

Classificação de sentimento no E-Commerce



com Regressão Logística



Contexto

Analise de Sentimento

Análise de sentimento é um termo que se refere ao uso de **processamento de linguagem natural**, análise de texto e linguística computacional para determinar a atitude de um falante ou escritor em relação a um tópico específico.

Basicamente, ajuda a determinar **se um texto expressa sentimentos** positivos, negativos ou neutros. A análise de sentimento é uma excelente maneira de descobrir como as pessoas, principalmente os consumidores, se sentem sobre um determinado tópico, produto ou ideia.



Origem dos Dados

Esses dados foram disponibilizados no site da Kaggle. Nessa base de dados há milhares de comentários feitos por clientes do Brasil inteiro em sites E-commerce.

Vamos minerar esses dados e criar um modelo de classificação de emoção.

<https://www.kaggle.com/olistbr/brazilian-ecommerce>

kaggle

Vamos utilizar uma base de dados da Kaggle

<https://www.kaggle.com/olistbr/brazilian-ecommerce>

Vamos importar as bibliotecas necessárias

```
[1] # Lib para Modelagem de Dados
import pandas as pd
# Lib para matrizes
import numpy as np
# Lib para plotagem gráfica
import matplotlib.pyplot as plt
# Lib para expressões regulares
import re
# Lib para trabalhar com palavras
import string
# Lib para processamento de linguagem
import nltk

# Função para extrair o radical da Palavra
from nltk.stem import RSLPStemmer
# Função para remover palavras sem relevância
from nltk.corpus import stopwords

# Libs para uso de ML
# Função do modelo logístico
from sklearn.linear_model import LogisticRegression
# Funções Estimadores
from sklearn.base import BaseEstimator, TransformerMixin
# Função de transformadores
from sklearn.pipeline import Pipeline
# Função para matriz
from sklearn.feature_extraction.text import CountVectorizer
# Funções para avaliação do modelo
from sklearn.model_selection import cross_val_score

# Ignorar avisos
import warnings
warnings.simplefilter("ignore")
```

Vamos instalar alguns pacotes e ler os dados

```
[2] # Download de alguns pacotes da biblioteca NLTK

# Pacote Radical da palavra
nltk.download('rslp')

# Pacote palavras sem relevância
nltk.download('stopwords')

# Grupo de Palavras sem relevância
Palavras_Sem_Relancia = stopwords.words('portuguese')

[nltk_data] Downloading package rslp to /root/nltk_data...
[nltk_data] Package rslp is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

[3] # Lendo a Base de Dados
Base = pd.read_csv('olist_order_reviews_dataset.csv')

# Verificando as primeiras linhas
Base.head()
```

	review_id	order_id	review_score	review_comment_title	review_comment_message	review_creation
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	NaN	NaN	2018-01-18 0
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5	NaN	NaN	2018-03-10 0
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	NaN	NaN	2018-02-17 0
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	NaN	Recebi bem antes do prazo estipulado.	2017-04-21 0
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	NaN	Parabéns lojas lannister adorei comprar pela l...	2018-03-01 0

Nessa base de dados vamos utilizar apenas o **Comentário** e **Avaliação do cliente**.

Vamos filtrar apenas as colunas essenciais

```
[4] # Filtrar as colunas essenciais
Base_Filtrada = Base[['review_comment_message', 'review_score']]

# Traduzir as colunas para ficar facil o entendimento
Base_Filtrada.rename( columns={
    'review_comment_message' : 'Comentário',
    'review_score' : 'Pontuação'
}, inplace=True )

# Verificando
Base_Filtrada.head()
```

	Comentário	Pontuação
0	NaN	4
1	NaN	5
2	NaN	5
3	Recebi bem antes do prazo estipulado.	5
4	Parabéns lojas lannister adorei comprar pela l...	5

Retirando os valores nulos

```
[5] # Retirando os valores Nulos
Base_Filtrada = Base_Filtrada.loc[
    ( Base_Filtrada['Comentário'].notnull() ) &
    ( Base_Filtrada['Pontuação'].notnull() ) ]

# Verificando os ajustes de registros
print('Antes:', len(Base) )
print('Depois:', len(Base_Filtrada) )
```

Antes: 100000
Depois: 41753

Vamos criar a classe, a variável que vamos tentar prever

```
[6] # Vamos utilizar a função 'CUT' do pandas para gerar nossa variavel de classificação
# Use cut quando precisar segmentar e classificar os valores dos dados em compartimentos

# Função CUT ira ter 2 Parametros
# Pontos [0 a 5]
# Classe [0 ou 1]

# Basicamente o pandas irá classificar entre 0 e 1 nossas frases com base na pontuação do cliente
# Assim vamos ter as classes para classificar nosso moldeo

# Lista com escala dos pontos
Pontos = [0, 2, 5]

# Classificação [ 0 - Comentarios Ruins ]
# Classificação [ 1 - Comentarios Bons ]
Label = [0, 1]

# Aplicando a função para gerar nossa Classe
Base_Filtrada['Classe'] = pd.cut( Base_Filtrada['Pontuação'],
                                bins=Pontos, # Modelo de score
                                labels=Label ) # Classificação

# Verificando
print( 'Registros Superiores', '\n', Base_Filtrada.head(10), '\n' )
print( 'Registros Inferiores', '\n', Base_Filtrada.tail(10), '\n' )
```

Registros Superiores			
	Comentário	Pontuação	Classe
3	Recebi bem antes do prazo estipulado.	5	1
4	Parabéns lojas lannister adorei comprar pela l...	5	1
9	aparelho eficiente. no site a marca do aparelh...	4	1
12	Mas um pouco ,travando...pelo valor ta Boa.\r\n	4	1
15	Vendedor confiável, produto ok e entrega antes...	5	1
16	GOSTARIA DE SABER O QUE HOUE, SEMPRE RECEBI E...	2	0
19	Péssimo	1	0
22	Loja nota 10	5	1
24	obrigado pela atencao amim dispensada	5	1
27	A compra foi realizada facilmente.\r\nA entreg...	5	1

Registros Inferiores			
	Comentário	Pontuação	Classe
99972	Bom dia \r\nDas 6 unidades compradas só recebi...	3	1
99975	Foto muito diferente principalmente a graninha...	2	0
99977	Produto original,prazo de entrega rápido.Super...	5	1
99978	Tive um problema na entrega em que o correio c...	2	0
99980	para este produto recebi de acordo com a compr...	4	1
99983	Entregou dentro do prazo. O produto chegou em ...	5	1
99990	O produto não foi enviado com NF, não existe v...	3	1
99996	Excelente mochila, entrega super rápida. Super...	5	1
99998	Solicitei a compra de uma capa de retrovisor c...	1	0
99999	meu produto chegou e ja tenho que devolver, po...	1	0

Para gerar a classe, usei o [seguinte racional](#).
Classificação **maior** que 3 será considerado ‘Positivo = 1 ’, **menor** que isso será ‘negativo = 0’

Vamos criar 3 Funções para tratar nossos dados

```
[7] # Classe com funções para limpar os Textos
class Funcao_Limpeza(BaseEstimator, TransformerMixin):

    # Função para ativar na chamada
    def fit(self, x, y=None):
        return self

    # Função para transformar os dados
    def transform(self, x, y=None):

        # Lista que ira receber os dados Transformado
        Texto_Transformado = []

        # Loop no Comentário
        for Texto in x:

            # retirando os paragrafos, numeros e afins
            Texto = re.sub('\n', ' ', Texto)
            Texto = re.sub('\r', ' ', Texto)
            Texto = re.sub(r'\d+(?:\.\d*(?:[eE]\d+))?', ' numero ', Texto)
            Texto = re.sub(r'R$', ' ', Texto)
            Texto = re.sub(r'W', ' ', Texto)
            Texto = re.sub(r'\s+', ' ', Texto)

            # Verificando se existe links no comentários
            URL_Texto = re.findall('(http|ftp|https)://([\w_-]+(?:\:([\w_-]+)+))([\w.,@?^=%&:/~+#-]*[\w@?^=%&:/~+#-])?', Texto)

            # Caso não exista uma URL vai passar
            if len(URL_Texto) == 0:
                pass

            # No caso de ter URL no Texto
            else:
                # Loop no Texto
                for url in URL_Texto:
                    # Loop na frase
                    for link in url:
                        # Retirando o Link
                        Texto = Texto.replace(link, '')

            # Retirando possveis ':' e '/'
            Texto = Texto.replace(':', '')
            Texto = Texto.replace('/', '')

            # Salvando a info na Lista
            Texto_Transformado.append(Texto)

        # Retorno da Função
        return Texto_Transformado
```

Essa função irá retirar os ‘**resíduos**’ das frases. Caracteres especiais, espaços, link e afins, não agregam em nada nos modelos ao contrário, podem piorar.

Assim vamos fazer essa limpeza.

Vamos Extrair o Radical da Palavra

```
[8] # Classe com funções para Extrair o Radical da Palabras
class Funcao_Radical(BaseEstimator, TransformerMixin):

    # Função para ativar na chamada
    def fit(self, x, y=None):
        return self

    # Extraíndo o Radical da Palavra
    def Aplicando_Radical(self, Loop):
        # Função do Radical
        Radical = RSLPStemmer()

        # Retornando o radical da palavra
        return list(map(lambda x: Radical.stem(x), [Palavra for Palavra in Loop.split()]))

    # Aplicando a Transformação
    def transform(self, x, y=None):
        # Retornando aplicando a função do Radical
        Texto_Transformado = list(map(lambda c: self.Aplicando_Radical(c), x))

        # Unificando as palavrvas
        Texto_Transformado = list(map(lambda x: ' '.join(x), Texto_Transformado))

        # Retornando o comentario tratado
        return Texto_Transformado
```

Caso não saiba o que é o Radical da palavra.
<https://querobolsa.com.br/enem/portugues/radical>

Vamos retirar pontuações e palavras sem relevâncias

```
[9] # Retirar palavras sem relevancia e pontuações
class Funcao_Palavras(BaseEstimator, TransformerMixin):

    def fit(self, x, y=None):
        return self

    # Revomendo Stops Words e Pontuações
    def Remover_Palavras(self, Texto):
        # Verificando se existe pontuações na Palavraas
        Sem_Pontuacao = [Palavra for Palavra in Texto if Palavra not in string.punctuation]

        # Unificando o texto
        Sem_Pontuacao = ''.join(Sem_Pontuacao)

        # Retornando a frase sem as stops words
        return [Sem_Relancia for Sem_Relancia in Sem_Pontuacao.split() if Sem_Relancia.lower() not in Palavras_Sem_Relancia]

    # Aplicando a transformação na frase
    def transform(self, x, y=None):
        # Removendo palavras e pontuações com a função 'Remover_Palavras'
        Texto_Transformado = list(map(lambda c: self.Remover_Palavras(c), x))

        # Unificando o texto
        Texto_Transformado = list(map(lambda x: ' '.join(x), Texto_Transformado))

        # Retorno da Função
        return Texto_Transformado
```

Explicação do que seria as Stops Words

<https://virtuati.com.br/cliente/knowledgebase/25/Lista-de-StopWords.html>

Vamos aplicar essas 3 funções nos dados

```
[10] # Unindo todas classes para serem passadas de uma vez pelo 'Pipeline'
Processamento_Geral = Pipeline([
    ('Limpeza', Funcao_Limpeza() ),
    ('Remover_Sem_Relevancia', Funcao_Palavras() ),
    ('Radical_Palavra', Funcao_Radical() ),
])

# Aplicando todas as funções
Texto_Processado = Processamento_Geral.fit_transform(
    Base_Filtrada['Comentário'] )
```

```
[11] # Verificando o Texto
Texto_Processado[0:10]

['receb bem ant praz estipul',
'parabém loj lannist ador compr internet segur prát parabém tod feliz pásco',
'aparelh efici sit marc aparelh impress numer desinfec cheg outr nom atual marc corret vez aparelh',
'pouc trav val ta boa',
'vend confi produt ok entreg ant praz',
'gost sab sempr receb compr agor decpcion',
'péss',
'loj not numer',
'obrig atença amim dispens',
'compr realiz facil entreg efetu ant praz dad produt começ ser us pres problem']
```

O que **aconteceu** ate aqui ?

Veja que as palavras ficaram com uma **certa abreviação**, certo? Esse processo é referente ao radical da palavra.
Outro detalhe que as frases **ficaram mais curtas**, isso se deve ao processo da remoção das stops words.

Quando treinamos um modelo de NLP (Linguagem de processamento natural) é essencial fazer esse tipo de treinamento para melhorarmos a performance do modelo.

Toda vez que for trabalhar com NLP é primordial fazer esse tipo de tratamento.

Vamos aplicar a vetorização nas Frases

```
[12] # Definindo os parametros para vetorização
# Essa função conveter uma coleção de documentos de texto em uma matriz de contagens de tokens
Vetorizacao = CountVectorizer( max_features=300,
                               stop_words=Palavras_Sem_Relancia)

# Aplicando a Vetorização
# Será retornando uma matriz em um formato de um Array
Dados_Matriz = Vetorizacao.fit_transform(Texto_Processado).toarray()
```

É denominado **vetorização** o processo geral de converter a coleção de textos em **vetores numéricos**. A estratégia de *tokenizar* (separar textos ou palavras em “blocos”), *contar* e *normalizar* (como já fizemos anteriormente) é chamado de representação **Bag of Words (BOW)**, ou na tradução literal **bolsa de palavras**, no qual são descritos pelas ocorrências das palavras enquanto são ignoradas suas posições relativas em todos os textos. Segue exemplo abaixo de como fica a estrutura, com a frequencia absoluta de cada palavra:

	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

fig.1 - Imagem adaptada de text-analytics-beginners-nltk bootcamp

Sugiro a leitura caso não entenda sobre Vetorização

<https://carlos-bonfim.medium.com/machine-learning-e-processamento-de-linguagem-natural-pln-com-modelos-lineares-d5aaaaf0efa5>

Vamos treinar o modelo e medir sua Acuracia

```
[14] # Definindo a Função Logística do nosso Modelo
Funcao_Logistica = LogisticRegression( max_iter=10000 )

# Treinando o Modelo
Funcao_Logistica.fit(x_Treino, y_Treino)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=10000,
                    multi_class='auto', n_jobs=None, penalty='l2',
                    random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                    warm_start=False)

[15] # Medir a acuracia do modelo no cross_val
# será feito 8 teste com dados aleatorios
Acuracia = cross_val_score( Funcao_Logistica, # Algoritmo
                             x_Testes, y_Testes, # Dados de testes
                             cv=8, # Interações
                             scoring='accuracy') # Tipo da medida

# Pegando a média
Acuracia.mean()

0.8677952823375319
```

Modelo foi muito bem !!!



Vamos plotar essas acurácias

```
[16] # Verificar os teste do modelo

# Definindo uma figura
fig = plt.figure( figsize=(12,6) )
# Titulo
fig.suptitle('Acuracia do Modelo - Testes')
# Plotando os Valores
plt.plot(Acuracia)
# Mostrando os gráfico
plt.show()
```



Hora dos Testes

```
[17] # Função testar as frases
def Testando_Modelo(Seu_Texto):
    # Variavel para receber a classe
    Classificação = ''

    # Aplicando a transformação
    Processamento_Texto = Processamento_Geral.fit_transform( Seu_Texto )

    # Aplicando a Vetorização
    Texto_Transformado = Vetorizacao.transform( Processamento_Texto ).toarray()

    # Fazendo a Previsão
    Previsao = Funcao_Logistica.predict( Texto_Transformado )

    # Condição para verificar Positivo ou False
    if Previsao[0] == 1:
        Classificação = 'Positivo'
    else:
        Classificação = 'Negativo'

    return print('Avaliação do Comentário: ', Classificação )
```

```
[18] Testando_Modelo( ['Excelente produto!'] )
```

Avaliação do Comentário: Positivo

```
[19] Testando_Modelo( ['produto de péssima qualidade!!!'] )
```

Avaliação do Comentário: Negativo



Mandamos muito bem !!!

Final

Esse guia foi elaborado para classificação de comentários.

Link do Colab

https://colab.research.google.com/drive/1CmC4SbPpjdmlcD2WAunoadXq6_rsjH1A?usp=sharing



Odemir Depieri Jr

Data Intelligence Analyst Sr
Tech Lead
Specialization AI