Análise da acurácia de um classificador Naive Bayes Gaussiano na identificação de espécies da flor de Íris – 2021/1*

*Nota: Projeto da discplina Aprendizado de Máquina do 1o. Semestre de 2021

Aluno: Carlos Fonseca
Programa de Pós Graduação em Eng. Elétrica
PPGEE/UFAM
Manaus/AM, Brasil
henrique_fonseca@hotmail.com

Aluno: Willian Guerreiro
Programa de Pós Graduação em Eng. Elétrica
PPGEE/UFAM
Manaus/AM, Brasil
wguerreiro31@gmail.com

***Resumo*—Este trabalho tem como objetivo implementar um classificador probabilístico Naive-Bayes Gaussiano para a identificação de flores de Íris entre 3 espécies: Setosa, Virginica e Versicolor, sendo a base de dados "íris_dataset" proveniente do software MATLAB. O conjunto de dados contendo 150 observações com 4 características cada, será armazenado e segmentado em 3 partes iguais. Foi utilizado o método k-fold em 5 pastas para a validação do modelo. Utilizou-se a matriz de confusão obtida a partir da classificação dos elementos do conjunto de teste de cada pasta. Após esse procedimento, as acurácias individuais de cada pasta foram utilizadas para o cálculo da acurácia média do modelo, possibilitando uma visão geral do classificador. O algoritmo responsável por essa implementação foi desenvolvido na linguagem Python com o suporte dos pacotes Numpy, Pandas e Matplotlib.**

***Index Terms*—Naive Bayes, classificador Gaussiano, k-fold, matriz de confusão**

I. Introdução

O problema de classificação de dados em categorias é comum em pesquisas em várias áreas do conhecimento, a exemplo de: filtros anti-SPAM para classificação e segregação de emails indesejados, classificação de etnias em algoritmos de reconhecimento facial, classificação de espécies de plantas com base em um conjunto de características físicas, classificação de tumores malignos ou benignos com base em imagens de ultrassom, classificação do risco de crédito com base no histórico de adimplência ou inadimplência de pagamentos, identificação de minerais rochosos a partir de suas propriedades radioativas, etc. Além da classificação, existe também o interesse em conhecer os fatores, ou variáveis preditoras, mais relevantes na caracterização e determinação de uma categoria ou classe, como é denominada a variável de saída do algoritmo.

Os modelos que consideram probabilidades (ou distribuições de probabilidades) na estimação da classificação recebem o nome de modelos probabilísticos. O algoritmo de classificação Naive Bayes utiliza as probabilidades de cada variável preditora pertencente a cada classe no conjunto de treinamento para prever a classe de novas observações. Há metodologias cujo objetivo também é o de classificar elementos em uma ou outra categoria distinta, mas que não se baseiam em teoria probabilística. Algumas delas são: Árvore de Classificação, Floresta Aleatória (Random Forest), Vizinhos mais próximos (k-Nearest Neighbor ou KNN), Redes Neurais Artificiais (ANN), dentre outras.

O trabalho de Yumei Li [**1**], por exemplo, utilizou classsificador de Naive Bayes Gaussiano e comparou sua performance com a técnica de Análise de Discriminantes Lineares (LDA), ambas aplicadas ao problema de identificação de fácies[1], a partir de 4 (quatro) características: raios gama (GR), porosidade de nêutrons (NPHI), densidade de formação (RHOB),e resistividade profunda (LLD). Com base nesse conjunto de dados, a pesquisa visa estimar cinco classes de fácies : duna de areia (SD), interduna (ID), marinho raso (SM), sabkha (SB) e folha de areia (SS). Neste estudo, o autor conclui que o classificador Naive Bayes apresenta resultados comparáveis à técnica LDA em termos de eficiência e consistência, comprovando sua viabilidade para o problema de identificação de fácies.

Da mesma forma, no presente estudo, propomos a aplicação do classificador Naive Bayes Gaussiano à base de dados *iris_dataset* disponibilizada no MATLAB, com o total de 150 observações. Esta base de dados tem origem no estudo de Fisher sobre as três espécies de flor de Íris (Setosa, Versicolor e Virgínica). Foram utilizadas 50 (cinquenta) amostras de cada espécie e tabuladas as medidas de comprimento e largura das sépalas e pétalas, conforme indicado nas fig.1 e fig. 2.

A partir dessas medidas, pretende-se estudar a acurácia de classificador Naive Bayes Gaussiano quando utilizado na classificação de espécies da flor de íris, a partir das medidas

[1]fácies: conjunto de rochas com determinadas características distintivas, quer paleontológicas (fósseis) quer litológicas, considerando qualquer aspecto composicional, químico ou mineralógico, morfológico, estrutural ou textural (assim como a forma, o tamanho, a disposição dos seus grãos e a sua composição de minerais) que ajudem a conhecer onde e quando se formou a rocha. *Glossário Geológico/SIGEP*

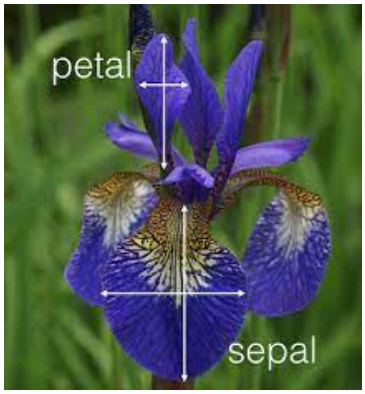

Figura 1. Sépalas e Pétalas da Flor de Íris

Setosa				Versicolor				Virginica			
Sétala		Pétala		Sétala		Pétala		Sétala		Pétala	
(C)ompr. x1	(L)arg. x2	(C)ompr. x3	(L)arg. x4	(C)ompr. x1	(L)arg. x2	(C)ompr. x3	(L)arg. x4	(C)ompr. x1	(L)arg. x2	(C)ompr. x3	(L)arg. x4
5,1	3,5	1,4	0,2	7	3,2	4,7	1,4	6,3	3,3	6	2,5
4,9	3	1,4	0,2	6,4	3,2	4,5	1,5	5,8	2,7	5,1	1,9
4,7	3,2	1,3	0,2	6,9	3,1	4,9	1,5	7,1	3	5,9	2,1
4,6	3,1	1,5	0,2	5,5	2,3	4	1,3	6,3	2,9	5,6	1,8
5	3,6	1,4	0,2	6,5	2,8	4,6	1,5	6,5	3	5,8	2,2
5,4	3,9	1,7	0,4	5,7	2,8	4,5	1,3	7,6	3	6,6	2,1
4,6	3,4	1,4	0,3	6,3	3,3	4,7	1,6	4,9	2,5	4,5	1,7
5	3,4	1,5	0,2	4,9	2,4	3,3	1	7,3	2,9	6,3	1,8
...	...	...	...	...	...	...	...	...	...	...	...

Figura 2. Conjunto de dados Iris

de suas pétalas e sépalas. Esse é um caso de aprendizado supervisionado, uma vez que para a obtenção dos modelos bayesianos que representam cada espécie, a partir da média e desvio padrão de suas características, faz-se necessária a catalogação e agrupamento prévio dos dados por espécie.

Na seção 2 será apresentada uma breve Fundamentação Teórica sobre o classificador Naive Bayes Gaussiano, na seção 3 a Metologia adotada para e implementação em Python, na seção 4 os Resultados encontrados e, por fim, na seção 5 as Conclusões sobre a aplicação do método no problema proposto.

II. Fundamentação Teórica

Um classificador Naive Bayes é um classificador probabilístico baseado na aplicação do Teorema de Bayes, equação 1, com a premissa ingênua (*naive*) de que as características, ou variáveis preditoras, são linearmente independentes. Isso signifca que, o classificador assume que o valor de uma característica do objeto em estudo não possui relação com as demais.

Apesar de sua simplicidade, o classificador Naive Bayes, em geral, apresenta bons resultados em problemas complexos do mundo real. No entanto, um estudo comparativo realizado em 2006[**2**], foi superado por outros métodos de classificação, a exemplo do *Random Forest* e *Boosted Tree*. Importante destacar que o classicador Naive Bayes possui como vantagem a necessidade de um reduzido conjunto de dados de treinamento para a obtenção dos parâmetros (médias e desvios padrões) utilizados no processo de classificação. A premissa ingênua de caracterícticas independentes elimina a necessidade do uso de matriz de covariâncias entre as variáveis preditoras.

A. Teorema de Bayes

O matemático e inglês Thomas Bayes (1701 – 1761) é o autor da formulação do Teorema expresso pela equação 1, a seguir:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \tag{1}$$

E podemos interpretá-la, de uma forma mais intuitiva, como:

$$Prob.Posterior = \frac{Prob.Anterior.Verossimilhança}{Evidencia}$$

- $P(A|B)$: chamada de **probabilidade posterior**, indica o nível de confiança na hipótese A após a análise dos dados observados em B;
- $P(A)$: probabilidade de hipótese A ser verdadeira, chamada de **probabilidade anterior**.
- $P(B|A)$: conhecido por verossimilhança (do inglês, *likelihood*) expressa o quão bem a hipótese A escolhida é capaz de reproduzir os dados observados em B.
- $P(B)$: probabilidade da observação B, independente de hipótese.

B. Estimação de Parâmetros

Os parâmetros do modelo (ou seja, classes anteriores e distribuições de probabilidade das variáveis preditoras) podem ser calculados a partir dos vetores x do conjunto de dados de treinamento. Para estimar os parâmetros de distribuição de uma característica (ou variável preditora), lidando com dados contínuos, um suposição típica é que os valores contínuos associados a cada classe c são distribuídos de acordo com uma distribuição normal ou gaussiana.

A partir dessa premissa, segmentam-se os dados em classes, e calcula-se a média e o desvio padrão de x do conjunto de caracteríticas pertencentes a cada classe. Seja μ a média dos valores de x e σ o desvio padrão dos valores associados a classe c.

As equações 2, 3, 4, a seguir, sintetizam o modelo Naive Bayes para eventos contínuos, em que as variáveis preditoras são os vetores x_i, com i de 1 a M (M: número de características ou preditores) e as classes observadas c_j, com j de 1 a N (N: número de classes).

$$p(x_i|c_j) = \frac{1}{\sqrt{2\pi\sigma_{i,j}^2}} e^{-\frac{1}{2}\left(\frac{x_i-\mu_{i,j}}{\sigma_{i,j}}\right)^2} \tag{2}$$

$$p(c_j \mid x) \overset{\text{(Bayes)}}{=} \frac{p(x_i \mid c_j) \cdot p(c_j)}{p(x)} \tag{3}$$

$$p(c_j \mid x) \overset{\text{(Naive)}}{=} \frac{\prod_{i=a}^{M} p(x_i \mid c_j) \cdot p(c_j)}{p(x)} \tag{4}$$

$$p(c_j \mid x) \overset{\text{(Gauss)}}{=} \frac{\prod_{i=a}^{M} \frac{1}{\sqrt{2\pi\sigma_{i,j}^2}} e^{-\frac{1}{2}\left(\frac{x_i-\mu_{i,j}}{\sigma_{i,j}}\right)^2} \cdot p(c_j)}{p(x)} \tag{5}$$

Dessa forma, o vetor de características x submetido à testes do algoritmo pertencerá à classe c_i que apresentar a maior probabilidade a posteriori:

$$p(c_i \mid x) > p(c_j \mid x) \tag{6}$$

Nota: como os denominadores das frações comparadas são os mesmos: $p(x)$, esse termo se cancela e não necessita, portanto, ser calculado

C. Matriz de confusão

De acordo com [**3**], uma matriz de confusão para um problema de n classes é uma matriz M de ordem nXn, no qual o elemento M_{ij} corresponde ao número de classificações corretas para a classe i, quando i = j. Para $i \neq j$, o elemento M_{ij} corresponde ao número de classificações da classe i que foram definidas incorretamente como classe j, e vice-versa.

Como estabelecido por [**4**], o número de acertos para cada classe se localiza na diagonal principal da matriz M.

Usualmente podem ser depreendidas algumas métricas de avaliação do modelo de classificação apenas com base na sua matriz de confusão. Neste trabalho, será utilizada apenas a acurácia, definida como uma porcentagem baseada no número de classificações corretas dividido pelo número total de tentativas de classificação, tal como expresso na equação 6.

$$acurácia = \frac{acertos}{número\ de\ classificações} \tag{7}$$

III. Metodologia

Nesta seção serão descritos os procedimentos metodológicos necessários à solução do problema proposto. Serão apresentadas as ferramentas de software, estratégia de partição do conjunto de treinamento do modelo de classificação e por fim a métrica de avaliação utilizada.

A. Ferramentas computacionais

1) Biblioteca Numpy: A biblioteca NumPy é fundamental para a computação científica em Python. Através desta biblioteca é possível manipular objetos do tipo array, com múltiplas dimensões, incluindo suas variações como é o caso das matrizes. Além disso, são fornecidas operações de álgebra linear, estatística e muitas outras[**5**].

2) Biblioteca Pandas: Pandas é uma biblioteca de código livre que fornece métodos práticos de se operar com estruturas de dados [**6**]. Para o estudo de ciência de dados, é especialmente útil na manipulação de datasets. Dentre os recursos oferecidos estão: redimensionamento, agrupamento, substituições, entre outros. As extensões de arquivos permitidas incluem os formatos ".xlsx", ".xls", ".csv', por exemplo.

3) Biblioteca Matplotlib: Matplotlib é uma biblioteca para visualização de dados estatísticos e criação de gráficos[**7**]. Pyplot é um módulo do matplotlib que fornece uma interface semelhante ao Matlab, com a vantagem de ser de código aberto e totalmente gratuito. Esta biblioteca pode ser integrada a scripts desenvolvidos na linguagem Python.

B. Descrição do problema proposto

A base de dados "iris_dataset", que pode ser encontrada em domínios públicos, é muito conhecida na literatura de reconhecimento de padrões. Esta base de dados é composta de 50 amostras de cada uma das três espécies da flor de íris, sendo elas: Iris setosa, Iris virginica e Iris versicolor. Cada amostra é composta de quatro atributos: comprimento e largura das pétalas e sépalas em centímetros.

Neste trabalho, os 4 atributos serão utilizados como um vetor de entrada de um classificador probabilístico de Naive-Bayes, tendo como objetivo classificar determinada flor de íris em sua respectiva espécie, dada as dimensões de suas pétalas e sépalas. Este modelo deve ser treinado e testado de acordo com o método de validação cruzada, ou seja, o dataset será dividido em 5 pastas distintas, estando estes conjuntos particionados em subconjunto de treino e subconjunto de teste.

Como métrica de avaliação, para cada conjunto de teste, será elaborada uma matriz de confusão, visando extrair a acurácia local. Ao final do experimento, deve-se apresentar os valores médios e desvios padrões das acurácias determinadas em cada uma das pastas.

C. Implementação do modelo

A implementação deste trabalho foi realizada através do desenvolvimento de um script na linguagem Python, tendo o suporte dos pacotes Numpy e Pandas. Para a edição e execução do código foi utilizado o ambiente de desenvolvimento Visual Studio Code.

Inicialmente foi realizada a leitura do dataset através do método *read_excel* do pacote Pandas, seguida do agrupamento das 3 classes em 3 matrizes chamadas W1, W2 e W3. O treinamento ocorreu em 5 pastas, sendo utilizado o método de validação cruzada como o intuito de variar o conjunto de treinamento ao longo de todo o dataset, e avaliar qual possui melhor desempenho. A tabela I apresenta a divisão dos conjuntos de teste (demarcados em azul) e conjuntos de treinamento (demarcados em verde) para as 5 pastas.

Tabela I
DIVISÃO DOS CONJUNTOS DE TREINO E TESTE PARA CADA CLASSE

150 Flores de Íris						
	Classe W1		Classe W2		Classe W3	
	Teste	Treino	Teste	Treino	Teste	Treino
Exper. 1	1-10	11-50	51-60	61-100	101-110	111-150
Exper. 2	11-20	1-10 U 21-50	61-70	51-60 U 71-100	111-120	101-110 U 121-150
Exper. 3	21-30	1-20 U 31-50	71-80	51-70 U 81-100	121-130	101-120 U 131-150
Exper. 4	31-40	1-30 U 41-50	81-90	51-80 U 91-100	131-140	101-130 U 141-150
Exper. 5	41-50	1-40	91-100	51-90	141-150	101-140

*Os conjuntos de coloração azul serão utilizados para a validação dos seus respectivos experimentos e os de coloração verde para treinamento.

Para cada classe separou-se um conjunto de teste contendo 10 elementos. Como se trata de 3 classes, o conjunto de teste possui 30 elementos em sua totalidade. À medida que os experimentos foram avançando, os subconjuntos de teste (representados pelas células azuis da tabela I) foram sendo

deslocados ao longo de suas respectivas classes, restando para o conjunto de treino a união dos intervalos remanescentes.

O procedimento de treinamento no classificador Naive-Bayes iniciou-se com a obtenção dos valores de média e desvio padrão dos dados de cada classe. A partir de então, calculou-se a probabilidade a posteriori de um elemento x pertencer às classes W1, W2 ou W3 através da expressão 5 da seção II. O maior valor de probabilidade define a que classe o elemento pertence.

O fluxograma exibido na figura 3 apresenta o procedimento completo de treinamento da máquina, obtendo a cada pasta a acurácia de classificação para o conjunto de dados apresentados. Por fim, obteve-se a média das acurácias alcançadas em cada uma das pastas, bem como a sua respectiva matriz de confusão, a fim de analisar o desempenho médio do classificador.

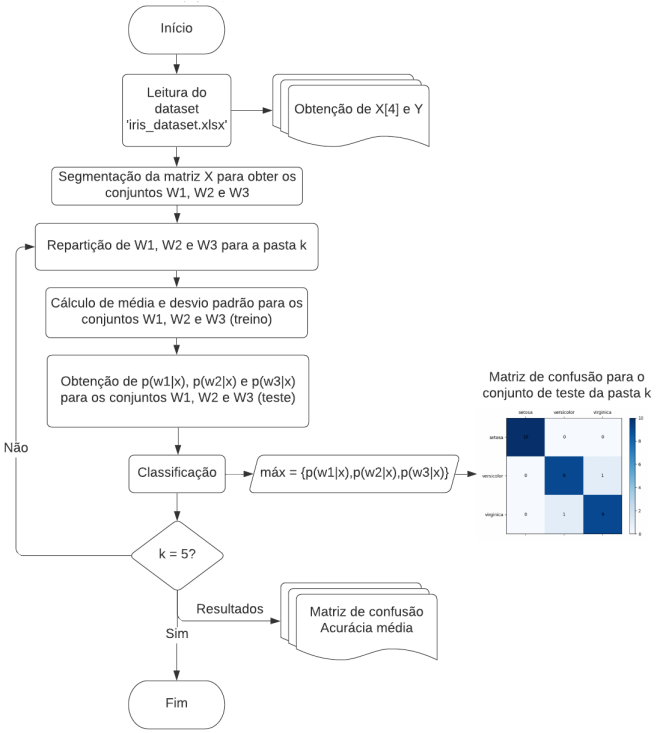

Figura 3. Fluxograma do código desenvolvido

IV. Resultados

A partir da obtenção das probabilidades de uma observação X pertencer às classes W1, W2 e W3, foi possível classificar esta observação em determinada classe. A tabela II ilustra a taxa de acerto do classificador para cada classe dentro do seu conjunto de teste ao longo dos 5 experimentos. É também exibida a média das acurácias alcançadas em cada uma das pastas.

Tabela II
Média das acurácias alcançadas em cada pasta

Classe	Exper. 1	Exper. 2	Exper. 3	Exper. 4	Exper. 5
Setosa (W1)	100%	100%	100%	100%	100%
Versicolor (W2)	90%	100%	80%	100%	100%
Virginica (W3)	90%	90%	100%	80%	100%
μ (Média)	93,33%	96,67%	93,33%	93,33%	100%
σ (Desvio Padrão)	2,67%				
Acurácia Média	95,33%				

*A acurácia média dos 5 experimentos assumiu o valor de 95,33% com um desvio padrão de 2,67%.

Como maneira de expressar graficamente o resultado das classificações e facilitar o processo de análise dos resultados, foram geradas as matrizes de confusão para cada pasta através dos métodos da biblioteca matplotlib. As figuras 4, 5, 6, 7 e 8 ilustram esse processo.

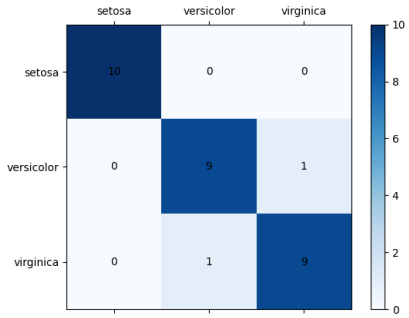

Figura 4. Matriz de confusão para a pasta 1

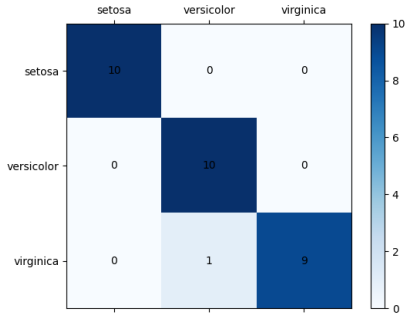

Figura 5. Matriz de confusão para a pasta 2

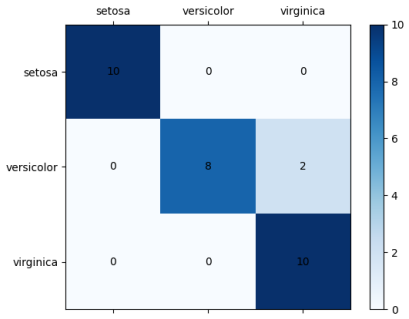

Figura 6. Matriz de confusão para a pasta 3

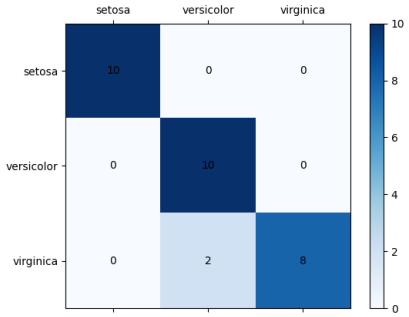

Figura 7. Matriz de confusão para a pasta 4

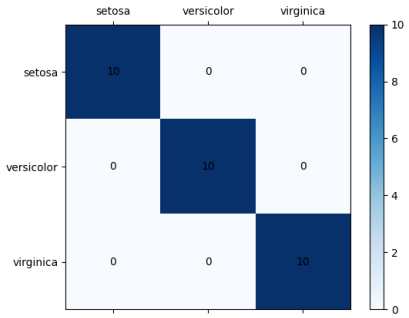

Figura 8. Matriz de confusão para a pasta 5

V. Conclusões

A implementação do classificador de Naive-Bayes para 3 classes teve como característica a simplicidade matemática, uma vez que foram necessárias apenas operações elementares para os cálculos de média, desvio padrão e exponenciação, no caso da função de densidade de probabilidade Gaussiana. O método de validação cruzada seguiu os mesmos princípios abordados em trabalhos anteriores, adicionando o fato de que os conjuntos de teste e treinamento foram balanceados para as 3 classes envolvidas. Os cinco experimentos realizados seguiram a distribuição exibida na tabela 1, tendo como resultado as acurácias dispostas na tabela 2. A acurácia média de 95,33% indica um nível de assertividade aceitável, visto que em um conjunto de 150 observações, em média 7 seriam classificadas erroneamente. A pasta 5 obteve acurácia máxima, as demais oscilaram entre 1 e 2 erros de classificação. Portanto, o classificador de Naive Bayes mostra-se robusto e de boa performance para dados reais, apesar de inferir uma independência entre as características.

Referências

[**1**]L. Yumei and A.s.Richard "Facies identification from well logs: A comparison of discriminant analysis and naïve Bayes classifier". ScienceDirect, 2006.

[**2**]R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms". Proceedings of the 23rd international conference on Machine learning, 2006.

[**3**]S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification,"in Pacific-Asia conference on knowledge discovery and data mining. Springer, 2004.

[**4**]M. Douglas C, Applied statistics and probability for engineers. John Wiley Sons, Inc, 2002.

[**5**]"Numpy: About Us," 2020. Accessed on: Apr. 16, 2021. [Online]. Available: https://numpy.org/about/

[**6**]"Pandas documentation," 2021. Accessed on: Apr. 16, 2021. [Online]. Available: https://pandas.pydata.org/docs/

[**7**]"Matplotlib: Visualization with Python," 2021. Accessed on: Apr. 16, 2021. [Online]. Available: https://matplotlib.org/stable/index.html

```
################### PPGEE 2021/01 ####################
############### Aprendizado de maquina ################
# Trabalho 2: Classificador probabil stico Naive Bayes
# Alunos: Carlos Henrique
#         Willian Guerreiro Colares

#Import de bibliotecas para manipulacao e visualizacao de dados
import math
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Funcao para calculo de densidade de probabilidade Gaussiana
#Par metros: x->feature, u-> m dia, sigma->desvio padr o
def gauss_distribution(x,u,sigma):
    p_x_w = math.exp(-0.5*pow(x-u,2)/pow(sigma,2))/(sigma*math.sqrt(2*math.pi))
    return p_x_w

#Funcao para calculo de media
#Aux eh uma matriz Mx4
def average(aux):
    soma = 0
    for i in range(len(aux)):
        soma = soma + aux[i]
    return soma/len(aux)

#Funcao para calculo de desvio padrao
#Aux eh uma matriz Mx4
def standard_dev(aux):
    soma = 0
    std_dev = 0
    for i in range(len(aux)):
        soma = soma + pow(aux[i] - average(aux),2)
    std_dev = soma/len(aux)
    for i in range(4):
        std_dev[i] = math.sqrt(std_dev[i])
    return std_dev

#Passo 1: Leitura do Dataset
dataset = pd.read_excel('iris_dataset.xlsx')

#Obtencao do numero de observacoes N=150
N = dataset.shape[0]

#Passo 2: Tratamento dos dados
dataset['species'].replace('setosa', 1, inplace=True)
dataset['species'].replace('versicolor', 2, inplace=True)
dataset['species'].replace('virginica', 3, inplace=True)

graph_labels = ['setosa','versicolor','virginica']

#Passo 3: Obtencao das caracteristicas, matriz de caracteristicas e labels
X_meas1 = np.array((dataset['meas1'])).reshape((-1,1))
X_meas2 = np.array((dataset['meas2'])).reshape((-1,1))
X_meas3 = np.array((dataset['meas3'])).reshape((-1,1))
X_meas4 = np.array((dataset['meas4'])).reshape((-1,1))
#Matriz de caracteristicas
X = np.concatenate([X_meas1,X_meas2,X_meas3,X_meas4],axis = 1)
#Labels
Y = np.array((dataset['species'])).reshape((-1,1))

#Passo 4: Segmentacao da matriz de caracteristicas em 3 classes W1, W2 e W3 (50 elementos cada)
#Classe 1 : 1   ao 50   elemento
w1 = X[:50]
```

```
#Classe 2 : 51   at   o primeiro dos 50   ltimos
w2 = X[50:-50]
#Classe 3: Os 50   ltimos   elementos
w3 = X[-50:]

#N mero de pastas
K = 5

acuracia_media = 0

for p in range(K):
    print('Pasta k=',p+1)
    # [0;c[   [c;d[    [d;N[      sendo   [c;d[ o subset de teste   e  d-c = testing_size = 10
    c = 10*p
    d = 10*(p+1)

    #Obtencao dos conjuntos de teste
    w1_testing = w1[c:d]
    w2_testing = w2[c:d]
    w3_testing = w3[c:d]
    #Obtencao dos conjuntos de treinamento
    w1_trainning = np.concatenate((w1[0:c],w1[d:50]),axis=0)
    w2_trainning = np.concatenate((w2[0:c],w2[d:50]),axis=0)
    w3_trainning = np.concatenate((w3[0:c],w3[d:50]),axis=0)

    #Etapa de treinamento (obtencao de medias e desvios padroes)
    w1_u = average(w1_trainning)
    w1_sigma = standard_dev(w1_trainning)

    w2_u = average(w2_trainning)
    w2_sigma = standard_dev(w2_trainning)

    w3_u = average(w3_trainning)
    w3_sigma = standard_dev(w3_trainning)

    #Declaracao de matrizes 10x3
    w1_test = np.ones((10,3))
    w2_test = np.ones((10,3))
    w3_test = np.ones((10,3))

    for i in range(10):
        for j in range(4):
            #Obtencao de p(w1|x), p(w2|x) e p(w3|x) no conjunto de teste W1
            w1_test[i][0] = w1_test[i][0] * gauss_distribution(w1_testing[i][j],w1_u[j],w1_sigma[j])
            w1_test[i][1] = w1_test[i][1] * gauss_distribution(w1_testing[i][j],w2_u[j],w2_sigma[j])
            w1_test[i][2] = w1_test[i][2] * gauss_distribution(w1_testing[i][j],w3_u[j],w3_sigma[j])

            #Obtencao de p(w1|x), p(w2|x) e p(w3|x) no conhunto de teste W2
            w2_test[i][0] = w2_test[i][0] * gauss_distribution(w2_testing[i][j],w1_u[j],w1_sigma[j])
            w2_test[i][1] = w2_test[i][1] * gauss_distribution(w2_testing[i][j],w2_u[j],w2_sigma[j])
            w2_test[i][2] = w2_test[i][2] * gauss_distribution(w2_testing[i][j],w3_u[j],w3_sigma[j])

            #Obtencao de p(w1|x), p(w2|x) e p(w3|x) no conhunto de teste W3
            w3_test[i][0] = w3_test[i][0] * gauss_distribution(w3_testing[i][j],w1_u[j],w1_sigma[j])
            w3_test[i][1] = w3_test[i][1] * gauss_distribution(w3_testing[i][j],w2_u[j],w2_sigma[j])
            w3_test[i][2] = w3_test[i][2] * gauss_distribution(w3_testing[i][j],w3_u[j],w3_sigma[j])

    #Obtencao de p(w1|x), p(w2|x) e p(w3|x) dos conjuntos W1,W2 e W3
    #Probabilidade a priori = 1/3
    w1_test = w1_test/3
    w2_test = w2_test/3
    w3_test = w3_test/3
    print(w1_test)
    print(w2_test)
    print(w3_test)
    #Variaveis para armazenar os casos de acerto na classificacao
    predicted_w1 = 0
```

```
    predicted_w2 = 0
    predicted_w3 = 0

    confusion_matrix = np.zeros((3,3))

    #Contabilizacao de p(w1|x) > p(w2|x) e p(w1|x) > p(w3|x)
    for element in w1_test:
        i = np.where(element == max(element))
        confusion_matrix[0][i]+=1
        if i[0][0] == 0:
            predicted_w1 += 1

    #Contabilizacao de p(w2|x) > p(w1|x) e p(w2|x) > p(w3|x)
    for element in w2_test:
        i = np.where(element == max(element))
        confusion_matrix[1][i]+=1
        if i[0][0] == 1:
            predicted_w2 += 1

    #Contabilizacao de p(w3|x) > p(w2|x) e p(w3|x) > p(w1|x)
    for element in w3_test:
        i = np.where(element == max(element))
        confusion_matrix[2][i]+=1
        if i[0][0] == 2:
            predicted_w3 += 1

    print(confusion_matrix)

    #Bloco de codigo para estilizacao e visualizacao da matriz de confusao
    window_name = 'Pasta ' + str(p+1)
    fig = plt.figure(window_name)
    ax = fig.add_subplot(111)
    cax = ax.matshow(confusion_matrix,cmap = 'Blues')
    for (c,r),label in np.ndenumerate(confusion_matrix):
        ax.text(r,c,int(label),ha='center',va='center')
    ax.set_xticklabels(['']+graph_labels)
    ax.set_yticklabels(['']+graph_labels)
    fig.colorbar(cax)
    plt.show()

    #Classificacoes corretas das classes 1,2 e 3
    #print(predicted_w1,predicted_w2,predicted_w3)
    #Calculo da acuraria para os 30 elementos do conjunto de testes
    acuracia = 100*(predicted_w1+predicted_w2+predicted_w3)/30
    print('Acuracia ',acuracia)
    #Acumulo da acuracia para cada pasta
    acuracia_media+=acuracia

#Calculo da acuracia media
acuracia_media = acuracia_media/5
print('Acuracia media ',acuracia_media)

if True:
    f1, (ax1,ax2) = plt.subplots(1,2)
    f1.canvas.set_window_title('Conjunto de dados reais')

    #Measure1, Measure2 e Species
    scatter = ax1.scatter(X_meas1, X_meas2, c = Y)
    ax1.set_xlabel(dataset.columns[0])
    ax1.set_ylabel(dataset.columns[1])
    handles, labels = scatter.legend_elements()
    labels = ['setosa','versicolor','virginica']
    ax1.legend(handles, labels, loc='best')

    #Measure1, Measure3 e Species
    scatter = ax2.scatter(X_meas1, X_meas3, c = Y)
    ax2.set_xlabel(dataset.columns[0])
```

```
ax2.set_ylabel(dataset.columns[2])
handles, labels = scatter.legend_elements()
labels = ['setosa','versicolor','virginica']
ax2.legend(handles, labels, loc='best')


f2, (ax3,ax4) = plt.subplots(1,2)
f2.canvas.set_window_title('Conjunto de dados reais')

#Measure1, Measure4 e Species
scatter = ax3.scatter(X_meas1, X_meas4, c = Y)
ax3.set_xlabel(dataset.columns[0])
ax3.set_ylabel(dataset.columns[3])
handles, labels = scatter.legend_elements()
labels = ['setosa','versicolor','virginica']
ax3.legend(handles, labels, loc='best')

#Measure2, Measure3 e Species
scatter = ax4.scatter(X_meas2, X_meas3, c = Y)
ax4.set_xlabel(dataset.columns[1])
ax4.set_ylabel(dataset.columns[2])
handles, labels = scatter.legend_elements()
labels = ['setosa','versicolor','virginica']
ax4.legend(handles, labels, loc='best')

f3, (ax5,ax6) = plt.subplots(1,2)
f3.canvas.set_window_title('Conjunto de dados reais')

#Measure2, Measure4 e Species
scatter = ax5.scatter(X_meas2, X_meas4, c = Y)
ax5.set_xlabel(dataset.columns[1])
ax5.set_ylabel(dataset.columns[3])
handles, labels = scatter.legend_elements()
labels = ['setosa','versicolor','virginica']
ax5.legend(handles, labels, loc='best')

#Measure3, Measure4 e Species
scatter = ax6.scatter(X_meas3, X_meas4, c = Y)
ax6.set_xlabel(dataset.columns[2])
ax6.set_ylabel(dataset.columns[3])
handles, labels = scatter.legend_elements()
labels = ['setosa','versicolor','virginica']
ax6.legend(handles, labels, loc='best')


plt.show()
```