

Atividade Prática 3 - Base de dados Trip Advisor Hotel Reviews

Summary

1. Limpeza da base de dados: Remover stopwords, caracteres especiais e aplicar stemming.
2. Transformação em atributos numéricos: Utilizar CountVectorizer para converter o texto em um vetor de termos/term frequency, e então aplicar TfidfTransformer para normalizar os dados.
3. Avaliação com classificadores: Utilizar árvore de decisão e random forest para avaliar os dados transformados.

```
In [ ]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from nltk.tokenize import word_tokenize
import re
from sklearn.model_selection import train_test_split
```

```
In [ ]: # Carregar os dados
df = pd.read_csv('tripadvisor_hotel_reviews.csv')
df.head(5)
```

```
Out[ ]:
```

| | Review | Rating |
|---|---|--------|
| 0 | nice hotel expensive parking got good deal sta... | 4 |
| 1 | ok nothing special charge diamond member hilt... | 2 |
| 2 | nice rooms not 4* experience hotel monaco seat... | 3 |
| 3 | unique, great stay, wonderful time hotel monac... | 5 |
| 4 | great stay great stay, went seahawk game aweso... | 5 |

```
In [ ]: # Função atualizada de limpeza de dados
def Limpeza_Dados(Texto):
    # Remoção de links
    etapa_01 = re.sub('www\S+|http\S+', '', Texto)
    # Transformar para minúsculo
    etapa_02 = etapa_01.lower()
    # Remoção de caracteres especiais
    etapa_03 = re.sub(r'[!~@#$$%^&*()+=|{}[\]:;<.>?/\\"'-]', ' ', etapa_02)
    # Remoção de números
    etapa_04 = re.sub('[0-9]', '', etapa_03)
    # Remove espaços extras
    etapa_05 = re.sub(r'\s+', ' ', etapa_04)
```

```

    return etapa_05

# Aplicar a limpeza nos dados
df['Review_cleaned'] = df['Review'].apply(Limpeza_Dados)

```

```
In [ ]: df.head(5)
```

```
Out[ ]:
```

| | Review | Rating | Review_cleaned |
|---|---|--------|---|
| 0 | nice hotel expensive parking got good deal sta... | 4 | nice hotel expensive parking got good deal sta... |
| 1 | ok nothing special charge diamond member hilt... | 2 | ok nothing special charge diamond member hilt... |
| 2 | nice rooms not 4* experience hotel monaco seat... | 3 | nice rooms not experience hotel monaco seattle... |
| 3 | unique, great stay, wonderful time hotel monac... | 5 | unique great stay wonderful time hotel monaco ... |
| 4 | great stay great stay, went seahawk game aweso... | 5 | great stay great stay went seahawk game awesom... |

```
In [ ]: exemplo = df['Review'][10]
        print(exemplo)
        Limpeza_Dados(exemplo)
```

poor value stayed monaco seattle july, nice hotel priced 100- 150 night not, hotel takes beating quotient, experience simply average, nothing exceptional paying 300+ not terrible disappointed, wife stayed nicest suites 200/night felt like overpaying, best advice shop, quality-wise league average marriott nice best western,

```
Out[ ]: 'poor value stayed monaco seattle july nice hotel priced night not hotel
        takes beating quotient experience simply average nothing exceptional pay
        ing not terrible disappointed wife stayed nicest suites night felt
        like overpaying best advice shop quality wise league average marriott
        nice best western '
```

```
In [ ]: df.drop('Review', axis=1, inplace=True)
```

```
In [ ]: df.shape
```

```
Out[ ]: (20491, 2)
```

```
In [ ]: df.rename(columns={"Rating": "class", "Review_cleaned": "text"}, inplace =
```

```
In [ ]: df.columns
```

```
Out[ ]: Index(['class', 'text'], dtype='object')
```

```
In [ ]: df['class'].isnull().sum()
```

```
Out[ ]: 0
```

```
In [ ]: df['class'].value_counts()
```

```
In [ ]: # Reduzir target para variável binária
```

```
In [ ]: # ALtera na Base de Dados (Dataframe)
df['class'] = df['class'].apply(change_rating)
```

```
Out[ ]: class
Positive Rating    15093
Negative Rating    5398
Name: count, dtype: int64
```

```
In [ ]: # Concatenar todas as palavras em uma única string
all_words = ''.join(df['text'])
```

```
Out[ ]: <matplotlib.image.AxesImage at 0x7f7e253a8e10>
```



```
In [ ]: df['class'].unique()
```

```
Out[ ]: array(['Positive Rating', 'Negative Rating'], dtype=object)
```

```
In [ ]: # Avaliação por review positivo e negativo

review_positive = ''
review_negative = ''

# Verifying each class (positive and negative)
for c in df['class'].unique():
    # Filter the dataframe for the current class
    filter_df = df[df['class'] == c]

    # Iterate over the filtered dataframe
    for text in filter_df['text']:
        # If the class is 'Positive Rating', concatenate to review_positi
        if c == 'Positive Rating':
            review_positive += ' ' + ' '.join(text.split())
        # If the class is 'Negative Rating', concatenate to review_negati
        elif c == 'Negative Rating':
            review_negative += ' ' + ' '.join(text.split())

# Trim the leading spaces
review_positive = review_positive.strip()
review_negative = review_negative.strip()
```

```
In [ ]: print(
    'Quantidade de Palavras: \n',
    f'Best_review: { len(review_positive) }',
    f'Worst_review: { len(review_negative) }'
)
```

```
Quantidade de Palavras:
Best_review: 10177358 Worst_review: 4223226
```

```
In [ ]: # Classe Positiva

# Instanciando a Nuvem de Palavras
Nuvem_Palavras = WordCloud().generate(review_positive)

# Tamanho
Figura, Eixo = plt.subplots()
# Plot
Eixo.imshow(Nuvem_Palavras)
```

```
Out[ ]: <matplotlib.image.AxesImage at 0x7f7dff3e5190>
```


StopWords

```
In [ ]: nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /home/a10/nltk_data...  
[nltk_data]   Package stopwords is already up-to-date!
```

```
Out[ ]: True
```

```
In [ ]: df['text'][1000]
```

```
Out[ ]: 'shame hotel wasnt good restaurant arrived clift late afternoon struggle  
luggage bags reception staff unhelpful uninterested eventually managed s  
orted shown room th floor room suite tried make separate living room put  
ting curtain inbetween bedroom living room bathroom tiny dirty stayed mu  
m unfortunatley night didnt feel suffering bad foot decided phone recept  
ion ask doctor come hotel told ther wasnt local receptionist closest tol  
d phone eventually decided hospital just safe came hospital evening door  
men talking girls outside let following night ate hotel restaurant aisa  
cuba fantastic think hotel intrest restaurant bar end day sleeping ignor  
ed wouldnt stay '
```

```
In [ ]: lista_stopwords = nltk.corpus.stopwords.words('english')
```

```
In [ ]: def Remover_StopWords(Texto):  
  
    Lista_Palavras = Texto.split()  
  
    # Texto sem as stopwords  
    nova_frase = ''  
  
    for word in Lista_Palavras:  
        if word not in lista_stopwords:  
            nova_frase = nova_frase + ' ' + word  
  
    return nova_frase
```

```
In [ ]: exemplo = df['text'][10]  
print(len(exemplo))  
saida = Remover_StopWords(exemplo)  
print(len(saida))
```

```
306  
298
```

```
In [ ]: df['text'] = df['text'].apply(Remover_StopWords)
```

Extração do Radical

```
In [ ]: Stem = nltk.stem.RSLPStemmer()
```

```
In [ ]: def Extrair_Radical(Texto):  
  
    Lista_Palavras = Texto.split()  
    nova_frase = ''  
  
    for word in Lista_Palavras:
```

```

    radical = Stem.stem(word)
    nova_frase = nova_frase + ' ' + radical
    return nova_frase

```

```

In [ ]: exemplo = df['text'][10]
        print(exemplo)
        Extrair_Radical(exemplo)

```

poor value stayed monaco seattle july nice hotel priced night hotel takes beating quotient experience simply average nothing exceptional paying n ca n terribly disappointed wife stayed nicest suites night felt like overpaying best advice shop quality wise league average marriott nice best western

```

Out[ ]: ' po valu stayed monac seattl july nic hotel priced night hotel tak beat
        ing quotient experienc simply averag nothing except paying n ca n terrib
        ly disappointed wif stayed nicest suit night felt lik overpaying best ad
        vic shop quality wis leag averag marriott nic best western'

```

```

In [ ]: df['text'] = df['text'].apply(Extrair_Radical)

```

Tokenização

```

In [ ]: nltk.download('punkt')

```

[nltk_data] Downloading package punkt to /home/a10/nltk_data...
[nltk_data] Package punkt is already up-to-date!

```

Out[ ]: True

```

```

In [ ]: from nltk.tokenize import word_tokenize

```

```

In [ ]: exemplo = df['text'][10]
        pd.DataFrame(word_tokenize(exemplo)).T

```

```

Out[ ]:
   0    1    2    3    4    5    6    7    8    9  ...  34  35
0  po  valu  stayed  monac  seattl  july  nic  hotel  priced  night  ...  advic  shop  qual

```

1 rows × 44 columns

Modelo

```

In [ ]: # Separar entre previsores e classe
        X = df['text']
        y = df['class']

```

```

In [ ]: # Dividir os dados em conjuntos de treino e teste
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,

```

```

In [ ]: X_train.shape

```

```

Out[ ]: (14343,)

```

```

In [ ]: X_test.shape

```

```

Out[ ]: (6148,)

```

```
In [ ]: from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

# Inicializar CountVectorizer
vect = CountVectorizer()
vect.fit(X_train)
X_train_vect = vect.transform(X_train)
X_test_vect = vect.transform(X_test)

# Inicializar TfidfTransformer
Tfidf = TfidfTransformer()
X_train_vect = Tfidf.fit_transform(X_train_vect)
X_test_vect = Tfidf.fit_transform(X_test_vect)
```

```
In [ ]: X_train_vect.shape
```

```
Out[ ]: (14343, 35046)
```

```
In [ ]: X_test_vect.shape
```

```
Out[ ]: (6148, 35046)
```

```
In [ ]: vect.get_feature_names_out()
```

```
Out[ ]: array(['_', '___', '____', ..., 'üè', 'üè_', 'üèc'], dtype=object)
```

```
In [ ]: from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Inicializar e treinar o classificador de árvore de decisão
dt_clf = DecisionTreeClassifier()
dt_clf.fit(X_train_vect, y_train)
dt_predictions = dt_clf.predict(X_test_vect)

# Avaliar árvore de decisão
print("Árvore de Decisão - Relatório de Classificação:\n", classification_report(dt_predictions, y_test))

# Inicializar e treinar o classificador Random Forest
rf_clf = RandomForestClassifier()
rf_clf.fit(X_train_vect, y_train)
rf_predictions = rf_clf.predict(X_test_vect)

# Avaliar Random Forest
print("Random Forest - Relatório de Classificação:\n", classification_report(rf_predictions, y_test))
```


Árvore de Decisão - Relatório de Classificação:

| | precision | recall | f1-score | support |
|-----------------|-----------|--------|----------|---------|
| Negative Rating | 0.56 | 0.58 | 0.57 | 1600 |
| Positive Rating | 0.85 | 0.84 | 0.84 | 4548 |
| accuracy | | | 0.77 | 6148 |
| macro avg | 0.70 | 0.71 | 0.71 | 6148 |
| weighted avg | 0.77 | 0.77 | 0.77 | 6148 |

Random Forest - Relatório de Classificação:

| | precision | recall | f1-score | support |
|-----------------|-----------|--------|----------|---------|
| Negative Rating | 0.96 | 0.41 | 0.57 | 1600 |
| Positive Rating | 0.83 | 0.99 | 0.90 | 4548 |
| accuracy | | | 0.84 | 6148 |
| macro avg | 0.89 | 0.70 | 0.74 | 6148 |
| weighted avg | 0.86 | 0.84 | 0.82 | 6148 |

In []: