ML_shaz13_CPU_V2

July 2, 2024

2. (M/D) Comparar um conjunto de algoritmos de aprendizado de máquina (pelo menos cinco) em aplicações relacionadas ao seu projeto ou alguma outra aplicação real sugerida por você. Neste caso, espera-se que a aplicação seja útil e conte com uma base de dados interessante e de tamanho que permita a aplicação de algoritmos de aprendizado de máquina.

1 Coleta de Dados

1.0.1 https://www.kaggle.com/datasets/shaz13/real-world-documents-collections

[24]:
```
import pandas as pd
import os
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from typing import List
```

[26]:
```
# https://github.com/kaiomudkt/projeto_Machine_Learning/blob/main/
 ↪ETL_download_dataset_kaggle.py
from ETL_download_dataset_kaggle import get_dataset

path_dataset_base: str = "dataset/real_world_documents_collections"
path_dataset: str = f"{path_dataset_base}/docs-sm"
link_kaggle_dataset = 'shaz13/real-world-documents-collections'
path_zip: str = "./dataset/compactados"
zip_file_name: str = f'{path_zip}/real-world-documents-collections.zip'
# obtem o dataset para poder ser processado
get_dataset(zip_file_name, link_kaggle_dataset, path_zip, path_dataset_base)
print('Download do dataset já foi concluido.')
```

```
Arquivo ./dataset/compactados/real-world-documents-collections.zip descompactado
com sucesso em dataset/real_world_documents_collections
Download do dataset já foi concluido.
```

[27]:
```
# https://github.com/kaiomudkt/projeto_Machine_Learning/blob/main/
 ↪ETL_process_image.py
from ETL_process_image import process_dataset, draw_bounding_boxes
```

```
required_folders_classes: dict = {'form': 0, 'resume': 1, 'letter': 2,␣
 ↪'invoice': 3, 'questionnaire': 4}
path_df_parquet: str = './DF_shaz13_real_world_documents_collections_V4.parquet'
# processa o dataset de imagens, extraindo textos via OCR e limpando os texto␣
 ↪para poder usar no Aprendizado de Máquina
df = process_dataset(f'./{path_dataset}', path_df_parquet,␣
 ↪required_folders_classes)
df.columns
```

[27]:
```
Index(['text', 'class_img', 'name', 'dict_ocr', 'class_number'], dtype='object')
```

2 Análise Exploratória de Dados (EDA)

[28]:
```
df.columns
```

[28]:
```
Index(['text', 'class_img', 'name', 'dict_ocr', 'class_number'], dtype='object')
```

[29]:
```
print(df.head(5))
```

```
                                                text      class_img  \
0  bellomycarrigginc job university parkway aa ex…  questionnaire
1  inc broadway york screening questionnatre time…  questionnaire
2                                              ozle  questionnaire
3  alan november account smoking manufacture desc…  questionnaire
4  market monttor office use interviewer resp res…  questionnaire

                      name                                               dict_ocr  \
0  505527865_505527879.jpg  {'block_num': [0, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2…
1    89583564_89583573.jpg  {'block_num': [0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
2  503543120_503543128.jpg  {'block_num': [0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3…
3           501525861.jpg  {'block_num': [0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2…
4            71224770.jpg  {'block_num': [0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2…

   class_number
0             4
1             4
2             4
3             4
4             4
```

3 Limpeza dos Dados

[30]:
```
# Substitui strings vazias por NaN
df['text'] = df['text'].replace('', np.nan)
# Conta o número de valores NaN na coluna 'text'
num_nan = df['text'].isna().sum()
print(f'Quantidade de linhas vazias: {num_nan}')
```

```
Quantidade de linhas vazias: 206
```

[31]:
```
# Remove linhas onde a coluna 'text' está NaN
df = df.dropna(subset=['text'])
num_nan = df['text'].isna().sum()
print(f'Quantidade de linhas vazias após a remoção: {num_nan}')
```

```
Quantidade de linhas vazias após a remoção: 0
```

[32]:
```
def text_smaller_than(df, amount: int):
    # Filtrar linhas onde a string na coluna 'text' tem menos de 'amount'␣
 ↪caracteres
    short_text_rows = df[df['text'].str.len() < amount]
    # Contar o número de tais linhas
    num_short_text_rows = short_text_rows.shape[0]
    return num_short_text_rows
amount: int = 5
num_short_text_rows = text_smaller_than(df, amount)
print(f'Quantidade de textos que é menor que {amount} caracteres:␣
 ↪{num_short_text_rows}')
```

```
Quantidade de textos que é menor que 5 caracteres: 48
```

[33]:
```
# Remover as linhas onde a string na coluna 'text' tem menos de 'amount'␣
 ↪caracteres
df_filtered = df[df['text'].str.len() >= amount]
df = df_filtered
num_short_text_rows = text_smaller_than(df, amount)
print(f'Quantidade de textos que é menor que {amount} caracteres:␣
 ↪{num_short_text_rows}')
```

```
Quantidade de textos que é menor que 5 caracteres: 0
```

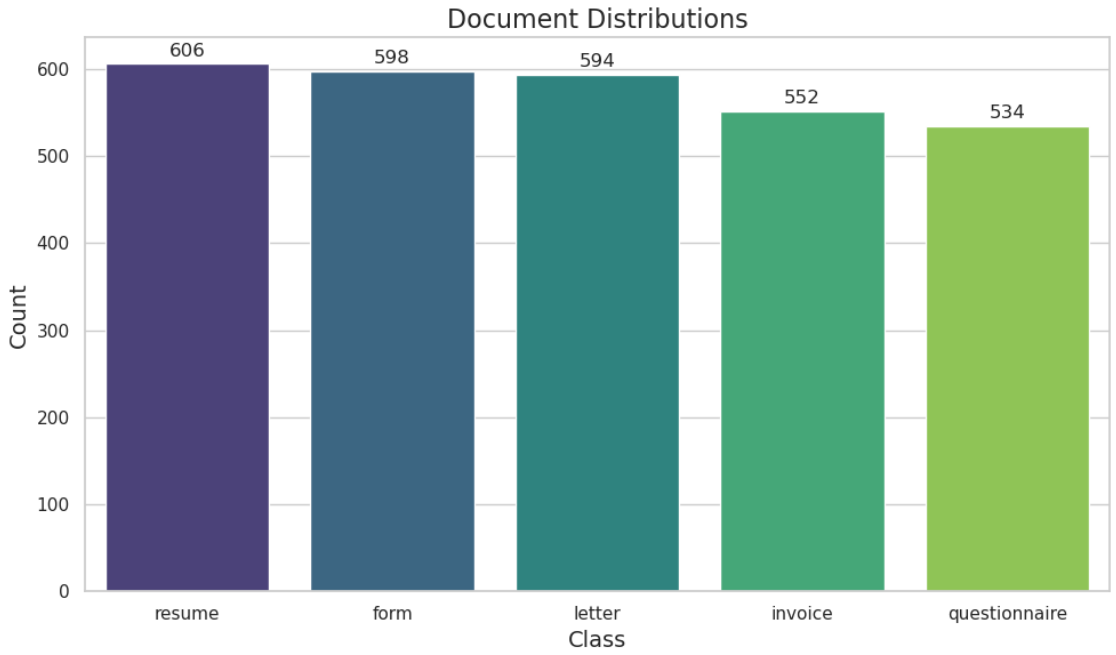
3.0.1 Analisar classes (target) do df

[34]:
```
# Conta a quantidade de linhas que têm o mesmo valor na coluna 'class_img'
class_img_counts = df['class_img'].value_counts()
print(f'Quantidade imagens de cada classe: {class_img_counts}')
```

```
Quantidade imagens de cada classe: class_img
resume           606
form             598
letter           594
invoice          552
questionnaire    534
Name: count, dtype: int64
```

[35]:
```python
# Contar as ocorrências de cada classe
class_counts = df['class_img'].value_counts()

# Configurar o estilo do gráfico
sns.set(style="whitegrid")

# Criar o gráfico de barras
plt.figure(figsize=(10, 6))
ax = sns.barplot(x=class_counts.index, y=class_counts.values, palette="viridis")

# Adicionar anotações sobre as barras
for i, value in enumerate(class_counts.values):
    ax.text(i, value + 5, str(value), ha='center', va='bottom', fontsize=12)

# Adicionar título e rótulos dos eixos
plt.title("Document Distributions", fontsize=16)
plt.xlabel("Class", fontsize=14)
plt.ylabel("Count", fontsize=14)

# Ajustar o layout
plt.tight_layout()

# Exibir o gráfico
plt.show()
```

```
/tmp/ipykernel_230/1263128678.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.barplot(x=class_counts.index, y=class_counts.values,
palette="viridis")
```



Nivelando aleatóriamente a quantidade de imagens de cada classe para ficar do mesmo tamanho que a menor

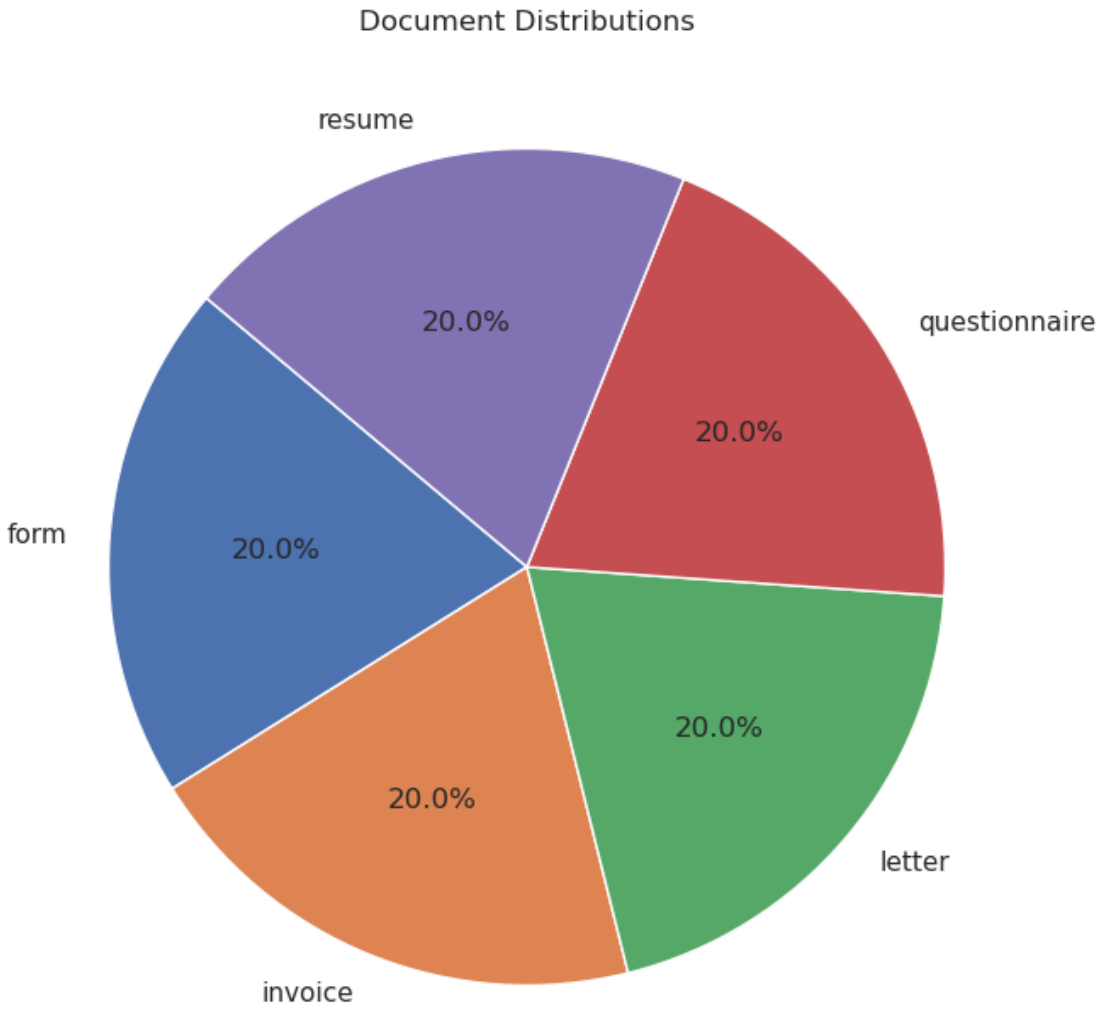
[36]:
```
# Define o número de linhas que cada classe deve ter
min_class_count = class_img_counts.min()
# Amostra aleatória de cada classe para ter a mesma quantidade que a menor␣
 ↪classe
balanced_df = df.groupby('class_img').apply(lambda x: x.
 ↪sample(min_class_count)).reset_index(drop=True)
# Verifica a quantidade de linhas em cada classe no DataFrame balanceado
balanced_class_img_counts = balanced_df['class_img'].value_counts()
df = balanced_df
print(f'Quantidade de imagens em cada classe após balanceamento:␣
 ↪\n{balanced_class_img_counts}')
```

```
Quantidade de imagens em cada classe após balanceamento:
class_img
form             534
invoice          534
letter           534
questionnaire    534
resume           534
Name: count, dtype: int64
```

```
/tmp/ipykernel_230/2612964233.py:4: DeprecationWarning: DataFrameGroupBy.apply
operated on the grouping columns. This behavior is deprecated, and in a future
version of pandas the grouping columns will be excluded from the operation.
```

```
Either pass `include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
  balanced_df = df.groupby('class_img').apply(lambda x:
x.sample(min_class_count)).reset_index(drop=True)
```

[37]:
```
# Contar as ocorrências de cada classe
class_counts = df['class_img'].value_counts()

# Criar um gráfico de pizza
plt.figure(figsize=(8, 8))
plt.pie(class_counts, labels=class_counts.index, autopct='%1.1f%%',␣
 ↪startangle=140)
plt.title("Document Distributions")
plt.show()
```



[38]:
```
print(df.head(5))
```

```
                                                text class_img  \
0  attorney general martin j barrington anthony b…      form
1  publication verification tam lew title publica…      form
2  crc contract determination aldehyde day ether wen      form
3                            report dept dept file      form
4  jan new york york murray bring company inc eug…      form

             name                                               dict_ocr  \
0  2071358315.jpg  {'block_num': [0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3…
1  2084090727.jpg  {'block_num': [0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2…
2  2505151976.jpg  {'block_num': [0, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2…
3  2084093448.jpg  {'block_num': [0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2…
4  2063304296.jpg  {'block_num': [0, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2…

   class_number
0             0
1             0
2             0
3             0
4             0
```

3.0.2 Lista de palavras que são consideradas stopwords (palavras vazias) para o idioma inglês.

[39]:
```
from nltk.corpus import stopwords
stopwords_list = set(stopwords.words("english"))

def check_for_stopwords(text: str, stopwords_list: List[str]) -> List[str] |␣
 ↪None:
    """
    Função para verificar se há stopwords em uma string
    """
    # Dividir a string em palavras
    words = text.split()
    # Verificar se há interseção entre as palavras e as stopwords
    stopwords_in_text = set(words) & stopwords_list
    if stopwords_in_text:
        return list(stopwords_in_text)  # Retorna lista de stopwords encontradas
    else:
        return None  # Retorna None se não há stopwords

# Aplicar a função check_for_stopwords à coluna 'text' e armazenar as stopwords␣
 ↪encontradas
```

```
df['has_stopwords'] = df['text'].apply(lambda x: check_for_stopwords(x,␣
 ↪stopwords_list))
# Filtrar o DataFrame para incluir apenas linhas onde 'has_stopwords' não é None
filtered_df = df[df['has_stopwords'].notna()]
# Exibir índice da linha, texto original e has_stopwords não vazio
# for index, row in filtered_df.iterrows():
#     print(f"Índice da linha: {index}")
#     print(f"Texto original: {row['text']}")
#     print(f"has_stopwords: {row['has_stopwords']}")
#     print()  # Linha em branco para separar os resultados
```

[40]:
```
df['has_stopwords'].notna()
```

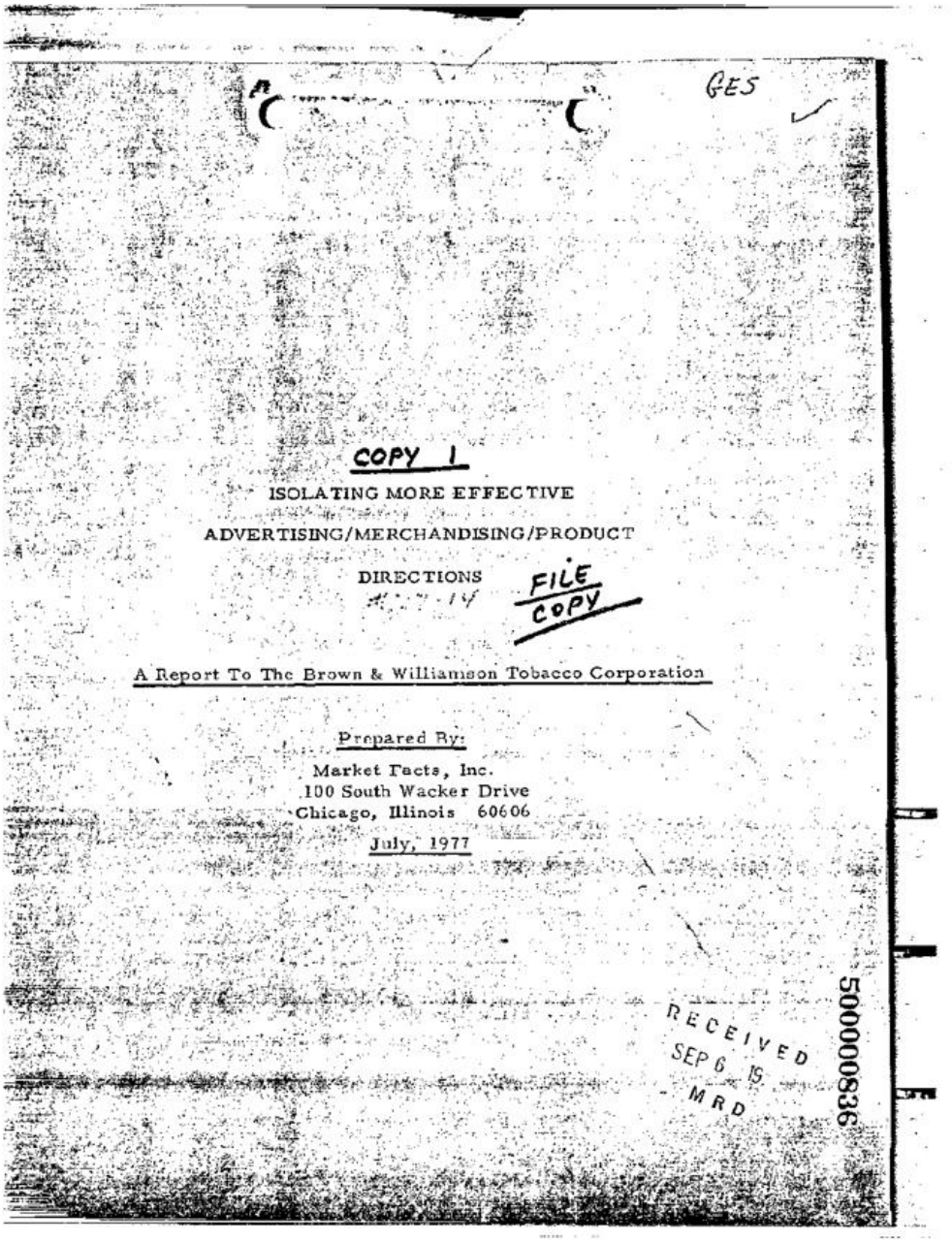
[40]:
```
0       False
1       False
2       False
3       False
4       False
        …
2665    False
2666    False
2667    False
2668     True
2669    False
Name: has_stopwords, Length: 2670, dtype: bool
```

3.1 Analisando a imagem

[41]:
```
from PIL import Image
class_img = 'questionnaire'
name_img = '0000002206.jpg'
# dataset/real_world_documents_collections/docs-sm/docs-sm/questionnaire/
 ↪0000002206.jpg
print(f"{path_dataset}/{class_img}/{name_img}")
image = Image.open(f"{path_dataset}/{class_img}/{name_img}")
image
```

```
dataset/real_world_documents_collections/docs-sm/questionnaire/0000002206.jpg
```

[41]:



[42]:
```
# Usar query para filtrar e iloc para pegar a primeira linha
filtered_row = df.query("class_img == @class_img and name == @name_img").iloc[0]
filtered_row
```

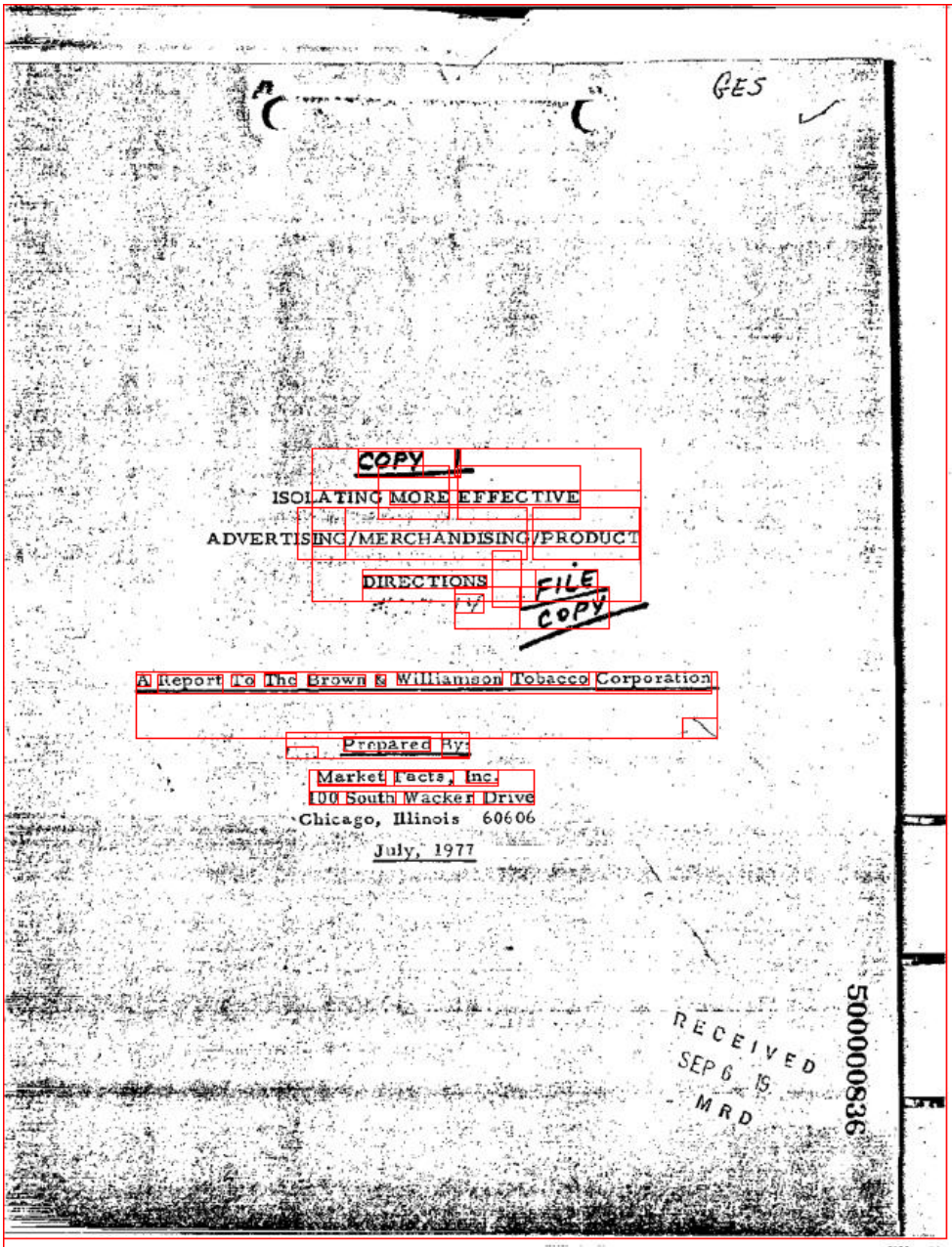
[42]:
```
text            effective brown tobacco corporation market inc…
class_img                                              questionnaire
name                                                0000002206.jpg
dict_ocr        {'block_num': [0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
class_number                                                    4
has_stopwords                                                None
Name: 1806, dtype: object
```

[43]:
```
# obtem a linha do DF onde essa imagem foi processada
ocr_df = pd.DataFrame(filtered_row['dict_ocr'])
ocr_df.columns
```

[43]:
```
Index(['block_num', 'conf', 'height', 'left', 'level', 'line_num', 'page_num',
       'par_num', 'text', 'top', 'width', 'word_num'],
      dtype='object')
```

[44]:
```
#
image = draw_bounding_boxes(image, ocr_df)
image
```

[44]:


GES
COPY 1
ISOLATING MORE EFFECTIVE
ADVERTISING/MERCHANDISING/PRODUCT
DIRECTIONS
FILE COPY
A Report To The Brown & Williamson Tobacco Corporation
Prepared By:
Market Facts, Inc.
100 South Wacker Drive
Chicago, Illinois 60606
July, 1977
RECEIVED
SEP 6 15
MRD
500000836

4 Separando os dados em treino e teste

4.0.1 Vetorizando dos dados

5 Ajuste de Hiperparâmetros e Avaliação de Modelos

5.0.1 O pipeline simplifica o fluxo de trabalho de processamento de texto e classificação em dois passos principais:

- Vectorização: O TfidfVectorizer transforma o texto em vetores TF-IDF, o que representa a importância das palavras em um documento.
- Classificação: O MultinomialNB classifica os documentos com base nos vetores TF-IDF.
- Tudo isso é encapsulado no objeto Pipeline, que permite o treinamento e a previsão em uma única chamada de método.

5.0.2 MultinomialNB

- O MultinomialNB é uma implementação do algoritmo Naive Bayes para dados distribuídos multinomialmente. É especialmente eficaz para problemas de classificação de texto, como a classificação de e-mails (spam vs. não spam) ou a categorização de documentos, onde as características são contagens de frequência de palavras.
- O Naive Bayes é um classificador probabilístico baseado no Teorema de Bayes com a suposição "ingênua" de independência entre cada par de características. Apesar da suposição simplista, ele funciona bem em muitos cenários práticos.
- Multinomial Naive Bayes, No contexto do MultinomialNB, o algoritmo calcula a probabilidade de uma classe com base nas contagens de frequência das características (como palavras em um documento). É adequado quando as características são representadas por contagens, como o número de vezes que uma palavra aparece em um documento.

5.0.3 Por que usar o MultinomialNB?

- Simplicidade: É fácil de implementar e interpretar.
- Eficiência: Rápido para treinar e fazer previsões, mesmo com grandes conjuntos de dados.
- Eficácia: Funciona bem em problemas de classificação de texto, onde a representação dos dados é baseada em contagens ou frequências de características.
- Em resumo, o MultinomialNB é uma escolha popular e eficaz para a classificação de texto e outros problemas onde os dados podem ser representados por contagens ou frequências.

5.0.4 Ajuste de hiperparâmetros para o modelo RandomForestClassifier

- Objetivo: A função grid_search_random_forest visa encontrar os melhores hiperparâmetros para o modelo RandomForestClassifier.
- Utilizando ajuste de hiperparâmetros para otimizar o desempenho do modelo RandomForestClassifier. A função grid_search_random_forest realiza essa tarefa ao definir uma grade de parâmetros, executar a busca e retornar os melhores parâmetros encontrados. Isso é crucial para melhorar a precisão do modelo e encontrar a configuração ideal de parâmetros que melhor se adapte aos dados fornecidos.

[45]:
```
# !python -m spacy download en_core_web_sm
```

```python
import spacy
# Carregar o modelo spaCy localmente
spacy.cli.download("en_core_web_sm")
nlp = spacy.load('en_core_web_sm')
```

```
Collecting en-core-web-sm==3.7.1
  Using cached https://github.com/explosion/spacy-
models/releases/download/en_core_web_sm-3.7.1/en_core_web_sm-3.7.1-py3-none-
any.whl (12.8 MB)
Requirement already satisfied: spacy<3.8.0,>=3.7.2 in
/opt/conda/lib/python3.11/site-packages (from en-core-web-sm==3.7.1) (3.7.5)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.0.9)
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (8.2.5)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.4.8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.4.1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.12.3)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (4.66.2)
```

```
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.7.4)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.11/site-packages
(from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.1.3)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.11/site-
packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (69.2.0)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (24.0)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in
/opt/conda/lib/python3.11/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.4.0)
Requirement already satisfied: numpy>=1.19.0 in /opt/conda/lib/python3.11/site-
packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.26.4)
Requirement already satisfied: language-data>=1.2 in
/opt/conda/lib/python3.11/site-packages (from
langcodes<4.0.0,>=3.2.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.2.0)
Requirement already satisfied: annotated-types>=0.4.0 in
/opt/conda/lib/python3.11/site-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.7.0)
Requirement already satisfied: pydantic-core==2.18.4 in
/opt/conda/lib/python3.11/site-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.18.4)
Requirement already satisfied: typing-extensions>=4.6.1 in
/opt/conda/lib/python3.11/site-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (4.10.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.11/site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.11/site-
packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.11/site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.11/site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2024.2.2)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in
/opt/conda/lib/python3.11/site-packages (from
thinc<8.3.0,>=8.2.2->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.7.11)
```

```
Requirement already satisfied: confection<1.0.0,>=0.0.1 in
/opt/conda/lib/python3.11/site-packages (from
thinc<8.3.0,>=8.2.2->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.1.5)
Requirement already satisfied: click>=8.0.0 in /opt/conda/lib/python3.11/site-
packages (from typer<1.0.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1)
(8.1.7)
Requirement already satisfied: shellingham>=1.3.0 in
/opt/conda/lib/python3.11/site-packages (from
typer<1.0.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in /opt/conda/lib/python3.11/site-
packages (from typer<1.0.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1)
(13.7.1)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in
/opt/conda/lib/python3.11/site-packages (from
weasel<0.5.0,>=0.1.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.18.1)
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in
/opt/conda/lib/python3.11/site-packages (from
weasel<0.5.0,>=0.1.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (7.0.4)
Requirement already satisfied: MarkupSafe>=2.0 in
/opt/conda/lib/python3.11/site-packages (from jinja2->spacy<3.8.0,>=3.7.2->en-
core-web-sm==3.7.1) (2.1.5)
Requirement already satisfied: marisa-trie>=0.7.7 in
/opt/conda/lib/python3.11/site-packages (from language-
data>=1.2->langcodes<4.0.0,>=3.2.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1)
(1.2.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/opt/conda/lib/python3.11/site-packages (from
rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1)
(3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/opt/conda/lib/python3.11/site-packages (from
rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1)
(2.17.2)
Requirement already satisfied: wrapt in /opt/conda/lib/python3.11/site-packages
(from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.1.0->spacy<3.8.0,>=3.7.2->en-
core-web-sm==3.7.1) (1.16.0)
Requirement already satisfied: mdurl~=0.1 in /opt/conda/lib/python3.11/site-
packages (from markdown-it-
py>=2.2.0->rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.1.2)
 Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
 Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
```

[46]:
```
import string
from spacy.lang.en.stop_words import STOP_WORDS
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import confusion_matrix
import pickle
import pandas as pd
from typing import Tuple, List, Dict, Any

def load_spacy_model() -> Tuple[spacy.language.Language, List[str]]:
    """
    Carrega o modelo spaCy e a lista de stopwords.

    Retorna:
        Tupla contendo o modelo NLP do spaCy e a lista de stopwords.
    """
    nlp = spacy.load('en_core_web_sm')
    stopwords = list(STOP_WORDS)
    return nlp, stopwords

def tokeniser(sentence: str, nlp: spacy.language.Language, stopwords:␣
 ↪List[str], punctuations: str) -> List[str]:
    """
    Tokeniza a sentença de entrada usando spaCy, removendo stopwords e␣
 ↪pontuações.

    Args:
        sentence: O texto de entrada a ser tokenizado.
        nlp: O modelo NLP do spaCy.
        stopwords: Lista de stopwords para remover.
        punctuations: String de pontuações para remover.

    Retorna:
        Lista de tokens processados.
    """
    doc = nlp(sentence)
    tokens = [token.lemma_.lower().strip() if token.lemma_ != "-PRON-" else␣
 ↪token.lower_ for token in doc]
    tokens = [token for token in tokens if token not in stopwords and token not␣
 ↪in punctuations]
    return tokens

def custom_tokeniser(text: str) -> List[str]:
    """
```

```
    Tokeniza o texto usando a função tokeniser com nlp, stopwords e pontuações␣
 ↪globais.

    Args:
        text: O texto de entrada a ser tokenizado.

    Retorna:
        Lista de tokens processados.
    """
    nlp, stopwords = load_spacy_model()
    punctuations = string.punctuation.replace('/', '').replace('-', '')
    return tokeniser(text, nlp, stopwords, punctuations)

def split_data(df: pd.DataFrame) -> Tuple[pd.Series, pd.Series, pd.Series, pd.
 ↪Series]:
    """
    Divide o DataFrame em conjuntos de treinamento e teste.

    Args:
        df: DataFrame contendo os dados de texto e os rótulos de classe.

    Retorna:
        Tupla contendo os conjuntos de treinamento e teste para recursos e␣
 ↪rótulos.
    """
    return train_test_split(df['text'], df['class_number'], test_size=0.2,␣
 ↪random_state=678)

def train_naive_bayes(X_train: pd.Series, y_train: pd.Series, tfvectorizer:␣
 ↪TfidfVectorizer) -> Pipeline:
    """
    Cria e treina um modelo Naive Bayes.

    Args:
        X_train: Conjunto de treinamento de recursos.
        y_train: Conjunto de treinamento de rótulos.
        tfvectorizer: Vetorizador TF-IDF.

    Retorna:
        Pipeline do modelo Naive Bayes treinado.
    """
    classifier_NB = MultinomialNB()
    model_pipe_NB = Pipeline([('vectorizer', tfvectorizer), ('classifier',␣
 ↪classifier_NB)])
    model_pipe_NB.fit(X_train, y_train)
    return model_pipe_NB
```

```python
def save_model(model: Pipeline, filename: str) -> None:
    """
    Salva um modelo treinado usando pickle.

    Args:
        model: O pipeline do modelo treinado.
        filename: O nome do arquivo para salvar o modelo.
    """
    with open(filename, 'wb') as file:
        pickle.dump(model, file)

def load_model(filename: str) -> Pipeline:
    """
    Carrega um modelo salvo usando pickle.

    Args:
        filename: O nome do arquivo do modelo salvo.

    Retorna:
        O pipeline do modelo carregado.
    """
    with open(filename, 'rb') as file:
        return pickle.load(file)

def evaluate_model(model: Pipeline, X_train: pd.Series, y_train: pd.Series,␣
 ↪X_test: pd.Series, y_test: pd.Series) -> Tuple[float, float, Any, Any, Any,␣
 ↪Any]:
    """
    Avalia um modelo e retorna a acurácia de treinamento, acurácia de teste e␣
 ↪as matrizes de confusão para os conjuntos de treinamento e teste.
    Args:
        model: O pipeline do modelo treinado.
        X_train: Conjunto de treinamento de recursos.
        y_train: Conjunto de treinamento de rótulos.
        X_test: Conjunto de teste de recursos.
        y_test: Conjunto de teste de rótulos.
    Retorna:
        Tupla contendo a acurácia de treinamento, acurácia de teste, matriz de␣
 ↪confusão do treinamento e matriz de confusão do teste.
    """
    train_accuracy = model.score(X_train, y_train)
    test_accuracy = model.score(X_test, y_test)
    preds_train = model.predict(X_train)
    preds_test = model.predict(X_test)
    conf_matrix_train = confusion_matrix(y_train, preds_train)
    conf_matrix_test = confusion_matrix(y_test, preds_test)
```

```
    return train_accuracy, test_accuracy, conf_matrix_train, conf_matrix_test

def grid_search_random_forest(X_train: pd.Series, y_train: pd.Series,␣
↪tfvectorizer: TfidfVectorizer) -> Tuple[GridSearchCV, Dict[str, Any], float]:
    """
    - Realiza uma busca em grade para encontrar os melhores hiperparâmetros␣
↪para RandomForestClassifier.
    - Objetivo: A função grid_search_random_forest visa encontrar os melhores␣
↪hiperparâmetros para o modelo RandomForestClassifier.
    - Modelo de Pipeline: É criado um pipeline model_pipe_RF que combina o␣
↪vetorizador tfvectorizer com o classificador RandomForestClassifier.
    - Parâmetros da Grade: A variável grid_param define uma grade de parâmetros␣
↪para o RandomForestClassifier. Neste caso, são testados diferentes valores␣
↪para n_estimators, criterion e bootstrap.
    - GridSearchCV: É utilizado o GridSearchCV para explorar todas as␣
↪combinações possíveis de parâmetros definidos em grid_param. Ele utiliza␣
↪validação cruzada (cv=3) para avaliar o desempenho do modelo em diferentes␣
↪divisões dos dados de treinamento.
    - Melhores Parâmetros e Pontuação: Após a busca, GridSearchCV retorna os␣
↪melhores parâmetros (best_params_) encontrados durante a busca e a melhor␣
↪pontuação (best_score_) obtida com esses parâmetros.
    Args:
        X_train: Conjunto de treinamento de recursos.
        y_train: Conjunto de treinamento de rótulos.
        tfvectorizer: Vetorizador TF-IDF.

    Retorna:
        Tupla contendo o objeto GridSearchCV, melhores parâmetros e melhor␣
↪pontuação.
    """
    classifier_RF = RandomForestClassifier()
    model_pipe_RF = Pipeline([('vectorizer', tfvectorizer), ('classifier',␣
↪classifier_RF)])

    grid_param = {
        'classifier__n_estimators': [50, 75, 100, 120, 200],
        'classifier__criterion': ['gini', 'entropy'],
        'classifier__bootstrap': [True, False]
    }

    gd_sr = GridSearchCV(estimator=model_pipe_RF, param_grid=grid_param,␣
↪scoring='accuracy', cv=3, n_jobs=-1)
    gd_sr.fit(X_train, y_train)

    best_parameters = gd_sr.best_params_
    best_score = gd_sr.best_score_
```

```
    return gd_sr, best_parameters, best_score

def main_comparison(df: pd.DataFrame) -> None:
    """
    Função principal para executar a comparação entre os modelos Naive Bayes e␣
 ↪RandomForest.

    Args:
        df: DataFrame contendo os dados de texto e os rótulos de classe.
    """
    tfvectorizer = TfidfVectorizer(tokenizer=custom_tokeniser)

    X_train, X_test, y_train, y_test = split_data(df)

    # Naive Bayes
    NB_model_5cat_data = 'NB_model_5cat_data.sav'
    if not os.path.exists(NB_model_5cat_data):
        model_pipe_NB = train_naive_bayes(X_train, y_train, tfvectorizer)
        save_model(model_pipe_NB, NB_model_5cat_data)
    loaded_model_NB = load_model(NB_model_5cat_data)

    nb_train_accuracy, nb_test_accuracy, conf_matrix_train_NB,␣
 ↪conf_matrix_test_NB = evaluate_model(loaded_model_NB, X_train, y_train,␣
 ↪X_test, y_test)

    print("Acurácia de Treinamento do Naive Bayes: ", nb_train_accuracy)
    print("Acurácia de Teste do Naive Bayes: ", nb_test_accuracy)
    print("Matriz de Confusão para o Treinamento do Naive Bayes:\n",␣
 ↪conf_matrix_train_NB)
    print("Matriz de Confusão para o Teste do Naive Bayes:\n",␣
 ↪conf_matrix_test_NB)

    # Random Forest com Busca em Grade
    RF_model_5cat_data = 'RF_model_5cat_data.sav'
    if not os.path.exists(RF_model_5cat_data):
        gd_sr, best_parameters, best_score = grid_search_random_forest(X_train,␣
 ↪y_train, tfvectorizer)
        model_pipe_RF = gd_sr.best_estimator_
        save_model(model_pipe_RF, RF_model_5cat_data)
    else:
        model_pipe_RF = load_model(RF_model_5cat_data)

    rf_train_accuracy, rf_test_accuracy, conf_matrix_train_RF,␣
 ↪conf_matrix_test_RF = evaluate_model(model_pipe_RF, X_train, y_train,␣
 ↪X_test, y_test)
```

```
    print("Acurácia de Treinamento do RandomForest: ", rf_train_accuracy)
    print("Acurácia de Teste do RandomForest: ", rf_test_accuracy)
    print("Matriz de Confusão para o Treinamento do RandomForest:\n",␣
 ↪conf_matrix_train_RF)
    print("Matriz de Confusão para o Teste do RandomForest:\n",␣
 ↪conf_matrix_test_RF)


# Executar a comparação
main_comparison(df)
```

```
Acurácia de Treinamento do Naive Bayes:  0.952247191011236
Acurácia de Teste do Naive Bayes:  0.951310861423221
Matriz de Confusão para o Treinamento do Naive Bayes:
 [[391   1  22   4   6]
 [  4 421   8   0   3]
 [  2   0 412   3   8]
 [  5   1   9 407   1]
 [  7   2  12   4 403]]
Matriz de Confusão para o Teste do Naive Bayes:
 [[101   0   7   1   1]
 [  0  95   2   0   1]
 [  0   0 108   0   1]
 [  5   1   1 103   1]
 [  1   0   4   0 101]]
Acurácia de Treinamento do RandomForest:  0.9915730337078652
Acurácia de Teste do RandomForest:  0.9925093632958801
Matriz de Confusão para o Treinamento do RandomForest:
 [[416   0   0   4   4]
 [  0 434   1   0   1]
 [  0   2 420   0   3]
 [  1   0   0 420   2]
 [  0   0   0   0 428]]
Matriz de Confusão para o Teste do RandomForest:
 [[109   0   0   1   0]
 [  0  97   0   0   1]
 [  0   0 108   0   1]
 [  1   0   0 110   0]
 [  0   0   0   0 106]]
```

Cada linha da matriz representa as instâncias reais de uma classe, enquanto cada coluna representa as instâncias previstas de uma classe.