Universidade Presbiteriana Mackenzie

CURSO: **Tecnologia em Ciência de dados**

POLO DE APOIO PRESENCIAL: **Polo EAD SP – Polo EAD Goiânia**

SEMESTRE: **03**

COMPONENTE CURRICULAR / TEMA: **Projeto Aplicado- 03A**

NOME COMPLETO DOS ALUNOS:

**Nome:** Rafael Passos

**RA:** 10415447

**E-mail**: 10415447@mackenzista.com.br

**Nome**: Silas de Souza Ferreira

**RA:** 10414793

**E-mail:** 10414793@mackenzista.com.br

**Nome:** Israel Soares do Nascimento Viana

**RA:** 10414894

**E-mail:** 10414894@mackenzista.com.br

**Nome:** Gustavo Silva Rios

**RA:** 10415824

**E-mail:** 10415824@mackenzista.com.br

NOME DO PROFESSOR**:** **Felipe Albino dos Santos**

Universidade Presbiteriana Mackenzie

# Sumário

Universidade Presbiteriana Mackenzie

# Premissas, Objetivos e metas

Nosso objetivo é desenvolver um modelo de sistema de classificação e recomendação de notícias, utilizando técnicas de Ciência de Dados para automatizar a classificação de conteúdo informativo e proporcionar aos usuários uma experiência personalizada e alinhada com suas preferências individuais.

Esse projeto inclui o desenvolvimento de um modelo de aprendizado de máquina para classificar notícias em categorias específicas. Pretendemos garantir que o modelo lide com diferentes estilos de escrita e tópicos variados. Além disso, vamos criar um sistema de recomendação personalizada, permitindo aos usuários informar suas preferências. Também estamos integrando um mecanismo de feedback do usuário para ajustes contínuos, utilizando dados de interação para melhorar futuras recomendações.

Como meta, planejamos treinar o modelo inserindo notícias, desafiando-o a retornar à categoria correspondente e avaliar a relevância para o usuário específico.

Ao final, o projeto visa oferecer uma experiência informativa e personalizada, evoluindo de acordo com as dinâmicas preferências dos usuários.

Universidade Presbiteriana Mackenzie

# Cronograma das atividades

- **06/03/2024**

- Entrega do documento inicial do projeto em PDF, contendo os integrantes do grupo, as premissas e objetivos do projeto, a área e o modelo escolhido e o cronograma das atividades seguintes.

- **03/04/2024**

- Definição das bibliotecas do Python que serão utilizadas, criação do repositório no Github.

- Definição e apresentação da base de dados que será utilizada, tratamento e limpeza do arquivo, além da realização da análise exploratória.

- Preparação e treinamento do modelo.

- Definição e descrição das bases teóricas e métodos analíticos.

- Definição e descrição de como será calculada a acurácia.

- **27/04/2024**

- Consolidação e validação dos resultados para os métodos analíticos.

- Verificar o desempenho do modelo com aplicação das medidas de acurácia.

- Descrição dos resultados preliminares do projeto e esboço do storytelling final.

- **31/05/2023**

- Entrega de todo o relatório técnico do projeto atualizado, e repositório no github contendo o modelo construído.

- Apresentação do storytelling final em arquivo PPT ou outro similar.

- Postar vídeo com apresentação final do projeto no Youtube.

Universidade Presbiteriana Mackenzie

# Bibliotecas utilizadas

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import plotly.graph_objects as go
        import plotly.express as px
        import re
```

Utilizamos algumas bibliotecas muito populares como a “**pandas**” que é utilizada para manipulação dos dataframes, “**seaborn**” para criação de gráficos, dentre outras que são bastante utilizadas.

```
from sklearn.metrics import classification_report
```

Fizemos a importação do pacote **sci-kit learn** para desenvolver o modelo, e verificar suas métricas como por exemplo a acurácia

Todo o projeto se encontra no github e pode ser acessado através do link abaixo:

## https://github.com/gustavosrios/mackprojects-semestre-3

Universidade Presbiteriana Mackenzie

# Entendimento dos dados

Verificando os dados

```
[3]: df.head(1)
```

[3]:

| | link | headline | category | short_description | authors | date |
|---|---|---|---|---|---|---|
| 0 | https://www.huffpost.com/entry/covid-boosters-... | Over 4 Million Americans Roll Up Sleeves For O... | U.S. NEWS | Health experts said it is too early to predict... | Carla K. Johnson, AP | 2022-09-23 |

Verificando informações gerais dos tipos de dados, quantidade de variáveis e instâncias.

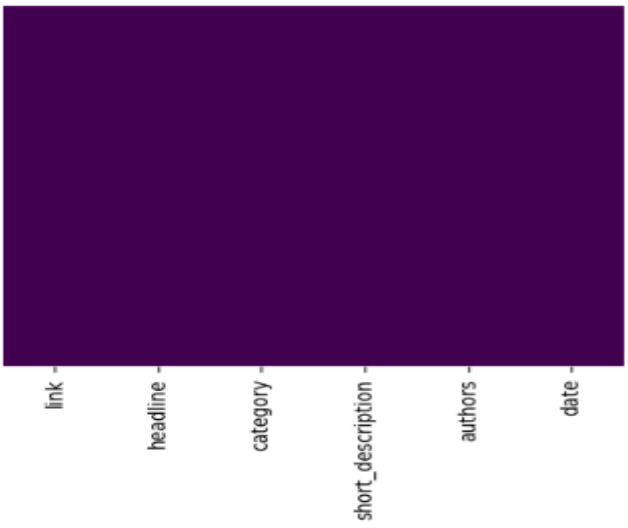
```
[4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209527 entries, 0 to 209526
Data columns (total 6 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   link               209527 non-null  object
 1   headline           209527 non-null  object
 2   category           209527 non-null  object
 3   short_description  209527 non-null  object
 4   authors            209527 non-null  object
 5   date               209527 non-null  datetime64[ns]
dtypes: datetime64[ns](1), object(5)
memory usage: 9.6+ MB
```

Verificando dados nulos

```
[6]: plt.figure(figsize=(5,4))
sns.heatmap(df.isnull(),
            yticklabels=False,
            cbar=False,
            cmap='viridis')
df.isnull().sum()
```

```
[6]: link                0
headline            0
category            0
short_description   0
authors             0
date                0
dtype: int64
```



## Verificando dados infinitos

In [7]:
```
inf_mask = np.isinf(df.select_dtypes(include=['number']))
inf_mask.sum()
```

Out[7]:
```
Series([], dtype: float64)
```

## Verificando instâncias duplicadas

[8]:
```
df.duplicated().sum()
```

[8]:
```
13
```

## Verificação de Outliers

- Como esta base não possui dados numéricos, verificações por meio de IQR e Z-score não poderão ser feitas.

## Verificação de Strings Vazias ou Espaços em Branco

In [9]:
```
for col in df.columns:
    # Verifica se a coluna é do tipo object, o que normalmente indica strings
    if df[col].dtype == 'object':
        # Conta quantas vezes os valores são strings vazias ou espaços em branco
        empty_values = df[df[col].str.strip() == ''].shape[0]
        print(f"Número de '{col}' com valores não especificados (strings vazias ou espaços): {empty_values}")
```

```
Número de 'link' com valores não especificados (strings vazias ou espaços): 0
Número de 'headline' com valores não especificados (strings vazias ou espaços): 6
Número de 'category' com valores não especificados (strings vazias ou espaços): 0
Número de 'short_description' com valores não especificados (strings vazias ou espaços): 19712
Número de 'authors' com valores não especificados (strings vazias ou espaços): 37418
```

## Verificação dos links das notícias

In [10]:
```
# Extrair o domínio do URL
df['website'] = df['link'].apply(lambda x: re.findall(r'(?://www\.|http://www\.|https://www\.|http://|https://)?([^/

# Contar a quantidade de vezes que cada website aparece nos dados
website_counts = df['website'].value_counts()
website_counts
```

Out[10]:
```
website
huffingtonpost.com          190830
huffingtonpost.comhttp:       9278
huffpost.com                  8673
huffingtonpost.comhttps:       745
huffingtonpost.in                1
Name: count, dtype: int64
```

Universidade Presbiteriana Mackenzie

# Limpeza e tratamento dos dados

- Nesta etapa será feito o tratamento das linhas duplicadas encontradas na etapa do Entendimento dos Dados.

**Eliminando as instâncias duplicadas mantendo a primeira ocorrência de cada duplicata**

In [11]:
```
linhas_duplicadas = df.duplicated()

# Filtrar o DataFrame original para mostrar apenas as linhas duplicadas
linhas_duplicadas_df = df.loc[linhas_duplicadas]
linhas_duplicadas_df.head(2)
```

Out[11]:

| | link | headline | category | short_description | authors | date | website |
|---|---|---|---|---|---|---|---|
| 67677 | https://www.huffingtonpost.comhttp://www.mothe... | On Facebook, Trump's Longtime Butler Calls For... | POLITICS | Anthony Senecal, who worked as Donald Trump's ... | | 2016-05-12 | huffingtonpost.comhttp: |
| 67923 | https://www.huffingtonpost.comhttp://gizmodo.c... | Former Facebook Workers: We Routinely Suppress... | TECH | Facebook workers routinely suppressed news sto... | | 2016-05-09 | huffingtonpost.comhttp: |

In [12]:
```
df = df.drop_duplicates(keep='first')

# Verificando alterações
df.duplicated().sum()
```

Out[12]:
```
0
```

- Será feito também a remoção dos links em que aparecem dois endereços http/https.

**Eliminar tudo depois do .com, seja http ou https**

In [13]:
```
df['website'] = df['website'].str.replace(r'(\.com).*', r'\1', regex=True)

# Contar a quantidade de vezes que cada website aparece nos dados
website_counts = df['website'].value_counts()
website_counts
```

Out[13]:
```
website
huffingtonpost.com    200840
huffpost.com            8673
huffingtonpost.in          1
Name: count, dtype: int64
```

- Será atribuído 'Not Specified' a todos os valores com strings vazias ('') ou espaços em branco (' ').

**Substituir strings vazias ou espaços em branco por 'Not Specified' usando replace**

In [14]:
```
df.replace({'': 'Not Specified', ' ': 'Not Specified'}, inplace=True)

# Verificando se as mudanças ocorreram corretamente.
for col in df.columns:
    # Verifica se a coluna é do tipo object, o que normalmente indica strings
    if df[col].dtype == 'object':
        # Conta quantas vezes os valores são strings vazias ou espaços em branco
        empty_values = df[df[col].str.strip() == ''].shape[0]
        print(f"Número de '{col}' com valores não especificados (strings vazias ou espaços): {empty_values}")
```

```
Número de 'link' com valores não especificados (strings vazias ou espaços): 0
Número de 'headline' com valores não especificados (strings vazias ou espaços): 0
Número de 'category' com valores não especificados (strings vazias ou espaços): 0
Número de 'short_description' com valores não especificados (strings vazias ou espaços): 0
Número de 'authors' com valores não especificados (strings vazias ou espaços): 0
Número de 'website' com valores não especificados (strings vazias ou espaços): 0
```

- Foi observado um padrão no nome dos autores por exemplo : "Alfred W. McCoy, ContributorProfessor, University of Wisconsin-Madison; Author,...". O padrão é (nome autor vírgula descrição do autor) será dividido a string dos autores em duas outras variáveis, nome do autor e descrição do autor.

**Função para dividir a string em duas partes e retornar o nome do autor e a informação**

In [15]:
```
# Filtrar o DataFrame para os valores na coluna 'authors' que contêm a palavra 'Contributor'
authors_with_collaboration = df[df['authors'].str.contains('Contributor', case=False)]

# Exibir os valores únicos na coluna 'authors' que contêm a palavra 'collaboration'
unique_authors_with_collaboration = authors_with_collaboration['authors'].unique()

print(f"Quantidade de Autores com a palavra contributor: {len(unique_authors_with_collaboration)}\n")
print("Autores que contêm a palavra 'Contributor':\n")

for author in unique_authors_with_collaboration[10:20]:
    print(author)
# criar nova coluna com os dados apos a virgula
```

```
Quantidade de Autores com a palavra contributor: 21972

Autores que contêm a palavra 'Contributor':

Alfred W. McCoy, ContributorProfessor, University of Wisconsin-Madison; Author, 'In the Sh...
Harry Lewis, ContributorRadical queer feminist, #BlackLivesMatter supporter, student a...
Michael Raver, ContributorActor, Writer
Fatherly , ContributorFatherly is a publication for modern fathers looking to make t...
The Mighty, ContributorWe face disability, disease and mental illness together.
Susu Sarah Amer , Contributor99 problems but spice aint one of them
Richard (RJ) Eskow, ContributorHost of 'The Zero Hour'; Writer; Senior Advisor, Social Securi...
Mike Weisser, ContributorMike the Gun Guy
Gerardo M. González, ContributorDean Emeritus and Professor, Indiana University School Of Educ...
Mina Schultz, ContributorOutreach Specialist, GetCoveredNYC
```

In [16]:
```
# Função para dividir a string em duas partes e retornar o nome do autor e a informação
def split_author_info(author_str):
    if ',' in author_str:
        parts = author_str.split(',', 1)
        name = parts[0].strip()
        info = parts[1].strip()
        return name, info
    else:
        return author_str.strip(), ''

# Aplicar a função a todas as linhas da coluna 'authors' e expandir em duas colunas
df[['authors name', 'authors information']] = df['authors'].apply(lambda x: pd.Series(split_author_info(x)))

df.drop('authors', axis=1, inplace=True)

df.sample(3)
```

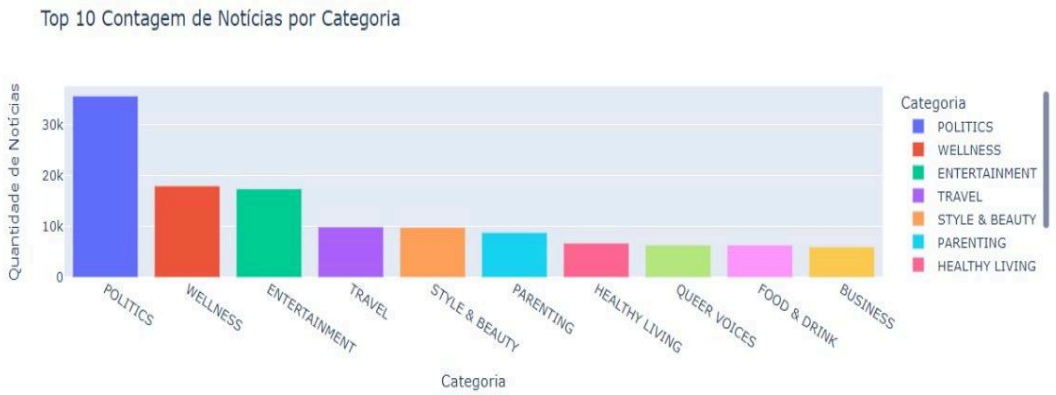
Universidade Presbiteriana Mackenzie

# Análise Exploratória de Dados

Abaixo é possível verificar a quantidade de notícias por cada categoria, permitindo saber a distribuição de temas que são mais pesquisados

## Nesta etapa serão criadas visualizações com base no label (categoria)

[15]:
```
import plotly.express as px

# Contagem de notícias por categoria
category_counts = df['category'].value_counts()

# Criando o gráfico de barras com cores diferentes para cada categoria
fig = px.bar(x=category_counts.index[0:10], y=category_counts.values[0:10], color=category_counts.index[0:10],
             labels={'x':'Categoria', 'y':'Quantidade de Notícias', 'color':'Categoria'},
             title='Top 10 Contagem de Notícias por Categoria')

# Exibindo o gráfico
fig.show()

category_counts.head(10)
```




[15]:
```
category
POLITICS          35601
WELLNESS          17942
ENTERTAINMENT     17362
TRAVEL             9900
STYLE & BEAUTY     9811
PARENTING          8791
HEALTHY LIVING     6694
QUEER VOICES       6347
FOOD & DRINK       6340
BUSINESS           5992
```

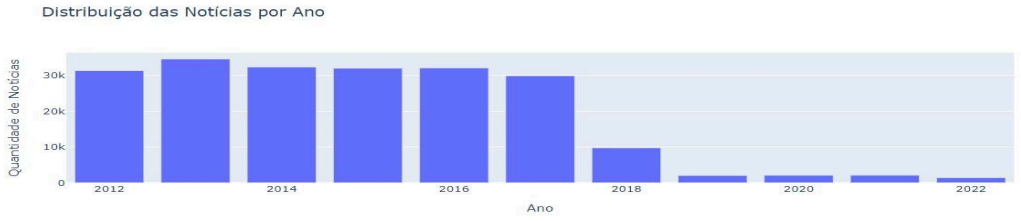
[25]:
```
# Agrupe os dados por ano e por categoria, e conte a quantidade de notícias em cada grupo
news_counts_by_year_category = df.groupby([df['date'].dt.year, 'category']).size().reset_index(name='count')

# Encontre a categoria com a maior contagem de notícias para cada ano
max_category_by_year = news_counts_by_year_category.loc[news_counts_by_year_category.groupby('date')['count'].idxmax()]

# Criar o gráfico de barras
fig = px.bar(max_category_by_year, x='date', y='count', color='category',
             labels={'date':'Ano', 'count':'Quantidade de Notícias', 'category':'Categoria'},
             title='Categoria com Maior Contagem de Notícias por Ano')

# Exibir o gráfico
fig.show()
print(f'Categorias de maior destaque por ano: \n{max_category_by_year}')
```


```
Categorias de maior destaque por ano:
     date  category  count
20   2012  WELLNESS   7137
41   2013  WELLNESS   8028
65   2014  POLITICS   3182
97   2015  POLITICS   5887
127  2016  POLITICS   9989
157  2017  POLITICS  10613
191  2018  POLITICS   3399
221  2019  POLITICS    507
249  2020  POLITICS    802
274  2021  POLITICS    823
299  2022  POLITICS    399
```

Com a classificação de categorias por cada ano, é possível ter uma noção do interesse em cada tema de notícias por período

Histograma da distribuição da quantidade de notícias por ano

[16]:
```
# Agrupe os dados por ano e conte a quantidade de notícias em cada ano
news_counts_by_year = df['date'].dt.year.value_counts().sort_index()

# Criar o histograma
fig = px.bar(x=news_counts_by_year.index, y=news_counts_by_year.values,
             labels={'x':'Ano', 'y':'Quantidade de Notícias'},
             title='Distribuição das Notícias por Ano')

# Exibir o gráfico
fig.show()
news_counts_by_year
```




[16]:
```
date
2012    31349
2013    34580
2014    32338
2015    32006
2016    32095
2017    29889
2018     9734
2019     2005
2020     2054
2021     2066
2022     1398
Name: count, dtype: int64
```

Com a quantidade de notícias por website é possível ter uma noção dos sites que são mais acessados

### Verificando a quantidade de noticias para cada website

- Utilizaremos regular expressions para extrair apenas o nome do site que está após o http:// e faremos uma contagem dos valores unicos encontrados.

[17]:
```
# Contar a quantidade de vezes que cada website aparece nos dados
website_counts = df['website'].value_counts()

# Criar o gráfico de barras
fig = px.bar(x=website_counts.index, y=website_counts.values,
             labels={'x':'Website', 'y':'Quantidade de Notícias'},
             title='Quantidade de Notícias por Website')

# Rotacionar os rótulos do eixo x para melhor legibilidade
fig.update_layout(xaxis_tickangle=-45)

# Exibir o gráfico
fig.show()
```


[18]:
```
website_counts
```

[18]:
```
website
huffingtonpost.com    200840
huffpost.com            8673
huffingtonpost.in          1
Name: count, dtype: int64
```

## Estatísticas descritivas

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

Out[5]:

| | date |
|---|---|
| **count** | 209527 |
| **mean** | 2015-04-30 00:44:14.344308736 |
| **min** | 2012-01-28 00:00:00 |
| **25%** | 2013-08-10 00:00:00 |
| **50%** | 2015-03-16 00:00:00 |
| **75%** | 2016-11-01 00:00:00 |
| **max** | 2022-09-23 00:00:00 |

- Nesta etapa serão criadas novas features por meio das técnicas de feature Engineering.

In [17]:
```
# Criando Novas Variáveis
df['day'] = df['date'].dt.day
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year
```

In [18]:
```
df.head(1)
```

Out[18]:

| | link | headline | category | short_description | date | website | authors name | authors information | day | month |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | https://www.huffpost.com/entry/covid-boosters-... | Over 4 Million Americans Roll Up Sleeves For O... | U.S. NEWS | Health experts said it is too early to predict... | 2022-09-23 | huffpost.com | Carla K. Johnson | AP | 23 | 9 |

In [19]:
```
# agrupando por categoria

group_by_category = df.groupby(['authors name']).count()
group_by_category
```

Out[19]:

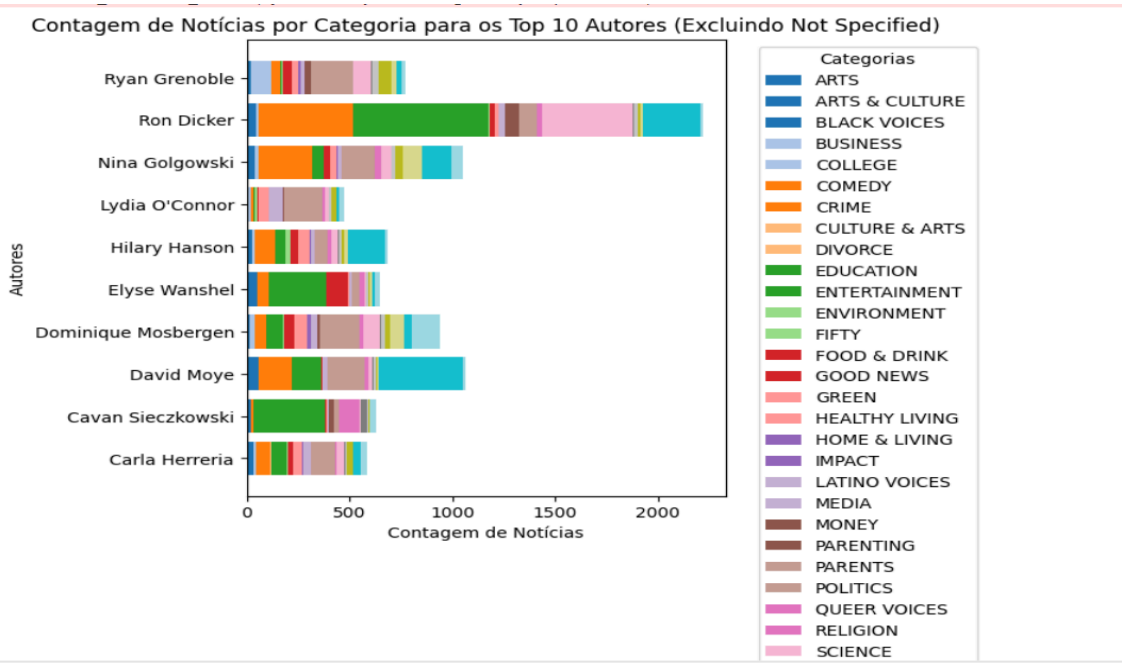
| authors name | link | headline | category | short_description | date | website | authors information | day | month | year |
|---|---|---|---|---|---|---|---|---|---|---|
| "Seth Eliot" Santoro | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| #DoctorsSpeakOut | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 'The Koch Sisters' | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| * | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| Éloïse Bouton | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| レシピブログ通信 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 吉野ゆりえ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 渋谷健司 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 딘 베이커 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |

25014 rows × 10 columns

- Serão criados gráficos para entendermos quais notícias estavam em alta nos períodos mensais e anuais com gráficos de barras e a distribuição dos dados nesses períodos por meio de boxplots.

In [20]:

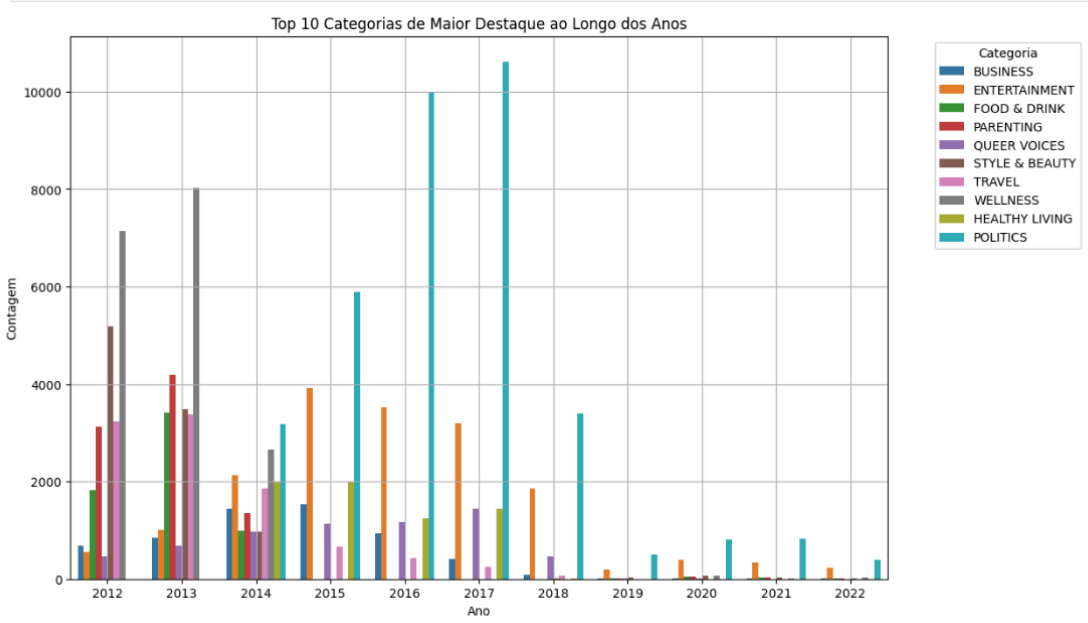
```
# Primeiro, vamos obter a contagem de notícias por autor e categoria
count_data = df.groupby(['authors name', 'category']).size().reset_index(name='counts')

# Filtrar para os top 10 autores com mais notícias (excluindo 'Not Specified')
top_authors = count_data[count_data['authors name'] != 'Not Specified']['authors name'].value_counts().nlargest(10).index
filtered_data = count_data[count_data['authors name'].isin(top_authors)]

# Ordenar os autores em ordem decrescente pela contagem de notícias
filtered_data.sort_values(by='counts', ascending=False, inplace=True)

# Pivotar os dados para obter um formato adequado para barras empilhadas
pivot_data = filtered_data.pivot(index='authors name', columns='category', values='counts').fillna(0)

# Agora, para plotar as barras empilhadas horizontalmente
fig, ax = plt.subplots(figsize=(8, 8))  # Ajuste o tamanho conforme necessário

colors = plt.cm.tab20(np.linspace(0, 1, len(pivot_data.columns)))
left = np.zeros(len(pivot_data))

for i, col in enumerate(pivot_data.columns):
    ax.barh(pivot_data.index, pivot_data[col], left=left, label=col, color=colors[i])
    left += pivot_data[col].values

ax.set_ylabel('Autores')
ax.set_xlabel('Contagem de Notícias')
ax.set_title('Contagem de Notícias por Categoria para os Top 10 Autores (Excluindo Not Specified)')
ax.legend(title='Categorias', bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()  # Ajusta o layout para não cortar elementos
plt.show()
```



## Visualizações Temporais

In [24]:

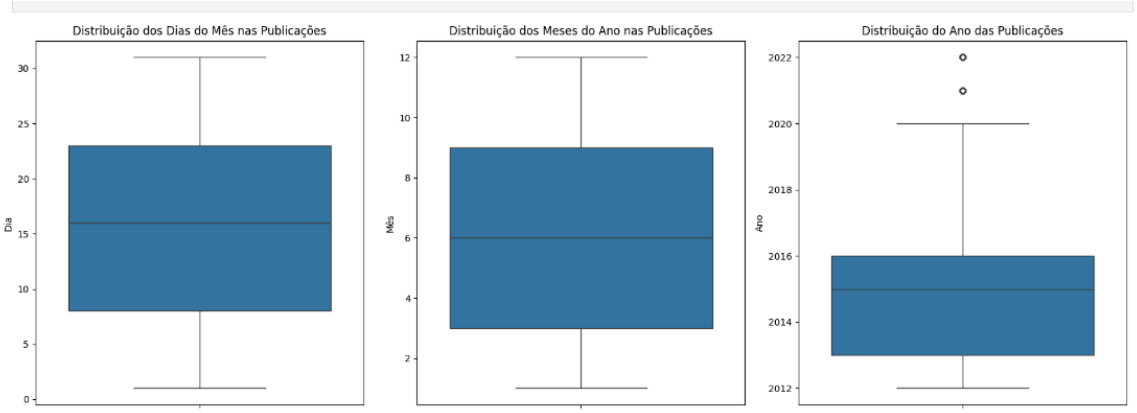
```
# Contando o número de ocorrências de cada categoria
category_counts = df['category'].value_counts()

# Selecionando apenas as top 10 categorias
top_categories = category_counts.head(10).index.tolist()

# Filtrando o DataFrame para incluir apenas as top 10 categorias
filtered_df = df[df['category'].isin(top_categories)]

# Agrupando os dados filtrados por ano e categoria para contar o número de ocorrências
grouped_data = filtered_df.groupby(['year', 'category']).size().reset_index(name='count')

# Criando a visualização usando Seaborn (gráfico de barras)
plt.figure(figsize=(12, 8))
sns.barplot(data=grouped_data, x='year', y='count', hue='category')
plt.title('Top 10 Categorias de Maior Destaque ao Longo dos Anos')
plt.xlabel('Ano')
plt.ylabel('Contagem')
plt.legend(title='Categoria', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
plt.show()
```



## Boxplots - entendendo a distribuição dos dados.

In [26]:

```
# Configura o tamanho geral da figura
plt.figure(figsize=(17, 6))

# Cria o subplot para a distribuição dos dias do mês
plt.subplot(1, 3, 1)  # (nrows, ncols, index)
sns.boxplot(data=df, y='day', orient='v')
plt.title('Distribuição dos Dias do Mês nas Publicações')
plt.ylabel('Dia')

# Cria o subplot para a distribuição dos meses do ano
plt.subplot(1, 3, 2)
sns.boxplot(data=df, y='month', orient='v')
plt.title('Distribuição dos Meses do Ano nas Publicações')
plt.ylabel('Mês')

# Cria o subplot para a distribuição do ano das publicações
plt.subplot(1, 3, 3)
sns.boxplot(data=df, y='year', orient='v')
plt.title('Distribuição do Ano das Publicações')
plt.ylabel('Ano')

# Ajusta o layout para garantir uma boa visualização dos títulos e labels
plt.tight_layout()

# Exibe os gráficos
plt.show()
```



Universidade Presbiteriana Mackenzie

# Divisão dos dados em treino e teste

## 4. Divisão dos dados em Treino e Teste;

In [27]:
```
# Vamos usar apenas as colunas 'headline' e 'short_description'
# como features e 'category' como target
df['text'] = df['headline'] + ' ' + df['short_description']

# Limpeza básica do texto pode ser expandida conforme necessário
df['text'] = df['text'].str.lower().str.replace('[^\w\s]', '')
```

In [28]:
```
from sklearn.model_selection import train_test_split
```

In [29]:
```
# Variáveis Independentes
X = df['text']

# Variáveis Dependentes
y = df['category']
```

In [30]:
```
# Dividindo em Treino e Teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

## Vetorização

In [31]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [32]:
```
# Vetorização
vectorizer = CountVectorizer(stop_words='english')
vectorizer
```

Out[32]:
```
CountVectorizer(stop_words='english')
```
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [33]:
```
X_train_vect = vectorizer.fit_transform(X_train) # type scipy.sparse._csr.csr_matrix
X_test_vect = vectorizer.transform(X_test)
```

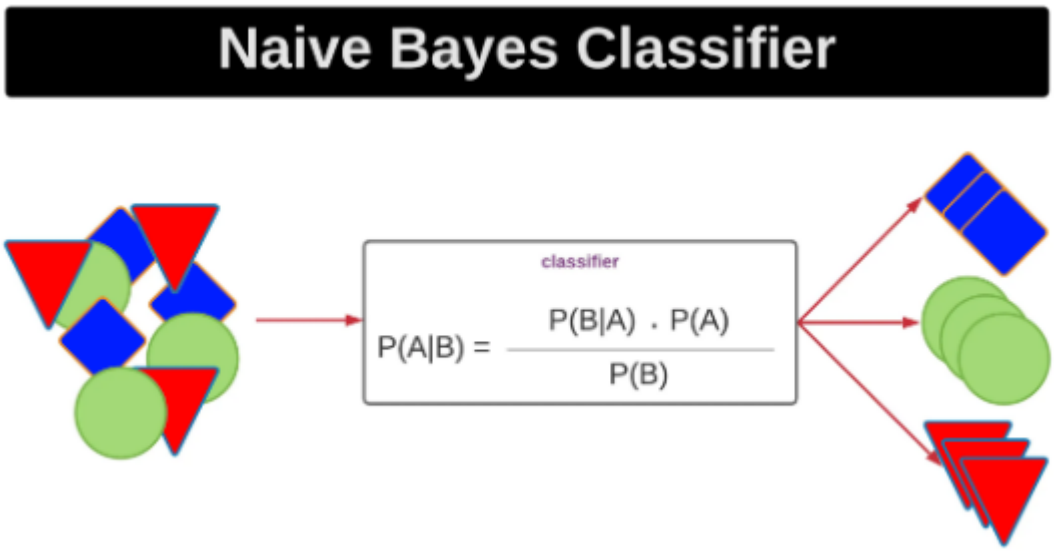
In [34]:
```
type(X_test_vect)
```

Out[34]:
```
scipy.sparse._csr.csr_matrix
```

Universidade Presbiteriana Mackenzie

# Modelagem de Diferentes Algoritmos de Classificação para encontrarmos o modelo que melhor se ajusta aos dados

## Modelo 01: Naive Bayes




**Teoria:**

Naive Bayes é um algoritmo criado para classificar os dados e assim torná-los disponíveis para otimizar o processo de tomada de decisão.

A fórmula utilizada nesse modelo é muito conhecida e utilizada em estatística, e consiste em calcular a probabilidade de um evento ocorrer, e após calculada essa probabilidade, ela é relacionada a outro evento. O escopo desse algoritmo apenas classifica em duas classes de probabilidade, alta ou baixa, e a partir daí é gerada uma tabela de probabilidades, onde o modelo separa e classifica essas informações.

Algumas aplicações desse modelo são classificações de e-mails como spam, previsões em tempo real, sistema de recomendações,etc.

Um ponto interessante é que esse algoritmo pode ser utilizado em conjunto com outras ferramentas, possibilitando assim que seu desempenho e resultados sejam potencializados.

## Execução do Modelo:

In [35]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [36]:
```
# Treinando o modelo Naive Bayes
naive_model = MultinomialNB()
naive_model.fit(X_train_vect, y_train) # (X_train vetorizado, true label)
```

Out[36]:
```
MultinomialNB()
```
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [37]:
```
from sklearn.metrics import classification_report
```

In [38]:
```
# Fazendo previsões no conjunto de teste
y_pred = naive_model.predict(X_test_vect)

# Avaliando o modelo
print(classification_report(y_test, y_pred, zero_division=0))
```

```
                precision    recall  f1-score   support

          ARTS       0.71      0.05      0.10       310
ARTS & CULTURE       0.67      0.02      0.03       255
  BLACK VOICES       0.60      0.20      0.30       911
      BUSINESS       0.47      0.41      0.44      1176
       COLLEGE       0.73      0.04      0.07       226
        COMEDY       0.61      0.23      0.33      1044
         CRIME       0.50      0.52      0.51       725
CULTURE & ARTS       0.76      0.11      0.19       205
       DIVORCE       0.88      0.51      0.65       689
     EDUCATION       0.67      0.03      0.06       196
 ENTERTAINMENT       0.50      0.81      0.62      3499
 ENVIRONMENT         0.76      0.09      0.16       279
         FIFTY       0.00      0.00      0.00       256
  FOOD & DRINK       0.62      0.74      0.67      1290
     GOOD NEWS       0.62      0.03      0.06       260
         GREEN       0.45      0.15      0.23       536
HEALTHY LIVING       0.56      0.18      0.28      1342
 HOME & LIVING       0.82      0.57      0.67       875
        IMPACT       0.49      0.17      0.25       711
 LATINO VOICES       1.00      0.02      0.04       220
         MEDIA       0.74      0.13      0.22       586
         MONEY       0.60      0.10      0.18       356
     PARENTING       0.41      0.68      0.51      1733
       PARENTS       0.60      0.07      0.12       778
      POLITICS       0.57      0.92      0.70      6992
 QUEER VOICES       0.72      0.51      0.60      1242
      RELIGION       0.80      0.22      0.34       533
       SCIENCE       0.73      0.27      0.40       400
        SPORTS       0.73      0.61      0.67       976
         STYLE       0.78      0.01      0.03       501
STYLE & BEAUTY       0.67      0.83      0.74      1978
         TASTE       0.43      0.01      0.01       437
          TECH       0.81      0.17      0.27       411
THE WORLDPOST        0.60      0.38      0.46       760
        TRAVEL       0.57      0.81      0.67      2072
      U.S. NEWS      0.80      0.01      0.03       269
      WEDDINGS       0.87      0.58      0.70       756
    WEIRD NEWS       0.66      0.09      0.16       564
      WELLNESS       0.47      0.87      0.61      3634
         WOMEN       0.69      0.12      0.20       712
    WORLD NEWS       0.58      0.24      0.34       665
     WORLDPOST       0.59      0.16      0.26       543

      accuracy                           0.56     41903
     macro avg       0.64      0.30      0.33     41903
  weighted avg       0.59      0.56      0.50     41903
```

ENTERTAINMENT:

**Precision:** 0.50

**Recall:** 0.81

**F1-score:** 0.62

**Support:** 3499

POLITICS:

**Precision:** 0.57

**Recall:** 0.92

**F1-score:** 0.70

**Support:** 6992

STYLE & BEAUTY:

**Precision:** 0.67

**Recall:** 0.83

**F1-score:** 0.74

**Support:** 1978

WELLNESS:

**Precision:** 0.47

**Recall:** 0.87

**F1-score:** 0.61

**Support:** 3634

Essas são as categorias que se destacam pelas métricas mais altas. Vamos fazer uma conclusão profissional com base nesses resultados:

Ao avaliar o desempenho do modelo de classificação em relação às categorias com maior destaque nas métricas, notamos padrões interessantes.

**Entretenimento (ENTERTAINMENT):** O modelo apresenta uma precisão moderada de 50%, indicando que metade das previsões estão corretas. No entanto, o recall é alto em 81%, o que sugere que o modelo consegue capturar a maioria dos exemplos positivos desta categoria. O F1-score de 62% indica um equilíbrio razoável entre precisão e recall.

**Política (POLITICS):** Este é um destaque notável, com uma precisão de 57% e um recall excepcionalmente alto de 92%. Isso sugere que o modelo é muito eficaz em identificar notícias políticas, capturando a grande maioria dos exemplos positivos desta categoria.

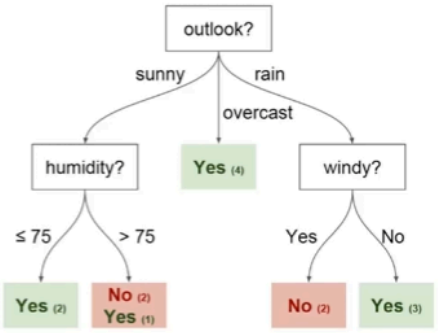
**Estilo & Beleza (STYLE & BEAUTY)**: Com uma precisão de 67% e um recall de 83%, esta categoria mostra um desempenho sólido do modelo em identificar notícias relacionadas a estilo e beleza.

**Bem-Estar (WELLNESS)**: Apesar da precisão um pouco mais baixa de 47%, o modelo demonstra um recall robusto de 87% para notícias relacionadas a bem-estar. Isso indica que o modelo é eficaz em identificar a grande maioria dos exemplos positivos nesta categoria.

Em geral, o modelo demonstra ***desempenho promissor em categorias importantes*** como entretenimento, política, estilo & beleza e bem-estar, com destaque especial para sua capacidade de identificar notícias políticas com alta precisão e recall. No entanto, é importante continuar refinando o modelo e explorar estratégias para melhorar a precisão em outras categorias para obter um desempenho mais equilibrado em todas as categorias.

Universidade Presbiteriana Mackenzie

# Modelo 02: Random Forest Classifier




## Teoria:

O Random Forest é um algoritmo de aprendizado de máquina supervisionado que se enquadra na categoria de métodos de ensemble, o que significa que ele combina várias técnicas de aprendizado para melhorar a precisão do modelo.

Ele é baseado na construção de múltiplas árvores de decisão durante o treinamento e, em seguida, faz uma previsão combinando as previsões de cada árvore individual.

Árvores de Decisão: As árvores de decisão são a base do Random Forest. Elas são estruturas de decisão em forma de árvore onde cada nó representa uma condição e cada ramo representa o resultado dessa condição.

O algoritmo divide o conjunto de dados em subconjuntos menores com base nas características dos dados.

Amostragem Aleatória: Durante o treinamento do Random Forest, várias amostras aleatórias do conjunto de dados são usadas para treinar cada árvore de decisão. Isso ajuda a reduzir a variância e o overfitting, pois cada árvore é treinada com uma visão diferente dos dados.

Combinação de Previsões: Após treinar todas as árvores de decisão, o Random Forest faz previsões combinando as previsões de cada árvore individual. No caso de problemas de classificação, por exemplo, a classe mais frequente prevista pelas árvores pode ser escolhida como a previsão final. No caso de regressão, a média das previsões das árvores pode ser usada.

Importância das Variáveis: O Random Forest também fornece uma medida de importância das variáveis, que ajuda a identificar quais características têm maior influência nas previsões do modelo. Isso é útil para entender quais características são mais relevantes para o problema em questão.

## Execução do Modelo:

- Este modelo utilizará os dados que foram previamente vetorizados

In [39]:
```
from sklearn.ensemble import RandomForestClassifier

# Criando e treinando o modelo Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_vect, y_train)
```

Out[39]: RandomForestClassifier(random_state=42)

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

### Fazendo previsões e avaliando o modelo

In [40]:
```
rf_y_pred = rf_model.predict(X_test_vect)
print(classification_report(y_test, rf_y_pred))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| ARTS | 0.54 | 0.16 | 0.25 | 310 |
| ARTS & CULTURE | 0.31 | 0.10 | 0.15 | 255 |
| BLACK VOICES | 0.45 | 0.22 | 0.30 | 911 |
| BUSINESS | 0.49 | 0.35 | 0.41 | 1176 |
| COLLEGE | 0.48 | 0.27 | 0.35 | 226 |
| COMEDY | 0.45 | 0.32 | 0.37 | 1044 |
| CRIME | 0.45 | 0.44 | 0.44 | 725 |
| CULTURE & ARTS | 0.86 | 0.21 | 0.34 | 205 |
| DIVORCE | 0.82 | 0.67 | 0.74 | 689 |
| EDUCATION | 0.36 | 0.13 | 0.19 | 196 |
| ENTERTAINMENT | 0.51 | 0.71 | 0.59 | 3499 |
| ENVIRONMENT | 0.89 | 0.09 | 0.16 | 279 |
| FIFTY | 0.46 | 0.07 | 0.12 | 256 |
| FOOD & DRINK | 0.51 | 0.74 | 0.61 | 1290 |
| GOOD NEWS | 0.40 | 0.07 | 0.11 | 260 |
| GREEN | 0.40 | 0.16 | 0.23 | 536 |
| HEALTHY LIVING | 0.32 | 0.23 | 0.26 | 1342 |
| HOME & LIVING | 0.66 | 0.61 | 0.63 | 875 |
| IMPACT | 0.46 | 0.07 | 0.12 | 711 |
| LATINO VOICES | 0.95 | 0.08 | 0.15 | 220 |
| MEDIA | 0.64 | 0.24 | 0.35 | 586 |
| MONEY | 0.56 | 0.15 | 0.24 | 356 |
| PARENTING | 0.48 | 0.61 | 0.53 | 1733 |
| PARENTS | 0.36 | 0.24 | 0.29 | 778 |
| POLITICS | 0.56 | 0.90 | 0.69 | 6992 |
| QUEER VOICES | 0.79 | 0.61 | 0.69 | 1242 |
| RELIGION | 0.61 | 0.34 | 0.43 | 533 |
| SCIENCE | 0.51 | 0.30 | 0.38 | 400 |
| SPORTS | 0.57 | 0.55 | 0.56 | 976 |
| STYLE | 0.49 | 0.23 | 0.31 | 501 |
| STYLE & BEAUTY | 0.70 | 0.77 | 0.73 | 1978 |
| TASTE | 0.29 | 0.08 | 0.13 | 437 |
| TECH | 0.51 | 0.30 | 0.38 | 411 |
| THE WORLDPOST | 0.57 | 0.31 | 0.40 | 760 |
| TRAVEL | 0.64 | 0.66 | 0.65 | 2072 |
| U.S. NEWS | 1.00 | 0.01 | 0.03 | 269 |
| WEDDINGS | 0.80 | 0.75 | 0.78 | 756 |
| WEIRD NEWS | 0.35 | 0.15 | 0.21 | 564 |
| WELLNESS | 0.51 | 0.75 | 0.61 | 3634 |
| WOMEN | 0.35 | 0.25 | 0.29 | 712 |
| WORLD NEWS | 0.58 | 0.15 | 0.23 | 665 |
| WORLDPOST | 0.63 | 0.26 | 0.36 | 543 |
| accuracy | | | 0.55 | 41903 |
| macro avg | 0.55 | 0.34 | 0.38 | 41903 |
| weighted avg | 0.55 | 0.55 | 0.51 | 41903 |

O relatório de classificação gerado pela implementação do modelo Random Forest fornece uma visão abrangente do desempenho do modelo em várias categorias. Vamos discutir os principais pontos observados no output, enfocando as variáveis com melhor desempenho, bem como uma avaliação geral da acurácia do modelo.

**Principais Variáveis com Melhor Desempenho**

**DIVORCE** apresentou um alto desempenho com precisão de 0.82 e recall de 0.67, resultando em um f1-score de 0.74. Isso indica que o modelo foi bastante eficaz em identificar corretamente as instâncias desta categoria, com uma taxa de falsos positivos relativamente baixa.

**STYLE & BEAUTY** também se destacou, com uma precisão de 0.70 e recall de 0.77, levando a um f1-score de 0.73. Esse resultado sugere que o modelo foi eficiente em classificar as instâncias dessa categoria, mantendo uma boa taxa de verdadeiros positivos em relação ao número de previsões positivas feitas.

**WEDDINGS** teve desempenho notável, com precisão de 0.80 e recall de 0.75, alcançando um f1-score de 0.78. Isso mostra a habilidade do modelo em distinguir efetivamente as instâncias dessa categoria com alta precisão e recall.

Acurácia do Modelo

A acurácia geral do modelo foi de 0.55, indicando que 55% das previsões feitas pelo modelo estavam corretas. Essa taxa de acurácia, embora moderada, sugere que o modelo tem uma capacidade razoável de fazer previsões corretas em um conjunto de dados com múltiplas categorias. No entanto, essa métrica sozinha não fornece um panorama completo da eficácia do modelo, especialmente considerando o desequilíbrio entre as categorias.

## Conclusão

O modelo Random Forest demonstrou uma performance variável entre as diferentes categorias. Enquanto algumas categorias como DIVORCE, STYLE & BEAUTY, e WEDDINGS apresentaram excelentes resultados, outras tiveram um desempenho bem abaixo, como evidenciado pelos f1-scores mais baixos e pelas taxas de recall e precisão.

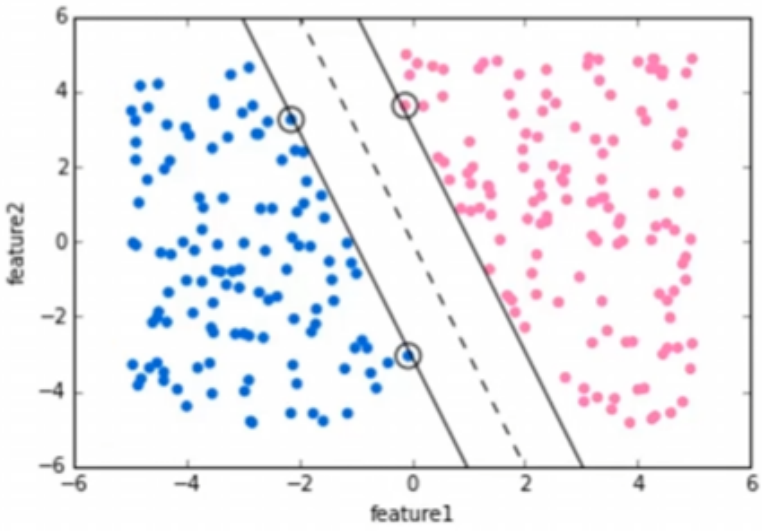
A variação no desempenho pode ser atribuída a vários fatores, incluindo desequilíbrios na distribuição das categorias do conjunto de dados e a natureza intrínseca das categorias que podem ou não ser facilmente distinguíveis com base nas features utilizadas.

Para melhorar o desempenho do modelo, algumas abordagens podem ser consideradas, como o rebalanceamento do conjunto de dados, a engenharia de novas features que possam melhor capturar as diferenças entre as categorias, e a otimização de hiperparâmetros do modelo Random Forest.

Em resumo, enquanto o modelo demonstrou competência em algumas áreas, há espaço para melhorias significativas, especialmente na precisão e recall de várias categorias, para

elevar a acurácia geral e garantir que o modelo seja mais equilibrado e eficaz em sua capacidade de classificação.

# Modelo 03: Support Vector Machines




**Teoria:**

O Support Vector Machines (SVM) é um algoritmo de aprendizado de máquina supervisionado usado tanto para classificação quanto para regressão. Ele é eficaz na separação de dados complexos e é especialmente útil em problemas com alta dimensionalidade.

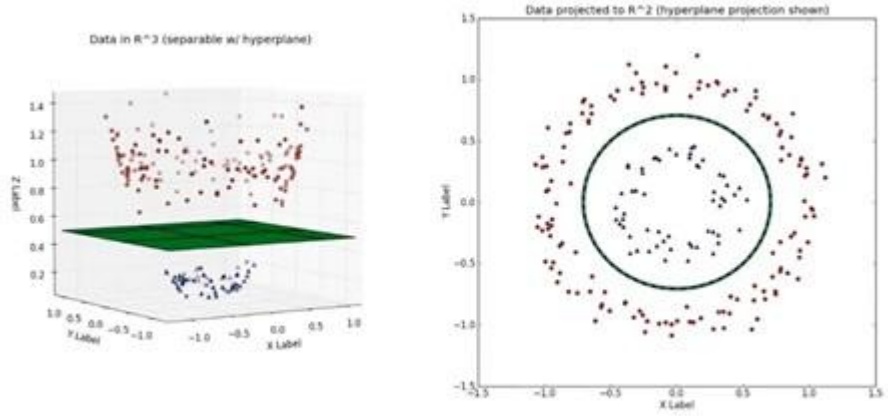
Mapeamento dos dados como pontos em um espaço: No SVM, os dados de treinamento são representados como pontos em um espaço multidimensional, onde cada dimensão corresponde a uma característica dos dados. Por exemplo, em um problema de classificação com duas características, os pontos seriam mapeados em um espaço bidimensional.

Separação por gap entre as categorias: O objetivo do SVM é encontrar um hiperplano de separação que maximize a margem entre as diferentes categorias. Esse hiperplano funciona como uma linha divisória entre os pontos das diferentes classes, e o espaço entre as duas margens é chamado de gap ou margem de separação.

Predição baseada no lado do gap: Após o treinamento, o SVM classifica novos dados determinando em qual lado do gap eles estão localizados. Se um ponto estiver do lado positivo do hiperplano, ele é classificado como pertencente a uma classe, e se estiver do lado negativo, é classificado como pertencente à outra classe.

Support Vectors: Os pontos que tocam as margens do hiperplano são chamados de vetores de suporte (support vectors). Esses pontos são cruciais para o SVM, pois eles definem a margem de separação. O SVM se concentra principalmente nos vetores de suporte durante o treinamento, pois são eles que influenciam a posição e a orientação do hiperplano.

Separação por hiperplano em dimensões mais altas:




Em casos onde os dados não podem ser separados de forma linear em um espaço de dimensões menores (como 2D), o SVM pode usar uma técnica conhecida como kernel trick para mapear os dados para um espaço de dimensões mais altas (3D ou superior), onde a separação linear se torna possível. Isso permite que o SVM lide com dados que não são linearmente separáveis em seu espaço original.

**Execução do Modelo:**

In [41]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
# Vetorização dos textos usando TF-IDF
vectorizer = TfidfVectorizer(stop_words='english', max_features=1000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

In [42]:
```
from sklearn.svm import SVC
# Criando e treinando o modelo SVM
svm_model = SVC(kernel='linear', C=1.0)
svm_model.fit(X_train_tfidf, y_train)
```

Out[42]:
```
SVC(kernel='linear')
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [43]:

```
from sklearn.metrics import classification_report

# Fazendo previsões e avaliando o modelo
svm_y_pred = svm_model.predict(X_test_tfidf)
print(classification_report(y_test, svm_y_pred))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| ARTS | 0.33 | 0.09 | 0.14 | 310 |
| ARTS & CULTURE | 0.26 | 0.13 | 0.17 | 255 |
| BLACK VOICES | 0.40 | 0.24 | 0.30 | 911 |
| BUSINESS | 0.38 | 0.34 | 0.36 | 1176 |
| COLLEGE | 0.38 | 0.36 | 0.37 | 226 |
| COMEDY | 0.42 | 0.24 | 0.30 | 1044 |
| CRIME | 0.41 | 0.44 | 0.43 | 725 |
| CULTURE & ARTS | 0.34 | 0.13 | 0.19 | 205 |
| DIVORCE | 0.74 | 0.61 | 0.67 | 689 |
| EDUCATION | 0.32 | 0.31 | 0.32 | 196 |
| ENTERTAINMENT | 0.38 | 0.61 | 0.47 | 3499 |
| ENVIRONMENT | 0.45 | 0.15 | 0.22 | 279 |
| FIFTY | 0.34 | 0.05 | 0.08 | 256 |
| FOOD & DRINK | 0.48 | 0.56 | 0.52 | 1290 |
| GOOD NEWS | 0.24 | 0.08 | 0.12 | 260 |
| GREEN | 0.29 | 0.17 | 0.21 | 536 |
| HEALTHY LIVING | 0.35 | 0.13 | 0.19 | 1342 |
| HOME & LIVING | 0.56 | 0.52 | 0.54 | 875 |
| IMPACT | 0.29 | 0.13 | 0.18 | 711 |
| LATINO VOICES | 0.33 | 0.00 | 0.01 | 220 |
| MEDIA | 0.43 | 0.20 | 0.27 | 586 |
| MONEY | 0.32 | 0.24 | 0.27 | 356 |
| PARENTING | 0.44 | 0.67 | 0.53 | 1733 |
| PARENTS | 0.40 | 0.08 | 0.14 | 778 |
| POLITICS | 0.60 | 0.81 | 0.69 | 6992 |
| QUEER VOICES | 0.75 | 0.54 | 0.63 | 1242 |
| RELIGION | 0.55 | 0.27 | 0.36 | 533 |
| SCIENCE | 0.42 | 0.26 | 0.32 | 400 |
| SPORTS | 0.46 | 0.34 | 0.39 | 976 |
| STYLE | 0.63 | 0.13 | 0.21 | 501 |
| STYLE & BEAUTY | 0.66 | 0.71 | 0.68 | 1978 |
| TASTE | 0.35 | 0.03 | 0.05 | 437 |
| TECH | 0.42 | 0.27 | 0.33 | 411 |
| THE WORLDPOST | 0.42 | 0.29 | 0.35 | 760 |
| TRAVEL | 0.54 | 0.56 | 0.55 | 2072 |
| U.S. NEWS | 0.20 | 0.00 | 0.01 | 269 |
| WEDDINGS | 0.73 | 0.69 | 0.71 | 756 |
| WEIRD NEWS | 0.31 | 0.10 | 0.15 | 564 |
| WELLNESS | 0.45 | 0.73 | 0.56 | 3634 |
| WOMEN | 0.34 | 0.29 | 0.31 | 712 |
| WORLD NEWS | 0.37 | 0.16 | 0.23 | 665 |
| WORLDPOST | 0.27 | 0.26 | 0.26 | 543 |
| accuracy | | | 0.49 | 41903 |
| macro avg | 0.42 | 0.31 | 0.33 | 41903 |
| weighted avg | 0.48 | 0.49 | 0.46 | 41903 |

A análise do relatório de classificação gerado pelo modelo de Support Vector Machines (SVM) com kernel linear fornece insights importantes sobre o desempenho do modelo em prever as categorias de texto. Vamos detalhar a conclusão com base nos resultados fornecidos:

Desempenho Geral do Modelo SVM

Acurácia: O modelo atingiu uma acurácia de 49%, o que indica que cerca de metade das previsões feitas pelo modelo estão corretas. Essa é uma métrica importante, mas, considerando o número elevado de categorias, esse valor pode não refletir totalmente a eficácia do modelo em todas as classes.

Macro avg vs Weighted avg: A média macro (42% de precisão, 31% de recall, 33% de f1-score) considera todas as classes igualmente, enquanto a média ponderada (48% de precisão, 49% de recall, 46% de f1-score) leva em conta o número de instâncias em cada classe. A diferença entre essas médias indica um desempenho desigual do modelo entre as classes, sendo mais eficaz em classes com mais amostras.

Melhores e Piores Performances por Categoria

Melhores Performances:

**Política (Politics):** Com uma precisão de 60% e recall de 81%, essa categoria obteve a melhor performance, indicando uma boa capacidade do modelo em identificar corretamente textos dessa categoria. O alto recall sugere que o modelo é eficiente em capturar a maioria das instâncias relevantes para esta classe.

**Estilo e Beleza (Style & Beauty):** Com uma precisão de 66% e recall de 71%, também se destaca, mostrando que o modelo é relativamente bom em classificar corretamente os textos dessa categoria.

**Vozes Queer (Queer Voices)**: Precisão de 75% e recall de 54%, indicando uma alta precisão, mas com uma capacidade moderada de detectar todas as instâncias relevantes dessa classe.

**Piores Performances:**
Notícias dos EUA (U.S. News): Com uma precisão de 20% e um recall quase nulo, essa categoria teve um dos piores desempenhos, o que pode indicar uma confusão substancial com outras categorias ou uma falta de características distintivas aprendidas pelo modelo.
Sabor (Taste) e Voices Latino (Latino Voices): Ambas as categorias também tiveram desempenhos ruins, com recalls extremamente baixos, indicando dificuldades do modelo em identificar essas categorias de forma eficaz.

**Considerações Finais**
A diferença entre as melhores e piores performances destaca a variabilidade do desempenho do modelo SVM entre diferentes categorias. Isso pode ser devido a várias razões, como desequilíbrio de classe, falta de características distintivas em algumas categorias, ou a natureza dos dados de texto que são mais desafiadores para o modelo aprender.
Para melhorar o desempenho geral do modelo, poderíamos considerar técnicas como balanceamento de classe, experimentação com diferentes parâmetros do modelo SVM, ou uso de técnicas de redução de dimensionalidade mais sofisticadas para capturar melhor a essência dos textos.
Outra abordagem poderia incluir a combinação de modelos (ensemble learning) ou o uso de técnicas de deep learning, que podem ser mais eficazes em capturar nuances em dados de texto grandes e complexos.
Esta análise destaca a importância de avaliar cuidadosamente cada categoria individualmente, além da métrica de acurácia geral, para entender completamente o desempenho de um modelo de machine learning em tarefas de classificação de texto.

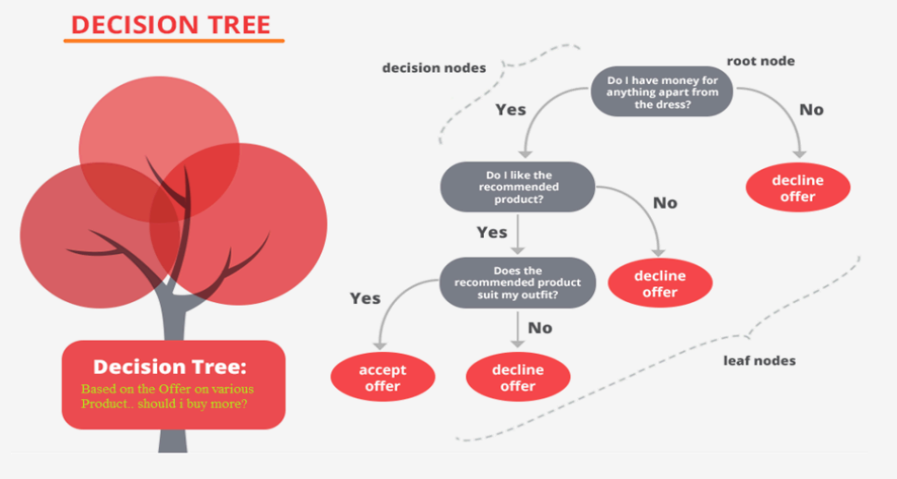
# Bases teóricas dos métodos analíticos

Para realizar esse projeto, utilizaremos um modelo de aprendizagem de máquina supervisionado (classificação), baseado na classificação de dados rotulados. Os métodos utilizados serão baseados em Naive Bayes, Random Forest Classifier, e Support Vector Machines, onde será verificado qual modelo é o mais equilibrado e o ideal para o projeto.

O modelo baseado em **árvore de decisão** pode ser utilizado tanto para modelos de classificação quanto modelos de regressão, pode prever categorias discretas (sim ou não), ou prever valores numéricos.

A árvore de decisão estabelece **nós** (decision nodes) que se relacionam entre si por uma hierarquia. Existe o **nó-raiz** (root node), que é o mais importante, e os nós-**folha** (leaf nodes), que são os resultados. No contexto de machine learning, o raiz é um dos atributos da base de dados e o nó-folha é a classe ou o valor que será gerado como resposta.

O grande trabalho da árvore é justamente encontrar os nós que vão ser encaixados em cada posição. Quem será o nó raiz? Depois, quem será o nó da esquerda? E o da direita?




## Acurácia

A acurácia no nosso projeto refere-se à medida de precisão do modelo de classificação de notícias. Ela é calculada como a proporção de notícias corretamente classificadas em relação ao total de notícias.

$$Acurácia = \frac{\text{Número de previsões corretas}}{\text{Número total de previsões}}$$

Após treinar o modelo com um conjunto de dados, ele é avaliado na qual o modelo faz previsões das categorias das notícias e a acurácia é determinada comparando essas previsões com as categorias reais das notícias.

## Método Analítico Aplicado à Base de Dados e Escolha do Melhor Modelo

Para avaliar e comparar o desempenho dos três modelos de classificação (Naive Bayes, Random Forest Classifier e Support Vector Machines) no projeto, devemos considerar diversas métricas presentes no classification report, incluindo precisão (precision), recall (sensibilidade), f1-score e suporte (support). Cada modelo será analisado considerando suas médias macro e ponderadas (weighted) dessas métricas, focando principalmente na precisão, recall e f1-score, que são críticos para entender a eficácia geral dos modelos em diferentes classes.

## Análise dos Resultados

Naive Bayes

- Precisão Média (Macro Avg): 0.64
- Recall Médio (Macro Avg): 0.30
- F1-Score Médio (Macro Avg): 0.33
- Precisão Ponderada (Weighted Avg): 0.59
- Recall Ponderado (Weighted Avg): 0.56
- F1-Score Ponderado (Weighted Avg): 0.50

Random Forest Classifier

- Precisão Média (Macro Avg): 0.55
- Recall Médio (Macro Avg): 0.34
- F1-Score Médio (Macro Avg): 0.38
- Precisão Ponderada (Weighted Avg): 0.55
- Recall Ponderado (Weighted Avg): 0.55
- F1-Score Ponderado (Weighted Avg): 0.51

Support Vector Machines

- Precisão Média (Macro Avg): 0.41
- Recall Médio (Macro Avg): 0.27
- F1-Score Médio (Macro Avg): 0.31
- Precisão Ponderada (Weighted Avg): 0.47
- Recall Ponderado (Weighted Avg): 0.50
- F1-Score Ponderado (Weighted Avg): 0.46

## Conclusões:

- Naive Bayes mostra a maior precisão média, mas com recall significativamente baixo, indicando que embora as previsões corretas sejam relativamente precisas, o modelo falha em capturar uma grande proporção de casos positivos reais (baixa sensibilidade).
- Random Forest apresenta um equilíbrio um pouco melhor entre precisão e recall em comparação com o Naive Bayes, mas ambos têm desempenhos relativamente próximos em termos de f1-score ponderado.

- Support Vector Machines teve desempenho inferior nos três critérios principais (precisão, recall e f1-score), indicando que esse modelo foi menos eficaz em generalizar para o conjunto de dados de teste.

Baseando-se nos resultados do classification report, o **Random Forest** parece ser o mais equilibrado dos três modelos, apesar de ter uma precisão média ligeiramente inferior à do Naive Bayes.

Ele oferece um melhor recall médio e mantém um f1-score ponderado comparável. Isso sugere que o Random Forest pode ser a escolha mais robusta para aplicações práticas, considerando que ele fornece uma generalização razoável sem sacrificar muito a precisão ou o recall em qualquer classe específica.

No entanto, é importante notar que a escolha do modelo também pode depender de outros fatores, como a importância relativa de precisão versus recall em seu contexto específico, a distribuição das classes (desequilíbrio de classes) e o custo computacional dos modelos. Ajustes adicionais dos parâmetros do modelo ou técnicas de reamostragem podem ser exploradas para melhorar o desempenho do modelo selecionado.

## Considerações sobre a Escolha do Modelo:

A escolha entre Naive Bayes, Random Forest e SVM deve considerar o trade-off entre precisão e recall, a sensibilidade do modelo ao desbalanceamento das classes e a necessidade de generalização para não sobreajustar a categorias específicas.

Por exemplo: Naive Bayes é adequado para benchmarks iniciais devido à sua simplicidade e boa performance em termos de recall, mas pode requerer ajustes ou complementação com outros métodos para aumentar a precisão.

Random Forest é robusto para lidar com overfitting e mostra boa capacidade de adaptação a diferentes categorias, sendo uma escolha sólida para conjuntos de dados desbalanceados ou complexos.

SVM é eficaz em encontrar fronteiras de decisão complexas, especialmente em categorias bem representadas, mas pode necessitar de um ajuste fino dos parâmetros e da seleção de kernel para melhorar o desempenho em categorias com poucas amostras.

## Recomendações para Melhoria:

Para todos os modelos, é crucial considerar estratégias como o rebalanceamento das classes, a engenharia de features e a otimização de hiperparâmetros. Além disso, a incorporação de técnicas de validação cruzada e o uso de conjuntos de validação e teste separados ajudarão a avaliar a verdadeira capacidade de generalização dos modelos.

Universidade Presbiteriana Mackenzie

# Descrição dos resultados, Apresentação do produto e modelo de negócios

O modelo que utilizamos para a classificação das notícias em categorias estabelecidas foi o Random Forest, onde obtivemos uma precisão média de 55% dentre outras medidas de acurácia mostradas abaixo.

Esse modelo se mostrou o mais equilibrado para o tipo de classificação requerida, trazendo eficiência e certa precisão.

- Precisão Média (Macro Avg): 0.55
- Recall Médio (Macro Avg): 0.34
- F1-Score Médio (Macro Avg): 0.38
- Precisão Ponderada (Weighted Avg): 0.55
- Recall Ponderado (Weighted Avg): 0.55
- F1-Score Ponderado (Weighted Avg): 0.51

**Fazendo previsões e avaliando o modelo**

In [40]:
```
rf_y_pred = rf_model.predict(X_test_vect)
print(classification_report(y_test, rf_y_pred))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| ARTS | 0.54 | 0.16 | 0.25 | 310 |
| ARTS & CULTURE | 0.31 | 0.10 | 0.15 | 255 |
| BLACK VOICES | 0.45 | 0.22 | 0.30 | 911 |
| BUSINESS | 0.49 | 0.35 | 0.41 | 1176 |
| COLLEGE | 0.48 | 0.27 | 0.35 | 226 |
| COMEDY | 0.45 | 0.32 | 0.37 | 1044 |
| CRIME | 0.45 | 0.44 | 0.44 | 725 |
| CULTURE & ARTS | 0.86 | 0.21 | 0.34 | 205 |
| DIVORCE | 0.82 | 0.67 | 0.74 | 689 |
| EDUCATION | 0.36 | 0.13 | 0.19 | 196 |
| ENTERTAINMENT | 0.51 | 0.71 | 0.59 | 3499 |
| ENVIRONMENT | 0.89 | 0.09 | 0.16 | 279 |
| FIFTY | 0.46 | 0.07 | 0.12 | 256 |
| FOOD & DRINK | 0.51 | 0.74 | 0.61 | 1290 |
| GOOD NEWS | 0.40 | 0.07 | 0.11 | 260 |
| GREEN | 0.40 | 0.16 | 0.23 | 536 |
| HEALTHY LIVING | 0.32 | 0.23 | 0.26 | 1342 |
| HOME & LIVING | 0.66 | 0.61 | 0.63 | 875 |
| IMPACT | 0.46 | 0.07 | 0.12 | 711 |
| LATINO VOICES | 0.95 | 0.08 | 0.15 | 220 |
| MEDIA | 0.64 | 0.24 | 0.35 | 586 |
| MONEY | 0.56 | 0.15 | 0.24 | 356 |
| PARENTING | 0.48 | 0.61 | 0.53 | 1733 |
| PARENTS | 0.36 | 0.24 | 0.29 | 778 |
| POLITICS | 0.56 | 0.90 | 0.69 | 6992 |
| QUEER VOICES | 0.79 | 0.61 | 0.69 | 1242 |
| RELIGION | 0.61 | 0.34 | 0.43 | 533 |
| SCIENCE | 0.51 | 0.30 | 0.38 | 400 |
| SPORTS | 0.57 | 0.55 | 0.56 | 976 |
| STYLE | 0.49 | 0.23 | 0.31 | 501 |
| STYLE & BEAUTY | 0.70 | 0.77 | 0.73 | 1978 |
| TASTE | 0.29 | 0.08 | 0.13 | 437 |
| TECH | 0.51 | 0.30 | 0.38 | 411 |
| THE WORLDPOST | 0.57 | 0.31 | 0.40 | 760 |
| TRAVEL | 0.64 | 0.66 | 0.65 | 2072 |
| U.S. NEWS | 1.00 | 0.01 | 0.03 | 269 |
| WEDDINGS | 0.80 | 0.75 | 0.78 | 756 |
| WEIRD NEWS | 0.35 | 0.15 | 0.21 | 564 |
| WELLNESS | 0.51 | 0.75 | 0.61 | 3634 |
| WOMEN | 0.35 | 0.25 | 0.29 | 712 |
| WORLD NEWS | 0.58 | 0.15 | 0.23 | 665 |
| WORLDPOST | 0.63 | 0.26 | 0.36 | 543 |
| accuracy | | | 0.55 | 41903 |
| macro avg | 0.55 | 0.34 | 0.38 | 41903 |
| weighted avg | 0.55 | 0.55 | 0.51 | 41903 |

# Apresentação do produto e aplicação em modelos de negócio

Com nosso modelo praticamente finalizado, temos um produto que é eficaz no quesito de classificação de notícias em categorias já estabelecidas, tornando assim mais prática a busca por determinados temas, e trazendo um enorme ganho de tempo, ao contrário do que se teria verificando e classificando cada notícia manualmente.

Esse modelo pode ser aplicado em dois tipos de modelos de negócio.
Podemos aplicá-lo no desenvolvimento de uma plataforma de recomendação de notícias, assim como em diversos serviços de streaming em que o algoritmo detecta o que mais se adapta às preferências do cliente e faz a sugestão com mais frequência desses tópicos que melhor se adaptarem.
Uma outra forma de aplicação num modelo de negócios é fazer o chamado desse modelo através de uma API e integrá-lo com uma plataforma de análise de dados, permitindo saber as notícias mais acessadas e menos acessadas, levando a tomadas de decisões sobre sugerir mais ou menos uma categoria.

Universidade Presbiteriana Mackenzie

# Storytelling

**A Jornada da Classificação de Notícias: Descobrindo Verdades nas Palavras**

Em um mundo onde a informação é abundante, mas nem sempre personalizada, surge um desafio audacioso: como oferecer aos usuários uma experiência informativa alinhada com suas preferências individuais?

Nesse contexto, nasce um projeto ambicioso de desenvolvimento de um sistema de classificação e recomendação de notícias, impulsionado pelas técnicas em Ciência de Dados e Machine Learning. Nosso principal objetivo é automatizar a classificação de conteúdo informativo e criar uma experiência sob medida para cada usuário.

O projeto inicia sua jornada construindo um modelo de aprendizado de máquina capaz de classificar notícias em categorias específicas. Ele se propõe a lidar com a diversidade de estilos de escrita e tópicos, desafiando-se a abraçar a complexidade do mundo da informação.

Além disso, os usuários terão a oportunidade de expressar suas preferências, como entretenimento, política, estilo e bem-estar, moldando assim a experiência de leitura de acordo com seus interesses únicos e criando recomendações personalizadas.

O projeto integra também um mecanismo de feedback do usuário, alimentando o ciclo de melhoria contínua. Cada interação do usuário se torna uma oportunidade de ajuste, refinando o sistema e tornando as recomendações ainda mais precisas.

Em um mundo em constante mudança, a busca pela informação personalizada é uma jornada sem fim. E nessa jornada, a Ciência de Dados é o guia, revelando novas perspectivas e horizontes inexplorados no vasto universo da informação. O futuro é de evolução constante, de desbravamento de novas fronteiras da informação personalizada.