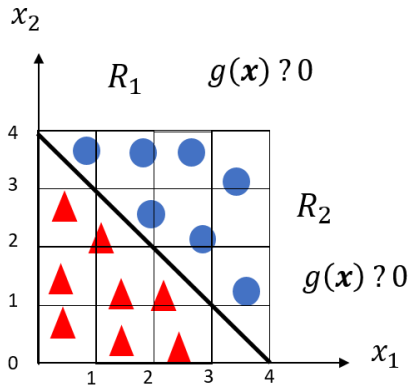
TP555 - AI/ML

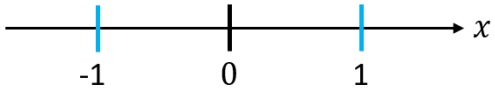
Lista de Exercícios #4

Classificação Linear: Parte 1

1. Dada a figura abaixo e a seguinte função discriminante: $g(\boldsymbol{x})$ = a0 + a1*x1 + a2*x2, encontre os pesos e as regiões de decisão.



2. Dada a seguinte figura, a qual representa os símbolos da modulação BPSK, encontre uma função discriminante linear, $g(\boldsymbol{x})$, que consiga classificar esses símbolos. Desenhe a função discriminante juntamente com os símbolos indicando as regiões de decisão.



bits	Símbolo
0	-1
1	$+1$

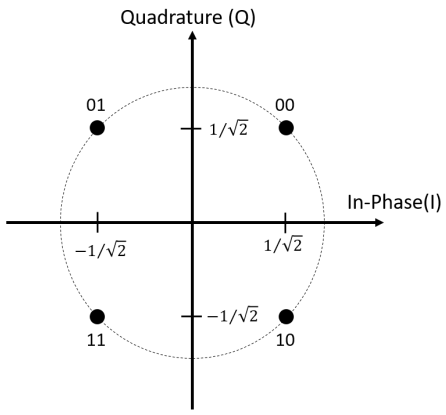
3. Dada a figura abaixo, a qual representa os símbolos da modulação QPSK, encontre as **funções discriminantes lineares**, que consigam classificar esses símbolos. Desenhe as funções discriminantes juntamente com os símbolos indicando as regiões de decisão. Em seguida, usando essas funções discriminantes faça o seguinte:
    a. Crie uma array com N=1000000 valores aleatórios, variando entre 0 e 3, com a função **numpy.random.randint**, passe esse vetor para a **função modulator** e armazena a saída da função em um vetor symbols.
    b. Adicione ruído gaussiano branco ao vetor de saída da função **modulator**. Varie a relação energia de símbolo (Es) por densidade espectral do ruído (N0) de -2 a 20 dB em passos de 2 dB.

c. Crie uma função **demodulator** que utiliza as **funções discriminantes** e calcule o erro de símbolo simulado para cada valor de Es/N0.

d. Em seguida, plote um gráfico comparando o taxa de erro de símbolo (SER) simulado com a taxa de erro de símbolo teórica, a qual é dada por

$$\text{SER} = erfc\left(\sqrt{\frac{Es}{2N0}}\right) - \frac{1}{4}erfc\left(\sqrt{\frac{Es}{2N0}}\right)^2$$

(**Dica**: As duas curvas devem coincidir quase que perfeitamente.)



bits	Símbolo (I + jQ)
00	$\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$
01	$-\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$
10	$\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$
11	$-\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$

4. Neste exercício você utilizará o teorema de Bayes. Considere dois exames médicos, A e B, para um vírus. O teste A é 95% eficaz no reconhecimento do vírus quando ele está presente, mas tem uma taxa de falso positivo de 10% (indicando que o vírus está presente, quando ele não está). O teste B é 90% eficaz no reconhecimento do vírus, mas possui uma taxa de falso positivo de 5%. Os dois testes usam métodos independentes para identificar o vírus. 1% de todas as pessoas possuem o vírus. Digamos que uma pessoa é testada para o vírus usando apenas um dos testes e que o teste é positivo para o vírus. Qual teste, retornando positivo, é mais indicativo de alguém realmente estar com o vírus?
5. Neste exercício você vai prever se Jair pagará o empréstimo que ele está solicitando junto a um banco para montar uma indústria farmacêutica especializada na produção de hidroxicloroquina. Jair possui os seguintes atributos: **Possui casa própria? Não - Estado civil: Casado - Experiência de trabalho: 3**. Portanto, dado estes atributos sobre Jair, qual a probabilidade de que ele pague o empréstimo? Qual a probabilidade de que ele não pague o empréstimo. Baseado nas duas probabilidades, caso você trabalhasse no banco, você autorizaria o empréstimo? Para calcular as probabilidades, utilize os dados da tabela abaixo. (**Dica**: utilize a teoria do classificador naive Bayes).
   OBS.: Todos os atributos são discretos, ou seja, assumem valores de um conjunto finito de valores. Por exemplo, o atributo experiência de trabalho assume apenas os seguintes valores: 0, 1, 2, 3, 4 e 5.

Possui casa própria?	Estado civil	Experiência de trabalho (0-5)	Pagou?
Sim	Solteiro	3	Sim
Não	Casado	4	Sim
Não	Solteiro	5	Sim
Sim	Casado	4	Sim
Não	Divorciado	2	Não
Não	Casado	4	Sim
Sim	Divorciado	2	Sim
Não	Casado	3	Não
Não	Casado	3	Sim
Sim	Solteiro	2	Não

6. Exercício sobre classificação de bayes: Usando a teoria Bayesiana de decisão e os dados de treinamento abaixo, encontre a probabilidade de uma pessoa com os seguintes atributos: **idade <= 30**, **renda = Média**, **estudante = Sim** e **classificação de crédito = boa**, comprar ou não um personal computer (PC). Baseado nas probabilidades encontradas, esta pessoa compraria ou não o PC? Apresente todos os cálculos necessários para se calcular as probabilidades.

Exemplo	Idade	Renda	Estudante	Classificação de crédito	Classificação: Compra o PC?
1	<= 30	Alta	Não	Boa	Não
2	<= 30	Alta	Não	Excelente	Não
3	31 a 40	Alta	Não	Boa	Sim
4	> 40	Média	Não	Boa	Sim
5	> 40	Baixa	Sim	Boa	Sim
6	> 40	Baixa	Sim	Excelente	Não
7	31 a 40	Baixa	Sim	Excelente	Sim
8	<= 30	Média	Não	Boa	Não
9	<= 30	Baixa	Sim	Boa	Sim
10	> 40	Média	Sim	Boa	Sim
11	<= 30	Média	Sim	Excelente	Sim
12	31 a 40	Média	Não	Excelente	Sim
13	31 a 40	Alta	Sim	Boa	Sim
14	> 40	Média	Não	Excelente	Não

7.

8. Neste exercício você vai prever, baseado em alguns atributos físicos de uma pessoa, se ela é do sexo masculino ou feminino. Dado os seguintes atributos físicos de uma pessoa: altura = 1.83 metros, peso = 58.97 Quilos e tamanho do calçado = 20.32 centímetros. Baseado nas informações anteriores, qual classe tem maior probabilidade, ou seja, qual dos 2 sexos teria a maior probabilidade? Para calcular as probabilidades, utilize os dados da tabela abaixo. **OBS**.: Apresente todos os cálculos feitos para se encontrar as probabilidades de cada classe, ou seja, neste exercício você não deve utilizar a biblioteca SciKit-learn.
   (**Dica**: Assuma que os as probabilidades condicionais dos atributos seguem uma distribuição Gaussiana).
   (**Dica**: Assuma que a probabilidade da pessoa ser do sexo masculino ou do feminino é de 0.5, respectivamente).
   (**Dica**: utilize a teoria do classificador naive Bayes e lembre-se que o numerador da equação do classificador não influencia na maximização das probabilidades).

Altura [m]	Peso [Kg]	Tamanho calçado [cm]	Sexo
1.83	81.65	30.48	masculino
1.80	86.18	27.94	masculino
1.70	77.11	30.48	masculino
1.80	74.84	25.40	masculino
1.52	45.36	15.24	feminino
1.68	68.04	20.32	feminino
1.65	58.97	17.78	feminino
1.75	68.04	22.86	feminino

9. Use um classificador Naive Bayes Gaussiano (***GaussianNB***) para separar os exemplos de duas classes gerados pelo trecho de código abaixo e faça o seguinte
   a. Plote uma figura com as duas classes. Use marcadores diferentes para diferenciar exemplos de cada classe.
   b. Divida o conjunto em conjuntos de treinamento (70%) e validação (30%).
   c. Treine o classificador com o conjunto de treinamento e calcule a acurácia com o conjunto de validação.
   d. Plote uma figura com as regiões de decisão.
   e. Plote a matriz de confusão.

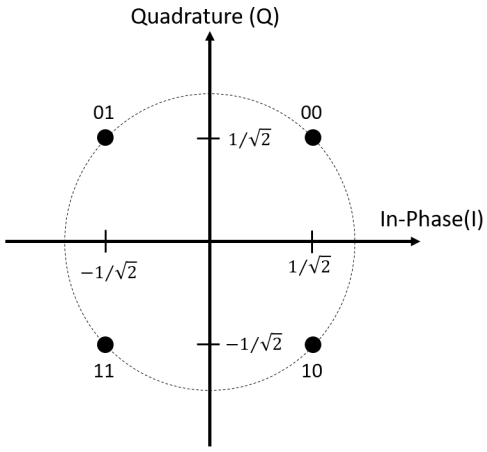
```
import numpy as np
from sklearn.datasets import make_circles

# Reset the PN sequence generator.
seed = 42
np.random.seed(seed)
```

```
# Number of examples.
N = 1000

# Create a 2-class dataset for classification.
X, y = make_circles(n_samples=N, factor=.5, noise=.05, random_state=seed)
```

10. Neste exercício você irá implementar um classificador linear, utilizando o classificador naive Bayes, para realizar a detecção de símbolos QPSK. Os símbolos QPSK são dados pela figura e tabela abaixo.



bits	Símbolo (I + jQ)
00	$\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$
01	$-\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$
10	$\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$
11	$-\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$

O resultado do seu classificador (neste caso, um detector) pode ser comparado com a curva da taxa de erro de símbolo (SER) teórica, a qual é dada por

$$\text{SER} = erfc\left(\sqrt{\frac{Es}{2N0}}\right) - \frac{1}{4}erfc\left(\sqrt{\frac{Es}{2N0}}\right)^2.$$

Utilizando a classe GaussianNB do módulo naive_bayes da biblioteca sklearn, faça o seguinte

A. Treine um classificador linear, utilizando o classificador naive Bayes, com uma relação sinal ruído elevada.
B. Use o modelo treinado para realizar a detecção dos símbolos QPSK.
    a. Gere N = 1000000 símbolos QPSK aleatórios.
    b. Passe os símbolos através de um canal AWGN.
    c. Detecte a probabilidade de erro de símbolo para cada um dos valores do vetor Es/N0 = [-2, 0, 2, 4, 6, 8, 10,12, 14, 16, 18, 20].
    d. Você pode utilizar o template abaixo para criar seu código.
C. Apresente um gráfico comparando a SER simulada e a SER teórica versus os valores de Es/N0 definidos acima.
D. Podemos dizer que a curva simulada se aproxima da curva teórica da SER?
E. Se as classes, ou seja, os símbolos, tivessem probabilidades diferentes, nós poderíamos dizer que o classificador ML é equivalente ao MAP?

(**Dica**: Como os símbolos são representados por números complexos e a classe GaussianNB não suporta tal representação, você terá que instanciar 2 objetos da classe, um para cada componente do símbolo, ou seja, um classificador para a parte real (i.e., In-phase - I) e outro para a parte imaginária (Quadrature - Q).
(**Dica**: A função ***erfc*** pode ser importada da seguinte forma: *from scipy.special import erfc*).
(**Dica**: Uma rápida revisão sobre taxa de erro de símbolo pode ser encontrada no link: http://www.dsplog.com/2007/11/06/symbol-error-rate-for-4-qam/).

```
# Template of a QPSK detection loop
# Import all necessary libraries.
import numpy as np
from scipy.special import erfc
from sklearn.naive_bayes import GaussianNB
import matplotlib.pyplot as plt

# Number of QPSK symbols to be transmitted.
N = 1000000

# Instantiate a Gaussian naive Bayes classifier for each one of the parts of a QPSK
symbol.
gnb_re = ????
gnb_im = ????

# Create Es/N0 vector.
EsN0dB = np.arange(-2,22,2)

ser_simu = np.zeros(len(EsN0dB))
ser_theo = np.zeros(len(EsN0dB))
for idx in range(0,len(EsN0dB)):

    print('Es/N0 dB:', EsN0dB[idx])

    EsN0Lin = 10.0**(-(EsN0dB[idx]/10.0))

    # Generate N QPSK symbols.
    ip =(2.0 * (np.random.rand(N, 1) >= 0.5) - 1.0) + 1j*(2.0 * (np.random.rand(N, 1) >=
0.5) - 1.0)
    # Normalization of energy to 1.
    s = (1/np.sqrt(2))*ip;

    # Generate noise vector with unitary variance.
    noise = np.sqrt(1.0/2.0)*(np.random.randn(N, 1) + 1j*np.random.randn(N, 1))

    # Pass symbols through AWGN channel.
    y = s + np.sqrt(EsN0Lin)*noise
```

```
    # Fit model for real part.
    ????
    # Fit model for imaginary parts.
    ????
    # Prediction for real part.
    detected_ip_re = ????
    # Prediction for imaginary part.
    detected_ip_im = ????

    # Simulated QPSK BER.
    error_re = (ip.real != detected_ip_re)
    error_im = (ip.imag != detected_ip_im)
    error = 0;
    for i in range(0, N):
        if(error_re[i]==True or error_im[i]==True):
            error = error + 1

    ser_simu[idx] = 1.0 * error / N

    # Theoretical BPSK BER.
    ser_theo[idx] = erfc( np.sqrt(  0.5*(10.0**(EsN0dB[idx]/10.0))   ) ) -
(1/4)*(erfc(np.sqrt(0.5*(10.0**(EsN0dB[idx]/10.0)))))**2.0
```

11. Neste exercício você fará a classificação de textos em uma das categorias que serão definidas. Utilize como base o exemplo: ***ClassifyingTextMultinomialNB.ipynb***. As categorias que devem ser classificadas pelo classificador são: 'comp.windows.x', 'comp.os.ms-windows.misc', 'misc.forsale' e 'rec.autos'. Treine e valide o classificador com os dados da base “20 Newsgroups corpus” da biblioteca scikit-learn. Plote a matriz de confusão. Analise e matriz de confusão e responda
    a. O que você percebe em relação à classe ‘comp.os.ms-windows.misc’?
    b. Qual uma possível explicação para o que você percebeu no item anterior?

    (**Dica**: Informações sobre matriz de confusão:
    https://en.wikipedia.org/wiki/Confusion_matrix
    https://dev.to/overrideveloper/understanding-the-confusion-matrix-264i)
12. Neste exercício você fará a classificação de algumas mensagens em duas categorias ‘spam’ e ‘ham’. Utilize as 6 mensagens abaixo e seus respectivos rótulos para treinar um classificador naive Bayes Bernoulli.

```
# Features.
x_train = np.array(['free great offer if you join, a great offer for free!',
        'great offer for free delivery',
        'uber is promoting a great offer for free',
        'try uber for free for your 1st ride',
        'earn your uber 10 credit for free by applying for the uber visa credit card',
        'uber receipt'])
```

```
# Labels.
y_train = np.array(['spam','spam','spam','ham','ham','ham'])
```

Use a classe ***CountVectorizer*** com o parâmetro ***binary=True*** para criar a matriz indicando a presença ou não de uma palavra, ou seja, uma matriz com valores booleanos. Em seguida, treine o classificador. De posse do modelo treinado, preveja a qual classe as 2 mensagens abaixo pertencem.

```
x_test = np.array(['Moonnight Trial', 'Limited offer: Free & Great Deal'])
```

(**Dica**: use como base o exemplo: SPAMClassificationBernoulliNB.ipynb)

13. Neste exercício você irá comparar as classificações naive Bayes Multinomial e Bernoulli. Utilize as mensagens abaixo e seus respectivos rótulos para treinar um classificador naive Bayes com distribuição de Bernoulli e outro classificador naive Bayes com distribuição Multinomial.

```
x_train = np.array(['Chinese Beijing Chinese',
                    'Chinese Chinese Shanghai',
                    'Chinese Macao',
                    'Tokyo Japan Chinese'])

y_train = np.array(['china','china','china','not china'])
```

Instancie um objeto da classe ***CountVectorizer*** com o parâmetro ***binary=True*** para o classificador naive Bayes com distribuição de Bernoulli. Para o classificador naive Bayes com distribuição Multinomial, instancie um objeto da classe ***CountVectorizer*** com o parâmetro ***binary=False***. Em seguida, treine os classificadores. Utilize os seguintes comandos para verificar os nomes dos atributos e a matriz com a contagem dos atributos para cada instância da classe ***CountVectorizer***, onde ***vect*** é o objeto da classe ***CountVectorizer*** e ***x_train_dtm*** é o matriz de contagem gerada pela execução do método ***fit_transform*** da classe ***CountVectorizer***. Não se esqueça de transformar a mensagem de validação, ***x_test***, com o método ***transform***, antes de predizer sua classe para cada classificador.

```
print(vect.get_feature_names())
print(x_train_dtm.toarray())
```

De posse dos modelos treinados, pede-se

A. Imprima o nome dos atributos e a matriz de contagem dos atributos para cada uma das instâncias de ***CountVectorizer***.
B. Utilize o método ***predict*** das classes ***BernoulliNB*** e ***MultinomialNB*** e preveja a qual classe a mensagem abaixo pertence para cada um dos classificadores.

   ```
   x_test = np.array(['Chinese Chinese Chinese Tokyo Japan'])
   ```

C. Calcule manualmente (ou seja, sem utilizar a biblioteca SciKit-learn) a probabilidade de cada classe, ou seja, ‘china’ e ‘not china’, dado a mensagem de teste para os 2 classificadores. Apresente os cálculos das probabilidades ***a priori*** e ***a posteriori***.

D. Utilize o método ***predict_proba*** das classes ***BernoulliNB*** e ***MultinomialNB*** para imprimir os resultados das probabilidades e confira se elas são iguais às que você encontrou manualmente no item (C). Utilize o comando abaixo para imprimir as probabilidades.

```
print(model.predict_proba(x_test_dtm))
```

E. Como você deve ter percebido, existe diferença na classificação feita pelos 2 classificadores. Explique o motivo da classificação feita por cada classificador. (**Dica**: Imprima o vetor de contagens de cada classificador com o comando,

```
print(x_test_dtm.toarray())
```

compare as contagens de cada palavra no vetor, além disso, o item (C) acima vai te ajudar a entender e responder este item).

Observação: quando vocês forem calcular as probabilidades condicionais, vocês irão se deparar com probabilidades nulas, e.g., P('japan' | 'china') = 0, e isso faria com que as respostas finais fossem zeradas. Uma solução para esse problema é utilizar a ***suavização de Laplace*** também conhecida como ***suavização adicione 1*** [1,2]. A suavização é diferente para os 2 classificadores estudados, i.e., MultinomialNB e BernoulliNB.

- **Suavização de Laplace para o caso do MultinomialNB**: Com a suavização as probabilidades condicionais se tornam

$$P(x_k \mid C_q) = \frac{contagem(x_k, C_q) + 1}{\sum_{l=1}^{K} contagem(x_l, C_q) + |V|},$$

onde $contagem(x_k, C_q)$é número de vezes que a palavra $x_k$aparece entre todas as palavras que pertencem à classeC_q, $\sum_{l=1}^{K} contagem(x_l, C_q)$ é a soma total de palavras pertencentes à classe C_qe $|V|$é o tamanho do vocabulário, ou seja, o número de atributos. Por exemplo, para o Classificador MultinomialNB P('japan' | 'china') = (0 + 1) / (8 + 6), onde $contagem(x_k = 'japan', C_q = 'china')$=0, $\sum_{l=1}^{K} contagem(x_l, C_q)$ = 8 e $|V|$= 6, pois os atributos são 'beijing', 'chinese', 'japan', 'macao', 'shanghai' e 'tokyo', ou seja, 6 termos/palavras.

- **Suavização de Laplace para o caso do BernoulliNB**: Para o caso do BernoulliNB, utilizando-se a ***suavização de Laplace***, $P(x_k \mid C_q)$ é calculada como

$$P(x_k \mid C_q) = \frac{M_{t,C_q} + 1}{M_C + 2},$$

onde M_{t,C_q} é o número de mensagens de treinamento da classe C_q que contém o termo/palavra x_k, enquanto M_C é o número total de mensagens de treinamento da categoria C_q e 2 pois existem dois casos a serem considerados para cada termo/palavra, ocorrência e não ocorrência.

Referências

[1] 'An Introduction to Naïve Bayes Classifier', https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf

[2] 'Additive smoothing', https://en.wikipedia.org/wiki/Additive_smoothing

14. **Exercício sobre classificação naive bayes**: Considere um conjunto de críticas (*reviews*) de filmes, ***R***, onde cada uma delas pertence a classe **Positive** (+) ou **Negative** (-). Dado um conjunto de treinamento com 6 críticas, crie 2 classificadores, um **Multinomial Naive Bayes (NB)** e outro **Bernoulli NB**, ambos com ***suavização de Laplace***, para classificar críticas não rotuladas como **Positive** ou **Negative**. Para este exercício, define-se o seguinte vocabulário com 4 palavras: ***V*** = ['good', 'poor, 'boring', 'great'].
    A. Para o classificador **MultinomialNB**, cada crítica, r_i, é representada como um vetor com 4 elementos, onde cada elemento define a frequência de cada palavra do alfabeto ***V*** na crítica r_i. Os dados de treinamento são apresentados a seguir como uma matriz, onde cada linha representa uma crítica.

Review	Vocabulary				Class
	"good"	"poor"	"boring"	"great"	
r1	3	0	0	3	Positive (+)
r2	0	1	1	2	Positive (+)
r3	1	0	1	2	Positive (+)
r4	1	3	2	0	Negative (-)
r5	1	5	0	2	Negative (-)
r6	0	2	2	0	Negative (-)

Agora, de posse dessas informações faça o seguinte:

a. Classifique **manualmente** as críticas de teste a seguir em **Positive** ou **Negative**. Apresente todos os cálculos necessários para se decidir a classe das 3 críticas de teste.
    i. r_1 =*A good direction. But boring locations. Overall a good movie.*
    ii. r_2 =*A good, good plot and great characters, but poor acting.*
    iii. r_3 =*Good plot, but poor acting.*

(**Dica**: com a classificação de texto naive Bayes, simplesmente ignoramos qualquer palavra que não ocorra, ou seja, que não esteja no conjunto de treinamento.)

b. Utilizando a classe **MultinomialNB** do scikit-learn, instancie e treine um classificador para decidir a classe das 3 críticas de teste. **OBS**.: Não é necessário utilizar a classe **CountVectorizer**. Para o treinamento, simplesmente use a matriz de treinamento dada acima com as frequências das palavras do vocabulário, em seguida, passe a matriz para o classificador (i.e., para o método fit) juntamente com os labels, **'Positive'** e '**Negative**', correspondentes a cada crítica da matriz de treinamento. Para o teste, crie uma matriz onde cada linha contém a frequência das palavras do vocabulário para cada crítica (i.e., frase) de teste.

c. Utilize o método ***predict*** da classe **MultinomialNB** e preveja a qual classe as críticas de teste pertencem.

d. Utilize o método **predict_proba** da classe **MultinomialNB** para imprimir as probabilidades encontradas e confira se elas são iguais às que você encontrou manualmente no item (A.a).

B. Para o classificador **BernoulliNB**, cada crítica, r_i, é representada como um vetor binário com 4 elementos, onde cada elemento define a presença ou ausência de cada palavra do alfabeto ***V*** na crítica r_i. Os dados de treinamento são apresentados a seguir como uma matriz, onde cada linha representa uma crítica.

Review	**Vocabulary**				**Class**
	"good"	**"poor"**	**"boring"**	**"great"**	
r1	1	0	0	1	Positive (+)
r2	0	1	1	1	Positive (+)
r3	1	0	1	1	Positive (+)
r4	1	1	1	0	Negative (-)
r5	1	1	0	1	Negative (-)
r6	0	1	1	0	Negative (-)

Agora, de posse dessas informações faça o seguinte:

a. Classifique **manualmente** as críticas de teste a seguir em **Positive** ou **Negative**. Apresente todos os cálculos necessários para se decidir a classe das 3 críticas de teste.

i. r_1 =*A good direction. But boring locations. Overall a good movie.*

ii. r_2 =*A good, good plot and great characters, but poor acting.*

iii. r_3 =*Good plot, but poor acting.*

b. Utilizando a classe **BernoulliNB** do scikit-learn, instancie e treine um classificador para decidir a classe dos documentos de teste. **OBS**.: Não é

necessário utilizar a classe **CountVectorizer**. Para o treinamento, simplesmente use a matriz de treinamento dada acima com as ocorrências das palavras do vocabulário, em seguida, passe a matriz para o classificador (i.e., para o método fit) juntamente com os labels, **'Positive'** e '**Negative**', correspondentes a cada crítica da matriz de treinamento. Para o teste, crie uma matriz onde cada linha contém a ocorrência ou não (i.e., 1 ou 0) das palavras do vocabulário para cada crítica (i.e., frase) de teste.

c. Utilize o método ***predict*** da classe **BernoulliNB** e preveja a qual classe as críticas de teste pertencem.
d. Utilize o método **predict_proba** da classe **BernoulliNB** para imprimir as probabilidades encontradas e confira se elas são iguais às que você encontrou manualmente no item (B.a).

OBSERVAÇÃO: Utilize a ***suavização de Laplace***, também conhecida como ***suavização adicione 1*** [1,2], quando vocês forem calcular as probabilidades condicionais. A suavização é diferente para os 2 classificadores estudados, i.e., **MultinomialNB** e **BernoulliNB**.

- **Suavização de Laplace para o caso do MultinomialNB**: Com a suavização as probabilidades condicionais se tornam

$$P(w_k \mid C_q) = \frac{contagem(w_k, C_q) + 1}{\sum_{l=1}^{K} contagem(w_l, C_q) + |V|},$$

onde $contagem(w_k, C_q)$é número de vezes que a palavra $w_k$aparece entre todas as palavras que pertencem à classeC_q, $\sum_{l=1}^{K} contagem(w_l, C_q)$ é a soma total de palavras pertencentes à classe C_qe $|V|$é o tamanho do vocabulário, ou seja, o número de atributos.

- **Suavização de Laplace para o caso do BernoulliNB**: Para o caso do classificador **BernoulliNB**, utilizando-se a ***suavização de Laplace***, $P(w_k \mid C_q)$ é calculada como

$$P(w_k \mid C_q) = \frac{M_{t,C_q} + 1}{M_C + 2},$$

onde M_{t,C_q}é o número de mensagens de treinamento da classe C_q que contém o termo/palavra w_k, enquanto M_Cé o número total de mensagens de treinamento da categoria C_qe 2 pois existem dois casos a serem considerados para cada termo/palavra, i.e., ocorrência e não ocorrência.

Referências:

[1] ‘An Introduction to Naïve Bayes Classifier’, https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf
[2] ‘Additive smoothing’, https://en.wikipedia.org/wiki/Additive_smoothing

15. **Exercício sobre classificação naive bayes**: Considere um conjunto de documentos, D, onde cada um deles pertence a classe **Esportes** (S) ou **Informática** (I). Dado um conjunto de treinamento com 11 documentos, crie 2 classificadores, um Multinomial Naive Bayes (NB) e outro Bernoulli NB, para classificar documentos não rotulados como S ou I. Para este exercício, define-se o seguinte vocabulário com 8 palavras:

$$V = \begin{bmatrix} w_1 = \text{goal} \\ w_2 = \text{tutor} \\ w_3 = \text{variance} \\ w_4 = \text{speed} \\ w_5 = \text{drink} \\ w_6 = \text{defence} \\ w_7 = \text{performance} \\ w_8 = \text{field} \end{bmatrix}$$

A. Para o classificador **MultinomialNB**, cada documento, d_i, é representado como um vetor de 8 dimensões (ou elementos), onde cada elemento define a frequência de cada palavra do alfabeto Vno documento d_i. Os dados de treinamento são apresentados a seguir como uma matriz para cada uma das 2 classes, onde cada linha representa um vetor do documento.

$$M^{\text{Esporte}} = \begin{bmatrix} 2 & 0 & 0 & 0 & 1 & 2 & 3 & 1 \\ 0 & 0 & 1 & 0 & 2 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 2 & 1 & 0 \\ 1 & 0 & 0 & 2 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 0 & 1 & 0 & 1 & 3 \\ 0 & 0 & 1 & 2 & 0 & 0 & 2 & 1 \end{bmatrix}$$

$$M^{\text{Informática}} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Agora, de posse dessas informações faça o seguinte:

a. Classifique **manualmente** os documentos de teste a seguir em **Esporte** ou **Informática**. Apresente todos os cálculos necessários para se decidir a classe dos 2 documentos de teste.
   i. $d_1 = [2,\ 1,\ 0,\ 0,\ 1,\ 2,\ 0,\ 1]$
   ii. $d_2 = [0,\ 1,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0]$

b. Utilizando a classe **MultinomialNB** do scikit-learn, instancie e treine um classificador para decidir a classe dos documentos de teste. **OBS**.: Não é necessário utilizar a classe **CountVectorizer**, apenas junte as 2 matrizes de treinamento e as passe para o classificador juntamente com os labels, **‘Esporte’** e ‘**Informatica**’, correspondentes a cada documento.

c. Utilize o método ***predict*** da classe **MultinomialNB** e preveja a qual classe os documentos de teste pertencem.
d. Utilize o método **predict_proba** da classe **MultinomialNB** para imprimir as probabilidades encontradas e confira se elas são iguais às que você encontrou manualmente no item (A.a).

B. Para o classificador **BernoulliNB**, cada documento, d_i, é representado como um vetor binário de 8 dimensões, onde cada elemento define a presença ou ausência de cada palavra do alfabeto Vno documento d_i. Os dados de treinamento são apresentados a seguir como uma matriz para cada uma das 2 classes, onde cada linha representa um vetor do documento.

$M^{\text{Esporte}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$	$M^{\text{Informática}} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$

Agora, de posse dessas informações faça o seguinte:

a. Classifique **manualmente** os documentos de teste a seguir em **Esporte** ou **Informática**. Apresente todos os cálculos necessários para se decidir a classe dos 2 documentos de teste.
   i. $d_1 = [1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 0,\ 1]$
   ii. $d_2 = [0,\ 1,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0]$
b. Utilizando a classe **BernoulliNB** do scikit-learn, instancie e treine um classificador para decidir a classe dos documentos de teste. **OBS**.: Não é necessário utilizar a classe **CountVectorizer**, apenas junte as 2 matrizes de treinamento e as passe para o classificador juntamente com os labels, **'Esporte'** e '**Informatica**', correspondentes a cada documento.
c. Utilize o método ***predict*** da classe **BernoulliNB** e preveja a qual classe os documentos de teste pertencem.
d. Utilize o método **predict_proba** da classe **BernoulliNB** para imprimir as probabilidades encontradas e confira se elas são iguais às que você encontrou manualmente no item (B.a).

OBSERVAÇÃO: Utilize a ***suavização de Laplace***, também conhecida como ***suavização adicione 1*** [1,2], quando vocês forem calcular as probabilidades condicionais. A suavização é diferente para os 2 classificadores estudados, i.e., MultinomialNB e BernoulliNB.

- **Suavização de Laplace para o caso do MultinomialNB**: Com a suavização as probabilidades condicionais se tornam

$$P(w_k \mid C_q) = \frac{contagem(w_k, C_q) + 1}{\sum_{l=1}^{K} contagem(w_l, C_q) + |V|},$$

onde $contagem(w_k, C_q)$é número de vezes que a palavra $w_k$aparece entre todas as palavras que pertencem à classeC_q, $\sum_{l=1}^{K} contagem(w_l, C_q)$ é a soma total de palavras pertencentes à classe C_qe $|V|$ é o tamanho do vocabulário, ou seja, o número de atributos.

- **Suavização de Laplace para o caso do BernoulliNB**: Para o caso do classificador **BernoulliNB**, utilizando-se a ***suavização de Laplace***, $P(w_k \mid C_q)$ é calculada como

$$P(w_k \mid C_q) = \frac{M_{t,C_q} + 1}{M_C + 2},$$

onde M_{t,C_q} é o número de mensagens de treinamento da classe C_q que contém o termo/palavra w_k, enquanto M_Cé o número total de mensagens de treinamento da categoria C_qe 2 pois existem dois casos a serem considerados para cada termo/palavra, i.e., ocorrência e não ocorrência.

Referências:

[1] ‘An Introduction to Naïve Bayes Classifier’, https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf

[2] ‘Additive smoothing’, https://en.wikipedia.org/wiki/Additive_smoothing