# Atividade: Implementação e Análise de Árvore AVL com Conjunto Iris

Material Didático – Estrutura de Dados II

Outubro 2025

## 3.1 Descrição da Atividade

Esta atividade propõe aos alunos do curso de Ciência da Computação, na disciplina de Estrutura de Dados II, a implementação de um programa em Python que utiliza **árvores AVL** para organizar e consultar dados do conjunto **Iris**. A atividade integra conceitos de estruturas de dados com manipulação de dados usando as bibliotecas `pandas`, `scikit-learn` e `scipy`.

Basicamente, a atividade consiste:

1. **Carregar e pré-processar** os dados do conjunto Iris.
2. **Construir árvores AVL** para cada espécie (setosa, versicolor, virginica).
3. **Realizar análise estatística** com intervalos de confiança.
4. **Implementar uma função de classificação** para novas amostras.
5. **Gerar um relatório** com a estrutura das árvores e resultados.

## 3.2 Objetivos de Aprendizado

Nesta atividade, vocês terão a oportunidade de mergulhar no fascinante mundo das estruturas de dados e da ciência de dados, aplicando conceitos teóricos de forma prática e criativa! O objetivo é que vocês não apenas compreendam as árvores AVL, mas também sintam a empolgação de usá-las para resolver problemas reais. Vamos juntos explorar como essas ferramentas podem transformar dados brutos em soluções poderosas! Aqui estão os principais objetivos que queremos que vocês alcancem:

- **Explorar e dominar árvores AVL**: Entendam como as árvores AVL organizam dados de forma eficiente e apliquem esse conhecimento para construir estruturas robustas. Vocês verão na prática como o balanceamento automático das AVLs faz toda a diferença em desempenho!
- **Manipular dados reais com confiança**: Usem a biblioteca `pandas` para trabalhar com o conjunto Iris e apliquem análises estatísticas com `scipy`. Essa é a chance de vocês

se sentirem como verdadeiros cientistas de dados, transformando números em insights valiosos!

- **Conectar mundos**: Integre o poder das estruturas de dados com ferramentas modernas de ciência de dados. Vocês descobrirão como combinar teoria e prática para criar soluções inovadoras, algo que é essencial no mercado de tecnologia atual.
- **Avaliar e refletir**: Analisem a eficiência das árvores AVL em buscas e balanceamento, comparando-as com outras abordagens. Essa é uma oportunidade para desenvolverem um olhar crítico e entenderem o impacto de escolhas técnicas em projetos reais.

Estou animados para ver como vocês vão abordar essa atividade! Cada passo que vocês derem aqui é uma chance de aprender, experimentar e crescer como futuros profissionais da Ciência da Computação. Confiem no processo, colaborem com seus colegas e não hesitem em explorar novas ideias!

## 3.3 Código Base

O código a seguir serve como ponto de partida:

**Código:** `Esqueleto do Código em Python`

```python
from avltree import AvlTree
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
from scipy.stats import norm

# Carregar o conjunto Iris
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = iris.target_names[iris.target]

# Funcao para calcular indice composto
def calculate_composite_index(row):
    return np.mean(row[iris.feature_names])

```

```
# Criar arvores AVL para cada especie
avl_setosa = AvlTree()
avl_versicolor = AvlTree()
avl_virginica = AvlTree()

# Dicionario para mapear especies as arvores
species_trees = {
    'setosa': avl_setosa,
    'versicolor': avl_versicolor,
    'virginica': avl_virginica
}

# Inserir dados nas arvores AVL
for index, row in df.iterrows():
    species = row['species']
    composite_index = calculate_composite_index(row)
    species_trees[species].insert(composite_index, index)

# Calcular intervalos de confianca (95%)
def calculate_confidence_interval(data):
    mean = np.mean(data)
    std = np.std(data, ddof=1)
    ci_lower, ci_upper = norm.interval(0.95, loc=mean, scale=std/np.sqrt(len(data)))
    return mean, ci_lower, ci_upper

# Exemplo: Intervalo de confianca para comprimento da petala
for species in df['species'].unique():
    species_data = df[df['species'] == species]['petal length (cm)']
    mean, ci_lower, ci_upper = calculate_confidence_interval(species_data)
    print(f"{species}: Media = {mean:.2f}, Intervalo de Confianca (95%) = [{ci_lower
        :.2f}, {ci_upper:.2f}]")

# Funcao de classificacao
def classify_sample(sample):
    composite_index = calculate_composite_index(sample)
    min_diff = float('inf')
    predicted_species = None
    for species, tree in species_trees.items():
        closest_node = tree.find_closest(composite_index)
```

```
        if closest_node and abs(closest_node.key - composite_index) < min_diff:
            min_diff = abs(closest_node.key - composite_index)
            predicted_species = species
    return predicted_species

# Exemplo de uso
sample = pd.Series([5.1, 3.5, 1.4, 0.2], index=iris.feature_names)
predicted = classify_sample(sample)
print(f"Amostra classificada como: {predicted}")

# Relatorio da estrutura da arvore
for species, tree in species_trees.items():
    print(f"Arvore para {species}: Altura = {tree.height()}, Nos = {tree.size()}")
```

> **Nota**
>
> A função `find_closest` não é padrão na biblioteca `avltree`. Será necessário implementá-la.

## 3.4 Critérios de Avaliação

A atividade valerá um ponto para a turma. Entretanto, os seguintes critérios serão observados pela excelência:

1. **Correção (40%)**: Implementação correta do carregamento de dados, inserção nas árvores AVL e classificação.
2. **Análise Estatística (20%)**: Cálculo correto de médias e intervalos de confiança, com integração na classificação.
3. **Relatório (20%)**: Clareza na apresentação da estrutura das árvores e resultados estatísticos.
4. **Eficiência e Organização do Código (20%)**: Código modular, comentado e eficiente.

## 3.5 Extensões Opcionais: Eleve Seu Projeto ao Próximo Nível!

Prontos para ir além e mostrar do que são capazes? Estas extensões opcionais são desafios empolgantes que vão testar sua criatividade e aprofundar suas habilidades como cientistas da computação! Cada uma dessas ideias é uma oportunidade de explorar novas possibilidades, impressionar a si mesmos e criar algo realmente especial. Vamos lá?

- **Visualização: Dê Vida às Suas Árvores AVL!** Transforme suas árvores AVL em representações visuais incríveis usando ferramentas como `graphviz`. Imagine poder *ver* a estrutura que você construiu, com cada nó e conexão ganhando forma! Essa é sua chance de unir código e arte, criando diagramas que mostram o poder do balanceamento das AVLs de forma clara e impactante.
- **Otimização: Prove a Superioridade das AVLs!** Coloque suas árvores AVL à prova comparando o desempenho de buscas nelas com uma busca linear em arrays. Você vai sentir a adrenalina de medir tempos de execução e descobrir, na prática, por que as AVLs são tão eficientes. Esse desafio é perfeito para quem quer entender o impacto real de uma boa estrutura de dados!

Esses desafios são o momento de vocês brilharem! Não tenham medo de experimentar, errar e aprender. Cada linha de código que vocês escreverem aqui é um passo para se tornarem profissionais ainda mais incríveis. Vamos transformar essas ideias em realidade e surpreender a todos com o que vocês podem fazer!