

Classificação de Textos com Algoritmos Tradicionais e BoW Features

> Ambiente

[] ↪ 2 cells hidden

✓ Dataset AmericanasBR

https://github.com/americanas-tech/b2w-reviews01/blob/main/b2wreviews01_stil2019.pdf

O dataset original foi simplificado, o campo 'overall_rating' foi usado como 'label':

- 4 e 5 representam positivo (label=1)
- 1 e 2 representam negativo (label=0)
- rating 3 não foi usado

O dataset resultante originou os dois conjuntos de dados:

- Treino: usando 5 mil instâncias de cada classe (positivo e negativo).
- Teste: usando 3 mil instâncias de cada classe excluindo-se instâncias de treino

Treino e Teste são disjuntos.

#baixando os dados de treino e teste

```
!curl https://www.inf.ufrgs.br/~viviane/DS/B2W-Reviews01_binario5000_TRAIN.csv > |
!curl https://www.inf.ufrgs.br/~viviane/DS/B2W-Reviews01_binario_TEST.csv > B2W-R
```

```

↻ % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
  Dload  Upload  Total    Spent    Left     Speed
100 1657k  100 1657k    0     0   815k      0  0:00:02  0:00:02 --:--:--  815k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
  Dload  Upload  Total    Spent    Left     Speed
100  981k  100  981k    0     0   671k      0  0:00:01  0:00:01 --:--:--  671k

```

```
df_train = pd.read_csv('B2W-Reviews01_binario5000_TRAIN.csv')
df_test = pd.read_csv('B2W-Reviews01_binario_TEST.csv')
```

✓ Explorando o dataset

```
df_train.sample(n=5)
```



	label	text	label_descr
4377	0	* Avarias no produto recebido * Dificil instal...	negativo
2429	0	Não pude avaliar o produto que deveria ter che...	negativo
7853	1	O melhor que já comprei! Preço justíssimo, poi...	positivo
8908	1	carrinho maravilhoso. Tenho ele ha 5 anos e co...	positivo
7212	1	Excelente, e chegou antes do prazo estabele...	positivo

```
#está balanceado e possui duas classes:
df_train.groupby('label_descr').count()
```



	label	text
label_descr		
negativo	5000	5000
positivo	5000	5000

```
df_test.sample(n=5)
```



	label	text	label_descr
4797	0	O produto tem tudo pra ser bom. O som é legal,...	negativo
3439	0	Não posso avaliar que não recebi! Infelizmente...	negativo
3353	0	realizei a compra e o vendedor não tinha o pr...	negativo
2832	1	O aparelho atende perfeitamente as minhas nece...	positivo
1528	1	Prático e ótimo custo benefício. Cumpre o que ...	positivo

```
#dataset de teste também está balanceado e possui duas classes:
df_test.groupby('label_descr').count()
```



	label	text
label_descr		
negativo	3000	3000
positivo	3000	3000

✓ Pré-processando

```
#função de pré-processamento
special_chars = "'!#$%&()*+,-./:;<=>?@[\\]^_`{|}~"
stop_words = stopwords.words('portuguese')

stop_words.remove('não') # mantém o não na lista de stopwords

def preprocess(x):
    new_x = x.replace("'", ' ')
    for c in special_chars:
        new_x = new_x.replace(c, ' ')
    new_x = ' '.join([word for word in nltk.word_tokenize(new_x.lower(), language=
    new_x = re.sub(r'[^\\w\\s]', ' ', new_x) #removendo pontuação do texto
    new_x = re.sub("http\\S+", ' ', new_x) # remove links
    new_x = re.sub("@\\w+", ' ', new_x) # remove contas com @
    new_x = re.sub("#\\S+", ' ', new_x) # hashtags
    new_x = re.sub('[0-9]+', ' ', new_x) # remove numeros e palavras com numeros
    new_x = unicode(new_x) #acentos
    new_x = re.sub("\\s+", ' ', new_x) # espaços
    new_x = new_x.strip()
    return new_x
```

```
#pré-processar datasets de treino e teste
df_train['text_original'] = df_train['text']
df_train['text'] = df_train['text'].apply(preprocess)
```

```
df_test['text_original'] = df_test['text']
df_test['text'] = df_test['text'].apply(preprocess)
```

```
df_train = df_train.sample(n=len(df_train)).copy() # embaralha treino
df_train.reset_index(drop=True, inplace=True)
len(df_train)
```

↗ 9935

```
# remove instâncias com texto com comprimento zero
df_train = df_train[df_train['text']!='']
df_train = df_train[~df_train['text'].isna()]
df_train.reset_index(drop=True, inplace=True) # reindexa dataframe
len(df_train)
```

↗ 9931

✓ Tarefa 1

1. **Verificação de Duplicidades Internas:** Verifique se existem sentenças duplicadas dentro dos conjuntos de treino e teste separadamente.
2. **Verificação de Duplicidades Entre Conjuntos:** Verifique se há sentenças duplicadas presentes tanto no conjunto de treino quanto no de teste.
3. **Remoção de Duplicidades:** Caso sejam encontradas duplicidades, remova-as tanto dentro de cada conjunto quanto entre os conjuntos de treino e teste.
4. **Análise de Impacto:** Apresente o impacto das remoções nos conjuntos de dados, destacando as alterações no tamanho e na distribuição das sentenças.

Modo estranho de usar o coalesce do C# em python

try:

 df_train_bkp

except NameError:

 df_train_bkp = df_train.copy()

try:

 df_test_bkp

except NameError:

 df_test_bkp = df_test.copy()

def redefinir_df():

 global df_train

 global df_test

 df_train = df_train_bkp.copy()

 df_test = df_test_bkp.copy()

def print_frase(frase):

 print("-----")

 print(frase)

 print("-----")

redefinir_df()

```
# [ SEU CÓDIGO AQUI ]
print(f"{df_train.duplicated().sum()} sentenças duplicadas em df_train")
print(f"{df_test.duplicated().sum()} sentenças duplicadas em df_test")
print(f"{pd.concat([df_train, df_test]).duplicated().sum()} sentenças duplicadas em df_concat")
print("-----")

print("Drop Duplicates")
df_train.drop_duplicates(inplace=True, keep='first')
df_test.drop_duplicates(inplace=True, keep='first')
print(f"{df_train.duplicated().sum()} sentenças duplicadas em df_train")
print(f"{df_test.duplicated().sum()} sentenças duplicadas em df_test")
print(f"{pd.concat([df_train, df_test]).duplicated().sum()} sentenças duplicadas em df_concat")
print("-----")
```

```
⇒ 65 sentenças duplicadas em df_train
   29 sentenças duplicadas em df_test
   136 sentenças duplicadas em df_train e df_test
   -----
   Drop Duplicates
   0 sentenças duplicadas em df_train
   0 sentenças duplicadas em df_test
   42 sentenças duplicadas em df_train e df_test
   -----
```

[SUA RESPOSTA AQUI]

Duplicados foram removidos porém ainda há 42 elementos em comum entre os grupos de treino e teste

```
0 sentenças duplicadas em df_train
0 sentenças duplicadas em df_test
42 sentenças duplicadas em df_train e df_test
-----
Drop Duplicates
0 sentenças duplicadas em df_train
0 sentenças duplicadas em df_test
42 sentenças duplicadas em df_train e df_test
-----
```

✓ Gerando Representação BoW com pesos TFIDF

Mais informações em https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer

# calcula features e os valores tfidf gerando os vetores dos documentos:
tfidf_vectorizer = TfidfVectorizer()

vetores_docs = tfidf_vectorizer.fit_transform(df_train['text'].values)
features = tfidf_vectorizer.get_feature_names_out()
print(features[1:100])
vetores_docs.shape, features.shape
```

```

→ ['000' '004' '004840' '0077' '008' '00x' '01' '02' '026' '03' '04' '05'
    '06' '07' '08' '080' '0800' '09' '0problemas' '10' '100' '1000' '10000'
    '1000w' '100kg' '100ml' '1020' '1056' '106' '1060' '1080p' '1080ti'
    '1090' '10a' '10cm' '10gb' '10h' '10minutos' '10mm' '10mp' '10muito'
    '10w' '10x' '10x10' '11' '110' '110kmh' '110v' '116' '11a' '11gb' '12'
    '120' '1200' '120km' '120v' '125' '1250ml' '127' '1276' '127v' '127w'
    '128gb' '1299' '12h' '12hs' '12mp' '12w' '12º' '13' '130' '133' '135g'
    '1360' '13912' '1399' '13gb' '13kg' '13mp' '14' '140' '1400' '142' '14cm'
    '14h' '15' '150' '1500ml' '1500w' '150kg' '152' '1599' '15cm' '16' '160'
    '1600' '165' '166' '1670']
((9935, 14527), (14527,))

print(tfidf_vectorizer.vocabulary_)
```

```

→ {'meu': 8996, 'produto': 11090, 'foi': 6677, 'faturado': 6396, 'no': 9579, 'd:
```

```
# # opções de préprocessamento que sobrescrevem funcoes do tfidftokenizer:

# tfidf_vectorizer = TfidfVectorizer(input='filename', max_features=200,
#                                   token_pattern='(?u)\\b[a-zA-Z]\\w{2,}\\b',
#                                   max_df=0.05,
#                                   stop_words='english',
#                                   ngram_range=(1, 3))

# def meu_preprocessamento(doc):
#     # tokeniza com algum tokenizador
#     # adiciona funcoes de préprocessamento
#     return doc
#
# tfidf = TfidfVectorizer(
#     analyzer='word',
#     tokenizer=meu_tokenizador,
#     preprocessor=meu_tokenizador,
#     token_pattern=None)
```

> Analisando os vetores gerados

[] ↳ 10 cells hidden

> Visualizando TFIDF

[] ↳ 8 cells hidden

✓ Treinando Modelo de Classificação


```
df_train.groupby(['label_descr', 'label']).count()
```



		text
label_descr	label	
negativo	0	4947
positivo	1	4988



```
# Gerando a representação vetorial para os textos da base de treino
vectorizer = TfidfVectorizer()
#fit_transform ajusta o vetorizador tfidf à base de treino e também transforma o
X_train_svc = vectorizer.fit_transform(list(df_train['text']))
y_train_svc = np.array(df_train['label'])
```

```
#efetuar o treinamento usando parâmetros predeterminados
#clf = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
#efetuar o treinamento usando parâmetros default
clf_svc = svm.SVC()
clf_svc.fit(X_train_svc, y_train_svc)
```



▼ SVC ⓘ ?

SVC()

```
# efetuar o treinamento fazendo refit com a melhor configuração:
#model = svm.SVC()
#parameters = {'kernel':['linear','rbf'], 'C':[1, 5]} # neste caso estaremos vari
#clf = GridSearchCV(estimator=model, param_grid=parameters, cv=5, verbose=4, scor
#clf.fit(X, y)
#print(f"Para {model} melhor score {clf.best_score_:.3f} para os seguintes parâme
```

✓ Prevendo a classe das instâncias de teste

```
# Gerando a representação vetorial para os textos da base de teste
X_true_svc = vectorizer.transform(df_test['text'].values) #somente transform usando
y_true_svc = df_test['label'].values
# gera as predições para os dados de teste:
y_pred_svc = clf_svc.predict(X_true_svc)
```

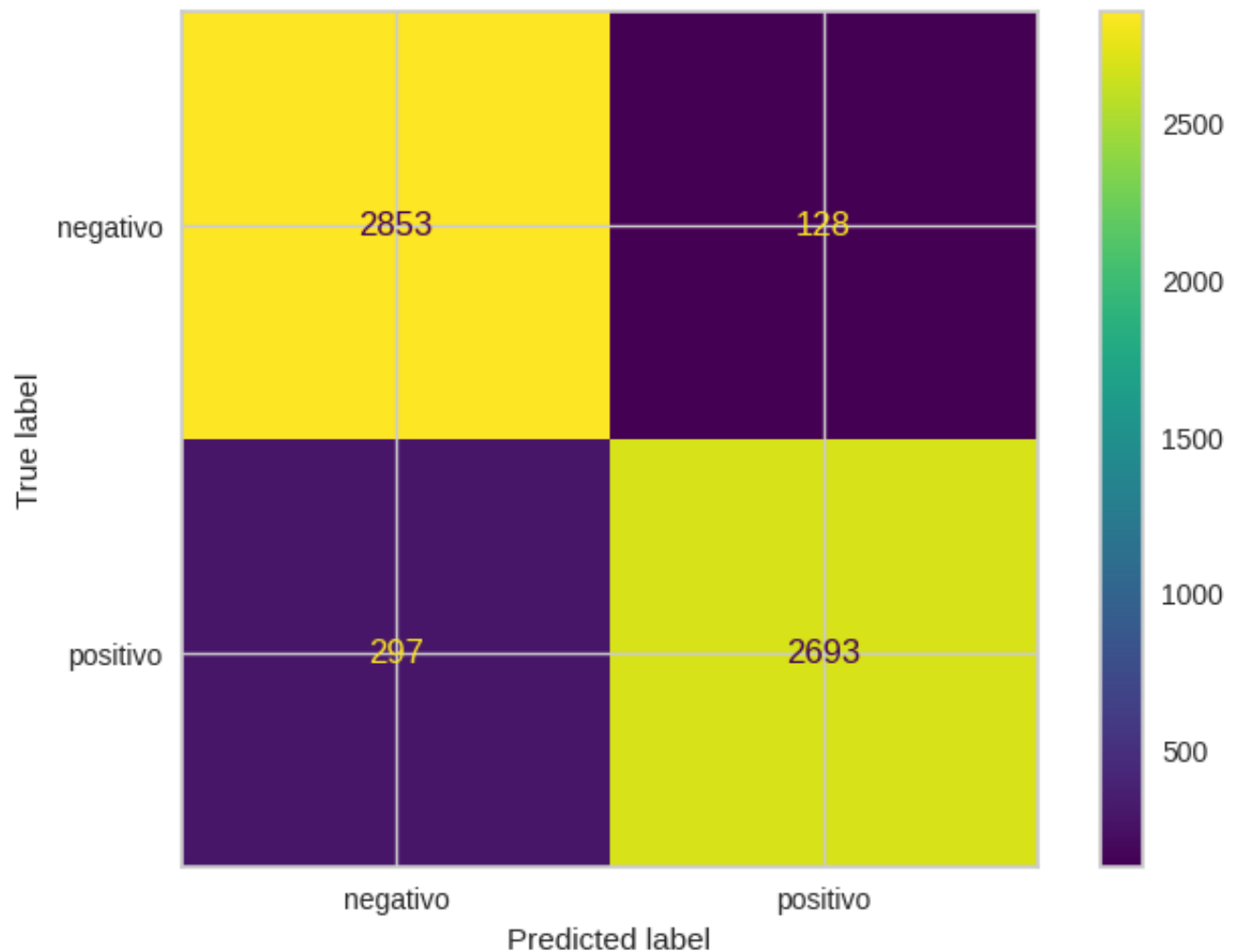
```
y_pred_svc[0:15]
```

```
→ array([1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1])
```

✓ Avaliando a Qualidade do Modelo

```
print(f"Acurácia: {accuracy_score(y_true_svc, y_pred_svc):.4f}")
print(f"F1-macro: {f1_score(y_true_svc, y_pred_svc, average='macro'):.4f}")
cm = confusion_matrix(y_true_svc, y_pred_svc)
cm_display = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = list(df
cm_display.plot()
plt.show()
```

→ Acurácia: 0.9288
F1-macro: 0.9288



```
# erros = list(zip(df_test['text'].values, df_test['text_original'].values, y_true))
erros_svc = list(zip(df_test['text'].values, df_test['label_descr'].values, y_true))

erros_svc = [item for item in erros_svc if item[2] != item[3]] #removendo as instâncias corretas
df_erros_svc = pd.DataFrame(erros_svc, columns = ['Texto', 'Original', 'True', 'Pred'])
```

```
## acrescentando colunas FP e FN no dataframe com os erros
df_erros_svc['FP'] = df_erros_svc.apply(lambda x: 1 if ((x['Pred']==1) & (x['True']==0)) else 0)
df_erros_svc['FN'] = df_erros_svc.apply(lambda x: 1 if ((x['Pred']==0) & (x['True']==1)) else 0)
print('Há ', len(df_erros_svc), ' instâncias mal classificadas.')
```

➡ Há 425 instâncias mal classificadas.

```
# Checando FN e FP
df_erros_svc[df_erros_svc['FN']==1]
```



	Texto	Original	True	Pred	FP	FN	
0	Chegou antes do prazo, a caneta é fininha e na...	positivo	1	0	0	1	
1	Comprei entrega a jato para entregar em 1 dia ...	positivo	1	0	0	1	
2	Funciona mesmo. Faz pipoca rápida, saudável e ...	positivo	1	0	0	1	
3	Eu fiquei apaixonada pelo anúncio, mas quando ...	positivo	1	0	0	1	
4	melhor que 50 tons , pô vocês mandam avaliar m...	positivo	1	0	0	1	
...	
292	A água da torneira de Porto Alegre/RS está com...	positivo	1	0	0	1	
293	A bicicleta é muito boa porém não pega câmbio ...	positivo	1	0	0	1	
294	melhor pc ever compreee.Oque se ta esperando v...	positivo	1	0	0	1	
295	Apesar da NVI não ser a minha tradução preferi...	positivo	1	0	0	1	
296	O produto é de qualidade e muito bom , porém o...	positivo	1	0	0	1	

297 rows x 6 columns

```
df_erros_svc[df_erros_svc['FP']==1]
```



	Texto	Original	True	Pred	FP	FN	
297	A piscina só enche com apoio pois as bordas sã...	negativo	0	1	1	0	
298	Meu bebê tem 3 anos e no segundo dia que ganho...	negativo	0	1	1	0	
299	Um excelente produto, ótimo preço, comprei par...	negativo	0	1	1	0	
300	Muito grande e muito pesado. Dei de presente a...	negativo	0	1	1	0	
301	Travesseiro muito alto, não funciona para bebê...	negativo	0	1	1	0	
...	
420	O equipamento é bom para atividades normais, m...	negativo	0	1	1	0	
421	MUITO DEMORADA A ENTREGA MILHARES DE FORNECED...	negativo	0	1	1	0	
422	O produto eh lindo. O problema foi a entrega. ...	negativo	0	1	1	0	
423	Gostei câmera, designer e ótimo ,não trava o q...	negativo	0	1	1	0	
424	Prnsei que fosse de vidro e qualidade bem ruim...	negativo	0	1	1	0	

```
#salvando csv com as instâncias mal classificadas
from google.colab import drive
drive.mount('/content/drive')
path = '/content/drive/MyDrive/Colab Notebooks/erros_americanas_svc.csv'
df_erros_svc.to_csv(path)
```

 Drive already mounted at /content/drive; to attempt to forcibly remount, call

✓ Tarefa 2

1. **Treinamento com Naive Bayes:** Realize o treinamento do corpus utilizando o algoritmo Naive Bayes.
2. **Comparação com SVM:** Compare os resultados obtidos com o classificador SVM treinado no mesmo corpus.
3. **Análise de Desempenho:** Avalie as diferenças de desempenho entre os dois classificadores com base nas seguintes análises:
 - Matriz de confusão
 - Métricas de F1-score
 - Acurácia
4. **Discussão das Diferenças:** Identifique e discuta as principais diferenças observadas nos resultados.

[SUA RESPOSTA AQUI]

```
# Separando X Y
X_train_nb = vectorizer.fit_transform(list(df_train['text']))
y_train_nb = np.array(df_train['label'])
```

```
# Classifier
clf_nb = naive_bayes.MultinomialNB()
clf_nb.fit(X_train_nb, y_train_nb)
```



▼ MultinomialNB ⓘ ?

MultinomialNB()

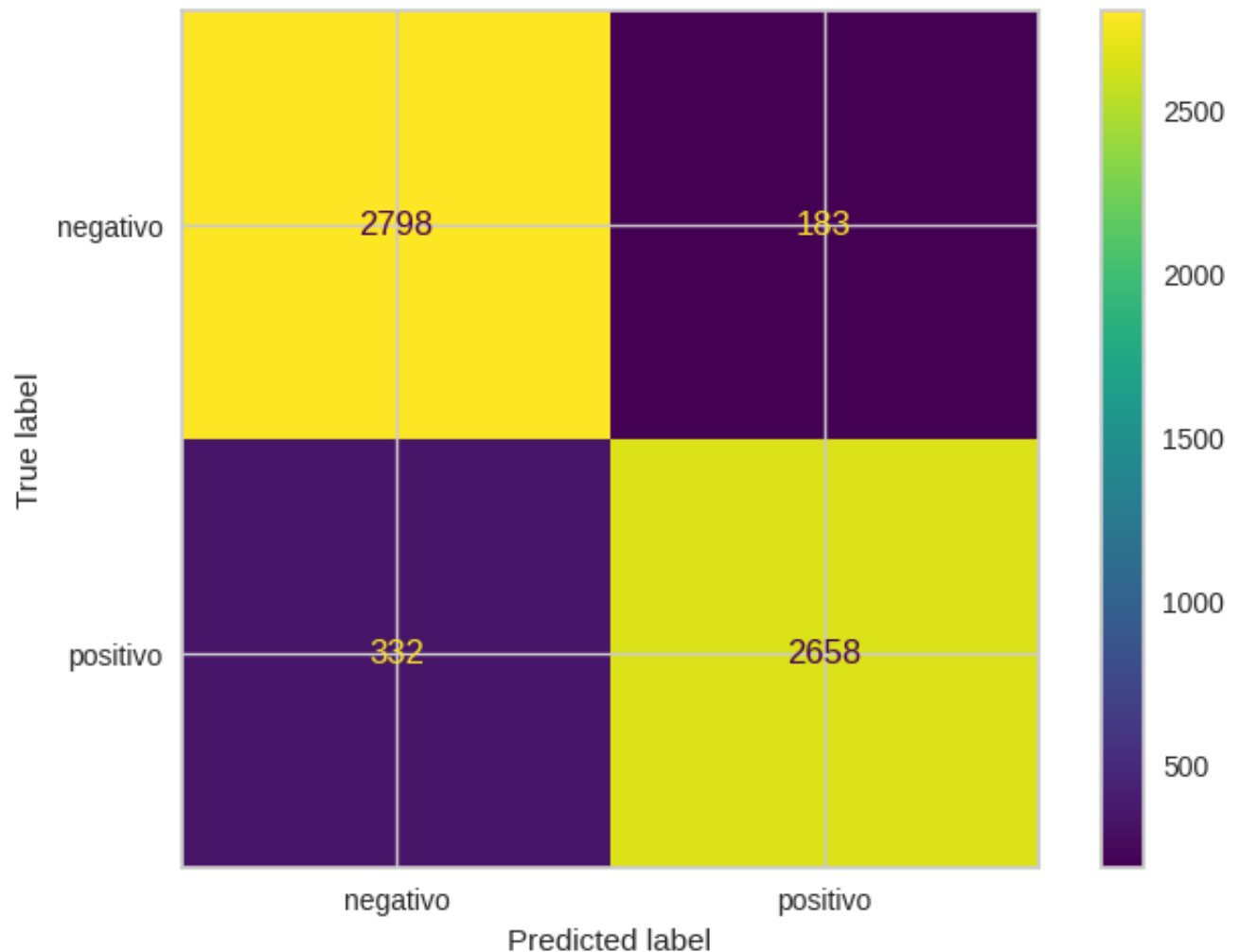
```
# Prevendo a classe
```

```
X_true_nb = vectorizer.transform(df_test['text'].values)  
y_true_nb = df_test['label'].values
```

```
y_pred_nb = clf_nb.predict(X_true_nb)
```

```
## Avaliando
print(f"Acurácia: {accuracy_score(y_true_nb, y_pred_nb):.4f}")
print(f"F1-macro: {f1_score(y_true_nb, y_pred_nb, average='macro'):.4f}")
cm = confusion_matrix(y_true_nb, y_pred_nb)
cm_display = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = list(
cm_display.plot()
plt.show()
```

→ Acurácia: 0.9137
F1-macro: 0.9137



✓ Examinando os as instâncias mal classificadas


```
# erros = list(zip(df_test['text'].values, df_test['text_original'].values, y_true))
erros_nb = list(zip(df_test['text'].values, df_test['label_descr'].values, y_true))

#removendo as instâncias corretas
erros_nb = [item for item in erros_nb if item[2] != item[3]]

#gerando um dataframe para ficar mais fácil de trabalhar
df_erros_nb = pd.DataFrame(erros_nb, columns = ['Texto', 'Original', 'True', 'Pred'])

#acrescentando colunas FP e FN no dataframe com os erros
df_erros_nb['FP'] = df_erros_nb.apply(lambda x: 1 if ((x['Pred']==1) & (x['True']!=1)) else 0)
df_erros_nb['FN'] = df_erros_nb.apply(lambda x: 1 if ((x['Pred']==0) & (x['True']==1)) else 0)

print('Há ', len(df_erros_nb), ' instâncias mal classificadas.')
```

→ Há 425 instâncias mal classificadas.


```
df_erros_nb[df_erros_nb['FN']==1]
```


→

	Texto	Original	True	Pred	FP	FN	
0	Chegou antes do prazo, a caneta é fininha e na...	positivo	1	0	0	1	
1	Comprei entrega a jato para entregar em 1 dia ...	positivo	1	0	0	1	
2	Funciona mesmo. Faz pipoca rápida, saudável e ...	positivo	1	0	0	1	
3	Eu fiquei apaixonada pelo anúncio, mas quando ...	positivo	1	0	0	1	
4	melhor que 50 tons , pô vocês mandam avaliar m...	positivo	1	0	0	1	
...	
292	A água da torneira de Porto Alegre/RS está com...	positivo	1	0	0	1	
293	A bicicleta é muito boa porém não pega câmbio ...	positivo	1	0	0	1	
294	melhor pc ever compreee.Oque se ta esperando v...	positivo	1	0	0	1	
295	Apesar da NVI não ser a minha tradução preferi...	positivo	1	0	0	1	
296	O produto é de qualidade e muito bom , porém o...	positivo	1	0	0	1	

297 rows x 6 columns

```
df_erros_nb[df_erros_nb['FP']==1]
```



	Texto	Original	True	Pred	FP	FN	
297	A piscina só enche com apoio pois as bordas sã...	negativo	0	1	1	0	
298	Meu bebê tem 3 anos e no segundo dia que ganho...	negativo	0	1	1	0	
299	Um excelente produto, ótimo preço, comprei par...	negativo	0	1	1	0	
300	Muito grande e muito pesado. Dei de presente a...	negativo	0	1	1	0	
301	Travesseiro muito alto, não funciona para bebê...	negativo	0	1	1	0	
...	
420	O equipamento é bom para atividades normais, m...	negativo	0	1	1	0	
421	MUITO DEMORADA A ENTREGA MILHARES DE FORNECED...	negativo	0	1	1	0	
422	O produto eh lindo. O problema foi a entrega. ...	negativo	0	1	1	0	
423	Gostei câmera, designer e ótimo ,não trava o q...	negativo	0	1	1	0	
424	Prnsei que fosse de vidro e qualidade bem ruim...	negativo	0	1	1	0	

```
# #salvando csv com as instâncias mal classificadas
# from google.colab import drive
# drive.mount('/content/drive')
path = '/content/drive/MyDrive/Colab Notebooks/erros_americanas_nb.csv'
df_erros_nb.to_csv(path)
```

✓ Tarefa 3

1. **Análise de Erros do Naive Bayes:** Realize uma análise detalhada dos erros cometidos pelo classificador Naive Bayes, seguindo o mesmo procedimento adotado anteriormente para o SVM.
2. **Verificação de Interseção de Erros:** Verifique se há interseção entre as classificações incorretas de ambos os classificadores (Naive Bayes e SVM), use o diagrama de Venn como apoio visual. Identifique quais frases foram classificadas de forma errada por ambos.
3. **Comparação de Resultados:** Analise se há alguma característica distinta nos erros de cada classificador. Discuta se o tipo de erro cometido pelo Naive Bayes difere dos erros cometidos pelo SVM e, se sim, explore as possíveis causas dessas diferenças.

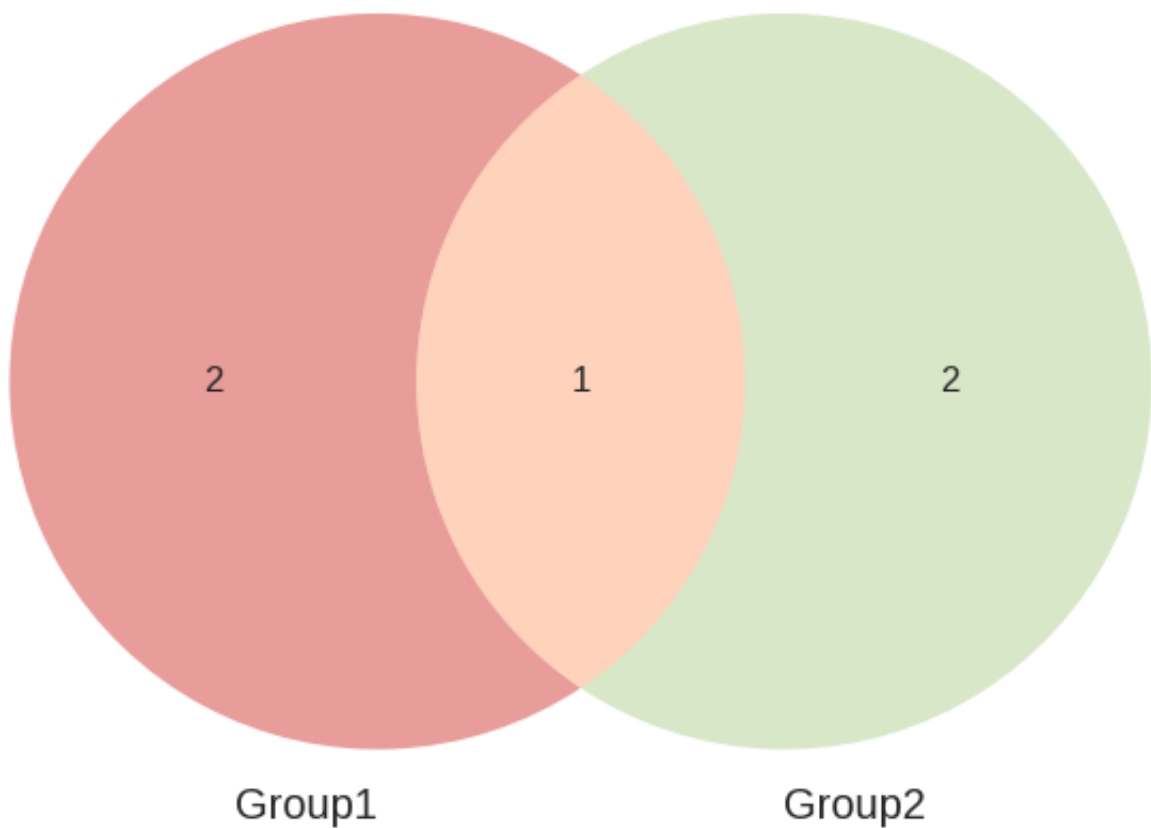
```
# Exemplo de código para a criação do diagram de Venn
```

```
import matplotlib.pyplot as plt  
from matplotlib_venn import venn2
```

```
set1 = set(['A', 'B', 'C'])  
set2 = set(['A', 'E', 'F'])
```

```
venn2([set1, set2], ('Group1', 'Group2'))
```

```
plt.show()
```



[SUA RESPOSTA AQUI]

```
df_erros_svc = pd.DataFrame(erros_svc, columns=["texto", "label_descr", "real", "previsto"])
print(df_erros_svc.head())
```

```
df_erros_nb = pd.DataFrame(erros_nb, columns=["texto", "label_descr", "real", "previsto"])
print(df_erros_nb.head())
```

```
↔
```

	texto	label_descr	real	\
0	Chegou antes do prazo, a caneta é fininha e na...	positivo	1	
1	Comprei entrega a jato para entregar em 1 dia ...	positivo	1	
2	Funciona mesmo. Faz pipoca rápida, saudável e ...	positivo	1	
3	Eu fiquei apaixonada pelo anúncio, mas quando ...	positivo	1	
4	melhor que 50 tons , pô vocês mandam avaliar m...	positivo	1	

	previsto
0	0
1	0
2	0
3	0
4	0

	texto	label_descr	real	\
0	Chegou antes do prazo, a caneta é fininha e na...	positivo	1	
1	Comprei entrega a jato para entregar em 1 dia ...	positivo	1	
2	Funciona mesmo. Faz pipoca rápida, saudável e ...	positivo	1	
3	Eu fiquei apaixonada pelo anúncio, mas quando ...	positivo	1	
4	melhor que 50 tons , pô vocês mandam avaliar m...	positivo	1	

	previsto
0	0
1	0
2	0
3	0
4	0

```
df_erros_nb.iloc[0]['texto'] == df_erros_svc.iloc[0]['texto']
```

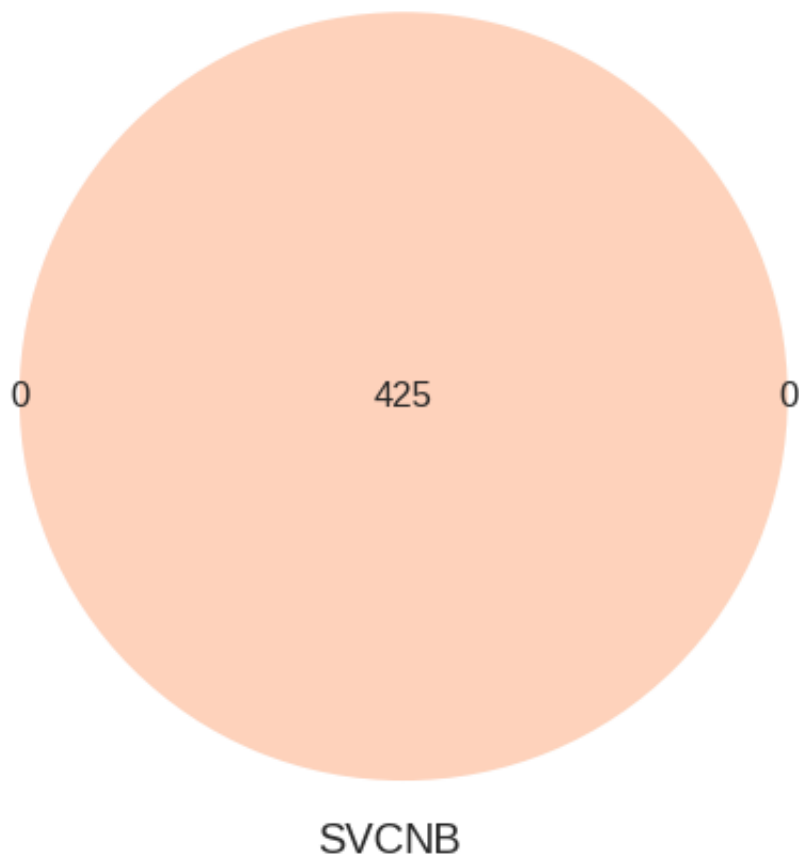
```
for i in range(len(df_erros_svc)):
    if df_erros_nb.iloc[i]['texto'] != df_erros_svc.iloc[i]['texto']:
        print(i)
        print(df_erros_nb.iloc[i]['texto'])
        print(df_erros_svc.iloc[i]['texto'])
```

```
# [ SEU CÓDIGO AQUI ]

set_errors_svc = set(df_erros_svc['texto'])
set_errors_nb = set(df_erros_nb['texto'])

venn2([set_errors_svc, set_errors_nb], ('SVC', 'NB'))


plt.show()
```



```

x = 10
for item in set_errors_svc.intersection(set_errors_nb):
    print(f">> {item}\n")
    x = x - 1
    if x == 0:
        break

```

 >> Malala Yousafzai, membro de uma tribo patchum, mora no vale do Swat, no Pa
 >> Agradei do aparelho. Vi alguns tutoriais na internet e acabei aproveitando
 >> Já possuo 1 máquina igual a esta e sei que o produto é bom, porém a loja nã
 >> Pedi diversos relógios e as romns desse e diferentes das outras do msm mode
 >> O produto tem tudo pra ser bom. O som é legal, mas um ruído chato fica inc
 >> Vale o investimento, faz o que promete para um produto de baixo valor agre
 >> gostei muito, achei bem pratica, além de não sujar panela, tb não utiliza
 >> Comprei para meu controle do Xbox one leva cerca de 12 a 15 horas para rec
 >> Adquiri o produto há duas semanas e estou viciado nele. Não sei como vivia
 >> COMPREI UM DESSE EM DEZEMBRO DE 2017, NUNCA CHEGOU! JA LANÇARAM OUTRO, JA

Resposta

Aparentemente os mesmos erros foram capturados com o classificador SVC e Naive Bayes

> O problema das palavras fora do vocabulário OOV

[] ↪ 6 cells hidden

