Naive Bayes NB

é um classificador probabilístico que utiliza o Teorema de Bayes para prever a probabilidade de uma instância pertencer a uma determinada classe. Ele é chamado de “naive” (ingênuo) porque assume que todas as características (ou atributos) são independentes entre si, o que raramente é verdade na prática, mas simplifica muito os cálculos.

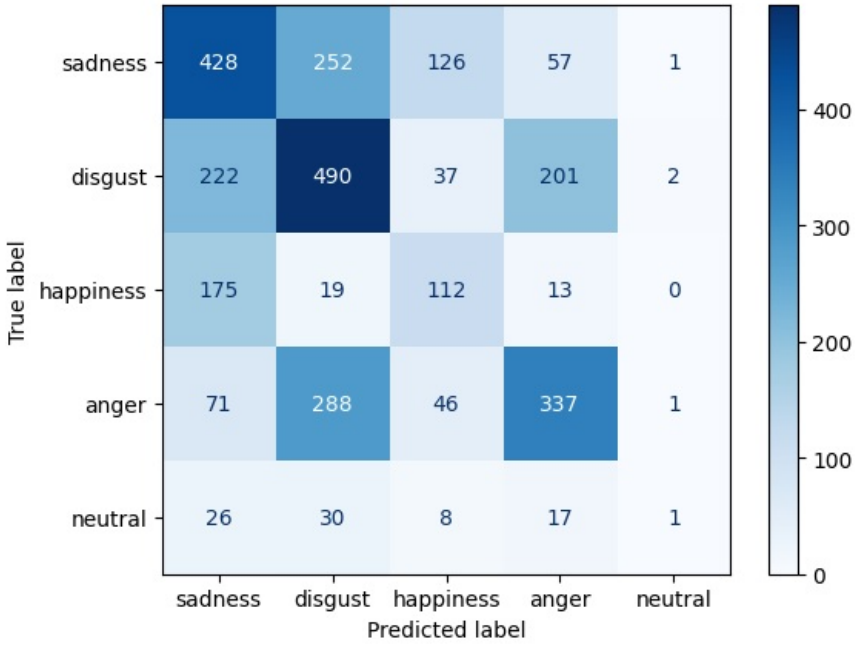
In [175…
```
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
dados_reviews = pd.read_csv('dados_reviews_tratados.csv', sep = ',')
dados_reviews['content'] = dados_reviews['content'].fillna('')
dados_reviews = dados_reviews[~dados_reviews['sentiment'].isin(['surprise', 'fear'])]
```

In [176…
```
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
from sklearn.model_selection import cross_val_score, cross_val_predict, StratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
from sklearn.naive_bayes import MultinomialNB

# Vectorização com Bag-of-Words
vectorizer_bow = CountVectorizer()
BoW_matriz = vectorizer_bow.fit_transform(dados_reviews['content'])
palavras_bow = vectorizer_bow.get_feature_names_out()
BOW_dataframe = pd.DataFrame(BoW_matriz.toarray(), columns=palavras_bow)

# Vectorização com TF-IDF
vectorizer_tfidf = TfidfVectorizer()
tfidf_matrix = vectorizer_tfidf.fit_transform(dados_reviews['content'])
palavras_tfidf = vectorizer.get_feature_names_out()
TFIDF_dataframe = pd.DataFrame(tfidf_matrix.toarray(), columns=palavras_tfidf)

# Definir as categorias
y = dados_reviews['sentiment']
y_polaridade = dados_reviews['sentiment_polarity']

# Configurar a avaliação cruzada
cv = StratifiedKFold(n_splits=5)


naive_bayes = MultinomialNB()
```

Validação Cruzada Sentimento BoW

In [177…
```
scores = cross_val_score(naive_bayes, BoW_matriz, y, cv=cv, scoring='accuracy')
print("Validação cruzada para Sentiment:", scores)
print("Média dos Scores:", scores.mean())

# Obter previsões de validação cruzada
predictions_BoW = cross_val_predict(naive_bayes, BoW_matriz, y, cv=cv)
print("Relatório de Classificação para Sentiment:")
print(classification_report(y, predictions_BoW, zero_division=0))

# matriz confusão
conf_matrix = confusion_matrix(y, predictions_BoW, labels=y.unique())
disp = ConfusionMatrixDisplay(conf_matrix, display_labels=y.unique())
disp.plot(cmap='Blues')
print("Matriz de Confusão para Sentiment:")
print(conf_matrix)
```

```
Validação cruzada para Sentiment: [0.38513514 0.49662162 0.39527027 0.45608108 0.5777027 ]
Média dos Scores: 0.4621621621621622
Relatório de Classificação para Sentiment:
              precision    recall  f1-score   support

       anger       0.54      0.45      0.49       743
     disgust       0.45      0.51      0.48       952
   happiness       0.34      0.35      0.35       319
     neutral       0.20      0.01      0.02        82
     sadness       0.46      0.50      0.48       864

    accuracy                           0.46      2960
   macro avg       0.40      0.37      0.36      2960
weighted avg       0.46      0.46      0.46      2960

Matriz de Confusão para Sentiment:
[[428 252 126  57   1]
 [222 490  37 201   2]
 [175  19 112  13   0]
 [ 71 288  46 337   1]
 [ 26  30   8  17   1]]
```

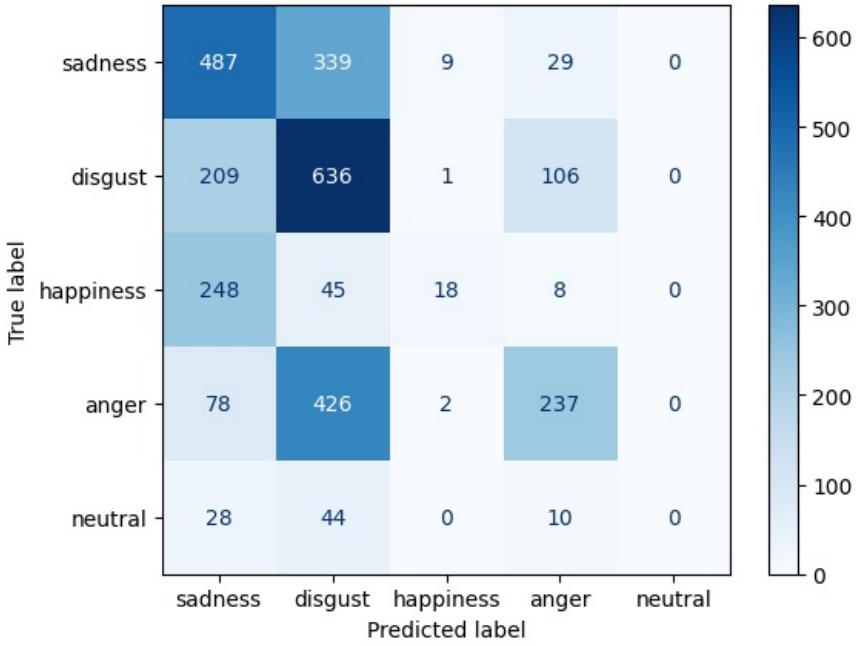


Validação Cruzada Sentimento TF-IDF

In [178…

```python
scores = cross_val_score(naive_bayes, tfidf_matrix, y, cv=cv, scoring='accuracy')
print("Validação cruzada para Sentiment:", scores)
print("Média dos Scores:", scores.mean())

# Obter previsões de validação cruzada
predictions = cross_val_predict(naive_bayes, tfidf_matrix, y, cv=cv)
print("Relatório de Classificação para Sentiment:")
print(classification_report(y, predictions, zero_division=0))

# matriz confusão
conf_matrix = confusion_matrix(y, predictions, labels=y.unique())
disp = ConfusionMatrixDisplay(conf_matrix, display_labels=y.unique())
disp.plot(cmap='Blues')
print("Matriz de Confusão para Sentiment:")
print(conf_matrix)
```

```
Validação cruzada para Sentiment: [0.50506757 0.47972973 0.39695946 0.44932432 0.49662162]
Média dos Scores: 0.4655405405405405
Relatório de Classificação para Sentiment:
              precision    recall  f1-score   support

       anger       0.61      0.32      0.42       743
     disgust       0.43      0.67      0.52       952
   happiness       0.60      0.06      0.10       319
     neutral       0.00      0.00      0.00        82
     sadness       0.46      0.56      0.51       864

    accuracy                           0.47      2960
   macro avg       0.42      0.32      0.31      2960
weighted avg       0.49      0.47      0.43      2960

Matriz de Confusão para Sentiment:
[[487 339   9  29   0]
 [209 636   1 106   0]
 [248  45  18   8   0]
 [ 78 426   2 237   0]
 [ 28  44   0  10   0]]
```

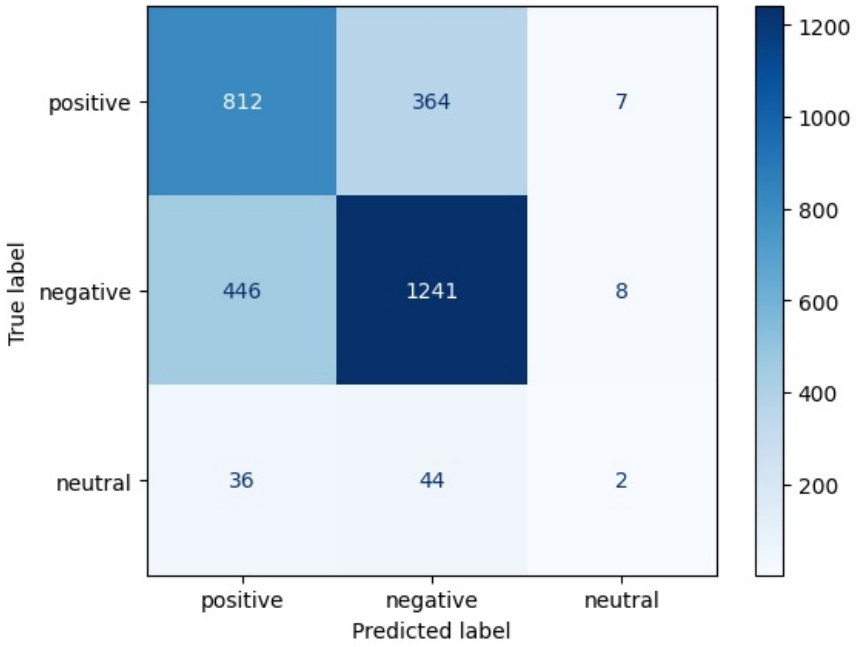


Validação Cruzada Polaridade BoW

In [179…

```python
# Avaliação cruzada para Sentiment Polarity
scores_polaridade = cross_val_score(naive_bayes, BoW_matriz, y_polaridade, cv=cv, scoring='accuracy')
print("Validação cruzada para Sentiment Polarity:", scores_polaridade)
print("Média dos Scores:", scores_polaridade.mean())

# Obter previsões de validação cruzada para polaridade
predictions_polaridade_BoW = cross_val_predict(naive_bayes, BoW_matriz, y_polaridade, cv=cv)
print("Relatório de Classificação para Sentiment Polarity:")
print(classification_report(y_polaridade, predictions_polaridade_BoW, zero_division=0))

# Gerar e mostrar a matriz de confusão para Sentiment Polarity
conf_matrix_polaridade = confusion_matrix(y_polaridade, predictions_polaridade_BoW, labels=y_polaridade.unique(
disp_polaridade = ConfusionMatrixDisplay(conf_matrix_polaridade, display_labels=y_polaridade.unique())
disp_polaridade.plot(cmap='Blues')
print("Matriz de Confusão para Sentiment Polarity:")
print(conf_matrix_polaridade)
```

```
Validação cruzada para Sentiment Polarity: [0.62331081 0.77027027 0.61993243 0.67905405 0.77871622]
Média dos Scores: 0.6942567567567568
Relatório de Classificação para Sentiment Polarity:
              precision    recall  f1-score   support

    negative       0.75      0.73      0.74      1695
     neutral       0.12      0.02      0.04        82
    positive       0.63      0.69      0.66      1183

    accuracy                           0.69      2960
   macro avg       0.50      0.48      0.48      2960
weighted avg       0.69      0.69      0.69      2960

Matriz de Confusão para Sentiment Polarity:
[[ 812  364    7]
 [ 446 1241    8]
 [  36   44    2]]
```

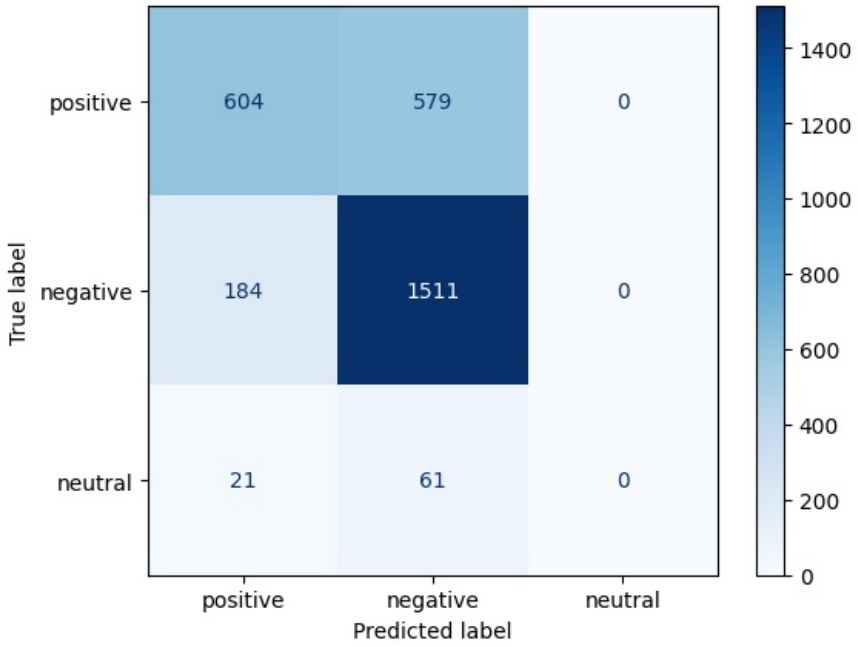


Validação Cruzada Polaridade TF-IDF

In [180…

```
# Avaliação cruzada para Sentiment Polarity
scores_polaridade = cross_val_score(naive_bayes, tfidf_matrix, y_polaridade, cv=cv, scoring='accuracy')
print("Validação cruzada para Sentiment Polarity:", scores_polaridade)
print("Média dos Scores:", scores_polaridade.mean())

# Obter previsões de validação cruzada para polaridade
predictions_polaridade_tfidf = cross_val_predict(naive_bayes, tfidf_matrix, y_polaridade, cv=cv)
print("Relatório de Classificação para Sentiment Polarity:")
print(classification_report(y_polaridade, predictions_polaridade_tfidf, zero_division=0))

# Gerar e mostrar a matriz de confusão para Sentiment Polarity
conf_matrix_polaridade = confusion_matrix(y_polaridade, predictions_polaridade_tfidf, labels=y_polaridade.unique
disp_polaridade = ConfusionMatrixDisplay(conf_matrix_polaridade, display_labels=y_polaridade.unique())
disp_polaridade.plot(cmap='Blues')
print("Matriz de Confusão para Sentiment Polarity:")
print(conf_matrix_polaridade)
```

```
Validação cruzada para Sentiment Polarity: [0.71283784 0.78040541 0.65033784 0.71114865 0.71790541]
Média dos Scores: 0.714527027027027
Relatório de Classificação para Sentiment Polarity:
              precision    recall  f1-score   support

    negative       0.70      0.89      0.79      1695
     neutral       0.00      0.00      0.00        82
    positive       0.75      0.51      0.61      1183

    accuracy                           0.71      2960
   macro avg       0.48      0.47      0.46      2960
weighted avg       0.70      0.71      0.69      2960

Matriz de Confusão para Sentiment Polarity:
[[ 604  579    0]
 [ 184 1511    0]
 [  21   61    0]]
```



Teste com avalições da Google Play

In [181…
```
naive_bayes_BoW = MultinomialNB()
naive_bayes_polaridade_BoW = MultinomialNB()
naive_bayes_BoW.fit(BOW_dataframe, y)
naive_bayes_polaridade_BoW.fit(BOW_dataframe, y_polaridade)
naive_bayes_tfidf = MultinomialNB()
naive_bayes_polaridade_tfidf = MultinomialNB()
naive_bayes_tfidf.fit(TFIDF_dataframe, y)
naive_bayes_polaridade_tfidf.fit(TFIDF_dataframe, y_polaridade)
```

Out[181…
```
MultinomialNB
MultinomialNB()
```

In [182…
```
import pandas as pd
teste_emocoes = pd.read_csv('teste_tratado.csv', sep = ',')
teste_emocoes.head(1)
```

Out[182…

	numero	content	sentiment_polarity	sentiment	app
0	1	último pedir app realmente último pra pe...	negative	disgust	iFood

In [183…
```
avaliacao_BoW = vectorizer_bow.transform(teste_emocoes['content'])
avaliacao_tfidf = vectorizer_tfidf.transform(teste_emocoes['content'])
```

Teste Naive Bayes com BoW sentimento

In [184…
```
emocao_predita = naive_bayes_BoW.predict(avaliacao_BoW)
print('emocao predita:')
print(emocao_predita)
print('emocao real:')
print(list(teste_emocoes['sentiment']))
```

```
emocao predita:
['anger' 'disgust' 'sadness' 'sadness' 'happiness' 'sadness' 'anger'
 'happiness' 'happiness']
emocao real:
['disgust', 'sadness', 'happiness', 'happiness', 'anger', 'sadness', 'anger', 'happiness', 'happiness']
/home/arthurwsl/classificao_textos/Classificacao_textos/myenv/lib/python3.12/site-packages/sklearn/base.py:493:
UserWarning: X does not have valid feature names, but MultinomialNB was fitted with feature names
  warnings.warn(
```

Teste Naive Bayes com TF-IDF sentimento

In [185…
```
emocao_predita = naive_bayes_tfidf.predict(avaliacao_tfidf)
```

```
print('emocao predita:')
print(emocao_predita)
print('emocao real:')
print(list(teste_emocoes['sentiment']))
```

```
emocao predita:
['disgust' 'disgust' 'sadness' 'sadness' 'anger' 'sadness' 'disgust'
 'sadness' 'sadness']
emocao real:
['disgust', 'sadness', 'happiness', 'happiness', 'anger', 'sadness', 'anger', 'happiness', 'happiness']
```

Teste Naive Bayes com BoW polaridade

In [186…

```
emocao_predita = naive_bayes_polaridade_BoW.predict(avaliacao_BoW)
print('emocao predita:')
print(emocao_predita)
print('emocao real:')
print(list(teste_emocoes['sentiment_polarity']))
```

```
emocao predita:
['negative' 'negative' 'positive' 'positive' 'negative' 'positive'
 'negative' 'positive' 'positive']
emocao real:
['negative', 'negative', 'positive', 'positive', 'negative', 'positive', 'negative', 'positive', 'positive']
```

Teste Naive Bayes com TF-IDF polaridade

In [187…

```
emocao_predita = naive_bayes_polaridade_tfidf.predict(avaliacao_tfidf)
print('emocao predita:')
print(emocao_predita)
print('emocao real:')
print(list(teste_emocoes['sentiment_polarity']))
```

```
emocao predita:
['negative' 'negative' 'negative' 'negative' 'negative' 'positive'
 'negative' 'positive' 'positive']
emocao real:
['negative', 'negative', 'positive', 'positive', 'negative', 'positive', 'negative', 'positive', 'positive']
```

In [ ]:

In [ ]: