de áudio em grave/agudo)

```
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt
import soundfile as sf

# Carregar arquivo de áudio original
audio_path = "audio_teste.wav"  # Substitua pelo seu arquivo
y, sr = librosa.load(audio_path, sr=None)

# Modificar pitch (graves e agudos)
y_grave = librosa.effects.pitch_shift(y, sr=sr, n_steps=-5)  # -5 semitons → mais grave
y_agudo = librosa.effects.pitch_shift(y, sr=sr, n_steps=+5)  # +5 semitons → mais agudo

# Salvar os áudios modificados
sf.write("audio_grave.wav", y_grave, sr)
sf.write("audio_agudo.wav", y_agudo, sr)

# Função para plotar gráficos
def plot_audio(y, sr, titulo):
    plt.figure(figsize=(10, 3))

    # Forma de onda
    plt.subplot(1, 3, 1)
    librosa.display.waveshow(y, sr=sr)
    plt.title(f"Onda - {titulo}")
    plt.xlabel("Tempo (s)")
    plt.ylabel("Amplitude")

    # Espectro de frequência (FFT) - Magnitude (Valor Absoluto)
    fft = np.fft.fft(y)
    freqs = np.fft.fftfreq(len(fft), 1/sr)
    magnitude = np.abs(fft[:len(fft)//2])  # Pegamos apenas a parte positiva e calculamos o valor
absoluto
    plt.subplot(1, 3, 2)
    plt.plot(freqs[:len(freqs)//2], magnitude)  # Magnitude (Valor Absoluto)
    plt.title(f"Frequências - {titulo}")
    plt.xlabel("Frequência (Hz)")
```

```
    plt.ylabel("Magnitude (Valor Absoluto)")

    # Espectrograma
    plt.subplot(1, 3, 3)
    D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
    librosa.display.specshow(D, sr=sr, x_axis="time", y_axis="log")
    plt.colorbar(format="%+2.0f dB")
    plt.title(f"Espectrograma - {titulo}")

    plt.tight_layout()
    plt.show()

# Plotando os três áudios (Original, Grave e Agudo)
plot_audio(y, sr, "Original")
plot_audio(y_grave, sr, "Grave (-5 Semitons)")
plot_audio(y_agudo, sr, "Agudo (+5 Semitons)")
```

2. Reconhecimento de texto e análise de intensidade de áudio:

```
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt
import speech_recognition as sr

# Carregar o arquivo de áudio
audio_path = "audio_teste.wav"  # Substitua pelo caminho correto do seu arquivo
y, sample_rate = librosa.load(audio_path, sr=None)

# Converter fala em texto (Speech-to-Text)
recognizer = sr.Recognizer()
try:
    with sr.AudioFile(audio_path) as source:
        audio_data = recognizer.record(source)
        texto = recognizer.recognize_google(audio_data, language='pt-BR')
        print("Texto extraído do áudio:", texto)
except sr.UnknownValueError:
    print("Não foi possível entender o áudio.")
except sr.RequestError:
    print("Erro ao acessar o serviço de reconhecimento de fala.")
except Exception as e:
    print(f"Erro inesperado: {e}")

# Análise de intensidade do som (RMS - Root Mean Square)
rms = librosa.feature.rms(y=y)[0]
tempo = librosa.times_like(rms, sr=sample_rate)  # Ajustando a variável 'sr' para 'sample_rate'

# Plotando a intensidade do som ao longo do tempo
plt.figure(figsize=(10, 4))
plt.plot(tempo, rms, label="Intensidade do Som (RMS)", color='b')
plt.xlabel("Tempo (s)")
plt.ylabel("Energia do Som")
plt.title("Análise de Intensidade do Áudio")
plt.legend()
plt.grid(True)
plt.show()
```

3. Gráfico com paleta de cores

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Criando dados fictícios
dados = pd.DataFrame({'Disciplina': ['Matemática', 'Física', 'História', 'Geografia', 'Português'],
                      'Nota Média': [7.5, 6.8, 8.2, 7.9, 9.0],
                      'Grupo': ['Exatas', 'Exatas', 'Humanas', 'Humanas', 'Linguagem']})

# Criando um gráfico de barras com cores agrupadas por área
plt.figure(figsize=(8,5))
```

```
ax = sns.barplot(x='Disciplina', y='Nota Média', data=dados, hue='Grupo', palette='coolwarm')

plt.title('Notas Médias por Disciplina (Agrupamento por Similaridade)')
plt.xlabel('Disciplina')
plt.ylabel('Nota Média')

# Posicionando a legenda fora do gráfico, no canto superior direito
plt.legend(title="Área do Conhecimento", bbox_to_anchor=(1.05, 1), loc='upper left',
borderaxespad=0.)

plt.show()
```

4. Exibindo uma matriz de confusão

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Definição das classes
classes = ["Muito Ruim", "Ruim", "Médio", "Bom", "Excelente"]

# Simulando valores reais e preditos com maior precisão (mais acertos na diagonal)
np.random.seed(42)
y_true = np.random.choice([0, 1, 2, 3, 4], size=200, p=[0.1, 0.2, 0.4, 0.2, 0.1])
y_pred = y_true.copy()  # Primeiro, assumimos que todas as previsões estão corretas

# Introduzindo alguns erros (desvio para classes vizinhas)
error_indices = np.random.choice(len(y_pred), size=40, replace=False)  # 40 exemplos errados
# Pequeno desvio %5 serve para garantir que as previsões permaneçam dentro dos índices válidos (0,
1, 2, 3, 4)
y_pred[error_indices] = (y_pred[error_indices] + np.random.choice([-1, 1], size=40)) % 5

# Criando a matriz de confusão
cm = confusion_matrix(y_true, y_pred)

# Normalizando por linha para obter percentuais
cm_percent = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis] * 100  # De 0 a 100%

# Plotando a matriz normalizada
plt.figure(figsize=(6,5))
sns.heatmap(cm_percent, annot=True, fmt=".2f", cmap="Blues", xticklabels=classes,
yticklabels=classes)
plt.xlabel("Previsto")
plt.ylabel("Real")
plt.title("Matriz de Confusão Normalizada (%) - Modelo com Boa Precisão")
plt.show()
```

5. Exemplo do cálculo de acurácia, Precisão, Revocação, F1-Score:

```
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Gerando dados aleatórios para as classes reais e preditas
np.random.seed(42)
y_true = np.random.choice([0, 1, 2, 3, 4], size=100)  # Classes reais
y_pred = np.random.choice([0, 1, 2, 3, 4], size=100)  # Classes preditas

# Calculando as métricas
acuracia = accuracy_score(y_true, y_pred)
precisao = precision_score(y_true, y_pred, average='macro', zero_division=0)
revocacao = recall_score(y_true, y_pred, average='macro', zero_division=0)
f1 = f1_score(y_true, y_pred, average='macro', zero_division=0)

# Exibindo os resultados
print(f"Acurácia: {acuracia * 100:.2f}%")
print(f"Precisão: {precisao * 100:.2f}%")
print(f"Revocação: {revocacao * 100:.2f}%")
print(f"F1-Score: {f1 * 100:.2f}%")
```