Documentação do Projeto: Classificação Iris com Naive Bayes

1. Visão Geral e Definição do Problema

O objetivo deste projeto é desenvolver um modelo de aprendizado de máquina supervisionado capaz de classificar flores da espécie *Iris* em três subespécies distintas com base em suas dimensões físicas.

O Dataset Iris

O problema utiliza o clássico conjunto de dados Iris. Ele contém amostras de três espécies de flores:

1. **Iris Setosa**
2. **Iris Versicolor**
3. **Iris Virginica**

Para cada flor, quatro características (features) foram medidas:

- Comprimento da Sépala (Sepal Length)
- Largura da Sépala (Sepal Width)
- Comprimento da Pétala (Petal Length)
- Largura da Pétala (Petal Width)

O desafio do algoritmo é aprender os padrões dessas medidas para prever a espécie de uma flor não vista anteriormente.

2. A Solução: Naive Bayes Gaussiano

O projeto utiliza o classificador **Gaussian Naive Bayes**. Este é um algoritmo probabilístico baseado no Teorema de Bayes.

- **"Naive" (Ingênuo):** O algoritmo assume que as características (ex: largura da pétala e comprimento da sépala) são independentes entre si, o que simplifica o cálculo.
- **"Gaussian" (Gaussiano):** Assume-se que os dados contínuos de cada classe seguem uma distribuição normal (curva em sino).

3. Estrutura do Código

O script `naive_bayes.py` executa o pipeline completo de Machine Learning:

1. **Carregamento de Dados:** Importa o dataset Iris através do `sklearn.datasets`.
2. **Pré-processamento:** Separa os atributos (`X`) dos rótulos (`y`).
3. **Divisão (Split):** Divide os dados em dois grupos para garantir uma avaliação justa:
    - **Treino (70%):** Usado para ensinar o modelo.

    - **Teste (30%):** Usado para verificar a precisão do modelo (`random_state=42` garante reprodutibilidade).
4. **Treinamento:** O método `.fit()` ajusta o modelo Gaussiano aos dados de treino.
5. **Avaliação:** Gera métricas como Acurácia, Relatório de Classificação e Matriz de Confusão.

4. Exemplo de Uso Prático

Entrada (Novos Dados)

O sistema recebe uma lista de listas, onde cada sub-lista contém as 4 medidas: `[Sépala_Comp, Sépala_Larg, Pétala_Comp, Pétala_Larg]`

```
novos_dados = [
   [5.1, 3.5, 1.4, 0.2],  # Amostra 1
   [6.0, 2.9, 4.5, 1.5],  # Amostra 2
   [7.2, 3.6, 6.1, 2.5]   # Amostra 3
]
```

Execução da Previsão

O modelo processa esses dados através do método `.predict()`:

```
previsoes_novos = modelo.predict(novos_dados)
```

Saída Esperada (Output)

Ao executar o programa, o console exibirá a classificação prevista para cada uma das amostras inseridas:

```
Dado [5.1, 3.5, 1.4, 0.2] previsto como: setosa
Dado [6.0, 2.9, 4.5, 1.5] previsto como: versicolor
Dado [7.2, 3.6, 6.1, 2.5] previsto como: virginica
```

5. Avaliação de Desempenho

O script também imprime métricas cruciais para validar a qualidade do modelo:

- **Acurácia:** Porcentagem global de acertos.
- **Matriz de Confusão:** Uma tabela que mostra onde o modelo acertou e onde confundiu uma espécie com outra (ex: confundir *Versicolor* com *Virginica*).

6. Requisitos

Para executar este projeto, é necessário ter instalado:

- Python 3.x
- Scikit-Learn (`pip install scikit-learn`)
- Matplotlib (`pip install matplotlib`)