UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA – CIN
Curso de bacharelado em sistemas de informação (BSI)

Disciplina de Tópicos Avançados em SI 5

Soluções em Mineração de Dados - IF1018

Prof. Leandro Maciel Almeida

19/04/2022 – 2021.2

EQUIPE 01

DANIEL MORAES COSTA ANDRADE **- dmca@cin.ufpe.br**
GUSTAVO PRAZERES PAZ DO NASCIMENTO **- gppn@cin.ufpe.br**
VINICIUS LUIZ DA SILVA FRANÇA **- vlsf2@cin.ufpe.br**

Implementação de Projeto de processamento de dados utilizando o CRISP-DM

Tendo sido fornecido o dataset "Gender Gap in Spanish Wikipedia", disponível em https://archive.ics.uci.edu/ml/datasets/Gender+Gap+in+Spanish+WP, realizar um relatórioa respeito do processamento de dados utilizando as etapas do CRISP-DM para o problema levantado.

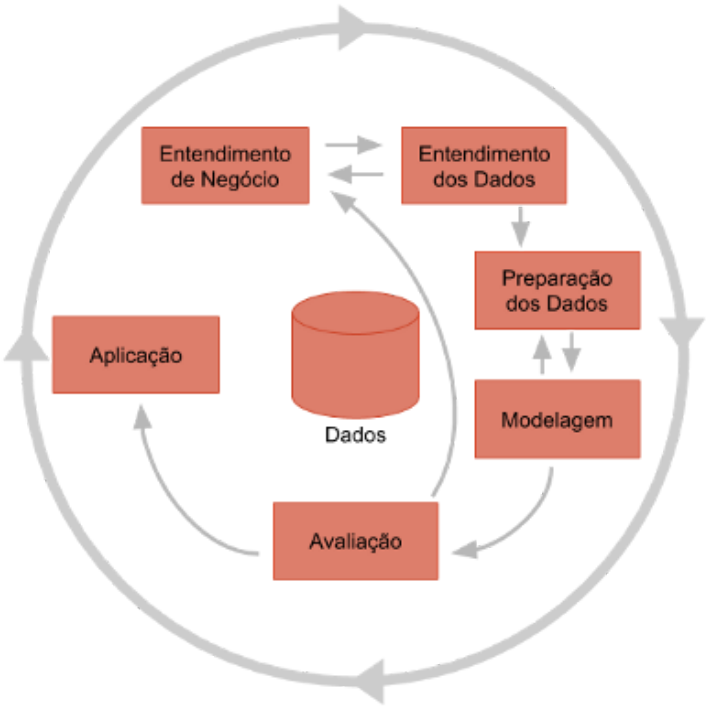
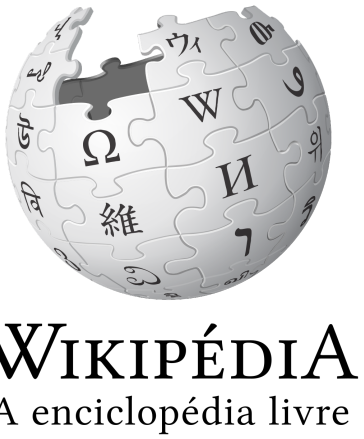


Figura 1 - Fases do modelo CRISP-DM

Fonte: https://medium.com/@kvmoura - Ciclo de vida dos dados #2

1. Entendimento de Negócio

1.1 Objetivos de negócio



A Wikipédia é uma das fontes de informação mais utilizadas no mundo e um dos dez sites mais populares da Espanha. É também o site mais utilizado no setor educacional, principalmente entre estudantes ou onde não há livros disponíveis (Aibar et al., 2015). A plataforma foi lançada em 2001 e hoje contém mais de 40 milhões de artigos em 301 idiomas. Devido à sua estrutura altamente vinculada, a Wikipedia geralmente aparece na primeira página de resultados de consulta em qualquer mecanismo de pesquisa (Singer et al., 2017).

A Wikipédia é uma "enciclopédia livre", escrita de forma colaborativa com o objetivo de se tornar "a soma de todo o conhecimento", como afirma a visão da organização: https://wikimediafoundation.org/about/vision/

Na contramão da ideia de colaboração e construção de conhecimento de forma social, a Wikipedia sofre de um grande viés de gênero, refletido tanto na comunidade envolvida no processo de edição (a maioria dos editores são homens) e no próprio conteúdo (as biografias de homens superam as de mulheres e tendem a ser mais extensas). Além disso, as editoras também estão sujeitas a mais conflitos, como reversão e bloqueio de suas edições, do que os homens (Minguillón et al., 2021, Hinnosaar, 2019; Massa; Zelenkauskaite, 2014; Wagner et al., 2016)

Em 2011, a Wikimedia Foundation reconheceu a desigualdade de gênero no site e afirmou estar buscando maneiras para resolvê-la, citando que "é inaceitável que as mulheres sejam excluídas deste recurso que impacta significativamente a educação".

1.2 Avaliação da situação

Dado o contexto acima, é de extrema importância realizar uma análise mais profunda para explorarmos os possíveis fatores que impedem as mulheres de se tornarem editoras da Wikipédia.

Existem diversas metodologias para se extrair informações sobre os autores e editores das páginas dos artigos publicados na Wikipédia:

- por meio da WikiMedia API (para os artigos em que os editores optam por identificar-se como mulheres ou homens);
- com base nas informações fornecidas nos perfis de usuários dos autores;
- os autores que fornecem um nome real que indique claramente o sexo;
- dedução do gênero gramatical através das expressões usadas pelos autores para se descrever;

Para este projeto, portanto, estaremos utilizando um dataset fornecido pela Julià Minguillón Alfonso, da Universidade Oberta de Catalunya - Barcelona, Espanha, no qual a descrição é "Conjunto de dados usado para estimar o número de mulheres editoras e suas práticas de edição na Wikipédia espanhola" (2021).

Para a criação deste dataset, a pesquisadora utilizou de um método interdisciplinar através da combinação dos procedimentos mencionados acima a fim de melhorar a estimativa do número de cada gênero, assim como explorar as diferenças de comportamento.

1.3 Determinação das metas de ciência de dados

Nossas metas de ciência de dados descrevem quais são as metas e objetivos para o projeto em termos técnicos e definição dos critérios de sucesso da ciência de dados. Temos portanto como metas de ciência de dados através da análise do conteúdo do dataset fornecido:

- Estimar qual o número de editores, artigos publicados, páginas criadas e editadas por cada gênero;
- Descobrir qual é a porcentagem de mulheres entre os editores ativos;

1.4 Plano de Projeto

Para atingirmos nossas metas, estaremos seguindo os passos das etapas do CRISP-DM, com o entendimento geral do negócio e do dataset, para em seguida aplicarmos algoritmos modernos para a identificação de padrões nos dados das páginas da Wikipédia.

Como ferramentas, estaremos utilizando a plataforma colaborativa do Google Collab, executando scripts em Python e bibliotecas modernas, como o Numpy, SciKit Learn, Pandas, dentre outras.

A fim de evitar repetidas execuções ou perda de dados através do Google Collab, foi definido que o projeto seria dividido em três etapas, com a utilização da biblioteca Pickle para exportar e importar variáveis e datasets por entre os Jupyter Notebooks. Abaixo, portanto, descrevemos o plano detalhado esperado para o projeto:

Parte 1 - Análise e Tratamento de dados

1. Importação de bibliotecas e dados
2. Análise exploratória dos dados
3. Preparação dos Dados
    - 3.1. Selecionando atributos relevantes
    - 3.2. Transformando dados
    - 3.3. Normalizando dados
    - 3.4. Separando o dataset entre gêneros conhecidos e desconhecidos
    - 3.5. Realizando o balanceamento do dataset
    - 3.6. Exportando dataset para realizar a predição e as variáveis de treino e teste

Parte 2 - Modelagem e Treino dos modelos

4. Modelagem dos dados
    - 4.1 Importação dos dados divididos em bases de treino, validação e teste
    - 4.2 Cross Validation
    - 4.3 Variação paramétrica - GridSearchCV
    - 4.4. Aplicação de scripts de experimentação e análise de dados
        - 4.4.1 KNn

2. Entendimento dos dados - Análise exploratória

Durante a etapa de Entendimento dos dados, realizamos a análise exploratória dos dados apresentando todas as informações do dataset.

2.1 Coleta de dados

Assim como comentado no ponto 1.2 [Avaliação da situação], o conjunto de dados utilizados para este projeto de processamento de dados foram resultados de um artigo de pesquisa realizado pela Julià Minguillón Alfonso, da Universidade Oberta de Catalunya - Barcelona. No artigo “Explorando a diferença de gênero na Espanha Wikipédia: Diferenças nas práticas de engajamento e edição” foi documentada toda a metodologia utilizada para a coleta dos dados.

O dataset foi, portanto, disponibilizado em um repositório de dados no website do Centro de Aprendizado de Máquina e Sistemas Inteligentes da Universidade da Califórnia [UCI Machine Learning Repository], disponível em
https://archive.ics.uci.edu/ml/datasets/Gender+Gap+in+Spanish+WP

Os dados foram disponibilizados em formato .csv na URL abaixo.
https://archive.ics.uci.edu/ml/machine-learning-databases/00619/data.csv
Após fazermos o download e importarmos os dados em nosso Google Collab, utilizamos a função read_csv da biblioteca pandas para a criação de um novo dataset `"dt"`

```
import pandas as pd
df = pd.read_csv('data.csv')
```

2.2 Descrição do conjunto de dados

Para determinar o gênero dos perfis de usuário, a pesquisadora utilizou um dump da Wikipédia espanhola (datado de 1º de outubro de 2017), contendo:

- 963.591 editores registrados
- 28.763 editores ativos

- 13.210 editores que desenvolveram sua própria página pessoal

● 5.651 editores através da extração de uma amostra
● 4.746 páginas pessoais que não estavam vazias, removidas ou bloqueadas

A amostra foi estratificada de forma a capturar igualmente todos os níveis de engajamento para obter uma amostra de dados aleatória.

Foi criado o campo “gender”, que foi determinado ou pelo gênero especificado no perfil do usuário (através da API MediaWiki) ou pelo gênero extraído da codificação de conteúdo. (ex: no perfil do usuário possui expressões como "sou advogada")

As métricas utilizadas para a definição final do valor de “gender” foram, portanto:
→ `C_API:` Dados da WikiMedia API
→ `C_MAN:` Gênero calculado pela codificação de conteúdo manual descrito no artigo
→ `GENDER`: Combinação de `C_API` e `C_MAN`, de acordo com a regra:
Se `C_MAN` = `C_API`, então valor final deve ser = `C_MAN`
Se `C_MAN` = unknown & `C_API` != unknown, então valor final deve ser = `C_API`
Se `C_MAN` != unknown & `C_API` = unkonwn, então valor final deve ser = `C_MAN`
Caso contrário, gera um erro, pois uma exceção ocorreu.

2.2.1 Informações dos atributos

gender → gênero, 0 (desconhecido), 1 (masculino), 2 (feminino)
C_api → gênero extraído da API WikiMedia, códigos como feminino, masculino ou desconhecido
C_man → gênero extraído da codificação do conteúdo, codificado como 1 (masculino) / 2 (feminino) / 3 (desconhecido)
E_NEds → Índice I do estrato IJ (0,1,2,3)
E_Bpag → Índice J do estrato IJ (0,1,2,3)
firstDay → data da primeira edição na Wikipédia espanhola (AAAAMMDDHHMMSS)
lastDay → data da última edição na Wikipédia espanhola (AAAAMMDDHHMMSS)
NEds → número total de edições
NDays → número de dias (últimoDia-primeiroDia+1)
NActDays → número de dias com edições
NPages → número de páginas diferentes editadas
NPcreated → número de páginas criadas
pagesWomen → número de edições em páginas relacionadas a mulheres
wikiprojWomen → número de edições em WikiProjects relacionadas a mulheres
ns_user → número de edições no usuário do namespace
ns_wikipedia → número de edições no namespace wikipedia
ns_talk → número de edições no namespace talk
ns_userTalk → número de edições no namespace user talk
ns_content → número de edições nas páginas de conteúdo
weightIJ → corrigindo o peso para o estrato IJ
NIJ → número de elementos no estrato IJ

2.2.2 Quantidade, formato e tipo dos dados

Para visualizarmos a quantidade de dados, utilizamos a função shape:

```
no_registros = df.shape[0]
no_atributos = df.shape[1]
```

```
print(f'O Dataset possui {no_registros} registros com {no_atributos} atributos cada\n')
```

Output:

O Dataset possui 4746 registros com 21 atributos cada

Pandas describe() é usado para visualizar alguns detalhes estatísticos básicos como percentil, média, std etc. de um quadro de dados ou uma série de valores numéricos

```
df.describe()
```

Output:

	gender	C_man	E_NEds	E_Bpag	firstDay	lastDay	NEds	NDays	NActDays	NPages	NPcreated	pagesWomen	wikiprojWomen	ns
count	4746.000000	4746.000000	4746.000000	4746.000000	4.746000e+03	4.746000e+03	4746.000000	4746.000000	4746.000000	4746.000000	4746.000000	4746.000000	4746.000000	4746.0
mean	0.737042	2.082807	1.484197	1.646228	2.009942e+13	2.015489e+13	2029.969448	2036.607880	183.162663	689.451960	43.479140	0.438896	0.439949	74.3
std	0.585355	0.964978	1.099795	1.079263	3.516337e+10	1.748104e+10	7793.300833	1336.119914	374.034481	3355.302483	297.395507	5.327440	17.832244	246.4
min	0.000000	1.000000	0.000000	0.000000	2.002011e+13	2.012010e+13	50.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	1.0
25%	0.000000	1.000000	1.000000	1.000000	2.007042e+13	2.014070e+13	95.000000	835.250000	24.000000	29.000000	1.000000	0.000000	0.000000	4.0
50%	1.000000	3.000000	1.000000	2.000000	2.009121e+13	2.016072e+13	218.000000	2035.500000	53.000000	68.000000	4.000000	0.000000	0.000000	14.0
75%	1.000000	3.000000	2.000000	3.000000	2.013040e+13	2.017073e+13	757.750000	3146.500000	154.000000	219.750000	14.000000	0.000000	0.000000	46.0
max	2.000000	3.000000	3.000000	3.000000	2.017093e+13	2.017100e+13	153193.000000	5349.000000	3843.000000	94142.000000	13394.000000	185.000000	949.000000	6041.0

A função info() é usada para imprimir um resumo conciso de um DataFrame. Ele imprime informações incluindo o dtype de índice e dtypes de coluna, valores não nulos e uso de memória.

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4746 entries, 0 to 4745
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   gender          4746 non-null   int64
 1   C_api           4746 non-null   object
 2   C_man           4746 non-null   int64
 3   E_NEds          4746 non-null   int64
 4   E_Bpag          4746 non-null   int64
 5   firstDay        4746 non-null   int64
 6   lastDay         4746 non-null   int64
 7   NEds            4746 non-null   int64
 8   NDays           4746 non-null   int64
 9   NActDays        4746 non-null   int64
 10  NPages          4746 non-null   int64
 11  NPcreated       4746 non-null   int64
 12  pagesWomen      4746 non-null   int64
 13  wikiprojWomen   4746 non-null   int64
 14  ns_user         4746 non-null   int64
 15  ns_wikipedia    4746 non-null   int64
 16  ns_talk         4746 non-null   int64
 17  ns_userTalk     4746 non-null   int64
 18  ns_content      4746 non-null   int64
 19  weightIJ        4746 non-null   float64
```

```
 20  NIJ             4746 non-null   int64
dtypes: float64(1), int64(19), object(1)
memory usage: 778.8+ K
```

2.4 Exploração dos dados

Correlação entre atributos

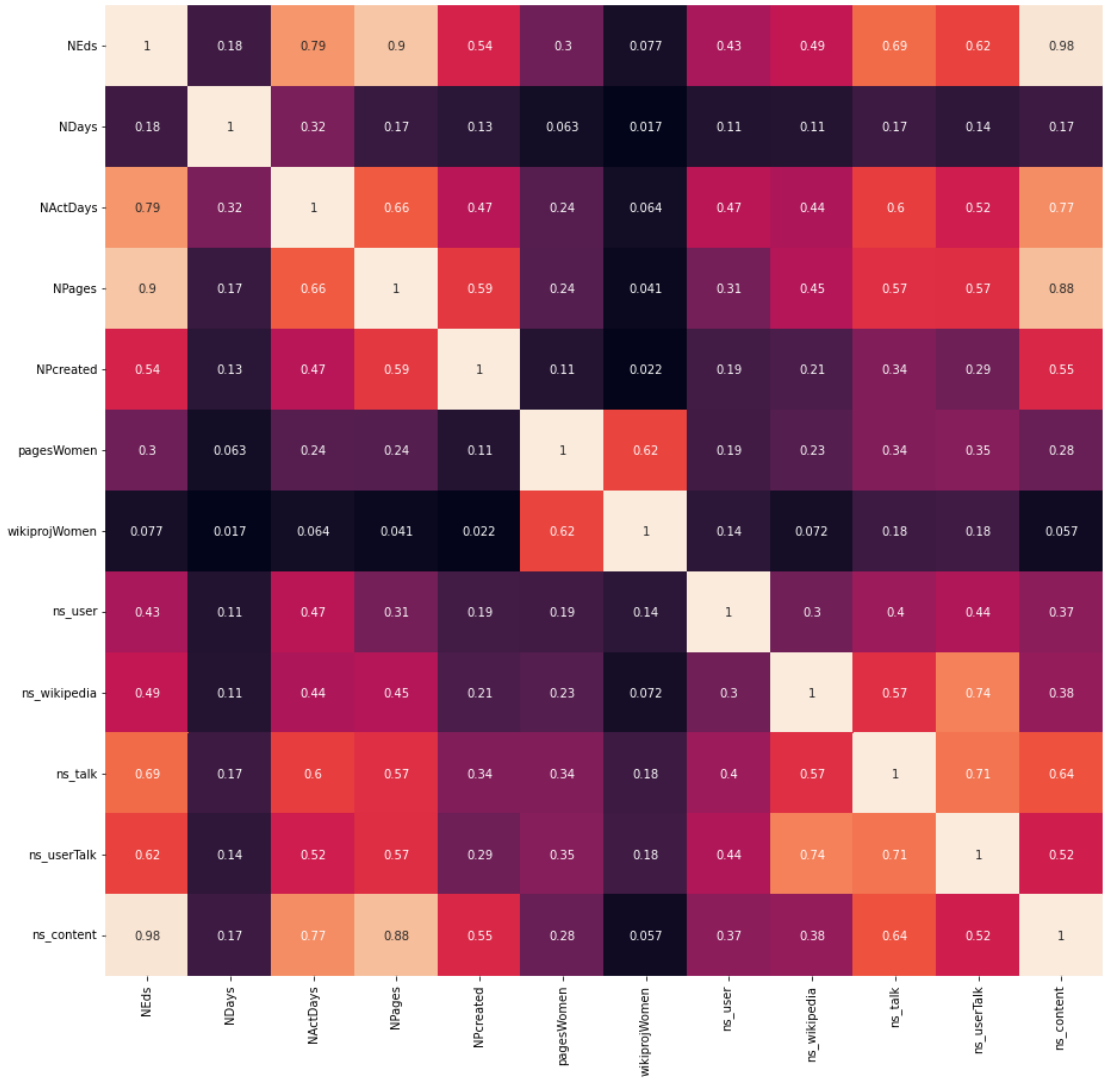
Os atributos altamente correlacionados são mostrados em cores mais claras, e aquelas que são negativamente correlacionadas são mostradas em cores escuras.

Ao analisar a matriz de correlação abaixo, é possível notar que não há correlações significativas a respeito da base de dados, visto que as correlações observadas apresentam informações previsíveis, como por exemplo, o número de edições nas páginas de conteúdo está fortemente relacionado ao número de páginas diferentes editadas, número total de edições e número de dias com edições.

Além disso, o número de edições em WikiProjects e em páginas relacionadas a mulheres não estão correlacionados com nenhum outro dado em específico.

```
plt.figure(figsize=(15,15))
sns.heatmap(df_graphics.corr(), annot=True, cbar=False)
```

Output:



Densidade por classe

O gráfico abaixo mostra a densidade de período da primeira edição na Wikipédia espanhola agrupado por gênero. Ao analisar esse gráfico, é possível notar que o número de mulheres na Wikipédia espanhola obteve um considerável aumento de 2009 em diante, o que não é visto pelo gênero masculino no mesmo período.

```
plt.figure(figsize=(15,7))
sns.kdeplot(df_days_male['firstDay'], label='Data da primeira
edição na Wiki espanhola - Masculino', color='b', shade=True)
sns.kdeplot(df_days_female['firstDay'], label='Data da primeira
edição na Wiki espanhola - Feminino', color='r', shade=True)
plt.legend()
```

Output:

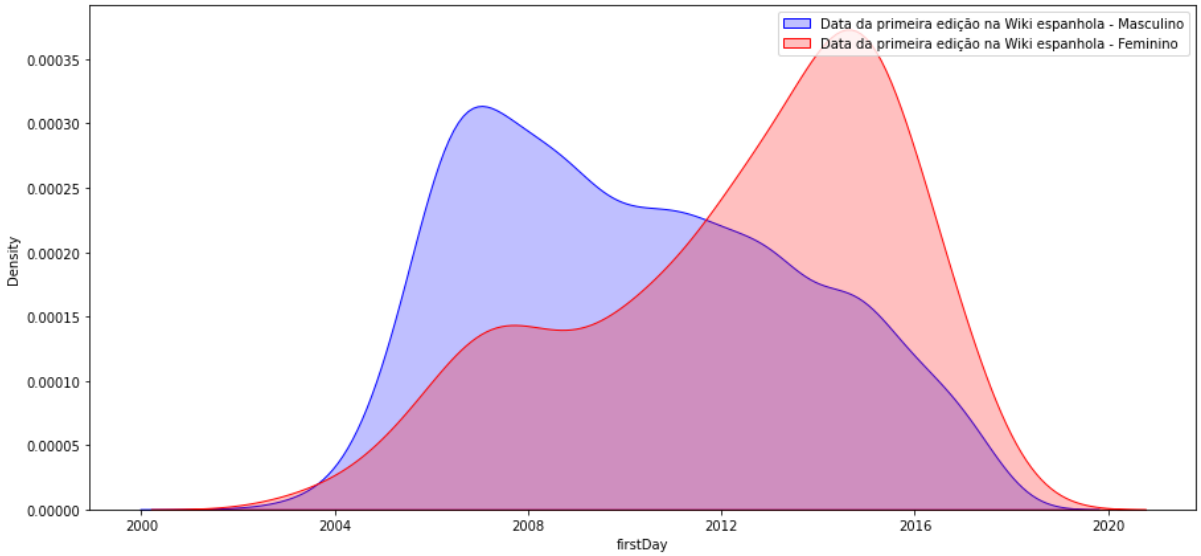


Gráfico de pizza para mensurar o total de edições

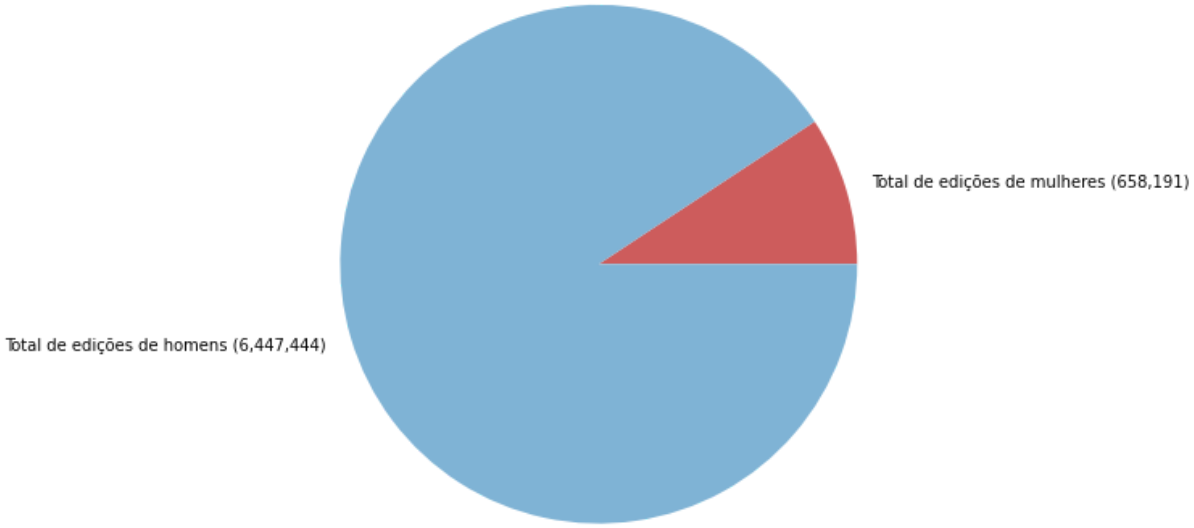
O gráfico de pizza é uma análise univariada e normalmente é usado para mostrar dados percentuais ou proporcionais, o que proporciona um resumo visual mais fácil de grandes pontos de dados, como também, o efeito e o tamanho das diferentes classes podem ser facilmente compreendidos.

```
data = [df_female['NEds'].sum(), df_male['NEds'].sum()]
tM, tF = "{:,}".format(data[1]), "{:,}".format(data[0])
labels = [f'Total de edições de mulheres ({tF})', f'Total de
edições de homens ({tM})']
colors= ['#CD5C5C', '#7FB3D5']

fig = plt.figure(figsize=(10, 10))
patches, texts = plt.pie(data, labels=labels, colors=colors)

plt.tight_layout()
```

output:



Ao analisar o dado de pizza, é possível notar a diferença entre o total de edições entre os gêneros, o que é afetado pela grande quantidade de editores masculinos em comparação ao feminino. Contudo, outra verificação foi realizada para visualizar a média de edições por pessoa agrupada por gênero.

```
qtd_F, qtd_M = df_female['gender'].count(),
df_male['gender'].count()
tF, tM = df_female['NEds'].sum(), df_male['NEds'].sum()

print(f'Os editores do gênero feminino apresentam em média
{round(tF/qtd_F, 1)} páginas editadas\n')
print(f'Os editores do gênero masculino apresentam em média
{round(tM/qtd_M, 1)} páginas editadas')
```

Output:

Os editores do gênero feminino apresentam em média 1864.6 páginas editadas
Os editores do gênero masculino apresentam em média 2309.3 páginas editadas

Análises iniciais

Visualização inicial da quantidade de gênero por registo no conjunto de dados

```
df_unknown = df[df['gender'] == 0]
df_male = df[df['gender'] == 1]
df_female = df[df['gender'] == 2]
for (l, c), gen in [(df_male.shape, 'Masculino'), (df_female.shape,
'Feminino'), (df_unknown.shape, 'Desconhecido')]:
  print(f'há {l} linhas de dados do gênero {gen}')
```

Output:

há 2792 linhas de dados do gênero Masculino
há 353 linhas de dados do gênero Feminino
há 1601 linhas de dados do gênero Desconhecido

2.5 Verificação da qualidade dos dados e ajustes necessários

Ao analisar a quantidade de valores nulos no conjunto de dados, nota-se que 100% dos valores estão preenchidos, assim, não sendo necessário tratamento para valores faltantes.

```
df.isnull().sum()
```

Output:

```
gender          0
C_api           0
C_man           0
E_NEds          0
E_Bpag          0
firstDay        0
lastDay         0
NEds            0
NDays           0
NActDays        0
NPages          0
NPcreated       0
pagesWomen      0
wikiprojWomen   0
ns_user         0
ns_wikipedia    0
ns_talk         0
ns_userTalk     0
ns_content      0
weightIJ        0
NIJ             0
   dtype: int64
```

Verificou-se a necessidade de se realizar um ajuste nos campos referente à datas, pois elas se encontram no formato AAAAMMDDhhmmss. A fim de facilitar a análise e mineração dos dados, identificamos que seria melhor transformar estas datas para o formato AAMM, e posteriormente, normalizar esses dados.

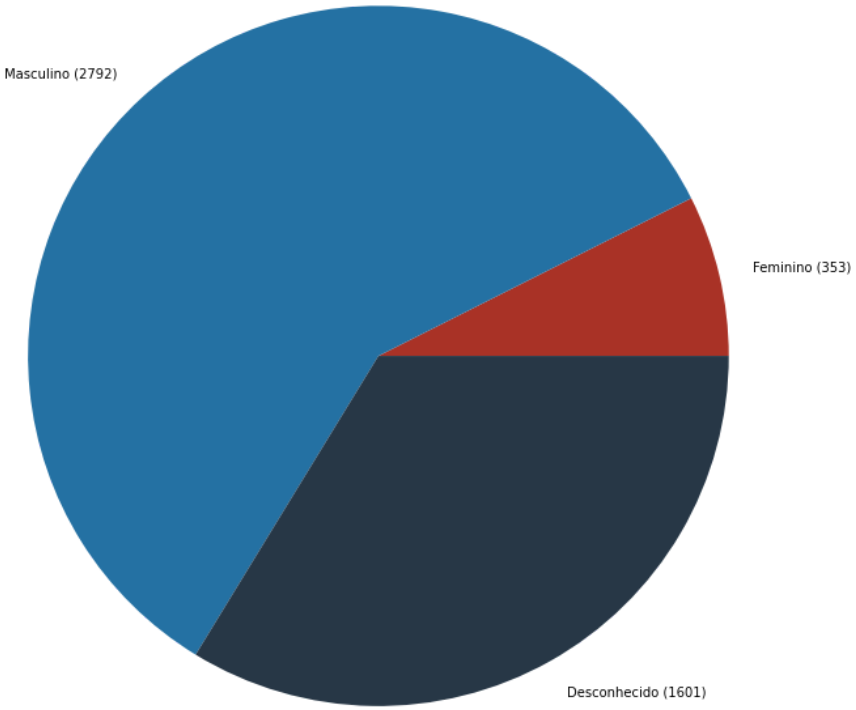
Os atributos **E_NEds**, **E_Bpag**, **weightIJ** e **NIJ** são informações referentes à técnica de amostragem de dados descrita no artigo de pesquisa, portanto, os consideramos como não necessários para realizar o projeto de mineração de dados.

Conforme dito no ponto 2.2 [Descrição do conjunto de dados], os atributos **C_API** e **C_MAN** deram suporte para a criação do atributo principal **gender** e, por isso,

também não serão necessários uma vez que a informação necessária já se encontra em gender.

2.6 Complexidade da base de dados

A base de dados possui uma distribuição desbalanceada de suas classes (Masculino e Feminino), o que pode afetar negativamente o treinamento dos modelos. Portanto, deve ser necessário utilizar técnicas de balanceamento dos dados para que os gêneros desconhecidos sejam atribuídos de forma mais sucinta possível para seus respectivos gêneros



3. Preparação dos Dados

3.1 Selecionando atributos relevantes

Assim como comentado no ponto 2.5 [Verificação da qualidade dos dados e ajustes necessários], os atributos [**E_NEds | E_Bpag | weightIJ | NIJ**] foram removidos porque são informações da técnica de amostragem de dados descritos no artigo, portanto, não é necessário para o treinamento dos modelos de predição de dados.

```
df = df[['gender', 'C_api', 'C_man', 'firstDay', 'lastDay', 'NEds',
'NDays', 'NActDays', 'NPages', 'NPcreated', 'pagesWomen',
'wikiprojWomen', 'ns_user', 'ns_wikipedia', 'ns_talk', 'ns_userTalk',
'ns_content']]
```

Removendo os atributos **C_api** e **C_man** para termos somente uma classe de gênero no dataset.

```
df = df[['gender', 'firstDay', 'lastDay', 'NEds', 'NDays', 'NActDays',
'NPages', 'NPcreated', 'pagesWomen', 'wikiprojWomen', 'ns_user',
'ns_wikipedia', 'ns_talk', 'ns_userTalk', 'ns_content']]
```

3.2 Transformando dados

Como observado na nossa exploração de dados, os atributos de data são de suma importância para o treinamento do modelo, visto que existe uma diferença entre os períodos observados na classe Masculino e Feminino. Portanto, vamos converter as datas que estão no formato AAAAMMDDhhmmss para AAMM, e posteriormente, aplicando a normalização desses dados.

```
def date_to_AAAAMMDD(date):
  return str(date)[:6]

df['firstDay'] = df['firstDay'].apply(date_to_AAAAMMDD)
df['lastDay']  = df['lastDay'].apply(date_to_AAAAMMDD)
df.head()
```

Output:

	gender	firstDay	lastDay	NEds	NDays	NActDays	NPages	NPcreated	pagesWomen	wikiprojWomen	ns_user	ns_wikipedia	ns_talk	ns_userTalk	ns_content
0	1	201705	201707	543	56	43	204	4	0	0	91	28	6	76	324
1	0	201103	201707	2764	2345	514	722	7	0	0	100	249	183	646	1526
2	1	200609	201409	57	2927	25	25	0	0	0	3	0	1	3	49
3	1	201210	201212	104	67	5	66	2	0	0	20	1	2	2	78
4	0	200703	201411	184	2798	27	125	0	0	0	26	10	5	24	112

3.3 Normalizando dados

Visto que os dados não são normalmente distribuídos, vamos a aplicar o Standard Scaler para realizar a normalização de dados. Essa normalização utiliza a média e o desvio padrão, que é mais útil quando existem muitos outliers na base de dados.

3.3.1 Separando os atributos previsores

```
previsores = ['firstDay', 'lastDay', 'NEds', 'NDays', 'NActDays',
'NPages', 'NPcreated', 'pagesWomen', 'wikiprojWomen', 'ns_user',
'ns_wikipedia', 'ns_talk', 'ns_userTalk', 'ns_content']

df_class = df[['gender']]
df       = df[previsores]
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)
```

O tipo de dado muda, não é mais um DF, e sim um Numpy Array

```
    min(df_scaled[0]), max(df_scaled[0]), type(df_scaled)
```

Output:

```
(-1.4825140715257943, 2.0221692216156013, numpy.ndarray)
```

3.3.2 Convertendo Numpy Array para Pandas Dataframe e concatenando a classe Gender

```
df_scaled = pd.DataFrame(data=df_scaled, columns = df.columns)
df_scaled = pd.concat([df_scaled, df_class], axis=1)
```

```
df_scaled.describe()
```

Output:

	firstDay	lastDay	NEds	NDays	NActDays	NPages	NPcreated	pagesWomen	wikiprojWomen	ns_user	ns_wikipedia	ns_talk	ns_userTalk	ns_content	gender
count	4.746000e+03	4.746000e+03	4.746000e+03	4.746000e+03	4.746000e+03	4.746000e+03	4.746000e+03	4.746000e+03	4.746000e+03	4.746000e+03	4.746000e+03	4.746000e+03	4.746000e+03	4.746000e+03	4746.000000
mean	5.595094e-16	-5.313152e-15	2.574379e-17	-1.375498e-17	2.579058e-17	1.919381e-17	7.565237e-17	2.721637e-16	-5.079165e-17	-4.982085e-17	-1.296137e-16	1.860957e-16	2.076112e-16	1.451524e-17	0.737042
std	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	0.585355
min	-2.255495e+00	-1.989587e+00	-2.540872e-01	-1.523682e+00	-4.870724e-01	-2.052049e-01	-1.462151e-01	-8.239270e-02	-2.467417e-02	-2.978024e-01	-1.326292e-01	-2.317421e-01	-1.763070e-01	-2.495564e-01	0.000000
25%	-8.248665e-01	-8.110295e-01	-2.483124e-01	-8.992340e-01	-4.255742e-01	-1.968590e-01	-1.428522e-01	-8.239270e-02	-2.467417e-02	-2.856262e-01	-1.326292e-01	-2.317421e-01	-1.744720e-01	-2.395538e-01	0.000000
50%	-2.332746e-01	3.332011e-01	-2.325280e-01	-8.292647e-04	-3.480331e-01	-1.852344e-01	-1.327636e-01	-8.239270e-02	-2.467417e-02	-2.450387e-01	-1.308458e-01	-2.131834e-01	-1.671322e-01	-2.247957e-01	1.000000
75%	8.816484e-01	9.053164e-01	-1.632625e-01	8.307705e-01	-7.797606e-02	-1.400027e-01	-9.913481e-02	-8.239270e-02	-2.467417e-02	-1.151587e-01	-1.183619e-01	-1.435880e-01	-1.359377e-01	-1.571136e-01	1.000000
max	2.033546e+00	9.224799e-01	1.939858e+01	2.479374e+00	9.785790e+00	2.785513e+01	4.489620e+01	3.464714e+01	5.319914e+01	2.421705e+01	4.336832e+01	2.198310e+01	2.248556e+01	1.869765e+01	2.000000

3.4 Separando o dataset entre gêneros conhecidos e desconhecidos

Vamos separar o dataset entre o conjunto de dados que possuem gêneros definidos (Masculino e Feminino) e o conjunto de dados que possuem gêneros desconhecidos. Essa separação se faz necessária para podermos utilizar a técnica de desbalaceamento do dataset posteriormente. Como também, as instâncias de classe "Desconhecida" vão ser utilizadas em produção após o treinamento do modelo.

```
df_scaled_unknown = df_scaled[df_scaled['gender'] == 0]
df_scaled_known   = df_scaled[df_scaled['gender'] != 0]

df_scaled_unknown.shape, df_scaled_known.shape
```

Output:

```
((1601, 15), (3145, 15))
```

3.5 Realizando o balanceamento do dataset

Visto que os dados estão desbalanceados, nós vamos aplicar a técnica de Oversampling para replicar as observações com menor quantidade para se equalizar em números de classificação.

Vamos utilizar o método SMOTE que encontra os vizinhos próximos para as classes em minoria para cada amostra das classificações e, em seguida, traça uma reta entre o ponto original e o vizinho para definir a localização da observação da observação genérica.

```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split

X, y = df_scaled_known[previsores], df_scaled_known['gender']

X_resampled, y_resampled = SMOTE().fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(X_resampled,
y_resampled, test_size = .25, random_state=1)
```

Verificando número de instâncias por classe

```
y_resampled.value_counts()
```

Output:

```
1    2792
2    2792
Name: gender, dtype: int64
```

```
print(f'{X_train.shape[0]} dados para treinar o modelo e {X_test.shape[0]}
para testar o modelo treinado.')
```

Output:

4188 dados para treinar o modelo e 1396 para testar o modelo treinado.

3.6 Exportando dataset para realizar a predição e as variáveis de treino e teste

```
import pickle
with open(path+'train_test.pkl', 'wb') as f:
    pickle.dump([X_train, X_test, y_train, y_test], f)
df_scaled_unknown.to_csv(path+'data_predict.csv')
```

4. Modelagem dos Dados

4.1 Importação dos dados divididos em bases de treino, validação e teste

Foi realizada a importação do conjunto de treino e teste de maneira que o mesmo conjunto de dados seja utilizado no treinamento de todos os modelos.

```
def open_train_test():
  with open('train_test.pkl','rb') as f:
    return pickle.load(f)
X_train, X_test, y_train, y_test = open_train_test()
```

4.2 Cross Validation

O Cross validation cria várias combinações diferentes para treino e teste, o que torna o resultado das métricas mais preciso e condizente com o desempenho real do algoritmo.

Para definir o número de CV no nosso algoritmo, vamos utilizar o StratifiedShuffleSplit, que fornece índices de treinamento/teste para dividir dados em conjuntos de treinamento/teste.

Definimos em 10 o número de iterações de embaralhamento do conjunto de treino e teste, divididos em 80% para treino e 20% para teste.

```
from sklearn.model_selection import StratifiedShuffleSplit

cv_sss = StratifiedShuffleSplit(n_splits = 10, test_size = 0.2)
```

4.3 Variação paramétrica - GridSearchCV

O GridSearchCV é um módulo do Scikit Learn e é amplamente usado para automatizar grande parte do processo de tuning. O objetivo primário do GridSearchCV é a criação de combinações de parâmetros para posteriormente avaliá-las.
Vamos visualizar os resultados dos testes utilizando 4 métricas diferentes:

- **Precision:** é usada para medir o desempenho do modelo ao medir a contagem de positivos verdadeiros da maneira correta de todas as previsões positivas feitas.
- **Recall:** é usada para medir o desempenho do modelo ao medir a contagem de positivos verdadeiros de maneira correta de todos os valores positivos reais.
- **Precision: é** usada para medir o desempenho do modelo ao medir a razão da soma de positivos verdadeiros e verdadeiros negativos de todas as previsões
- **F1-score:** é a média harmônica de precision e recall e é usada como uma métrica nos cenários em que a escolha de precisão ou pontuação de recall pode resultar em comprometimento em termos de modelo dando altos falsos positivos e falsos negativos, respectivamente.

Contudo, a métrica **rank_test_f1** foi definida como a principal métrica para se analisar os parâmetros, seguido das métricas: rank_test_accuracy, rank_test_recall, e rank_test_precision.

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer
from sklearn.metrics import precision_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import fbeta_score

metrics = {'accuracy' :make_scorer(accuracy_score),
           'recall'   :make_scorer(recall_score),
           'precision':make_scorer(precision_score),
           'f1'       :make_scorer(fbeta_score, beta = 1)}
```

Abaixo está uma demonstração de como o GridSearchCV será utilizado na realização dos testes de parâmetros e no relacionamento com o modelo em questão.

```
test_1_knn = GridSearchCV(estimator = knn,
                      param_grid = knn_parameters,
                      cv = cv_sss,
                      scoring = metrics,
                      refit=False)

test_1_knn.fit(X_train, y_train)
```

Criando um Dataframe pandas para melhor visualização dos resultados coletados.

```
df_knn_1 = pd.DataFrame(test_1_knn.cv_results_)[['params', 'rank_test_f1',
'rank_test_accuracy', 'rank_test_recall',
'rank_test_precision','mean_test_accuracy',
'mean_test_recall','mean_test_precision','mean_test_f1']]

df_knn_1.sort_values(by=['rank_test_f1', 'rank_test_accuracy',
'rank_test_recall', 'rank_test_precision'], inplace=True)

df_knn_1.reset_index(drop=True, inplace=True)
```

A tabela abaixo mostra o Dataframe do teste de parâmetros do modelo de Rede Neural, com isso, fica simples entender o rankeamento dos parâmetros testados.

	params	rank_test_f1	rank_test_accuracy	rank_test_recall	rank_test_precision	mean_test_accuracy	mean_test_recall	mean_test_precision	mean_test_f1
0	{'activation': 'relu', 'hidden_layer_sizes': (...	1	4	37	7	0.752625	0.734928	0.761291	0.747788
1	{'activation': 'relu', 'hidden_layer_sizes': (...	2	3	41	4	0.754535	0.729426	0.767178	0.747676
2	{'activation': 'tanh', 'hidden_layer_sizes': (...	3	1	51	1	0.755847	0.721531	0.774109	0.746758
3	{'activation': 'tanh', 'hidden_layer_sizes': (...	4	2	48	2	0.755728	0.722967	0.773969	0.746755
4	{'activation': 'tanh', 'hidden_layer_sizes': (...	5	4	44	5	0.752625	0.725598	0.766649	0.745090

4.4. Aplicação de scripts de experimentação e análise de dados

4.4.1 KNN

4.4.1.1 KNN - Teste 1

Na primeira etapa de teste do KNN, vamos utilizar o conjunto dos parâmetros possíveis para esse modelo. A partir dos resultados obtidos, vamos realizar o afunilamento de parâmetros para extrair a melhor combinação possível.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
```

```
knn_parameters = {
      'n_neighbors': (5, 7, 9, 11, 13, 15)
    , 'weights': ['uniform', 'distance']
    , 'algorithm': ['auto','ball_tree','kd_tree','brute']
    , 'p': (1,2)
    , 'metric': ['euclidean', 'manhattan', 'chebyshev', 'minkowski',
'wminkowski', 'seuclidean', 'mahalanobis']
}
```

Realizando a contagem dos parâmetros que foram utilizados nas 10 melhores classificações

Algorithm	
parâmetro	**Quantidade**
auto	3
ball_tree	3
brute	1
kd_tree	3

Metric	
Parâmetro	**Quantidade**
chebyshev	0
mahalanobis	0
manhattan	7
minkowski	3
seuclidean	0

n_neighbors	
Parâmetro	**Quantidade**
5	10
7	0
9	0
11	0
13	0
15	0

p	
Parâmetro	**Quantidade**
1	7
2	3

weights	
Parâmetro	**Quantidade**
distance	10
uniform	0

4.4.1.2 KNN - Teste 2

Vamos utilizar os parâmetros mais recorrentes nas melhores classificações e realizar uma segunda rodada de testes para determinar a melhor combinação de parâmetros.

```
knn_parameters = {
      'n_neighbors': (3, 4, 5)
    , 'weights': ['distance']
    , 'algorithm': ['auto', 'ball_tree','kd_tree']
    , 'p': (1,)
    , 'metric': ['manhattan']
}
```

Melhor conjunto de parâmetros

Parâmetro	**Valor**
algorithm	auto
metric	manhattan
n_neighbors	4
p	1
weights	distance

Indicadores

mean test accuracy	**mean test recall**	**mean test precision**	**mean test f1**
0.859189	0.753828	0.954295	0.842145

4.4.1.3 KNN - Classificação com os melhores parâmetros

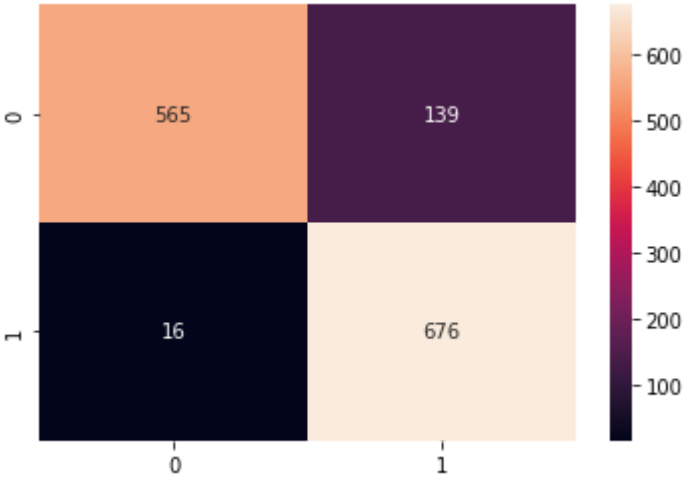
Métricas de desempenho do modelo

```
1: Masculino
2: Feminino
              precision    recall  f1-score   support

           1       0.97      0.80      0.88       704
           2       0.83      0.98      0.90       692

    accuracy                           0.89      1396
   macro avg       0.90      0.89      0.89      1396
weighted avg       0.90      0.89      0.89      1396
```

Matriz de confusão



4.4.2 Árvore de Decisão

4.4.2.1 Árvore de Decisão - Teste 1

Na primeira etapa de teste do Decision Tree, vamos utilizar o conjunto dos parâmetros possíveis para esse modelo. A partir dos resultados obtidos, vamos realizar o afunilamento de parâmetros para extrair a melhor combinação possível.

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
```

```
dt_parameters = {
       'criterion':  ['gini', 'entropy']
      ,'min_samples_split': [2,3,5,7]
      ,'max_depth': [3,5,6,7,9,11,13,15]
      ,'min_samples_leaf':  [2, 3]
}
```

Realizando a contagem dos parâmetros que foram utilizados nas 10 melhores classificações

Criterion	
parâmetro	**Quantidade**
gini	4
entropy	6

min_samples_leaf	
parâmetro	**Quantidade**
2	8
3	2

min_sample_split	
Parâmetro	**Quantidade**
2	2
3	2

5	3
7	3

max_depth	
Parâmetro	**Quantidade**
3	0
5	0
6	0
7	0
9	0
11	0
13	0
15	10

4.4.3.2 Árvore de Decisão - Teste 2

Vamos utilizar os parâmetros mais recorrentes nas melhores classificações e realizar uma segunda rodada de testes para determinar a melhor combinação de parâmetros

```
dt_parameters = {
       'criterion':  ['gini', 'entropy']
      ,'min_samples_split': [5,7,9]
      ,'max_depth': [15, 17, 19, 21, 23]
      ,'min_samples_leaf':  [2]
}
```

Melhor conjunto de parâmetros

Parâmetro	**Valor**
criterion	entropy
min_samples_split	5
max_depth	23
min_samples_leaf	2

Indicadores

mean test accuracy	**mean test recall**	**mean test precision**	**mean test f1**
0.844272	0.840191	0.847152	0.843239

4.4.3.3 Árvore de Decisão -Classificação com os melhores parâmetros

```
dt = DecisionTreeClassifier(criterion = criterion, min_samples_leaf =
min_samples_leaf, max_depth = max_depth, min_samples_split =
min_samples_split)
dt.fit(X_train, y_train);
y_pred = dt.predict(X_test)
```

Métricas de desempenho do modelo

```
print('1: Masculino\n2: Feminino\n',classification_report(y_test, y_pred))
```

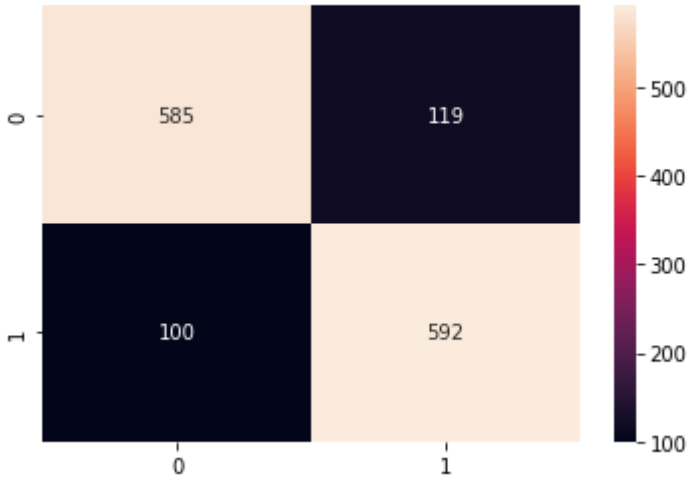
```
1: Masculino
2: Feminino
              precision    recall  f1-score   support

           1       0.85      0.83      0.84       704
           2       0.83      0.86      0.84       692

    accuracy                           0.84      1396
   macro avg       0.84      0.84      0.84      1396
weighted avg       0.84      0.84      0.84      1396
```

Matriz de confusão

```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='g')
```



4.4.4 Rede Neural

4.4.4.1 Teste 1

Na primeira etapa de teste da Rede Neural, vamos utilizar o conjunto dos parâmetros possíveis para esse modelo. A partir dos resultados obtidos, vamos realizar o afunilamento de parâmetros para extrair a melhor combinação possível.

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier()
```

```
mlp_parameters = {
       'solver': ['lbfgs', 'adam', 'sgd']
      ,'activation': ['identity', 'tanh', 'logistic','relu']
      ,'hidden_layer_sizes': [(14, 7, 2, 7, 14), (14, 2, 14), (8, 16, 32),
(2, 4, 8, 16), (20, 10, 20)]
      ,'learning_rate':  ['constant', 'adaptive', 'invscaling']
}
```

Realizando a contagem dos parâmetros que foram utilizados nas 10 melhores classificações

Solver	
parâmetro	**Quantidade**
lbfgs	8
adam	2
sgd	0

Activation	
Parâmetro	**Quantidade**
identity	0
tanh	5
logistic	0
relu	5

Hidden_layer_sizes	
Parâmetro	**Quantidade**
(14, 7, 2, 7, 14)	0
(14, 2, 14)	0
(8, 16, 32)	2
(2, 4, 8, 16)	0
(20, 10, 20)	8

Learning_rate	
Parâmetro	**Quantidade**
constant	2
adaptative	4
invscaling	4

4.4.4.2 Teste 2

Vamos utilizar os parâmetros mais recorrentes nas melhores classificações e realizar uma segunda rodada de testes para determinar a melhor combinação de parâmetros.

```
mlp_parameters = {
       'solver': ['lbfgs']
      ,'activation': ['tanh','relu']
      ,'hidden_layer_sizes': [(20, 10, 20), (8, 12), (12, 8), (20, 12, 8)]
      ,'learning_rate':  ['adaptive', 'invscaling']
}
```

Melhor conjunto de parâmetros

Parâmetro	Valor
solver	lbfgs
activation	tahn
hidden_layer_sizes	(20, 10, 20)
learning_rate	invscaling

Indicadores

mean test accuracy	mean test recall	mean test precision	mean test f1
0.757637	0.728230	0.773590	0.749838

4.4.4.3 Rede Neural - Classificação com os melhores parâmetros

```
mlp = mlp = MLPClassifier(solver = solver, activation = activation,
hidden_layer_sizes = hidden_layer_sizes, learning_rate = learning_rate)

mlp.fit(X_train, y_train);

y_pred = mlp.predict(X_test)
```

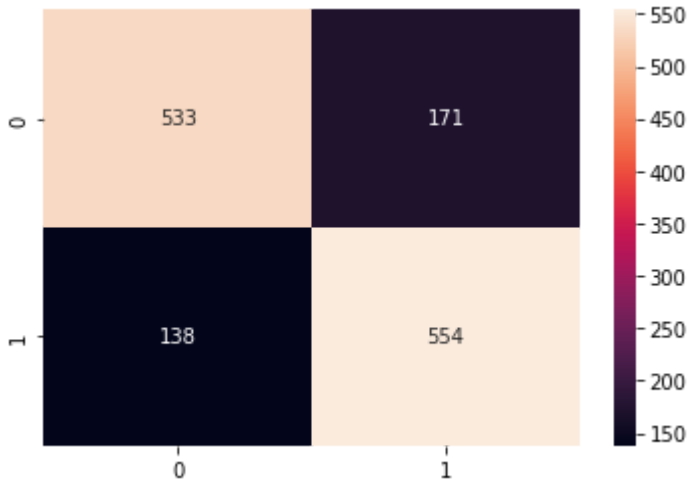
Métricas de desempenho do modelo

```
1: Masculino
2: Feminino
              precision    recall  f1-score   support

           1       0.79      0.76      0.78       704
           2       0.76      0.80      0.78       692

    accuracy                           0.78      1396
   macro avg       0.78      0.78      0.78      1396
weighted avg       0.78      0.78      0.78      1396
```

Matriz de confusão



4.4.5 Comitês de Redes Neurais

Na primeira etapa vamos importar os modelos treinados anteriormente, sendo eles: KNN, Rede Neural e Árvore de Decisão

```
with open(path+'KNN_model.pkl','rb') as f:
  KNN = pickle.load(f)[0]
```

```
with open(path+'DT_model.pkl','rb') as f:
  DT = pickle.load(f)[0]

with open(path+'MLP_model.pkl','rb') as f:
  MLP = pickle.load(f)[0]

estimators = [('DT', DT), ('KNN', KNN), ('MLP', MLP)]
```

```
from sklearn.ensemble import VotingClassifier
from sklearn.model_selection import cross_val_score

voting = VotingClassifier(estimators)
```

```
VOT = voting.fit(X_train, y_train)
y_pred = VOT.predict(X_test)
```

Métricas de desempenho do modelo

```
print('1: Masculino\n2: Feminino\n',classification_report(y_test, y_pred))
```

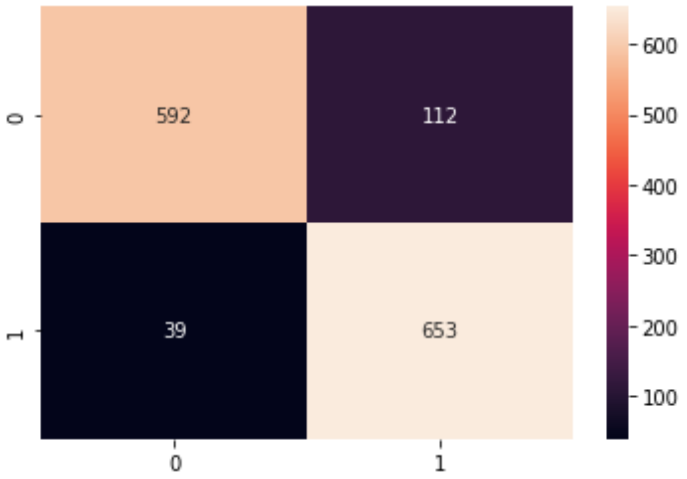
```
1: Masculino
2: Feminino
              precision    recall  f1-score   support

           1       0.94      0.84      0.89       704
           2       0.85      0.94      0.90       692

    accuracy                           0.89      1396
   macro avg       0.90      0.89      0.89      1396
weighted avg       0.90      0.89      0.89      1396
```

Matriz de confusão

```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='g')
```



5. Aplicação de scripts utilizando os modelos treinados

Vamos utilizar o classificador Ensemble Voting, visto que ele obteve os melhores resultados nos testes realizados. Nossa base de dados de produção foi tratada e separada na etapa de tratamento de dados, nela, contém todos os perfis em que o gênero é desconhecido.

```
df = pd.read_csv(path+'data_predict.csv')
df.drop(columns=['Unnamed: 0', 'gender'], inplace = True)
df.head()
```

Output:

	firstDay	lastDay	NEds	NDays	NActDays	NPages	NPcreated	pagesWomen	wikiprojWomen	ns_user	ns_wikipedia	ns_talk	ns_userTalk	ns_content
0	0.309966	0.905316	0.094197	0.230836	0.884603	0.009702	-0.122675	-0.082393	-0.024674	0.104014	0.311440	0.617321	1.009083	0.000675
1	-0.827711	-0.788145	-0.236891	0.569913	-0.417553	-0.168245	-0.146215	-0.082393	-0.024674	-0.196334	-0.114795	-0.208544	-0.132268	-0.231191
2	0.890181	-1.383145	-0.253189	-1.516946	-0.471029	-0.201628	-0.142852	-0.082393	-0.024674	-0.261274	-0.132629	-0.213183	-0.165297	-0.243489
3	0.315654	0.344643	-0.239971	-0.055845	-0.382793	-0.192984	-0.146215	-0.082393	-0.024674	-0.155746	-0.132629	-0.073993	-0.124928	-0.239718
4	0.045456	-0.238914	-0.250109	-0.215279	-0.388141	-0.195369	-0.146215	-0.082393	-0.024674	-0.167922	-0.132629	-0.231742	-0.176307	-0.244637

5.1 Aplicando o modelo

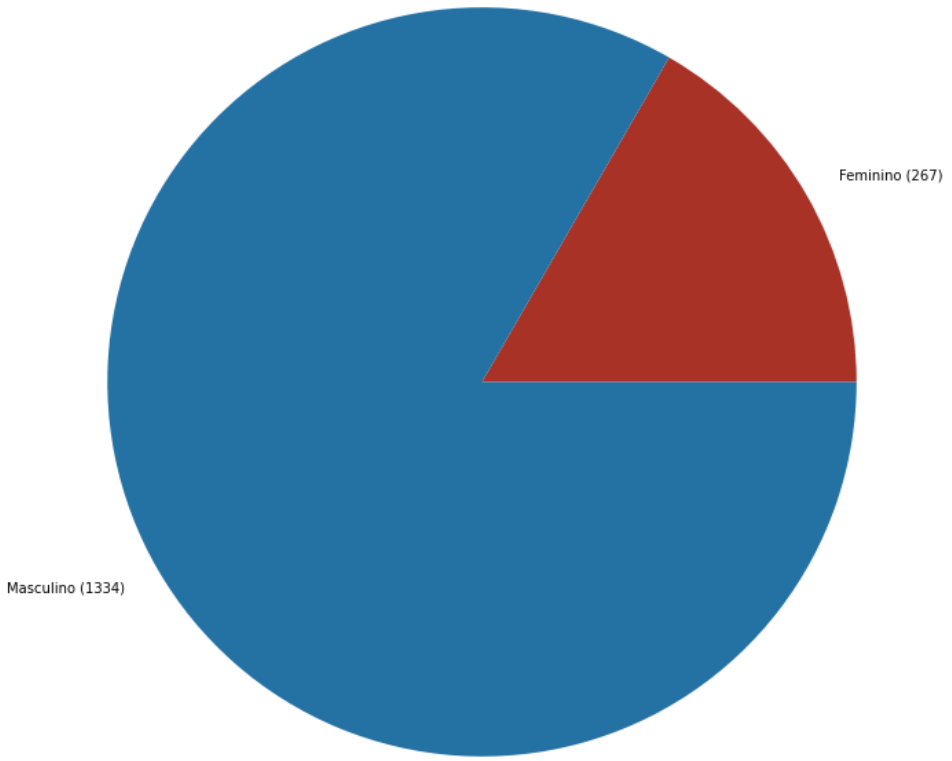
```
pred = voting_model.predict(df)
df_pred = pd.DataFrame(data=pred, columns = ['gender_ml'])
df_final = pd.concat([df, df_pred], axis=1)
```

Output:

	firstDay	lastDay	NEds	NDays	NActDays	NPages	NPcreated	pagesWomen	wikiprojWomen	ns_user	ns_wikipedia	ns_talk	ns_userTalk	ns_content	gender_ml
0	0.309966	0.905316	0.094197	0.230836	0.884603	0.009702	-0.122675	-0.082393	-0.024674	0.104014	0.311440	0.617321	1.009083	0.000675	1
1	-0.827711	-0.788145	-0.236891	0.569913	-0.417553	-0.168245	-0.146215	-0.082393	-0.024674	-0.196334	-0.114795	-0.208544	-0.132268	-0.231191	1
2	0.890181	-1.383145	-0.253189	-1.516946	-0.471029	-0.201628	-0.142852	-0.082393	-0.024674	-0.261274	-0.132629	-0.213183	-0.165297	-0.243489	2
3	0.315654	0.344643	-0.239971	-0.055845	-0.382793	-0.192984	-0.146215	-0.082393	-0.024674	-0.155746	-0.132629	-0.073993	-0.124928	-0.239718	1
4	0.045456	-0.238914	-0.250109	-0.215279	-0.388141	-0.195369	-0.146215	-0.082393	-0.024674	-0.167922	-0.132629	-0.231742	-0.176307	-0.244637	1

6. Avaliação dos resultados

Ao visualizar o gráfico de pizza, é possível notar que a proporção de homens e mulheres mantém o mesmo padrão dos perfis com gêneros já conhecidos.



6.1 Aplicação de PCA (principal component analysis) e visualização dos resultados

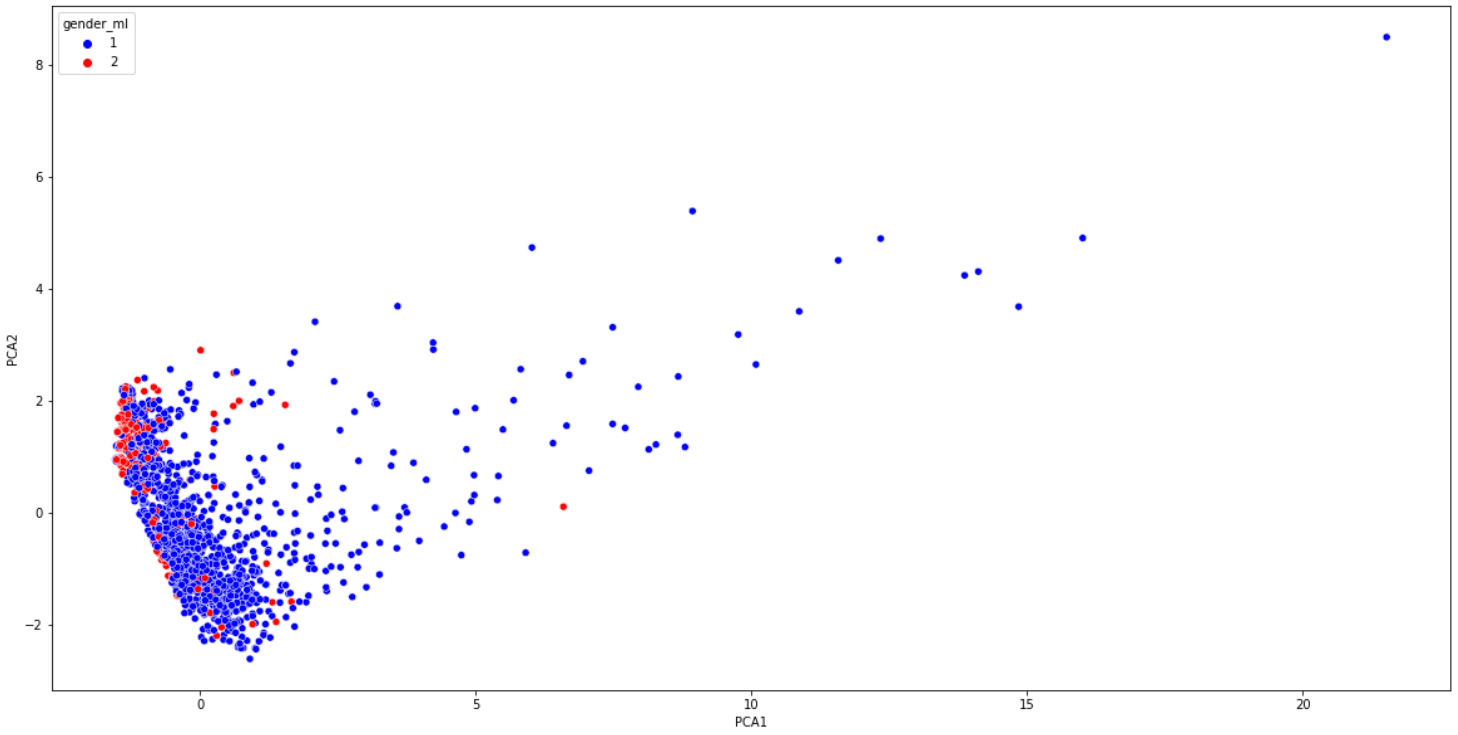
Vamos diminuir a dimensionalidade dos dados para visualizar no gráfico como os gêneros estão distribuídos.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
principal_comp = pca.fit_transform(df)
pca_df = pd.DataFrame(data = principal_comp, columns = ['PCA1','PCA2'])
pca_df = pd.concat([pca_df, class_df], axis=1)
```

Plotando o gráfico

```
sns.scatterplot(x = 'PCA1', y = 'PCA2', hue = 'gender_ml', data = pca_df,
palette = ['blue','red'])
```



7. Conclusão

No desenvolvimento do projeto de processamento de dados, conseguimos com sucesso aplicar a metodologia do CRISP-DM para a obtenção de resultados e análises referentes ao nosso domínio de dados.

Foi também possível responder às nossas metas de ciência de dados, definidas na primeira fase do CRISP-DM. Conseguimos estimar com sucesso:

- qual o número de editores, artigos publicados, páginas criadas e editadas por cada gênero;
- qual é a porcentagem de mulheres entre os editores ativos;
- analisar se homens e mulheres começaram a editar no Wikipedia em períodos semelhantes;

Finalizamos o projeto entendendo a suma importância das ferramentas propostas pelo CRISP-DM, uma vez que elas incentivam as melhores práticas e permitem a replicação de projetos. A metodologia também fornece uma estrutura uniforme para planejar, implantar e gerenciar um projeto de Data Science.

O CRISP-DM está se tornando o modelo de processo padrão da indústria para mineração de dados, com um número cada vez maior de aplicações, como em diagnósticos de qualidade, garantia e outros. No atual mercado global de extrema competitividade, os dados desempenham um papel muito importante e processos como mineração de dados ajudam a gerar insights valiosos, extrair padrões e identificar relacionamentos de grandes conjuntos de dados.

Referências

- CRISP-DM Phase 1: Business Understanding | by Zipporah Luna | Analytics Vidhya | Medium
- CRISP-DM Phase 2: Data Understanding | by Zipporah Luna | Analytics Vidhya | Medium
- CRISP-DM Phase 3: Data Preparation| by Zipporah Luna | Analytics Vidhya | Medium
- CRISP-DM Phase 4: Modeling Phase | by Zipporah Luna | Analytics Vidhya | Medium
- MinguillÃ³n J, Meneses J, Aibar E, Ferran-Ferrer N, FÃ bregues S (2021) Exploring the gender gap in the Spanish Wikipedia: Differences in engagement and editing practices.
- Ferran-Ferrer, Núria; Castellanos-Pineda, Patricia; Minguillón, Julià; Meneses, Júlio (2021). "A diferença de gênero na Wikipédia espanhola: ouvindo as vozes das mulheres editoras". Profesional de la información, v. 30, n. 5, e300516.
- Como lidar com dados desbalanceados em problemas de classificação | by Arthur Lamblet Vaz | Medium
- Accuracy, Precision, Recall & F1-Score – Python Examples | by Ajitesh Kumar
- Como usar o GridSearchCV | by AH Uyekita