

# Naive Bayes para Iris



Estou fazendo alguns testes de Machine Learning com o dataset Iris. Gostaria que você me fornecesse a implementação do algoritmo de Naive Bayes com parâmetros adequados para esse dataset. Não quero que use nenhuma biblioteca com esse algoritmo pronto, apenas bibliotecas auxiliares como pandas e numpy



Claro! Aqui está uma implementação do algoritmo de Naive Bayes para o dataset Iris, utilizando apenas `pandas` e `numpy`. O código inclui a leitura do dataset, a separação em conjuntos de treino e teste, e a implementação do classificador Naive Bayes.

## Código

python

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Carregar o dataset Iris
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
data = pd.read_csv(url, header=None, names=columns)

# Separar características e rótulos
X = data.iloc[:, :-1].values # características
y = data.iloc[:, -1].values # rótulos

# Dividir em conjunto de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Função para calcular a média e o desvio padrão
def summarize_dataset(X, y):
    summaries = {}
    for label in np.unique(y):
        X_label = X[y == label]
        summaries[label] = [(np.mean(column), np.std(column)) for column in zip(*X_label)]
    return summaries
```

```
# Função para calcular a probabilidade
def calculate_probability(x, mean, std):
    exponent = np.exp(-(x - mean) ** 2 / (2 * std ** 2))
    return (1 / (np.sqrt(2 * np.pi) * std)) * exponent

# Função para fazer previsões
def predict(summaries, input_vector):
    total_rows = sum([len(summaries[label]) for label in summaries])
    probabilities = {}

    for label, summary in summaries.items():
        probabilities[label] = 1
        for i in range(len(summary)):
            mean, std = summary[i]
            probabilities[label] *= calculate_probability(input_vector[i], mean, std)
        probabilities[label] *= (len(summaries[label]) / total_rows)

    return max(probabilities, key=probabilities.get)

# Resumir o dataset
summaries = summarize_dataset(X_train, y_train)

# Fazer previsões
predictions = [predict(summaries, row) for row in X_test]

# Avaliar o modelo
accuracy = accuracy_score(y_test, predictions)
print(f"Acurácia: {accuracy:.2f}")
```

## Explicação

1. **Carregamento do Dataset:** O código carrega o dataset Iris diretamente da UCI Machine Learning Repository.
2. **Divisão dos Dados:** Utiliza ``train_test_split`` para dividir o dataset em conjuntos de treino (80%) e teste (20%).
3. **Sumarização:** A função ``summarize_dataset`` calcula a média e o desvio padrão de cada característica para cada classe.

4. **Cálculo da Probabilidade:** A função ``calculate_probability`` calcula a probabilidade de um ponto de dados pertencer a uma classe com base na distribuição normal.
5. **Previsão:** A função ``predict`` utiliza as distribuições resumidas para prever a classe de novos dados.
6. **Avaliação:** Por fim, calcula a acurácia das previsões em relação aos rótulos reais do conjunto de teste.

Você pode executar esse código e ajustá-lo conforme necessário para seus testes! Se precisar de mais ajuda, é só avisar.



Algum parametro foi setado?



Na implementação do Naive Bayes que forneci, não há parâmetros específicos que você configure diretamente, já que o algoritmo é baseado em estatísticas descritivas (média e desvio padrão) das características para cada classe. No entanto, há algumas considerações que podem ser vistas como "parâmetros":

1. **Divisão do Conjunto de Dados:** A proporção de 80% para treino e 20% para teste foi escolhida. Você pode ajustar a ``test_size`` no ``train_test_split`` conforme necessário.
2. **Modelo de Distribuição:** O Naive Bayes assume que as características seguem uma distribuição normal (gaussiana). Se as características não se ajustarem bem a essa suposição, pode ser necessário considerar outros tipos de Naive Bayes (como o Multinomial ou Bernoulli) ou mesmo outros algoritmos.

3. **Semente Aleatória:** O parâmetro ``random_state`` na divisão dos dados assegura que a divisão seja reproduzível. Alterá-lo pode gerar diferentes conjuntos de treino e teste.

Se você quiser adicionar parâmetros específicos ou ajustes, como regularização, precisaria implementar isso manualmente, já que a versão básica do Naive Bayes não possui esses parâmetros. Se precisar de algo mais específico, me avise!



Perfeito! E a acurácia retornada esta em qual escala?



A acurácia retornada pelo código está na escala de 0 a 1, onde:

- **0** significa que o modelo não acertou nenhuma previsão.
- **1** significa que o modelo acertou todas as previsões.

Quando você imprime a acurácia com ``print(f"Acurácia: {accuracy:.2f}")``, o formato ``:.2f`` formata o número para duas casas decimais. Assim, a acurácia pode ser interpretada como uma porcentagem ao multiplicar por 100. Por exemplo, uma acurácia de ``0.90`` significa que o modelo acertou 90% das previsões.