



# ANÁLISE DE SENTIMENTO - PLN

BTG Pactual



## Controle do Documento

### Histórico de revisões

Data	Autor	Versão	Resumo da atividade
25/04/2023	Maria Luísa Maia	1.0	Parceiro de Negócios Definição do problema
26/04/2023	Maria Luísa Maia Vitor Oliveira	1.1	User stories
27/04/2023	Maria Luisa Maia Vitor Oliveira	1.2	Revisão dos textos Atualização da User story
28/04/2023	Maria Luísa Maia	1.3	Análise do negócio Explicação Matriz de Risco Explicação Proposta de Valor Revisão User Story
30/04/2023	Maria Luísa Maia Pedro Rezende Vitor Oliveira	1.4	Revisão geral dos textos Implementação da Matriz Oceano Azul Organização da Matriz de Risco Organização das User Stories
01/05/2023	Pedro Rezende	1.5	Revisão geral dos textos Organização da Matriz Oceano Azul
01/05/2023	Pedro Rezende Vitor Oliveira	1.6	Markdown no Github
11/05/2023	Maria Luísa Maia Daniel Barzilai	2.0	Análise descritiva
11/05/2023	Larissa Carvalho Rafael Moritz	2.1	Pré-processamento
11/05/2023	Pedro Rezende Vitor Oliveira	2.2	Modelo
12/05/2023 13/05/2023	Maria Luísa	2.3	Revisão e formatação no Google Docs
13/05/2023	Maria Luísa Larissa Carvalho	2.4	Markdown no Github

15/05/2023	Maria Luísa Larissa Carvalho	3.0	Revisão e melhorias de feedbacks sprint 1
27/05/2023	Maria Luísa	3.1	Revisão texto geral
27/05/2023	Maria Luísa	3.2	Pré - processamento - atualização de novos processos
28/05/2023	Larissa Carvalho	3.4	Comparação entre os modelos - seção 8.1 a 8.4
06/05/2023	Maria Luísa	4.0	Documentação da rede neural - sequência de palavras (9.7)
07/05/2023	Maria Luísa	4.1	Revisão e alteração de códigos (9.7)
08/05/2023	Maria Luísa	4.2	Revisão e alteração do código (9.6)
09/05/2023	Maria Luísa	4.3	Revisão e alteração do código (9.5)
10/06/2023	Larissa Carvalho	4.4	Rede Neural sem embedding - 10.8 Random Forest Word2vec - 10.9 Decision Tree - 10.11
11/06/2023	Maria Luísa	4.5	Comparação entre os modelos
18/06/2023	Maria Luísa	5.0	Modelos Naive Bayes e Random Forest com TF-IDF
21/06/2023	Larissa Carvalho	5.1	Modelo Random Forest com TF-IDF (base das novas features) - 10.14 Modelo Rede Neural com TF-IDF (base das novas features) - 10.15
22/06/2023	Larissa Carvalho	5.2	Modelo Naive Bayes com TF-IDF (base das novas features) - 10.16
22/06/2023	Maria Luísa	5.3	Arquitetura e Diagrama da solução Revisão final

# Sumário

<b>Sumário</b>	<b>4</b>
<b>Índice de figuras</b>	<b>8</b>
<b>Índice de tabelas</b>	<b>9</b>
<b>1. Introdução</b>	<b>10</b>
<b>2. Problema</b>	<b>10</b>
<b>3. Objetivos</b>	<b>11</b>
<b>4. Análise de Negócios</b>	<b>11</b>
4.1. Contexto da indústria	11
4.1.1 Principais players	12
4.1.2 Modelo de negócio	15
4.1.3 Tendências	15
4.1.4 Análise 5 forças de Porter:	16
4.2 Matriz de Avaliação de valor Oceano Azul	18
4.3 Análise Financeira do Projeto	20
4.4 Value Proposition Canvas	22
4.5 Matriz de Risco	26
<b>5. Análise de Experiência do Usuário</b>	<b>29</b>
5.1 Personas	29
5.2 Jornadas do Usuário	33
5.3 User Stories	36
<b>6. Análise descritiva</b>	<b>40</b>
6.1 Introdução	40
6.2 Método	40
6.3 Resultados	41
6.3.1 Autores	41
6.3.2 Tipos de interação	42
6.3.3 Classificação de sentimento	44
6.3.4 Frequência dos sentimentos por tipo de interação	45
6.4 Conclusão	47
<b>7. Criação de features</b>	<b>47</b>
7.1 Introdução	47
7.2 Método	47
7.3 Resultados	47
7.3.1 Extração e Contagem	47
7.3.2 Criação das Features	49
7.4 Conclusão	49
<b>8. Pré processamento</b>	<b>50</b>
8.1 Introdução	50
8.2 Método	50
8.3. Resultados	51
8.3.1 Tratamento dos dados	51
8.3.2 Tokenização	52
8.3.3 Tratamento de emoji	52

8.3.4 Remoção de alfanuméricos	53
8.3.5 Tratamento de abreviações	54
8.3.6 Remoção de StopWords	55
8.3.7 Lematização	56
8.3.8 Pipeline	56
8.4 Conclusão	57
<b>9. Vetorização</b>	<b>58</b>
9.1 Bag of Words	58
9.1.1 Introdução	58
9.1.2 Método	58
9.1.3 Resultados	58
9.1.4 Conclusão	60
9.2 Word2Vec	62
9.2.1 Introdução	62
9.2.2 Método	62
9.2.3 Resultados	62
9.2.4 Conclusão	63
9.3 TF - IDF	63
9.3.1 Introdução	63
9.3.2 Método	63
9.3.3 Resultados	63
9.3.4 Conclusão	65
9.4 Comparação entre os modelos de vetorização	66
9.4.1 Representação Vetorial	66
9.4.2 Tamanho da Representação	67
9.4.3 Semântica e Contexto	67
9.4.4 Flexibilidade	68
9.4.5 Uso de contexto local	69
<b>10. Modelos</b>	<b>70</b>
10.1 Naive Bayes + Bow	70
10.1.1 Introdução	70
10.1.2 Método	70
10.1.3 Resultados	70
10.1.3.1 Naive Bayes	70
10.1.3.1 Naive Bayes - Cross Validation	74
10.1.3.1 Naive Bayes - Grid Search	77
10.1.4 Conclusão	80
10.2 Rede Neural (Seqüência de palavras) - Word2Vec	80
10.2.1 Introdução	80
10.2.2 Método	80
10.2.3 Resultado	80
10.2.3.1 Leitura da base de dados	80
10.2.3.2 Separação de treino e teste	81
10.2.3.3 Criação do modelo	83
10.2.3.3.1 Construção da rede neural com a base tratada - Sprint 3	84
10.2.3.3.2 Construção da rede neural com Word2Vec + CBoW - Sprint 3	85

10.2.3.3.3 Construção da rede neural com Word2Vec + Embedding Layer - Sprint 3	87
9.2.3.3.4 Construção da rede neural com a base tratada - Sprint 4	88
10.2.3.3.5 Construção da rede neural com Word2Vec + CBoW - Sprint 4	89
10.2.3.3.6 Construção da rede neural com Word2Vec + Embedding Layer - Sprint 4	91
10.2.3.4 Exportação com a biblioteca pickle	93
10.2.4 Conclusão	93
10.3 Random Forest - Word2vec	93
10.3.1 Introdução	93
10.3.2 Método	93
10.3.2.1 Cross validation	93
10.3.3 Resultados	94
10.3.4 Conclusão	98
10.4 XGboost - Bag of Words e Word2Vec	98
10.4.1 Introdução	98
10.4.2 Método	99
10.4.3 Resultados	99
10.4.3.1 Cross Validation	101
10.4.3.1 Grid Search	102
10.4.4 Conclusão	104
10.5 Naive Bayes + TF-IDF	104
10.5.1 Introdução	104
10.5.2 Método	104
10.5.3 Resultados	105
10.5.4 Conclusão	108
10.6 Random Forest + TF-IDF	108
10.6.1 Introdução	108
10.6.2 Método	108
10.6.3 Resultados	109
10.6.4 Conclusão	113
10.7 Modelo com novas features Random Forest + TF-IDF	113
10.7.1 Introdução	113
10.7.2 Método	113
10.7.3 Resultados	114
10.7.4 Conclusão	117
<b>11. Comparação entre os modelos</b>	<b>117</b>
11.1 Introdução	117
11.2 Método	118
11.3 Tabela de comparação	118
11.5 Conclusão	128
<b>12. Arquitetura macro</b>	<b>128</b>
<b>13. Diagrama</b>	<b>129</b>
<b>Referências:</b>	<b>131</b>
<b>Anexos:</b>	<b>131</b>

## Índice de figuras

Figura 01: Comparativo do lucro dos principais bancos

Figura 02: Infográfico das 5 forças de Porter

Figura 03: Matriz de avaliação

Figura 04: Tabela de desenvolvimento

Figura 05: Tabela de custeamento do projeto

Figura 06: Tabela de receita

Figura 07: Tabela de retorno

Figura 08: Canvas Proposta de Valor Completo

Figura 09: Proposta de Valor

Figura 10: Perfil do Cliente

Figura 11: Matriz de risco

Figura 12: Matriz de risco - Ameaças

Figura 13: Matriz de risco - Oportunidades

Figura 14: Persona 1

Figura 15: Persona 2

Figura 16: Persona 3

Figura 17: Jornada de usuário 1

Figura 18: Jornada de usuário 2

Figura 19: Jornada de usuário 3

Figura 20: Gráfico “Autores que mais comentam”

Figura 21: Gráfico “Tipos de interação”

Figura 22: Gráfico “Tipos de sentimento” - Barras

- Figura 23: Gráfico “Tipos de sentimento com Interação” - Barras
- Figura 24: Demonstração do Bag of Words
- Figura 25: *Output* do código
- Figura 26: Demonstração do modelo pronto
- Figura 27: Nuvem de palavras
- Figura 28: Word2Vec com modelo pré-treinado
- Figura 29: Word2Vec treinado com o corpus
- Figura 30: Matriz de confusão Naive Bayes - Sprint 3
- Figura 31: Matriz de confusão Naive Bayes - Sprint 4
- Figura 32: Matriz de confusão Naive Bayes Cross Validation - Sprint 3
- Figura 33: Matriz de confusão Naive Bayes Cross Validation - Sprint 4
- Figura 34: Matriz de Confusão - base tratada Sprint 3
- Figura 35: Matriz de Confusão - Word2Vec + CBoW Sprint 3
- Figura 36: Matriz de Confusão - Word2Vec + Embedding Layer Sprint 3
- Figura 37: Matriz de Confusão - Base tratada Sprint 4
- Figura 38: Matriz de Confusão - Word2Vec + CBoW Sprint 4
- Figura 39: Matriz de Confusão - Word2Vec + Embedding Layer Sprint 4
- Figura 40: Matriz de Confusão Random Forest Sprint 3
- Figura 41: Matriz de Confusão Random Forest Sprint 3
- Figura 42: Matriz de Confusão XGBoost
- Figura 43: Matriz de confusão Naive Bayes - Sprint 4
- Figura 44: Matriz de confusão Random Forest - Sprint 4
- Figura 45: Matriz de confusão Random Forest - Sprint 5
- Figura 46: Arquitetura Macro da Solução
- Figura 47: Diagrama de implantação UML

## Índice de tabelas

Tabela 01: Tabela de Avaliação

Tabela 02: User Story 01

Tabela 03: User Story 02

Tabela 04: User Story 03

Tabela 05: User Story 04

Tabela 06: User Story 05

Tabela 07: Características de 3 modelos - Representação Vetorial

Tabela 08: Características de 3 modelos - Tamanho da Representação

Tabela 09: Características de 3 modelos - Semântica e Contexto

Tabela 10: Características de 3 modelos - Flexibilidade

Tabela 11: Características de 3 modelos - Uso de contexto local

## 1. Introdução

O BTG Pactual é uma instituição financeira brasileira que oferece uma ampla gama de produtos e serviços financeiros, desde investimentos em renda fixa e variável até soluções de previdência privada e gestão de patrimônio. Fundada em 1983, a empresa tem uma atuação global, com presença em vários países da América Latina, Europa, Ásia e Estados Unidos, sendo conhecida pela sua expertise no mercado financeiro, pela qualidade do seu atendimento e pela inovação constante em seus serviços.

Esse projeto se especifica na área de marketing do banco, que é responsável por desenvolver estratégias para promover a marca e os produtos da instituição financeira, atraindo e fidelizando clientes e investidores. Por isso, ela tem um papel fundamental no relacionamento com os clientes, pois é responsável por comunicar de forma clara e eficiente as vantagens e benefícios dos produtos e serviços financeiros oferecidos pelo BTG Pactual, bem como identificar as necessidades e demandas dos clientes para desenvolver soluções personalizadas e eficazes.

## 2. Problema

O *Instagram* é uma das principais redes sociais utilizadas pelos clientes do BTG Pactual para se comunicarem com a instituição financeira, seja para tirar dúvidas, obter informações ou expressar suas opiniões sobre os produtos e serviços oferecidos. Porém, uma das principais dificuldades enfrentadas pela área de *Marketing* do banco é entender as necessidades e demandas dos clientes de maneira fácil e rápida no *Instagram*. Isso se deve ao fato de que muitas vezes aquele comentário do cliente está em um post que não tem relação com o tema, ou ainda esse cliente ainda não entrou em contato com o banco e já mandou *direct* no *Instagram*.

Esse tópico se mostra de extrema importância, tendo em vista que marketing representa aproximadamente 13,6% do orçamento total de uma empresa em 2023, de acordo com a Pesquisa Anual de CMO da Deloitte. Demonstrando que há uma enorme

necessidade de investimentos e priorização dos canais de marketing, para que se mostrem de qualidade, tanto em questão de divulgação do banco, quanto também de recepção e cuidado com o cliente.

Assim, devido essa utilização em massa das redes sociais, a riqueza e volumetria dos dados gerados acaba resultando em altos investimentos em estratégias de marketing digital pelo banco, então, foi levantado questões de como a análise de dados de mídia social conseguiria fornecer informações que ajudariam a entender o que está funcionando e o que não está funcionando. Dessa forma, a partir da parceria do Banco BTG Pactual e do Instituto de Tecnologia e Liderança, o Inteli, foi apresentado a necessidade de uma solução que solucionasse esse tipo de problema/questões, de forma que ajudasse o time de Marketing a tomar as decisões de negócios corretas e refinar as estratégias novas ou já existentes, à medida que avança a criação de novas campanhas.

### 3. Objetivos

Tendo em vista o problema, o time de Automação juntamente com o time de Marketing do banco BTG Pactual, propôs o desenvolvimento de uma Inteligência Artificial (IA) utilizando processamento de linguagem natural (PLN), para fazer o monitoramento de campanhas de marketing.

A solução de PLN ajuda a equipe a entender as necessidades e demandas dos clientes de maneira fácil e rápida no Instagram. O objetivo principal da solução é rastrear os dados em tempo real, analisar e interpretar as mensagens e comentários enviados pelos clientes na rede social, a fim de identificar as necessidades e demandas de forma precisa e eficiente. Essa solução é implementada através de algoritmos de aprendizado de máquina, que são treinados para identificar palavras-chave e sentimentos nas mensagens e comentários enviados pelos clientes, de acordo com a campanha presente no post.

## 4. Análise de Negócios

### 4.1. Contexto da indústria

Os bancos desempenham um papel fundamental na intermediação financeira, captando recursos de depositantes / investidores e concedendo empréstimos a tomadores de empréstimos para financiar seus negócios e projetos. E com isso, eles oferecem uma ampla gama de produtos e serviços financeiros, como contas correntes, poupança, empréstimos, cartões de crédito, investimentos, seguros, entre outros.

A indústria bancária está em constante evolução, com mudanças significativas ocorrendo ao longo das últimas décadas. A redução de tarifas, a digitalização e a democratização são fatores importantes que têm afetado o setor bancário em todo o mundo, tópicos que serão explicados posteriormente.

#### 4.1.1 Principais players

A imagem a seguir demonstra quais são os principais players da indústria bancária e o lucro desde 2015. O lucro está sendo representado em bilhões de reais.

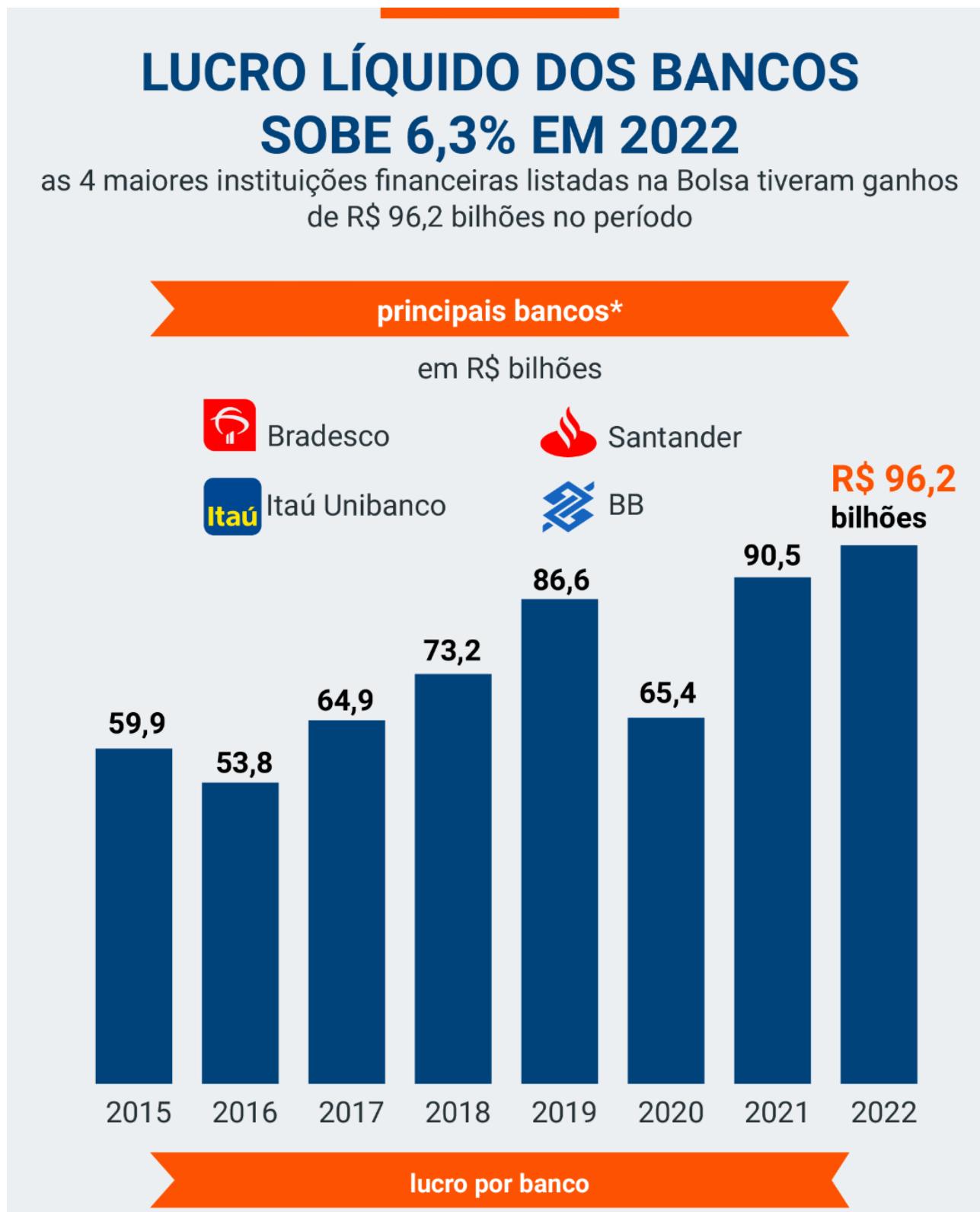


Figura 01: Comparativo do lucro dos principais bancos

Abaixo será descrito cada *player* que foi citado no gráfico como “principais bancos” e quais são os pontos fortes de cada um.

## **1. Bradesco**

O Bradesco é constituído em forma de sociedade anônima, fundada em 1934 no interior de São Paulo. Em 2021, entrou no *ranking* da Forbes entre os melhores empregadores do mundo, além de já ter recebido alguns prêmios por conta da sua cultura, como o Prêmio Valor Cultura Empresarial em 2020.

### **Pontos fortes:**

- Presença no mercado brasileiro: rede extensa de agências, mais de 4700, e clientes pelo Brasil, permitindo a possibilidade de atender a maior parte da população.
- Forte cultura: o banco é conhecido pela sua cultura interna muito forte, com princípios bem definidos, como a ética, integridade e compromisso com o cliente.
- Adaptabilidade: o banco conseguiu se adaptar muito bem com as últimas mudanças do mercado bancário, como a digitalização, onde foi criado a BIA, inteligência artificial do banco.

## **2. Itaú Unibanco**

O Itaú Unibanco foi criado a partir da fusão entre o Banco Itaú e o Unibanco em 2008, que na época eles estavam no topo das maiores instituições financeiras do Brasil, e hoje está presente no Brasil e em outros países da América Latina.

### **Pontos fortes:**

- Presença forte no mercado: forte presença no Brasil, sendo um dos mais importantes e influentes, e também uma presença tão forte quanto na América Latina;
- Governança corporativa: a empresa tem uma pontuação alta em *rankings* de sustentabilidade e governança.

## **3. Santander Brasil**

O Santander é uma empresa espanhola que possui algumas subsidiárias em outros países, como o Brasil que entrou em atividade em 1982. Além disso, o banco foi considerado o terceiro maior na indústria em 2022.

### **Pontos fortes:**

- Presença global: como já dito, o banco possui diversas subsidiárias em diferentes países;

- Foco na inovação: o banco tem investido bastante em soluções tecnológicas, como o Santander Way, um aplicativo de celular que permite transações financeiras.

#### **4. Banco do Brasil**

O Banco do Brasil é uma empresa de economia mista, ou seja, o governo tem uma parte de suas ações, que nesse caso é 50%. A instituição foi fundada em 1808, sendo o primeiro banco no Brasil.

##### **Pontos Fortes:**

- Presença nacional: ampla rede de agências no Brasil inteiro, tornando-se um banco democrático;
- Foco no agronegócio: como o país tem um forte mercado rural, o banco utiliza dessa característica para oferecer linhas de crédito e serviços para produtores rurais.

#### **4.1.2 Modelo de negócio**

O modelo de negócio do setor bancário pode ser resumido em duas atividades principais: captar recursos financeiros de clientes / investidores, e emprestar esses recursos a outros clientes que precisam de financiamento. O principal meio de captação de recursos é por meio de depósitos e emissão de títulos. Em seguida, eles utilizam esses recursos para emprestar a outras pessoas, empresas ou governos, com o objetivo de obter lucro com juros e outras taxas cobradas sobre esses empréstimos. Para reter e atrair clientes, os bancos oferecem outros serviços, como contas correntes, cartões de crédito, investimentos, seguros, entre outros.

Outro aspecto importante do modelo de negócio do setor bancário é a gestão de riscos. Os bancos devem avaliar cuidadosamente o risco de cada empréstimo, a fim de minimizar as perdas decorrentes de uma falta de cumprimento ou outros eventos adversos. Para isso, eles utilizam uma série de ferramentas e técnicas de análise de risco.

#### **4.1.3 Tendências**

A seguir serão apresentados três tendências principais que foram citadas no estudo global “Digital Banking Maturity 2022” feito pela Deloitte.

##### **1. Digitalização**

Com a evolução da tecnologia e a popularização da internet, os bancos têm buscado se adaptar a esse novo cenário, oferecendo serviços bancários cada vez mais digitais e automatizados. O objetivo principal é oferecer mais conveniência e facilidade para os clientes, que podem realizar transações bancárias a qualquer hora e em qualquer lugar, utilizando dispositivos móveis. Além disso, os bancos também

buscam reduzir seus custos operacionais, com a automatização de processos e a redução da necessidade de agências físicas.

## 2. Liberalização de investimentos

A tendência de liberalização de investimentos dos bancos tem ganhado força nos últimos anos, com a flexibilização das regulações em diversos países ao redor do mundo, permitindo uma gama mais ampla de produtos e serviços de investimento, que antes eram restritos a instituições especializadas, como corretoras e gestoras de fundos. Com isso, os bancos podem oferecer aos seus clientes opções de investimento em diversas classes de ativos, fazendo com o que os clientes tenham mais opções para diversificar suas carteiras de investimentos, e tendo a opção de contar com o suporte e a experiência dos bancos na hora de tomar decisões de investimento.

## 3. Democratização

O objetivo principal da democratização é tornar o acesso aos serviços financeiros mais acessível a um público mais amplo e diverso, que inclui pessoas de baixa renda, pequenas empresas e empreendedores, que muitas vezes não têm acesso a serviços bancários tradicionais. Essa tendência de democratização está diretamente relacionada com a primeira tendência explicada, a digitalização.

### 4.1.4 Análise 5 forças de Porter:

As 5 forças de Porter é uma ferramenta de análise de mercado criada por Michael Porter, para avaliar a intensidade da concorrência em um setor e a força das forças competitivas que determinam a atratividade de um mercado. Essas 5 forças são:

- Rivais Competitivos: a concorrência direta e indireta de outras empresas no setor.
- Poder de negociação de compradores: o poder de barganha dos compradores em relação ao preço, qualidade e outros requisitos.
- Poder de negociação de fornecedores: o poder de barganha dos fornecedores em relação a preços e condições de fornecimento.
- Ameaça de novos entrantes: a facilidade de entrada de novas empresas no mercado, incluindo barreiras à entrada, economias de escala e recursos financeiros.
- Ameaça de produtos substitutos: a presença de produtos ou serviços substitutos que possam afetar a demanda pelos produtos atuais do setor.

Essas 5 forças são utilizadas para avaliar o potencial de lucro de um setor e para identificar as principais fontes de pressão competitiva, ajudando as empresas a tomar decisões estratégicas e a se posicionar de forma mais forte no mercado. Abaixo segue o infográfico das 5 forças de Porter do BTG Pactual.

# Infográficos das 5 forças de Porter - BTG Pactual



Figura 02: Infográfico das 5 forças de Porter

## 1. Poder de barganha com clientes

O poder de barganha dos clientes do setor bancário é relativamente baixo, uma vez que muitas vezes os clientes são dependentes dos serviços oferecidos pelos bancos e têm poucas alternativas para escolher, já que alguns bancos são dominantes em certas regiões. Porém essa métrica pode mudar nos próximos anos, já que os bancos estão no processo de digitalização, permitindo com que os clientes, independente de onde mora, possam acessar bancos diferentes.

## 2. Poder de barganha com fornecedores

O poder de barganha dos fornecedores do setor bancário pode ser visto de duas formas: 1. Setor no geral, que tem um poder de barganha relativamente baixo, já que os bancos geralmente lidam com um grande número de fornecedores e têm a capacidade de negociar com eles em grande escala. 2. A área específica de fornecimento de tecnologias, onde, nesse caso, os fornecedores podem ter maior poder de negociação e impor condições de venda menos favoráveis.

## 3. Ameaça de novos concorrentes

Apesar de ser um mercado com um regulamento muito complexo e ter empresas que já tem uma confiança dos clientes estabelecidas, nos últimos anos foi observado que empresas digitais conseguem entrar no mercado facilmente. Esses novos entrantes podem representar uma ameaça aos bancos estabelecidos, especialmente quando conseguem conquistar um grande número de clientes e estabelecer uma marca forte e confiável.

## 4. Ameaça de produtos substitutos

A ameaça de produtos substitutos no setor bancário é moderada, pois existem algumas alternativas disponíveis para os serviços financeiros oferecidos pelos bancos, principalmente com a tendência de digitalização do setor e a entrada de novos

concorrentes. Ou seja, essas novas empresas digitais oferecem soluções financeiras alternativas, como aplicativos de pagamento móvel, que podem competir com os serviços bancários tradicionais.

Porém, os bancos ainda mantêm uma vantagem competitiva, pois possuem uma rede de agências bancárias, uma base de clientes estabelecida e um histórico de confiabilidade e segurança. Além disso, os bancos estão investindo em tecnologias inovadoras e melhorando a experiência do cliente para se igualar com novas soluções de empresas recentes.

### **5. Rivalidade entre concorrentes:**

A rivalidade entre concorrentes no setor bancário é alta, devido ao grande número de empresas no mercado e à competição intensa pelos clientes. Os bancos usam estratégias de marketing para atrair / manter clientes, oferecendo diversas promoções. Além disso, eles competem em termos de inovação e tecnologia, com o objetivo de fornecer aos clientes serviços mais rápidos, seguros e convenientes. A rivalidade entre concorrentes é ainda mais intensa em regiões onde há um grande número de bancos, o que aumenta a pressão sobre as instituições financeiras para conquistar e manter seus clientes.

## **4.2 Matriz de Avaliação de valor Oceano Azul**

A matriz de Oceano Azul é uma ferramenta que permite, através do comparativo entre atributos comuns, a identificação de possíveis oportunidades que não estão no radar da concorrência. Também são analisados esses atributos a fim de que seja agregado mais valor para os usuários. Utilizando essa ferramenta, é possível moldar o modelo de quatro vieses, os quais visam identificar diferentes oportunidades em diferenciação dos concorrentes, isso acontece a partir das seguintes ações:

- Reduzir;
- Eliminar;
- Aumentar;
- Criar;

Com base na solução proposta da equipe BT G3, realizou-se a matriz de avaliação de valor “oceano azul”, com base em 3 concorrentes, Google Cloud NLP, Amazon Comprehend NLP e Open AI NLP (GPT-3) e, além disso, foi baseada em 8 atributos principais.

Atributos	BT-G3 NLP	Google Cloud NLP (CNL)	Amazon Comprehend NLP	Open AÍ NLP (GPT-3)
Melhor preço	10	5	8	1
Qualidade de treinamento	1	5	5	10
Tecnologia	5	10	10	10
Análise de	5	5	10	5

palavras-Chave				
Análise de sentimentos	5	10	5	10
Tradução PT-BR	8	10	0	10
Customização	10	8	5	5
Praticidade	10	1	8	5
Controle	10	0	0	0

Tabela 01: Tabela de Avaliação

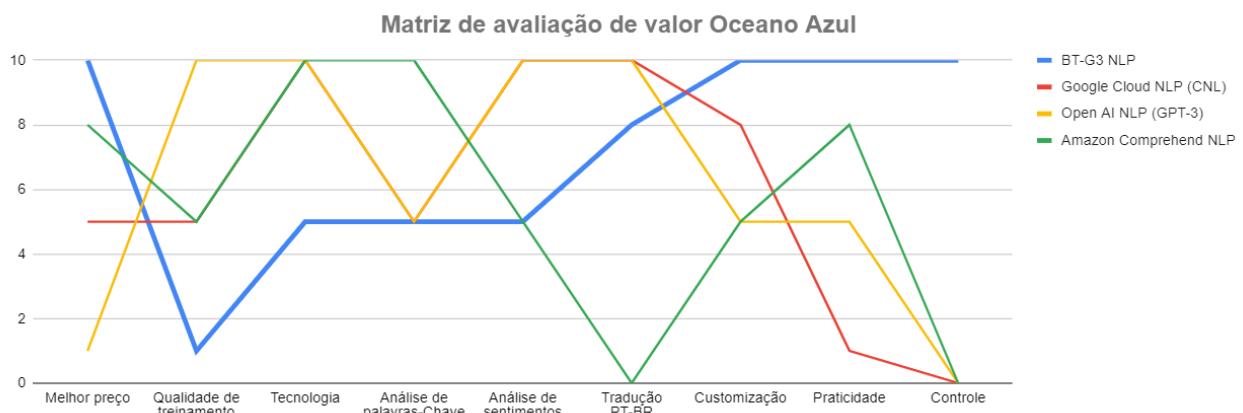


Figura 03: Matriz de avaliação

#### 4.2.1 Criar

A partir do desenvolvimento do modelo BT-G3, será possível **criar** um controle completo da ferramenta devido ao fato de ser uma tecnologia interna, permitindo melhor customização e praticidade ao mesmo tempo. Assim, a partir desse controle, o time de automação pode analisar dados em tempo real das campanhas, permitindo então, ajustes imediatos em parâmetros, tratamento da base e monitoramento dos processos até o *output* final. Além disso, a partir da customização da IA, é possível segmentar a análise para certas campanhas mais alarmantes, satisfazendo o time de Marketing. Por último, vale ressaltar que **criou** uma independência parcial dos ecossistemas fechados de *Cloud*, como é o da AWS e o da Google.

#### 4.2.2 Aumentar

Deve-se destacar que o modelo BT-G3 **aumentou** os atributos “Melhor preço”, pois independe de taxas de requisição de API por ser uma ferramenta interna. Além disso, é possível aumentar a “Customização”, já que há a possibilidade de uso dos modelos, parâmetros e base de treinamento conforme a necessidade, disponibilidade e propriedade do código-fonte. Também, por se tratar de um produto interno, o atributo “Praticidade” foi amplificado pela irrestritividade a sistemas *Cloud* de terceiros, que podem forçar um ecossistema fechado.

#### 4.2.3 Eliminar

Diante dos parâmetros listados, a equipe desenvolvedora do modelo BT-G3, optou por não incluir integrações com meios tecnológicos, como Power BI, para focar

no desenvolvimento da análise de sentimentos, entregando os aspectos que estavam no escopo de forma qualitativa.

#### 4.2.4 Reduzir

Também foi responsável por **reduzir** o modelo BT-G3 os atributos “Qualidade de treinamento” e “Tecnologia”, visto que os modelos desenvolvidos pelas demais empresas contam com investimentos massivos em treinamento. Em exemplo disso, o modelo GPT-3 obteve um investimento estimado de 4 milhões de dólares, desenvolvido pela OpenAI, que possui um aporte muito maior para o desenvolvimento da ferramenta comparativamente ao grupo BT-G3.

### 4.3 Análise Financeira do Projeto

A precificação de um projeto de *Machine Learning* envolve vários fatores. Dentre esses fatores, estão:

- Escopo do projeto;
- Quantidade de dados a serem processados;
- Tempo necessário para a implementação;
- Valor agregado que a solução trará ao cliente.

Abaixo estão alguns aspectos que envolvem o projeto a ser desenvolvido:

Projeto	
Duração do projeto	10 semanas (17/04/2023 - 23/06/2023)
Número de integrantes do time	6 integrantes no time
Horas totais	232h 30 min
Remuneração por hora	R\$ 13,93

Figura 04: Tabela de desenvolvimento

O projeto envolve a análise de sentimento e identificação de palavras-chave em textos de redes sociais, utilizando técnicas de PLN e Machine Learning. O seu desenvolvimento também inclui a coleta de dados, o pré-processamento, análise dos dados, treinamento do modelo de Machine Learning, validação e ajuste do modelo e implementação da solução em um ambiente de produção.

Como mostra a Figura 05, pela complexidade, necessidade de softwares, número de funcionários, manutenção do sistema e grande carga horária de trabalho, o valor estimado é em torno de R\$150.000,00 - R\$200.000,00.

Custos	
Salário de cada técnico de automação júnior	R\$ 3.238,25
Gastos com desenvolvedores (6)	R\$ 19.429,50
Custo total do projeto	R\$ 200.000,00

Figura 05: Tabela de custeamento do projeto

Como mostra a Figura 06, o valor estimado por cliente é de R\$1.250,00 para o uso de uma solução de Data Science podendo ser justificado de diversas maneiras.

Primeiramente, vale ressaltar que a implementação de projetos de machine learning requer um alto grau de especialização e conhecimento técnico. A criação de modelos preditivos, envolve a utilização de algoritmos complexos e a análise de grandes volumes de dados. Portanto, o valor cobrado por uma solução de PLN reflete, a expertise necessária para executar esse tipo de trabalho.

Além disso, é preciso levar em conta o valor agregado que a solução oferece ao cliente. Projetos de machine learning podem trazer insights valiosos para as empresas, permitindo uma melhor tomada de decisão e aumentando a eficiência de processos. Dessa forma, o preço cobrado pode ser visto como um investimento que trará retorno ao cliente no longo prazo.

Em resumo, o valor de R\$1.250,00 por cliente para o uso de uma solução que utiliza a inteligência artificial, é uma forma de especificação justificável, que reflete tanto a expertise necessária para a implementação do projeto quanto o valor agregado que a solução oferece ao cliente.

Receita		
Meses	Nº de clientes por mês	Pagamento
1º mês	1 cliente	1.250,00
2º mês	2 clientes	2.500,00
3º mês	4 clientes	5.000,00
4º mês	6 clientes	7.500,00
5º mês	8 clientes	10.000,00
6º mês	10 clientes	12.500,00
7º mês	12 clientes	15.000,00
8º mês	14 clientes	17.500,00
9º mês	16 clientes	20.000,00
10º mês	18 clientes	22.500,00
11º mês	20 clientes	25.000,00
12º mês	22 clientes	27.500,00
		Total: R\$166.250,00
13º mês	20 clientes	25.000,00
14º mês	20 clientes	25.000,00
15º mês	22 clientes	27.500,00
16º mês	22 clientes	27.500,00
		Total: R\$105.000,00
		Receita Total: R\$271.250,00

Figura 06: Tabela de receita

Dado o valor estimado de R\$1.250,00 por cliente e que a cada mês há um aumento de dois clientes, em 1 ano (12 meses), a receita para uma empresa que desenvolve esse tipo de solução seria em média R\$166.250,00.

Considerando que a empresa mantenha o número de clientes como 20 no 13º e 14º mês, e tenha 22 clientes no 15º mês e 16º mês, o total seria uma receita de R\$105.000,00 em 4 meses.

Portanto, em um período de 16 meses, essa empresa já conseguiria ter o retorno do que foi investido inicialmente, tendo uma receita de R\$271.250,00.

Retorno	
Receita que o projeto traz em um período de um ano	R\$ 173.750,00
Receita que recupera o valor investido	16º mês - R\$ 271.250,00

Figura 07: Tabela de retorno

Link disponibilizado para análise financeira do projeto:

<https://docs.google.com/spreadsheets/d/1anx53miwDWVmi6IDCjUSv-PWB7V6-em45LE6V90IdHE/edit?usp=sharing>

## 4.4 Value Proposition Canvas

Canvas Proposta de Valor é uma ferramenta de gestão estratégica que auxilia na criação e desenvolvimento de novos produtos, serviços e negócios. Essa ferramenta é dividida em dois blocos: o perfil do cliente e a proposta de valor.

No bloco de proposta de valor, é necessário definir qual é o produto desenvolvido, quais são os principais criadores de ganho que o cliente tem ao adquirir o produto e quais são os aliviadores de dores que o cliente terá. Já no perfil do cliente, deve-se identificar quais são as tarefas que ele irá realizar, quais são os principais ganhos e dores do atual processo. Abaixo segue o Canvas Proposta de Valor Completo.

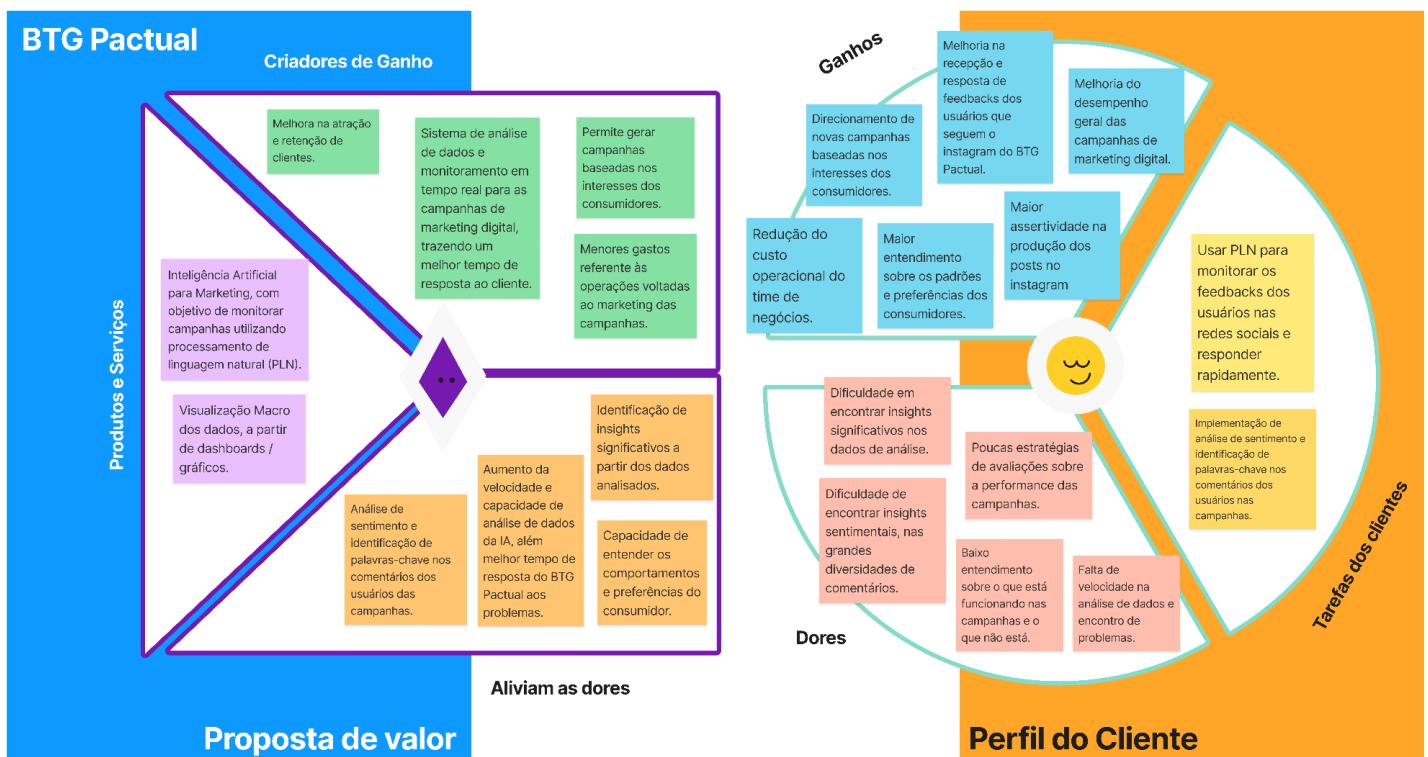


Figura 08: Canvas Proposta de Valor Completo

#### **4.4.1 Proposta de valor:**

A solução que está sendo descrita neste documento é a utilização da inteligência artificial para a área de Marketing do BTG Pactual, que tem como objetivo monitorar as campanhas utilizando PLN. Além disso, também será desenvolvida uma visualização macro dos modelos desenvolvidos a partir de dashboards e gráficos. Os principais criadores de ganhos são:

- **Melhora na atração e retenção de clientes:** hoje em dia, a área de marketing não consegue entender com clareza as necessidades dos clientes, então eles não conseguem agir antes de perder ou receber menos clientes;
- **Sistema de análise de dados e monitoramento em tempo real para as campanhas de marketing digital, trazendo um melhor tempo de resposta ao cliente:** hoje em dia, a área de marketing não consegue entender com rapidez as necessidades dos clientes, ou seja, caso tenha algo que não esteja agradando, na maioria das vezes, eles não conseguem agir rápido para resolver;
- **Permite gerar campanhas baseadas nos interesses dos consumidores:** ter a possibilidade de criar um relacionamento mais personalizado com o cliente, ou seja, caso os usuários estejam pedindo algo específico, a área de produtos pode desenvolvê-lo;
- **Menores gastos referente às operações voltadas ao marketing das campanhas:** já que o tratamento pode ser mais personalizado e claro, a empresa não gastará recursos para campanhas que provavelmente não irão fazer sucesso.

Os principais aliviadores de dores são:

- **Análise de sentimento e identificação de palavras-chaves nos comentários dos usuários das campanhas:** como é difícil identificar o que os usuários estão falando, ao não ser que tenha um comentário de alguém influente ou com diversos *likes*, e qual é o sentimento gerado, a solução propõe identificar as palavras-chaves para ajudar a análise;
- **Aumento da velocidade e capacidade de análise de dados da IA, além melhor tempo de resposta do BTG Pactual aos problemas:** como toda a análise é feita por uma pessoa, o colaborador responsável tem que ler cada comentário e *direct* para responder de acordo com a necessidade do usuário, então o processo é demorado;
- **Identificação de *insights* significativos a partir dos dados analisados:** uma das soluções desenvolvidas é uma interface que será composta por elementos gráficos, tornando mais fácil identificar *insights* sobre as campanhas e seus comentários;
- **Capacidade de entender os comportamentos e preferências do consumidor:** a grande dificuldade hoje do banco é entender com clareza quais são as necessidades do consumidor, e a solução promove uma visualização clara das necessidades, aliviando essa dor.

# BTG Pactual

## Criadores de Ganho

### Produtos e Serviços

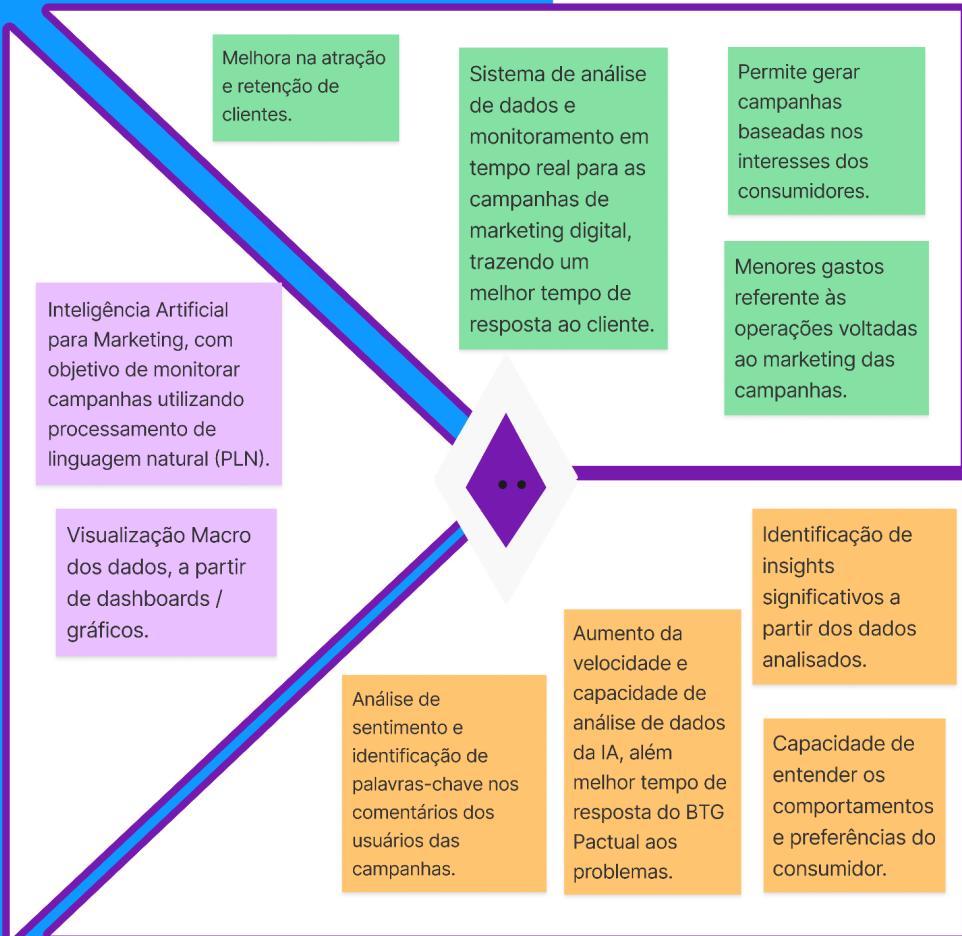


Figura 09: Proposta de Valor

### 4.4.2 Perfil do Cliente:

O cliente deverá realizar algumas tarefas como o uso da solução para monitorar quais necessidades os clientes estão dizendo nas redes sociais e respondê-las. Além disso, implementar a análise de sentimento e identificação das palavras-chaves. Os principais ganhos são:

- **Redução no custo operacional do time de negócios:** isso se dá com a automação de um processo hoje feito por uma pessoa, então esse colaborador poderá ser alocado para outra área;
- **Direcionamento de novas campanhas baseadas nos interesses dos consumidores:** já que a visualização de quais campanhas estão dando

certo será mais clara, a área de *marketing* e de produtos poderão direcionar recursos para aquelas que os usuários gostam mais;

- **Melhoria do desempenho geral das campanhas de *marketing digital*:** como o banco poderá direcionar melhor suas campanhas, isso faz com que o desempenho de campanhas que o público já gosta irá se manter elevado;
- **Maior entendimento sobre os padrões e preferências dos consumidores e maior assertividade na produção dos posts no Instagram:** com a visualização clara de quais são os tópicos que os clientes mais falam sobre, os colaboradores poderão entender padrões de comportamento mais intuitivamente.

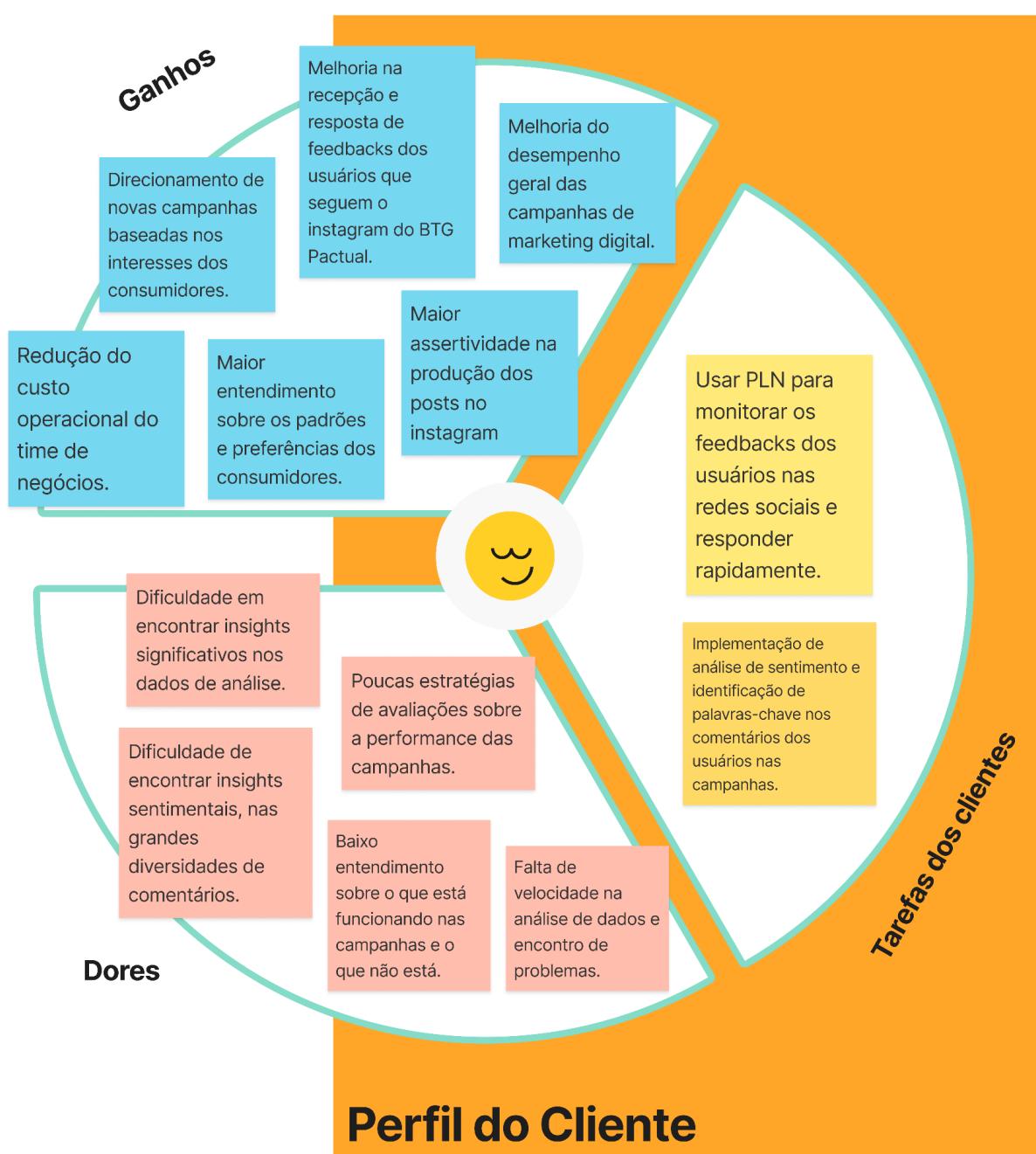


Figura 10: Perfil do Cliente

## 4.5 Matriz de Risco

A matriz de risco é dividida em duas partes: Ameaças e Oportunidades. A primeira, o grupo que está desenvolvendo aquele projeto deve colocar tópicos que acreditam que são ameaças pensando na entrega do projeto. Já a segunda é o oposto, ou seja, o grupo deve colocar quais tópicos são vistos como positivos dentro do projeto. As duas partes são classificadas de acordo com a sua probabilidade de acontecer (10% até 90%) e qual é o impacto que irá trazer (muito baixo - muito alto).

Essa ferramenta permite que as organizações visualizem e priorizem os riscos de acordo com sua probabilidade e impacto, o que ajuda a tomar decisões mais informadas sobre como gerenciá-los. Por exemplo, riscos com alta probabilidade e alto impacto devem receber mais atenção e esforço de gerenciamento do que aqueles com baixa probabilidade e baixo impacto.

Probabilidade	Ameaças					Oportunidades				
	Muito Baixo	Baixo	Moderado	Alto	Muito Alto	Muito Baixo	Baixo	Moderado	Alto	Muito Alto
	Impacto					Impacto				
90%										
70%										
50%										
30%										
10%										
	Muito Baixo	Baixo	Moderado	Alto	Muito Alto	Muito Baixo	Baixo	Moderado	Alto	Muito Baixo

Figura 11: Matriz de risco

O grupo definiu como **ameaças**:

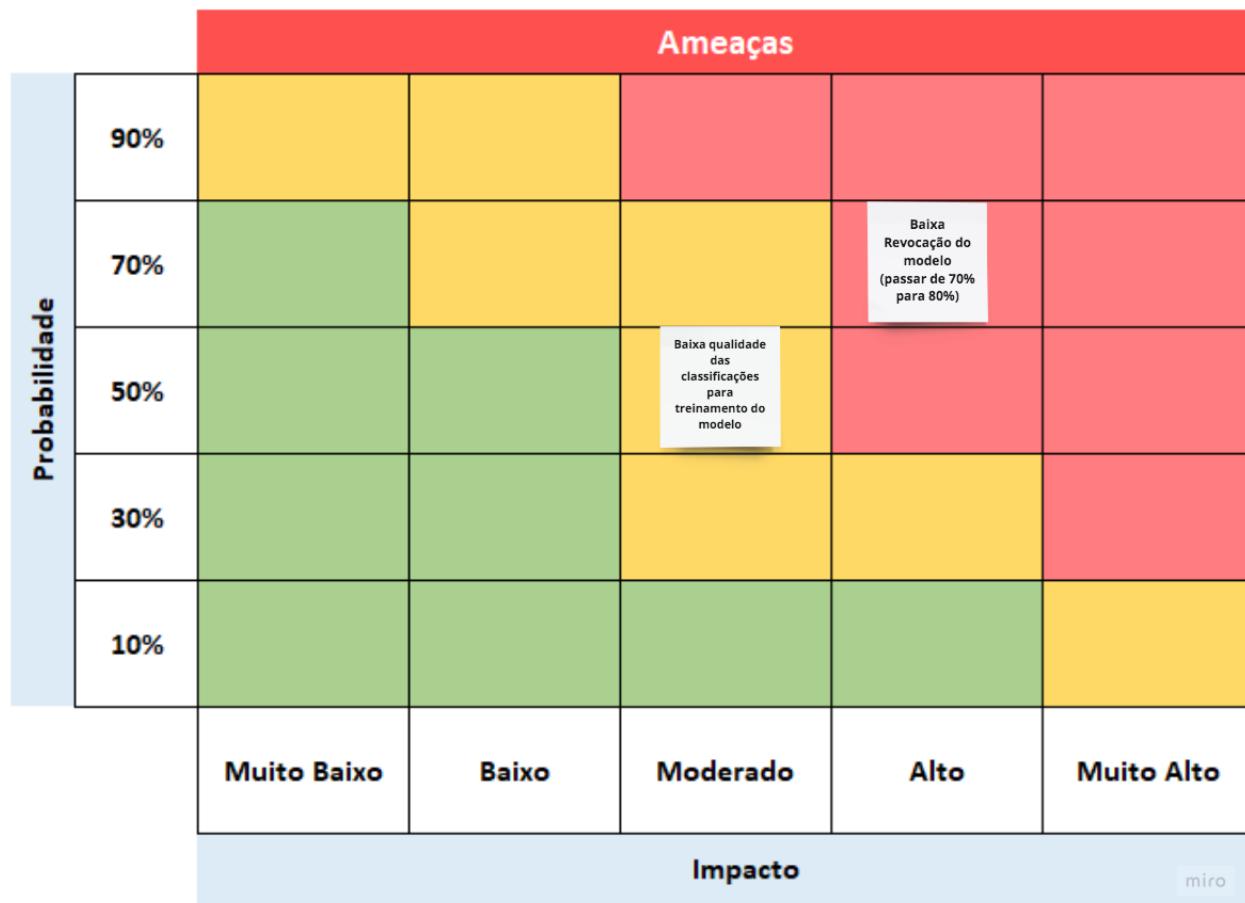


Figura 12: Matriz de risco - Ameaças

- Baixa revocação do modelo - Probabilidade: **70%**; Impacto: Alto - Apesar de ter uma chance da revocação ficar baixa, entendemos que o valor entre 70% - 80% não é pouco satisfatório.
- Baixa qualidade das classificações para treinamento do modelo - Probabilidade: **50%**; Impacto: Moderado - É a mesma linha de raciocínio do anterior.

### PLANO DE AÇÃO:

- Baixa revocação do modelo e Baixa qualidade das classificações para treinamento do modelo -> Garantir que o grupo faça o melhor trabalho de pré processamento para evitar isso. (Pedro)

O grupo definiu como **oportunidades**:

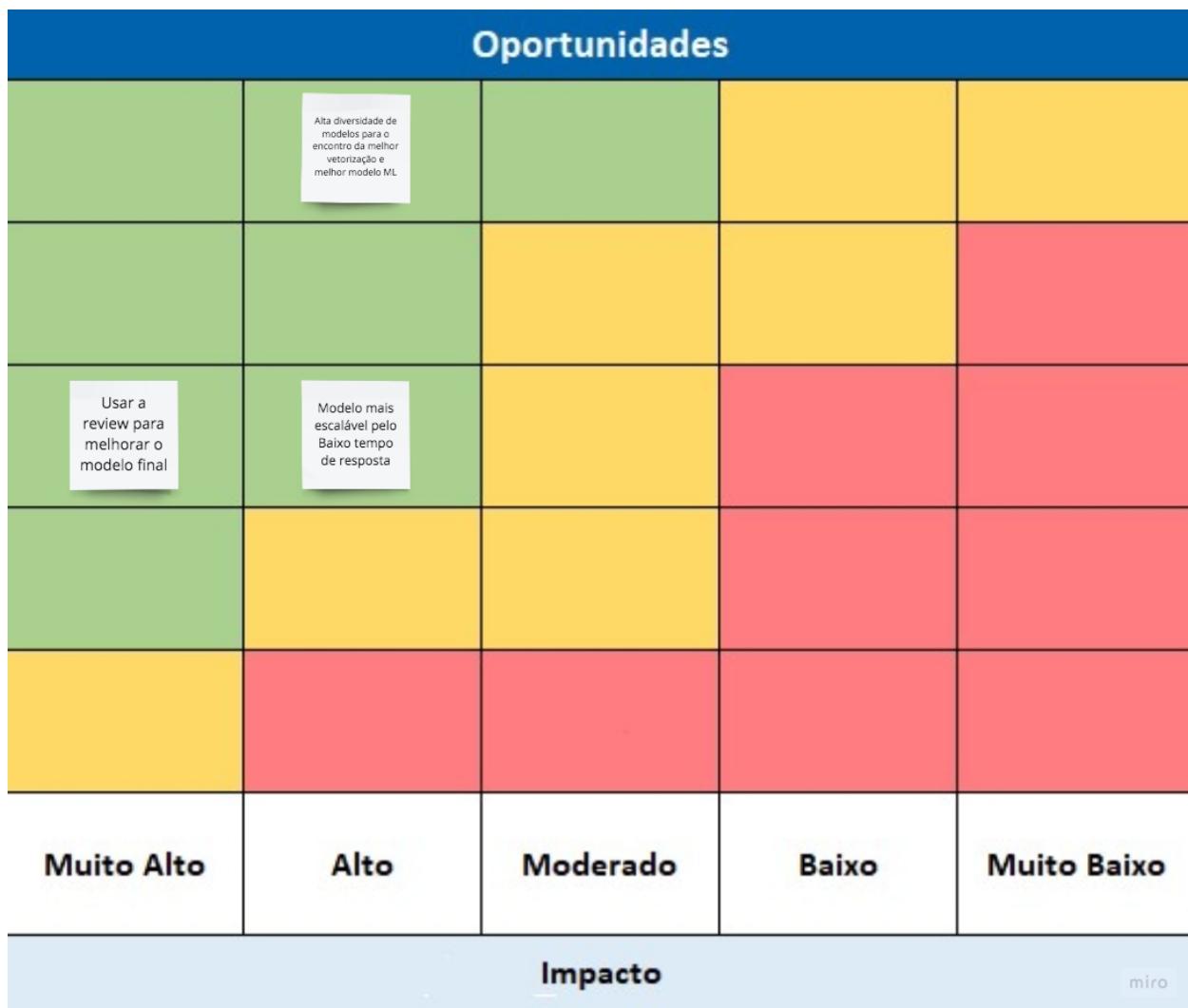


Figura 13: Matriz de risco - Oportunidades

- Usar a *review* para melhorar o modelo final - Probabilidade: **50%**; Impacto: Muito alto - Utilizamos a última *review* para tirar *insights* dos modelos e melhorar o desempenho;
- Modelo mais escalável pelo baixo tempo de resposta - Probabilidade: **50%**; Impacto: Alto - como o nosso modelo não demora para rodar, acreditamos que o projeto pode ser mais escalável;
- Alta diversidade de modelos para o encontro da melhor vetorização e melhor modelo ML - Probabilidade: **90%**; Impacto: Alto - Durante as 10 semanas foi realizado o teste de mais de 15 modelos para que o grupo entenda qual é o melhor;

#### **PLANO DE AÇÃO:**

- Usar a *review* para melhorar o modelo final -> anotar *feedbacks* do parceiro e dos professores; (Malu)
- Modelo mais escalável pelo baixo tempo de resposta-> tentar deixar o modelo rápido para ser rodado; (Vitor)
- Alta diversidade de modelos para o encontro da melhor vetorização e melhor modelo ML -> Garantir a diversidade de modelos; (Pedro)

## 5. Análise de Experiência do Usuário

### 5.1 Personas

Personas é uma técnica comum de pesquisa de mercado e design de experiência do usuário (UX), que tem como função o entendimento das necessidades, desejos, objetivos e comportamentos de um determinado grupo de pessoas. Com esse conhecimento, os desenvolvedores conseguem criar soluções mais adequadas, já que é possível entender melhor as dores, comportamentos e motivações dos usuários.

#### 5.1.1 Marcos Santos - Persona 1

A primeira Persona apresentada é o Marcos Santos, analista de Marketing do BTG Pactual. Conversando com os stakeholders foi entendido que a pessoa que ocupa esse cargo sente que as estratégias sobre as performances das campanhas que o banco faz, não é claro para ele, sendo assim ele não consegue ter uma visualização intuitiva do processo. Em consequência desse fato, a equipe de marketing não consegue entender quais são as campanhas que já estão fazendo sucesso e as que não, para assim medir recursos para cada uma.

Por último, o tópico mais difícil, para o Marcos, é entender quais são os sentimentos que os usuários estão tentando demonstrar por meio dos comentários e mensagens, para assim criar um serviço personalizado. Para resolver as dores, a solução proposta possui uma visualização macro, para ver o andamento e sucesso das campanhas, e uma micro, para entender quais são as necessidades dos clientes naquela campanha.

# MARCOS SANTOS

A N A L I S T A   D E   M A R K E T I N G



NOME: Marcos Santos

IDADE: 32 anos

GÊNERO: Masculino

OCUPAÇÃO: Analista de Marketing

## BIOGRAFIA

Marcos é um profissional de marketing digital que trabalha no BTG Pactual. Ele cresceu no interior de São Paulo, estudou marketing na faculdade USP e se destacou em agências de marketing antes de ser contratado pelo banco. No BTG Pactual, ele lidera a equipe de marketing digital e é responsável por desenvolver estratégias para aumentar a presença online e atrair mais clientes.

## CARACTERÍSTICAS

- Comunicativo;
- Criativo;
- Habilidade analítica;
- Visão estratégica;

## DORES

- Atualmente tem poucas estratégias de avaliação com relação a performance das campanhas.
- Possui um baixo entendimento sobre o que está sendo efetivo nas campanhas e o que não está, das estratégias já existentes.
- Possui dificuldade em realizar uma análise aprofundada a respeito dos sentimentos e o engajamento dos seguidores do instagram do BTG Pactual.

## FORMAÇÃO

- SUPERIOR EM MARKETING PELA USP COM PÓS GRADUAÇÃO NA ÁREA DE TECNOLOGIA PELA FGV.

## NECESSIDADES

- Marcos necessita de uma melhoria do engajamento dos usuários que seguem o instagram do BTG Pactual.
- Necessita de uma solução que traga entendimento sobre os comportamentos e preferências do consumidor.
- Deseja ter uma visualização macro e micro em relação aos dados, a partir de dashboards/gráficos.
- Necessita de uma melhoria no desempenho geral das campanhas de marketing digital.

Figura 14: Persona 1

### 5.1.2 Gabriela Ferreira - Persona 2

A segunda Persona é a Gabriela Ferreira, que trabalha como analista de produtos do BTG Pactual, e com o processo atual da empresa, ela acaba sofrendo de algumas dores, como por exemplo a dificuldade de identificar quais produtos os clientes gostariam de obter de forma prática e visual. Além disso, é importante, para os analistas, que os produtos que já estão no mercado, recebam algum tipo de feedback, para assim ser possível de entender quais foram os motivos para o sucesso / falha. Para isso, a solução promove uma ferramenta onde é possível visualizar quais foram

os sentimentos das pessoas de acordo com a campanha e, consequentemente, o produto.



**NOME:** Gabriela Ferreira

**IDADE:** 30 anos

**GÊNERO:** Feminino

**OCUPAÇÃO:** Analista de produto

#### BIOGRAFIA

Gabriela é uma analista de produtos do banco BTG Pactual, que tem se destacado no mercado financeiro pela sua habilidade em desenvolver soluções criativas e eficientes para seus clientes. Ela baseia seus produtos e decisões de negócios em métricas que ela acompanha pela internet e newsletters.

#### CARACTERÍSTICAS

- Comunicativa;
- Criativa;
- Metódica;

#### DORES

- Não há um rastreamento em tempo real da recepção do público-alvo a novos produtos desenvolvidos.
- Ela não consegue identificar, de forma prática, novas oportunidades de produtos e serviços que estão sendo demandadas pelo público nas redes.
- Falta de um meio que dê suporte aos resultados de venda de um produto desenvolvido, permitindo o entendimento da razão de sucesso ou falha de um lançamento

#### FORMAÇÃO

- GRADUAÇÃO EM ADMINISTRAÇÃO DE EMPRESAS NA FGV-SP
- 6 ANOS DE EXPERIÊNCIA COMO PRODUCT MANAGER

#### NECESSIDADES

- Necessita de uma solução que notifique ela de demandas feitas nas redes sociais, por sentenças, para poder atuar o mais rápido possível, o que permitiria tempo para impulsionar o hype de um novo produto.
- Uma ferramenta que monitore os sentimentos das pessoas nas redes em relação a um produto que foi divulgado em uma campanha de marketing

### 5.1.3 Amir Abdullah - Persona 3

A terceira e última Persona é o Amir Abdullah, técnico de automação do BTG Pactual, que será responsável pela manutenção e atualização do banco de dados da solução. Para ele, as dores estão mais relacionadas com o banco de dados, ou seja, a falta de métricas, de organização e de tratamento de emojis são as principais dores.

Figura 15: Persona 2

Para resolver isso, o modelo tem uma etapa de pré - processamento que irá estruturar, organizar e limpar os dados.

# AMIR ABDULLAH

TÉCNICO DE AUTOMAÇÃO



**NOME:** Amir Abdullah

**IDADE:** 29 anos

**GÊNERO:** Masculino

**OCUPAÇÃO:** Técnico de Automação do BTG Pactual

**CARACTERÍSTICAS**

- Curioso;
- Paciente;
- Detalhista;
- Observador.

**FORMAÇÃO**

- BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO
- MESTRADO EM CIÉNCIA DE DADOS

**BIOGRAFIA**

Amir vem de uma família de imigrantes, trabalha no BTG Pactual há 5 anos, e recentemente teve uma promoção na área de Automação. Ele é responsável por desenvolver e manter sistemas automatizados utilizados nas operações de trading e investimentos do banco. Possui uma grande habilidade em implementar soluções eficientes. Em seu tempo livre, seu hobby favorito é escalar montanhas e viajar para fazenda de sua família.

**DORES**

- Não existe um tratamento de emoticons no sistema de análise de sentimentos;
- Falta de métricas e dados organizados;
- Dificuldade de compreensão de um escopo micro dos dados, além do entendimento aprofundado sobre os sentimentos dos clientes do BTG.

**NECESSIDADES**

- Precisa de um sistema capaz de realizar a análise de comentários das campanhas de marketing do BTG, identificando e classificando a intenção de cada comentário
- Necessita de um sistema mais eficaz e preciso para o entendimento de emoções.
- Deseja obter insights significativos a partir dos dados analisados, para um melhor direcionamento das campanhas, obtendo um melhor engajamento.

Figura 16: Persona 3

## 5.2 Jornadas do Usuário

Jornada do usuário é o caminho que um cliente percorre ao interagir com um produto ou serviço, desde a descoberta até a finalização. É importante entender essa jornada para melhorar a experiência do usuário de acordo com suas dores e responsabilidades, a fim de que o planejamento não se torne apenas um mapeamento de processos, mas um mapa de valor.

### 5.2.1 Jornada 1 - Analista de Marketing:

No caso abaixo, a jornada está dividida em 5 fases:

1. Identificação com o problema - o cliente entende quais são suas dores;
2. Descoberta - onde o cliente é exposto ao produto;
3. Utilização do modelo - o cliente começa a utilizar a solução;
4. Compreensão dos dados - o cliente começa a entender o que os dados estão mostrando à ele;
5. Ajustes - a solução cumpre o seu papel e o marketing consegue ajudar de acordo com as necessidades dos clientes.

A primeira Jornada do Usuário é sobre o Marcos Santos, que espera que a solução consiga trazer uma melhoria na área de Marketing e que a empresa consiga entender cada dia mais sobre suas campanhas. Ele acredita que, com a solução, a obtenção de dados se torna mais valiosa, sendo capaz de transformar a maneira como o Marketing atua dentro da empresa.



**Marcos Santos**

**Cenário:** Marcos necessita de uma solução que faça uma análise de sentimento dos dados e que forneça insights sobre o comportamento e preferências dos usuários que seguem o Instagram do BTG Pactual.

#### Expectativas

Marcos espera uma melhoria do engajamento dos usuários que seguem o Instagram do BTG Pactual e no desempenho geral das campanhas de marketing digital.

FASE 1 (Identificação do problema)	FASE 2 (Descoberta da solução)	FASE 3 (Utilizando o modelo de PLN)	FASE 4 (Compreensão dos dados a partir da IA)	FASE 5 (Ajustes nas campanhas de marketing)
<p>1. Baixo entendimento sobre quais elementos das campanhas estão gerando resultados positivos e quais não estão.</p> <p>2. Dificuldade em encontrar informações sentimentais significativas em meio a uma grande variedade de comentários.</p>	<p>1. Utilizar Inteligência Artificial para Marketing, com objetivo de monitorar campanhas utilizando processamento de linguagem natural (PLN).</p> <p>2. Solução de análise de sentimento e identificação de palavras-chave.</p> <p><i>Essa análise poderia ser realizada com o uso de algoritmos de aprendizado de máquina</i></p>	<p>1. Em uma interface, será possível visualizar os resultados que a inteligência artificial gerará sobre os dados analisados.</p> <p>2. O dashboard mostrará as informações sentimentais coletadas, permitindo uma fácil compreensão dos insights gerados pela ferramenta.</p> <p><i>Muito bom ter a visualização sobre os dados, a partir de dashboards/gráficos.</i></p>	<p>1. A partir da análise feita com PLN, será possível o entendimento sobre o que as campanhas têm sido positivo e o que tem sido negativo.</p> <p><i>É possível um maior entendimento sobre os padrões e preferências dos consumidores.</i></p>	<p>1. A partir das informações obtidas com a utilização do PLN, será possível um melhor direcionamento de novas campanhas baseadas nos interesses dos consumidores.</p> <p>2. As postagens serão mais assertivas e atrativas para os usuários do Instagram, além do aumento do engajamento.</p>

#### Oportunidades

As oportunidades seriam obter informações valiosas sobre a preferência e comportamento dos usuários. Com isso, seria possível a identificação dos tipos de conteúdo que geram maior engajamento, bem como os pontos fracos que precisam ser melhorados, resultando em um aumento geral do engajamento e desempenho das campanhas.

#### Responsabilidades

A responsabilidade será atribuída tanto pelos desenvolvedores do modelo de processamento de linguagem natural (PLN), quanto aos analistas de marketing que utilizarão essa ferramenta para melhorar a eficiência das campanhas do Instagram do BTG Pactual.

Figura 17: Jornada de usuário 1

### 5.2.2 Jornada 2 - Analista de Produto:

No caso abaixo, a jornada está dividida nas seguintes fases:

1. Criação e divulgação de um novo produto - escolha e desenvolvimento de um produto;
2. Identificação de problemas e sugestões - referente à campanha do produto desenvolvido;
3. Recolhimento de feedbacks do público - rastreamento e coleta dos dados;
4. Criação de um novo produto ou melhoria do próprio - a partir dos dados, buscar melhorias e/ou inovações;
5. Nova campanha de divulgação do produto - a partir da melhoria ou inovação, fazer uma nova campanha.

A segunda Jornada do Usuário é sobre a Gabriela Ferreira, que espera que a solução consiga trazer uma melhoria na área de Produtos e que junto com a área de Marketing consigam entender cada dia mais sobre as necessidades do cliente. Ela acredita que, com a solução, a obtenção de dados se torna mais valiosa, sendo capaz de transformar como a empresa trata o desenvolvimento de novos produtos ou a melhoria daqueles que já existem.

	<b>Gabriela Ferreira</b> <b>Cenário:</b> Necessita de uma ferramenta que monitore e notifique demandas feitas nas redes sociais em relação a um novo produto divulgado, ou algum já existente, em campanhas de marketing, permitindo uma atuação rápida para impulsionar o hype.	<b>Expectativas</b> Receber feedbacks rápidos e práticos dos produtos que foram divulgados pelas campanhas de marketing, para agir rapidamente em qualquer situação.		
<b>FASE 1</b> (Criação e divulgação de um novo produto)	<b>FASE 2</b> (Identificação de problemas e sugestões)	<b>FASE 3</b> (Recolhimento de feedbacks do público)	<b>FASE 4</b> (Criação de um novo produto ou melhoria do próprio)	<b>FASE 5</b> (Nova campanha de divulgação do produto)
1. Após identificar uma oportunidade de mercado de acordo com as demandas de cliente, a equipe de desenvolvimento do BTG desenvolverá o novo produto; 2. É definida uma estratégia de lançamento, para que seja publicado a campanha de um novo produto nas redes sociais do BTG.	1. Baixo monitoramento do lançamento e identificação de problemas e sugestões dos usuários; 2. Dificuldade em categorizar os problemas e sugestões, a fim de entender os principais problemas dos usuários e sugestões relevantes para campanha.	1. Definição dos objetivos da coleta de feedback, como tendências que devem ser observadas para criação do novo produto, da campanha e o que deve ser descartado; 2. Análise dos resultados com o intuito de identificar o que melhorar, pontos fortes e fracos e sugestões para melhorias do produto.	1. Com base nos resultados da análise, será realizada a implementação de melhorias do produto existente; 2. Após a realização de testes e avaliação desse novo produto, será desenvolvida a versão final.	1. Com base nos resultados gerados pela análise dos comentários, é realizado uma nova campanha de divulgação do novo produto sugerido pelo público.
<b>Oportunidades</b> As oportunidades serão que os analistas de produto terão uma análise mais precisa sobre os feedbacks das postagens do BTG, portanto terão seus comentários filtrados para a melhoria do produto e de sua campanha.	<b>Responsabilidades</b> O analista de produto deve entender as necessidades dos clientes e identificar oportunidades de lançamento do novo produto, além de desenvolver um novo conceito de produto que atenda às necessidades dos clientes.			

Figura 18: Jornada de usuário 2

## 5.2.2 Jornada 3 - Técnico de Automação:

No caso abaixo, a jornada está dividida nas seguintes fases:

1. Recolhimento dos dados - Rastreamento e coleta de dados;
2. Pré-processamento dos dados - transferência dos dados e tratamento dos mesmos;
3. Criação e modelagem da IA - Desenvolvimento da IA e utilização da mesma;
4. Revisão e validação dos resultados - Obter resultados e validá-los;
5. Interface de visualização e envio para o Marketing - Meio de visualização final dos dados;

A terceira Jornada do Usuário é sobre Amir Abdullah, que espera que a solução consiga trazer uma melhoria na área de Automação, a partir de uma melhor visualização dos dados, e que junto com a área de Marketing consigam entender cada dia mais sobre as necessidades do cliente. Ele acredita que, com a solução, a obtenção de dados das campanhas de Marketing pode mudar o direcionamento/acompanhamento da empresa.



**Amir Abdullah**

**Cenário:** Amir necessita de um sistema capaz de realizar a análise de sentimento nos comentários das campanhas de marketing do BTG, que os classifica em 3 categorias: positivo, negativo e neutro, e com isso entender as necessidades dos usuários.

### Expectativas

Amir espera automatizar o entendimento das reações do público em relação as campanhas de marketing e obter insights significativos a partir dos dados analisados.

FASE 1 (Recolhimento dos dados)	FASE 2 (Pré-processamento dos dados)	FASE 3 (Criação e modelagem da IA)	FASE 4 (Revisão e validação dos resultados)	FASE 5 (Interface de visualização e envio para o Marketing)
1. A API coleta os comentários das postagens do perfil do BTG e os coloca em uma base de dados.  Gostaria de receber os comentários coletados em um banco de dados	1. Os dados serão transferidos manualmente do banco de dados para o ambiente de tratamento; 2. Um desenvolvedor realiza a normalização dos dados.  Gostaria tratar os dados coletados para serem utilizados futuramente na criação do modelo preditivo	1. Criação e desenvolvimento da IA; 2. Escolha dos requisitos e métricas para treinamento da IA; 3. Testes de precisão.  Gostaria de ter um sistema capaz de realizar a análise de sentimentos completa dos comentários	1. Obtenção dos resultados de teste e visualização da precisão; 2. Após a validação, para casos de falha da IA há a necessidade de retornar para a FASE 3.  Gostaria de obter os resultados do teste, e verificar se é necessário realizar os testes novamente	1. Um Dashboard contendo os resultados e a precisão atingida pelo modelo; 2. Envio dos resultados para a área de marketing  Com isso, posso criar uma interface de visualização, para enviar à área de marketing

### Oportunidades

A grande oportunidade é que os analistas de marketing terão uma análise mais precisa sobre os comentários das postagens do BTG, que os separa entre positivo e negativo. Com isso, será possível criar Dashboards precisos e assim, aumentar a relevância de futuras postagens.

### Responsabilidades

A responsabilidade será atribuída para os desenvolvedores que irão utilizar o sistema de análise dos dados. Já o sistema é responsável por realizar todo o processo de tratamento e processamento dos dados, além de uma interface de visualização.

miro

Figura 19: Jornada de usuário 3

## 5.3 User Stories

User stories são descrições curtas e simples de funcionalidades que um usuário precisa para alcançar um objetivo específico em relação a um produto ou sistema. Elas ajudam fornecendo uma descrição clara e concisa dos requisitos que o sistema deve ter, mantendo o foco na solução das dores do usuário.

Esse modelo é dividido em: 1. Número - usado para a identificação; 2. Título - também usado para a identificação; 3. Personas - cita quais pessoas estão envolvidas nessa user story; 4. História - descrição da funcionalidade; 5. Critérios de aceitação - quais são as etapas necessárias para que a história aconteça; 6. Testes de aceitação - verificações de aceite e recusa caso algo aconteça.

<b>Número</b>	01
<b>Título</b>	Pré - processamento
<b>Personas</b>	Amir Abdullah - Técnico de Automação
<b>História</b>	Eu, como técnico de automação, quero poder realizar o pré-processamento com uma base maior de dados, para que eu consiga atualizar frequentemente as análises.
<b>Critérios de aceitação</b>	<ol style="list-style-type: none"><li>1. Entendimento dos dados</li><li>2. Limpeza</li><li>3. Análise</li><li>4. Organização / Estruturação</li></ol>
<b>Testes de aceitação</b>	<p>Critério 1:</p> <ul style="list-style-type: none"><li>- Todas as colunas foram processadas<ul style="list-style-type: none"><li>- Aceitou: Correto, começar o próximo critério</li><li>- Recusou: Errado, analisar origem do erro e resolvê-lo</li></ul></li><li>- Todas as linhas foram processadas<ul style="list-style-type: none"><li>- Aceitou: Correto, começar o próximo critério</li><li>- Recusou: Errado, analisar origem do erro e resolvê-lo</li></ul></li></ul> <p>Critério 2:</p> <ul style="list-style-type: none"><li>- O código executa todas as técnicas de pré processamento sem erros<ul style="list-style-type: none"><li>- Aceitou: Correto, começar o próximo critério</li><li>- Recusou: Errado, revisar o código</li></ul></li><li>- O código executa uma (ou mais) técnica (s) de pré processamento com erros<ul style="list-style-type: none"><li>- Aceitou: Errado, revisar o código</li><li>- Recusou: Correto, começar o próximo critério</li></ul></li></ul> <p>Critério 3:</p> <ul style="list-style-type: none"><li>- Todos os elementos gráficos para a análise foram executados com sucesso {elementos gráficos: infográficos; tabelas; gráficos}<ul style="list-style-type: none"><li>- Aceitou: Correto, começar o próximo critério</li><li>- Recusou: Errado, revisar o código</li></ul></li></ul>

	<ul style="list-style-type: none"> <li>- A Persona conseguiu tirar insights dos dados           <ul style="list-style-type: none"> <li>- Aceitou: Correto, começar o próximo critério</li> <li>- Recusou: Errado, revisar o código ou analisar novamente</li> </ul> </li> </ul> <p>Critério 4:</p> <ul style="list-style-type: none"> <li>- Os dados foram estruturados e transformados em valores numéricos apenas           <ul style="list-style-type: none"> <li>- Aceitou: Correto, começar a próxima user story;</li> <li>- Recusou: Errado, revisar se o processo de tokenização foi realizado.</li> </ul> </li> </ul>
--	---

Tabela 02: User Story 01

Com a base de dados devidamente preparada, é então possível o desenvolvimento dos modelos de processamento de linguagem natural:

<b>Número</b>	02
<b>Título</b>	Modelagem
<b>Personas</b>	Amir Abdullah - Técnico de Automação
<b>História</b>	<p>Eu, como técnico de automação, quero fazer a modelagem dos dados, separando os dados em treino e teste, para poder validar o modelo e os parâmetros usados.</p>
<b>Critérios de aceitação</b>	<ol style="list-style-type: none"> <li>1. Separação dos dados em teste e treino</li> <li>2. Exposição dos resultados</li> </ol>
<b>Testes de aceitação</b>	<p>Critério 1:</p> <ul style="list-style-type: none"> <li>- Os dados foram separados em amostra de treino e teste entre os intervalos 80/20 e 70/30:           <ul style="list-style-type: none"> <li>- Aceitou: Correto, começar o próximo critério.</li> <li>- Recusou: Errado, revisar os intervalos descritos no código.</li> </ul> </li> </ul> <p>Critério 2:</p> <ul style="list-style-type: none"> <li>- O modelo foi treinado apenas com os tokens de treino           <ul style="list-style-type: none"> <li>- Aceitou: Correto, realizar o segundo teste.</li> <li>- Recusou: Errado, o .fit deve ser realizado apenas nos tokens de treino.</li> </ul> </li> <li>- O modelo foi treinado incluindo os tokens de treino           <ul style="list-style-type: none"> <li>- Aceitou: Errado, revisar o código.</li> <li>- Recusou: Correto, avançar para a user story 3</li> </ul> </li> </ul>

Tabela 03: User Story 02

Depois da modelagem, deve ser avaliado e comparado os modelos para a prevalência dos melhores modelos com os melhores parâmetros:

<b>Número</b>	03
<b>Título</b>	Validação
<b>Personas</b>	Amir Abdullah - Técnico de Automação
<b>História</b>	<p>Eu, como técnico de automação, quero fazer a validação dos resultados gerados para poder comparar a acurácia das diferentes épocas e diferentes modelos existentes</p>
<b>Critérios de aceitação</b>	<ol style="list-style-type: none"> <li>1. Análise de métricas</li> <li>2. Escolha do melhor modelo</li> <li>3. O modelo deve ter os hiperparâmetros ajustados</li> </ol>
<b>Testes de aceitação</b>	<p>Critério 1:</p> <ul style="list-style-type: none"> <li>- O conjunto de teste foi usado para determinar sua precisão e desempenho geral.             <ul style="list-style-type: none"> <li>- Aceitou: Correto, realizar o segundo teste.</li> <li>- Recusou: Errado, revisar o código.</li> </ul> </li> <li>- O conjunto de teste não foi usado para determinar sua precisão e desempenho geral.             <ul style="list-style-type: none"> <li>- Aceitou: Errado, revisar o código.</li> <li>- Recusou: Correto, começar o próximo critério.</li> </ul> </li> </ul> <p>Critério 2:</p> <ul style="list-style-type: none"> <li>- O modelo foi escolhido conforme o melhor desempenho em perda e acurácia.             <ul style="list-style-type: none"> <li>- Aceitou: Correto, começar o próximo critério.</li> <li>- Recusou: Errado, revisar a escolha do modelo</li> </ul> </li> </ul> <p>Critério 3:</p> <ul style="list-style-type: none"> <li>- Diferentes hiperparâmetros foram testados para a escolha do conjunto ideal             <ul style="list-style-type: none"> <li>- Aceitou: Correto, avançar para a user story 4.</li> <li>- Recusou: Errado, revisar o código.</li> </ul> </li> </ul>

Tabela 04: User Story 03

Posteriormente, os times de produto e marketing poderão usufruir da automatização de análises provenientes do modelo:

<b>Número</b>	04
<b>Título</b>	Visualização dos dados - desenvolvimento
<b>Personas</b>	Gabriela Ferreira - analista de produtos

<b>História</b>	Eu, como analista de produtos, quero ter uma visualização objetiva com <i>insights</i> sobre as necessidades e sentimento dos clientes, para que me apoie no processo de elaboração de novos produtos.
<b>Critérios de aceitação</b>	<ol style="list-style-type: none"> <li>1. Desenvolvimento da interface</li> <li>2. Ranking de palavras-chaves</li> <li>3. Todas as palavras-chaves são relevantes</li> </ol>
<b>Testes de aceitação</b>	<p>Critério 1:</p> <ul style="list-style-type: none"> <li>- Acesso a interface com os elementos gráficos sem problemas:           <ul style="list-style-type: none"> <li>- Aceitou: Correto, começar o próximo critério.</li> <li>- Recusou: Errado, verificar a disponibilidade do site.</li> </ul> </li> <li>- A interface mostra os gráficos dentro do padrão:           <ul style="list-style-type: none"> <li>- Aceitou: Correto, avançar para a User Story 5</li> <li>- Recusou: Errado, revisar o código ou a conexão com a <i>Internet</i>.</li> </ul> </li> </ul> <p>Critério 2:</p> <ul style="list-style-type: none"> <li>- A interface exibe as 5 palavras-chave mais mencionadas pelos comentários das campanhas:           <ul style="list-style-type: none"> <li>- Aceitou: Correto, avançar para a User Story 5</li> <li>- Recusou: Errado, revisar o código.</li> </ul> </li> </ul> <p>Critério 3:</p> <ul style="list-style-type: none"> <li>- De 5 palavras-chave escolhidas, nenhuma é semântica recorrente de frases em qualquer outro contexto, ou seja, irrelevante.           <ul style="list-style-type: none"> <li>- Aceitou: Correto, avançar para a User Story 5.</li> <li>- Recusou: Errado, ajustar o modelo.</li> </ul> </li> </ul>

Tabela 05: User Story 04

Apesar da mesma interface e mesma análise, o analista de marketing e a analista de produtos têm prioridades diferentes de visualização na plataforma por se tratar de dores distintas.

<b>Número</b>	05
<b>Título</b>	Visualização dos dados
<b>Personas</b>	Marcos Santos - analista de marketing
<b>História</b>	Eu, como analista de marketing, quero poder acessar a interface de forma imediata, para que eu tenha acesso às análises de comportamento e palavras-chave das demandas no Instagram.

<b>Critérios de aceitação</b>	1. Rastreamento dos sentimentos por campanha 2. Comparação de sentimentos entre campanhas
<b>Testes de aceitação</b>	<p>Critério 1:</p> <ul style="list-style-type: none"> <li>- A interface exibe um escopo macro dos sentimentos dos clientes por campanha             <ul style="list-style-type: none"> <li>- Aceitou: Correto.</li> <li>- Recusou: Errado, revisar o código.</li> </ul> </li> </ul> <p>Critério 2:</p> <ul style="list-style-type: none"> <li>- A interface exibe uma comparação entre campanhas para estabelecer o produto mais bem sucedido             <ul style="list-style-type: none"> <li>- Aceitou: Correto.</li> <li>- Recusou: Errado, revisar o código.</li> </ul> </li> </ul>

Tabela 06: User Story 05

## 6. Análise descritiva

### 6.1 Introdução

A análise descritiva é uma técnica estatística que pode ser aplicada em diferentes áreas, incluindo a análise de sentimentos. No contexto do projeto proposto pelo BTG de análise de sentimentos realizado a partir de comentários de usuários em publicações no *Instagram* do banco, a análise descritiva é utilizada para descrever e resumir as principais características dos dados coletados.

Por meio desta, é possível obter informações sobre: 1. O número total de comentários coletados; 2. A distribuição de sentimentos positivos, negativos e neutros expressos pelos usuários; 3. As palavras mais frequentes nos comentários; 4. Os usuários que mais realizaram comentários. Essas informações são cruciais para compreender melhor a percepção dos usuários em relação ao banco e para orientar futuras estratégias de comunicação e relacionamento com o público, garantindo uma maior assertividade em futuras publicações do banco BTG.

### 6.2 Método

Os comentários realizados pelos usuários nas publicações do banco BTG são uma fonte valiosa de informações para entender como os clientes se sentem em relação aos serviços oferecidos pela instituição financeira. Para realizar a análise desses dados, foram utilizados diversos métodos de tratamento de dados, que serão descritos no tópico 6.2.2 Pré - processamento.

Para visualizar as informações de maneira clara e acessível, foram utilizados gráficos de barra e pizza. Os gráficos de barra foram utilizados para a identificação dos autores mais ativos e o tipo de sentimento causado (positivo, neutro ou negativo). Já nos gráficos de pizza destacam os tipos de interação mais utilizados, alternando entre comentários, menções e replies e também tipos de sentimentos expressos pelos usuários.

Para realizar a análise e a visualização desses dados, foram utilizadas bibliotecas como: *Matplotlib*, que é uma biblioteca de visualização de dados em Python, além de bibliotecas notáveis como é o caso do *pandas*, *numpy* e a *nltk*. Com essas ferramentas, foi possível obter *insights* valiosos sobre a percepção dos clientes em relação ao Banco BTG e identificar áreas que precisam de melhorias.

## 6.3 Resultados

Na análise descritiva dos dados, foram explorados 2 tipos de gráficos: gráficos de pizza e barras. Utilizando técnicas de visualização, foi possível apresentar informações relevantes e obter *insights* sobre os dados em questão. Abaixo serão descritos os gráficos e os resultados obtidos.

### 6.3.1 Autores

A seguir é mostrado o código utilizado para plotar o primeiro gráfico de barras, com o objetivo de demonstrar quais são os usuários (autores) que mais comentam nos *posts* do BTG Pactual.

```
autor_counts = df['autor'].explode().value_counts()
plt.figure(figsize=(10, 6))
autor_counts.head(20).plot(kind='bar')
plt.xlabel('Autores')
plt.ylabel('Contagem')
plt.title('Autores que mais comentam')
plt.show()
```

Na primeira linha é criada uma nova variável chamada `autor_counts`. Ela utiliza a coluna "autor" do *dataframe* `df`. A função `explode()` é aplicada para transformar uma coluna de listas em várias linhas, e em seguida, a função `value_counts()` é aplicada para contar a ocorrência de comentários daqueles usuários. A segunda linha especifica um tamanho para a figura.

Na terceira linha é utilizado o `head(20)` para selecionar somente os 20 primeiros valores da variável `autor_counts`. Em seguida, o método `plot` é chamado com o parâmetro `kind= 'bar'`, indicando que um gráfico de barras deve ser criado.

Na quarta e quinta linha é criado um rótulo no eixo x e y do gráfico com o texto "Autores" e "Contagem", respectivamente. Na sexta linha é definido qual é o título do gráfico: "Autores que mais comentam". Por último, a sétima linha exibe o gráfico abaixo.

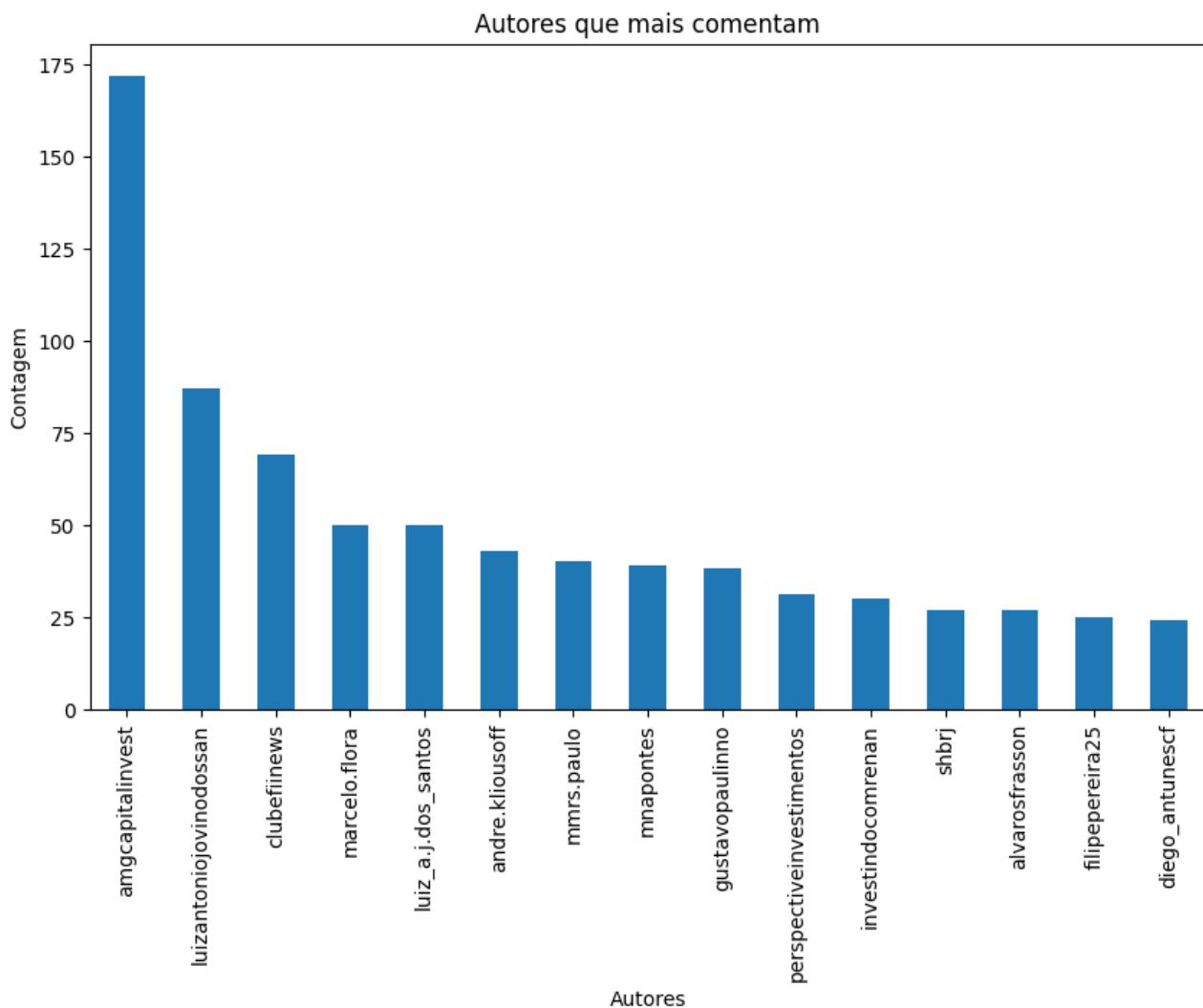


Figura 20: Gráfico “Autores que mais comentam”

Com esse gráfico foi possível observar que, na maioria das vezes, tem um padrão muito claro de frequência de comentários, o que significa que a empresa mantém um público específico que também é muito engajado. Apesar disso, foi criada a hipótese de que, pelo fato do primeiro usuário (@amgcapitalinvest) ser uma empresa credenciada pelo BTG, ela marca o banco nos seus posts, referenciando os créditos, é possível interpretar que talvez não sejam somente comentários.

### 6.3.2 Tipos de interação

A seguir é mostrado o código utilizado para plotar o segundo gráfico, que com o objetivo demonstrar a diferença entre os tipos de interação presentes no *dataframe*.

```
count_interactions = df['tipoInteracao'].value_counts()
plt.figure(figsize=(8, 6))
count_interactions.plot(kind='pie', autopct='%.1f%%')
plt.title('Tipos de Interação')
plt.ylabel('')
plt.show()
```

Na primeira linha, é criada uma nova variável chamada `count_interactions`, que tem como função utilizar a coluna "tipoInteracao" do `dataframe df`. E com isso, a função `value_counts()` é aplicada para contar a ocorrência de cada tipo de interação. A seguir é especificado qual é o tamanho do gráfico.

Na terceira linha, a variável `count_interactions` usa o método `plot` com o parâmetro `kind='pie'`, que indica o tipo de gráfico que deve ser gerado, nesse caso de pizza. Além disso, o parâmetro `autopct='%.1f%%'` é utilizado para exibir a porcentagem de cada fatia no gráfico.

Na quarta linha é definido um título para o gráfico, com o texto "Tipos de Interação". E a seguir, na quarta linha, o rótulo do eixo y é removido, por ser um gráfico de pizza e as porcentagens já estão sendo mostradas. Por último, o gráfico abaixo é exibido na saída.

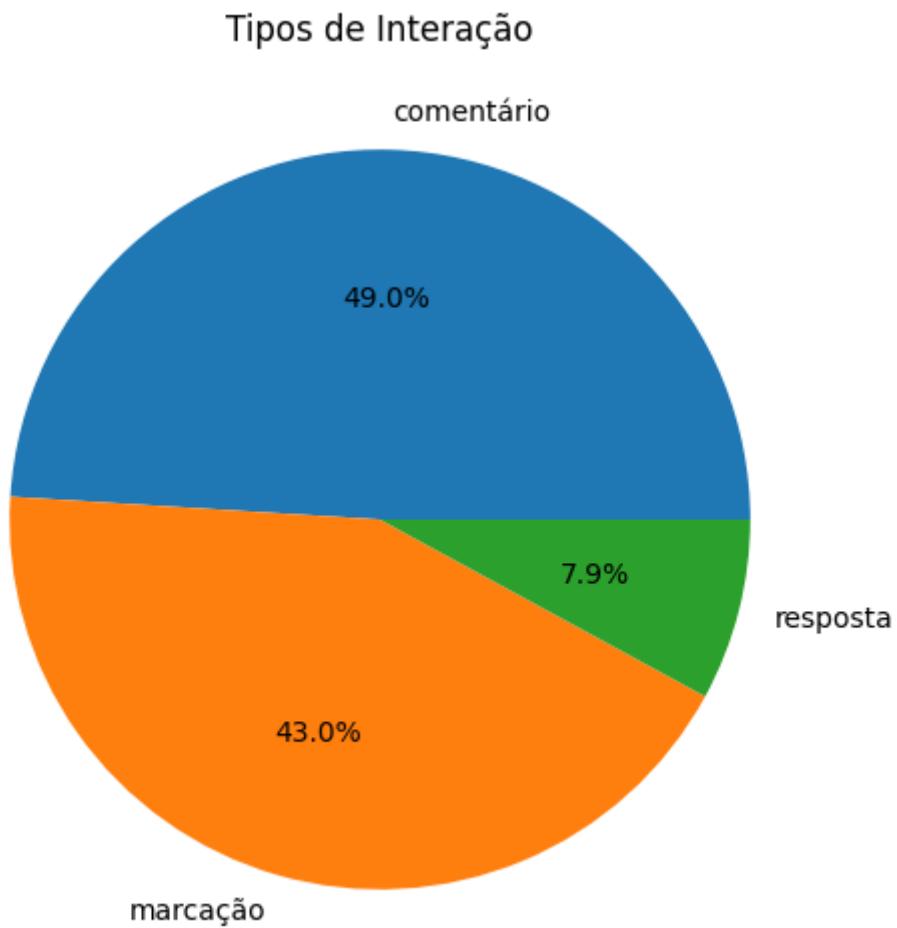


Figura 21: Gráfico “Tipos de interação”

O gráfico acima demonstra que, caso a hipótese de ter *repost* dos *posts* do BTG esteja certa, o `dataframe` está, em sua maioria com esses casos, o que torna preocupante, já que a ideia é que o projeto analise comentários dos *posts*. Além disso, pode-se observar uma diferença significativa entre “comentários” e “resposta”.

### 6.3.3 Classificação de sentimento

Para esse tipo de gráfico foi criado um gráfico de barra. O código e o gráfico serão apresentados abaixo.

```
count_sentimentos = df['sentimento'].value_counts()  
  
plt.figure(figsize=(10, 6))  
  
count_sentimentos.plot(kind='bar')  
  
plt.xlabel('Sentimentos')  
  
plt.ylabel('Contagem')  
  
plt.title('Tipos de Sentimento')  
  
plt.show()
```

A primeira linha é criada uma nova variável chamada `count_sentimentos`, que utiliza a coluna "sentimento" do `dataframe` `df`. A função `value_counts()` é aplicada para contar a ocorrência de cada tipo de sentimento. Essa linha pertence aos 2 tipos de gráficos, isso porque ela só está definindo a variável e função que serão utilizadas posteriormente.

A segunda linha define qual será o tamanho da figura que será gerada no final do código. A seguir, a variável `count_sentimentos` é plotada utilizando o método `plot` com o parâmetro `kind= 'bar'`, indicando o tipo de gráfico, essa linha que diferencia os tipos de gráficos. As próximas 3 linhas são usadas para definir os rótulos dos eixo x e y e o título do gráfico. A última linha exibe o gráfico a seguir na saída.

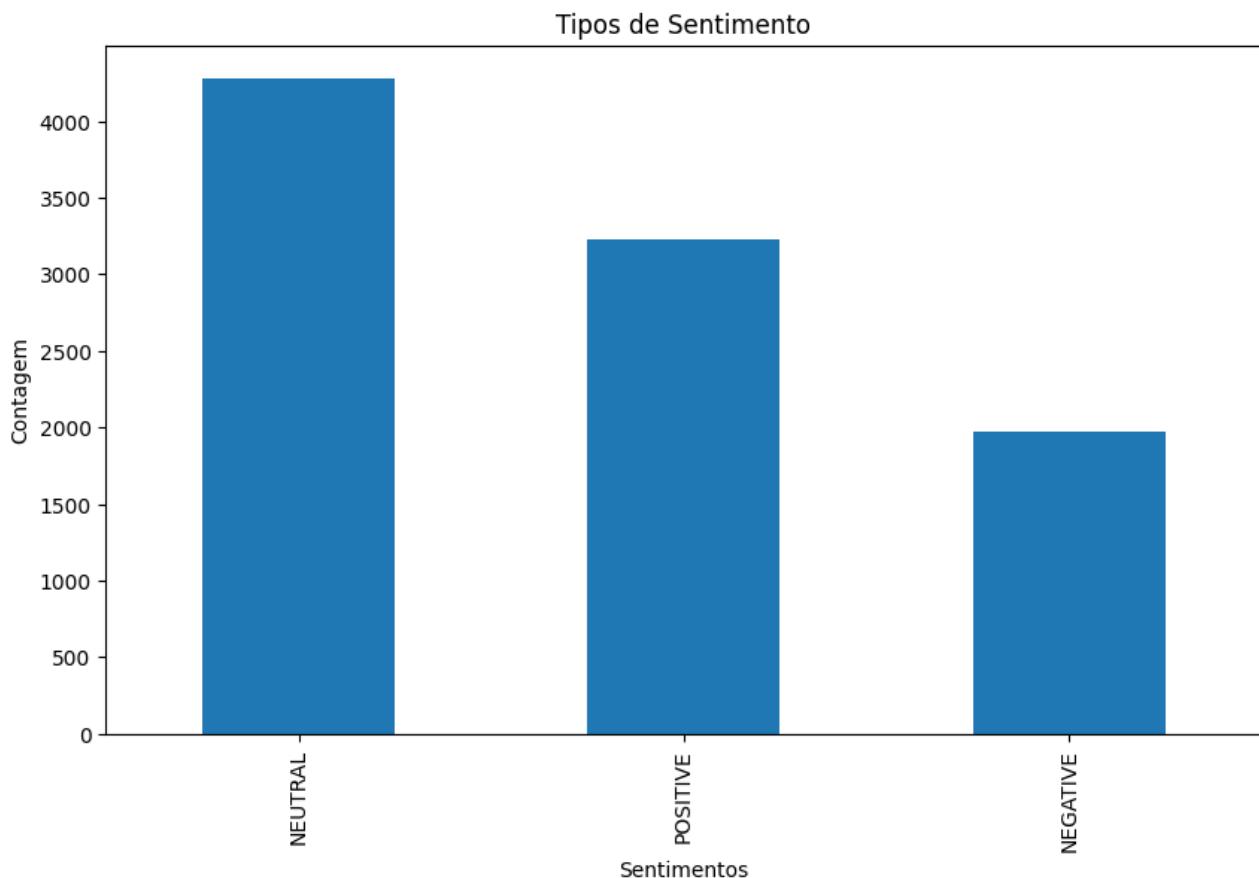


Figura 22: Gráfico “Tipos de sentimento” - Barras

Analisando o gráfico é possível observar que a quantidade de comentários neutros é maior que os outros dois, pode-se interpretar que essa métrica é ruim para os dados, e com isso podemos chegar em duas hipóteses: 1. mais de 4000 comentários não causam nenhum tipo de sentimento para as pessoas; ou 2. a classificação feita está equivocada, caso os *posts* causem algum tipo de sentimento. Além disso, a quantidade de comentários positivos é muito maior do que negativo, o que se pode referir que os usuários estão se sentindo contentes com os serviços prestados.

#### 6.3.4 Frequência dos sentimentos por tipo de interação

Para esse tipo de gráfico foi criado um gráfico de barra. O código e o gráfico serão apresentados abaixo.

```

contagem = df.groupby(['tipoInteracao',
'sentimento']).size().unstack(fill_value=0)

plt.figure(figsize=(10, 6))

contagem.plot(kind='bar', stacked=True)

plt.xlabel('Sentimentos por Interação')

plt.ylabel('Contagem')

plt.title('Tipos de Sentimento com Interação')

```

```
plt.show()
```

A primeira linha é criada uma nova variável chamada contagem, que utiliza as colunas "tipoInteracao" e "sentimento" do *dataframe* df. A função groupby() agrupa os dados pelos valores das colunas e a função size() conta o número de ocorrências para cada grupo. Em seguida, a função unstack() transforma os resultados em um *dataframe*, onde cada tipo de sentimento é uma coluna e cada tipo de interação é um índice. O parâmetro fill\_value=0 preenche os valores ausentes com zero.

A segunda linha define qual será o tamanho da figura que será gerada no final do código. A seguir, a variável contagem é plotada utilizando o método *plot* com o parâmetro kind='bar', indicando o tipo de gráfico, essa linha que diferencia os tipos de gráficos e o parâmetro stacked=True, que empilha as barras a fim de visualizar a contagem total. As próximas 3 linhas são usadas para definir os rótulos dos eixo x e y e o título do gráfico. A última linha exibe o gráfico a seguir na saída.

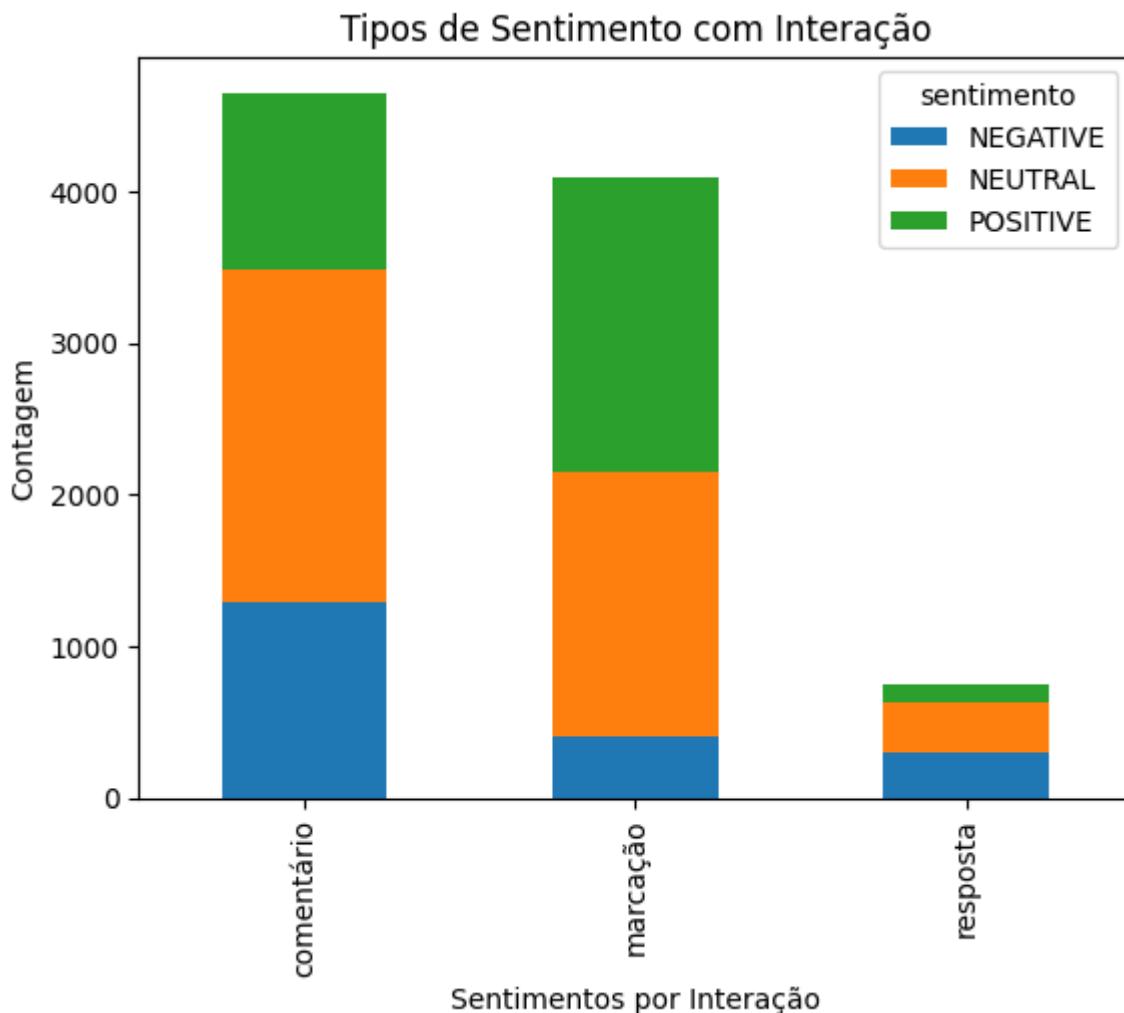


Figura 23: Gráfico “Tipos de sentimento com Interação” - Barras

Analisando o gráfico é possível observar que a divisão entre os sentimentos por interação, por exemplo na interação “comentário” a maior parte gerou sentimento

neutro, coisa que é replicada em todas as interações e também no gráfico analisado anteriormente. O *insight* mais interessante é que a interação “marcação” tem uma quantidade significativa de comentários positivos.

## 6.4 Conclusão

Esta análise descritiva dos gráficos proporciona uma compreensão mais profunda dos dados, permitindo identificar *insights* e tomar decisões. É importante ressaltar que as conclusões obtidas são interpretadas considerando o contexto específico dos dados e as questões de pesquisa em análise.

# 7. Criação de features

## 7.1 Introdução

A criação de features, também conhecida como engenharia de feature, é um processo que envolve a transformação ou combinação dos dados brutos em formatos que possam ser mais facilmente compreendidos e utilizados pelos algoritmos de aprendizado de máquina. Essa etapa pode desempenhar um papel fundamental no sucesso e desempenho dos modelos.

## 7.2 Método

No caso desse projeto, foi criado um dicionário com todos os emojis relevantes que estavam no *dataframe*, que foram transformados para o seu significado em português. Com isso, cada emoji identificado se tornou uma coluna.

## 7.3 Resultados

### 7.3.1 Extração e Contagem

Os códigos abaixo realizam a extração e a contagem dos emojis, para depois criar as features. A primeira linha cria um conjunto vazio, chamado de `emojis_encontrados`, para armazenar os emojis. A segunda linha define uma expressão regular que encontra os emojis, em diferentes categorias, como mostra os comentários feitos.

```
emojis_encontrados = set()

emoji_pattern = re.compile([
    u"\U0001F600-\U0001F64F" # emoticons
    u"\U0001F300-\U0001F5FF" # símbolos
    e pictogramas
    u"\U0001F680-\U0001F6FF" #
    transporte e símbolos de mapas
    u"\U0001F1E0-\U0001F1FF" # bandeiras
    do mundo
```

```
"] + ", flags=re.UNICODE)
```

A primeira linha realiza a iteração de cada linha da coluna “texto” do *dataframe* df. Já a segunda, aplica a expressão regular definida anteriormente para encontrar todos os emojis do *dataframe*. Além disso, a lista dos emojis\_encontrados será convertida para lista\_de\_emojis, e o conjunto vazio emo é criado novamente.

```
for text in df['texto']:
    emojis_encontrados.update(set(emoji_pattern.findall(text)))

lista_de_emojis = list(emojis_encontrados)

emojis_separados = []
```

O código se inicia iterando novamente a variável lista\_de\_emojis. A segunda linha, cada emoji é separado para ser um caractere único, que é adicionado na lista emojis\_encontrados, lista que, por meio do método print, é impresso a seguir.

```
for emoji in lista_de_emojis:
    emojis_separados.extend(list(emoji))

print('Lista de emojis separados:')
print(emojis_separados)

output (primeiros 5 emojis):
Lista de emojis separados:
['📸', '🙏', '💕', '🚀', '💻', ]
```

A frequência dos emojis da lista emojis\_separados é contada por meio do método Counter, a próxima linha realiza a iteração do objeto frequencia\_de\_emojis, que são os emojis e a frequência de cada um. Por último, os emojis e suas respectivas frequências são impressas.

```
frequencia_de_emojis = Counter(emojis_separados)

print('Frequência de cada emoji:')
for emoji, frequencia in frequencia_de_emojis.items():
    print(f'{emoji}: {frequencia}')

output (primeiros 5 emojis):
Frequência de cada emoji:
📸: 2
🙏: 94
💕: 8
🚀: 212
```

A próxima célula é realizado o dicionário de emojis, deixando claro que alguns emojis que tiveram uma frequência baixa e são bem específicos não foram adicionados no dicionário. Abaixo há os 5 primeiros emojis para demonstrar como é feito o dicionário. É importante ressaltar que alguns emojis tem o mesmo significado, e por isso a palavra correspondente é a mesma. A lista emoji\_dict tem 131 emojis.

```
emoji_dict = {
    '😊': 'sorriso',
    '😂': 'risos',
    '😅': 'envergonhado',
    '😆': 'sorridente',
    '🤣': 'sorriso'}
```

len(emoji\_dict)

output: 131

### 7.3.2 Criação das Features

O código abaixo define a função verifica\_emoji que recebe dois parâmetros (“texto” e “emoji”), que tem como função a verificação se o “emoji” está presente no texto. Caso essa afirmação seja verdadeira, o bloco `if` será executado, onde será retornado o valor 1 caso o emoji esteja na lista. Caso a afirmação seja negativa, o bloco `else` será executado, onde será retornado o valor 0. A seguir, é realizada uma iteração no dicionário definido anteriormente, o emoji\_dict. Na próxima linha, uma coluna é adicionada para cada significado no dicionário, e o valor da célula é definido de acordo com a função verifica\_emoji.

```
def verifica_emoji(texto, emoji):
    if emoji in texto:
        return 1
    else:
        return 0

for emoji, descricao in emoji_dict.items():
    df[descricao] = df['texto'].apply(lambda x: verifica_emoji(x, emoji))

df
```

## 7.4 Conclusão

Esse processo foi realizado pensando em melhorar resultados dos modelos que serão desenvolvidos, isso porque os emojis e suas frequências são um ponto crucial

para definir o sentimento gerado. Os códigos apresentados acima realizam dois processos: extração e análise dos emojis e criação de features.

## 8. Pré processamento

### 8.1 Introdução

O pré-processamento de dados no contexto do PLN refere-se a uma série de etapas de preparação que os dados textuais devem passar antes de serem usados em um modelo de aprendizado de máquina. Essas etapas visam limpar, organizar e estruturar os dados textuais para que sejam mais facilmente compreendidos pelo modelo. Algumas etapas importantes do pré- processamento são: 1. Tokenização; 2. Tratamento de abreviações; 3. Tratamento de emoji; 4. Remoção de stopwords; 5. Remoção de alfanuméricos; 6. Lematização. Além disso, foi realizado um tratamento dos dados e a definição de uma função pipeline.

### 8.2 Método

A etapa de pré - processamento, como dito anteriormente, foi dividida em 6 etapas:

**1. Tratamento de dados:** processo que envolve a manipulação, limpeza, enriquecimento e transformação de dados de forma a torná-los mais úteis e adequados para a análise. Onde foram realizados os processos de: Mudança dos nomes das colunas - retirada das aspas (""), retirada de algumas colunas que se mostraram não necessárias para o projeto, remoção dos comentários do banco (autor: abtgpactual) e a verificação de valores nulos (dropna());

**2. Tokenização:** processo de dividir um texto em unidades menores chamadas tokens. Esses tokens podem ser palavras individuais ou partes menores de palavras, como prefixos ou sufixos;

**3. Tratamento de emoji:** processo de transformar símbolos de emojis para o seu significado;

**4. Remoção de alfanuméricos:** processo de retirar os caracteres de pontuação para reduzir o tamanho do vocabulário e evitar ruídos;

**5. Tratamento de abreviações:** processo de transformar palavras abreviadas, muito utilizada em redes sociais, para sua expansão;

**6. Remoção de stopwords:** palavras que podem ser consideradas irrelevantes para o conjunto de resultados a ser exibido;

**7. Lematização:** processo de normalização das palavras, reduzindo a variabilidade e simplificando a análise e compreensão do texto;

**8. Pipeline:** sequência de etapas ou processos interligados que são aplicados aos dados durante o fluxo de trabalho.

Para essa etapa foi utilizada a biblioteca do *NLTK*, para a tokenização e remoção de stopwords, a biblioteca *emoji* para o tratamento do mesmo, o *spacy*, para a lematização e o *pandas* para a leitura e tratamento dos dados.

## 8.3. Resultados

Abaixo serão descritos cada etapa do pré - processamento.

### 8.3.1 Tratamento dos dados

Um dos primeiros tratamentos de dados que foi utilizado, foi o tratamento que retira as aspas duplas ("") dos nomes das colunas da base de dados, já que anteriormente as colunas estavam da seguinte forma: “*texto*”, após esse tratamento, ficou apenas *texto*, como demonstra o código abaixo:

```
data = data.rename(columns={'anomalia' : 'anomalia',
                           'dataPublicada' : 'dataPublicada', 'autor' : 'autor',
                           'texto' : 'texto', 'sentimento' : 'sentimento',
                           'tipoInteracao' : 'tipoInteracao', 'probabilidadeAnomalia' :
                           'probabilidadeAnomalia', 'linkPost' : 'linkPost',
                           'processado' : 'processado', 'contemHyperlink' :
                           'contemHyperlink'})
```

Esse tratamento facilita o trabalho de chamar os textos das colunas de uma maneira mais simples, sem a necessidade de ter que colocar aspas, podendo chamar o texto diretamente.

O segundo tratamento realizado utilizou o método `data.dtypes`, e com isso é possível entender quais são os tipos de cada coluna. Com isso, pode-se entender como as etapas do pré-processamento podem atuar em cada coluna.

```
data.dtypes

output:
id                  int64
dataPublicada      object
autor               object
texto               object
sentimento         object
tipoInteracao      object
anomalia            int64
probabilidadeAnomalia  int64
linkPost             object
processado          int64
contemHyperlink     int64
```

Com isso, foi possível descartar essa coluna utilizando a função `data.drop()`. Vale ressaltar que todas as colunas que foram removidas da base de dados, não possuem tanta relevância para uma análise de sentimento que tem como principal embasamento os textos, como mostra o código abaixo.

```
data_dropado = data.drop(['id', 'dataPublicada', 'anomalia',
    'probabilidadeAnomalia', 'linkPost', 'processado',
    'contemHyperlink'], axis=1)
```

```
data_dropado
```

O terceiro tratamento realizado foi a remoção do autor `btgpactual`, da coluna “autor”, foi possível removê-lo através de uma função que remove apenas o autor mencionado.

```
data_limpo = data_dropado.loc[data_dropado['autor'] != 'btgpactual']
data_limpo
```

Esse tratamento é necessário para o projeto, pois os comentários vindos desse autor não são tão relevantes para análise de sentimentos, uma vez que a maioria são respostas a comentários ou legendas dos *posts*.

Por último, foi realizada uma confirmação de não nulos dentro do *dataframe*, e caso tenha, é removido no código abaixo.

```
df = data_limpo.dropna()
df
```

### 8.3.2 Tokenização

Para começar o pré - processamento pensando no modelo de análise de sentimento, é necessário separar as palavras dos textos em *tokens*, e o código abaixo define a função necessária para realizar esse processo.

```
def tokenizer(comment):
    if isinstance(comment, str):
        tokens = nltk.word_tokenize(comment)
        return tokens
    else:
        return []
```

A função acima realiza o processo descrito referenciando a biblioteca `nltk.word_tokenize`.

### 8.3.3 Tratamento de emoji

A função `demojize_tokens` recebe uma lista de tokens e tem como objetivo realizar o processo de "demojize", ou seja, remover emojis e substituí-los por sua representação textual.

```

def demojize_tokens(words, emoji_dict):
    demojized = []
    for word in words:
        if isinstance(word, list): # Verifica se é uma lista de tokens
            processed_word = []
            for token in word:

                if emoji.emoji_count(token) > 0:
                    if token in emoji_dict:
                        processed_word.append(emoji_dict[token])
                    else:
                        processed_word.append(emoji.demojize(token))
                else:
                    processed_word.append(token)

            demojized.append(processed_word)
        else:
            if emoji.emoji_count(word) > 0:

                if word in emoji_dict:
                    demojized.append(emoji_dict[word])
                else:
                    demojized.append(emoji.demojize(word))
            else:
                demojized.append(word)

    demojized = [[token.replace(":", "").replace("_", "") if any(c in
token for c in [":", "_"]) else token for token in sublist] if
isinstance(sublist, list) else sublist.replace("-", "_") if "-" in sublist
else sublist for sublist in demojized]

    return demojized

```

#### 8.3.4 Remoção de alfanuméricos

Da mesma forma que algumas palavras não têm importância para a análise, a pontuação, caracteres especiais, links, menções e hashtags também não tem, por isso a função abaixo retira esses caracteres.

```

def removendo_alfanumericos(tokens):
    output_tokens = []
    for sentence in tokens:
        output_list = []

```

```

        for palavra in sentence:
            if palavra.strip(): # Verifica se a palavra não é uma
                string vazia
                    palavra_sem_marcacao = re.sub(r'\@\w*'), '',
                palavra)
                    palavra_sem_hashtag = re.sub(r'#\w*'), '',
                palavra_sem_marcacao)
                    palavra_sem_hyperlink = re.sub(r'https\S*', '',
                palavra_sem_hashtag)
                    palavra_sem_www = re.sub(r'\bwww\.[^\s]*', '',
                palavra_sem_hyperlink)
                    palavra_sem_numeros = re.sub(r'[0-9]', '',
                palavra_sem_www)
                    palavra_sem_btg = re.sub(r'\bbtg\b'), '',
                palavra_sem_numeros)
                    palavra_sem_btgpactual = re.sub(r'\bpactual\b'), '',
                palavra_sem_btg)
                    output_list.extend(re.findall(r'\w+', 
                palavra_sem_btgpactual)) # analisar se não é melhor usar o
                append em vez de extend
            output_tokens.append(output_list)
        return output_tokens
    
```

### 8.3.5 Tratamento de abreviações

Para tornar mais fácil a análise de sentimento, foi feito um tratamento de abreviações, para que palavras como: “vcs” se torne “vocês”. O código abaixo define um dicionário de gírias e abreviações usado para normalização de texto.

```

dicionario_girias = {'vc': 'você', 'vcs': 'você', 'Vc': 'você',
'pq': 'porque', 'Pq': 'porque', 'tbt': 'também', 'q': 'que', 'td':
'tudo', 'blz': 'beleza', 'flw': 'falou', 'kd': 'cadê', 'Gnt': 'gente',
'gnt': 'gente', 'to': 'estou', 'mt': 'muito', 'cmg': 'comigo', 'ctz':
'certeza', 'jah': 'já', 'naum': 'não', 'ta': 'está', 'eh': 'é', 'vdd':
'verdade', 'vlw': 'valeu', 'p': 'para', 'sdds': 'saudades', 'qnd':
'quando', 'msm': 'mesmo', 'fzr': 'fazer', 'ss': 'sim', 'Ss': 'sim',
'pdc': 'pode crer', 'nn': 'não', 'Nn': 'não', 'pls': 'please', 'obg':
'obrigado', 'agr': 'agora'}
    
```

A função comentarios\_normalizados abaixo tem como objetivo normalizar os comentários representados por tokens, levando em consideração o dicionário de gírias e abreviações e um conjunto de palavras desconsideradas, os dois citados acima.

```
def comentarios_normalizados(tokens, dicionario_girias):
```

```

tokens_normalizados = []

for sentence in tokens:
    treated = []

    for palavra in sentence:
        if palavra in dicionario_gírias:
            palavra_normalizada = dicionario_gírias.get(palavra,
palavra)
            treated.append(palavra_normalizada)
        else:
            treated.append(palavra)

    treated = [palavra.replace(' ', '') if '_' in palavra else
palavra for palavra in treated]
    tokens_normalizados.append(treated)

return tokens_normalizados

```

### 8.3.6 Remoção de StopWords

Já que as palavras que são consideradas como *stopwords* não tem uma importância para o sentido do texto e elas ocupam a maior parte dos *tokens*, essa etapa foi realizada por meio do código abaixo:

```

def remove_stopwords(tokens):
    filtered_tokens = []
    for sentence in tokens:
        filtered = [palavra for palavra in sentence if palavra not
in stopwords]
        filtered_tokens.append(filtered)
    return filtered_tokens

```

A função acima referencia a biblioteca para que as palavras classificadas sejam removidas do conjunto de tokens. No entanto, deve-se notar que o uso isolado da biblioteca não contempla o dialeto comum nos comentários de internet devido à ausência de abreviaturas e gírias similares ao conjunto de *stopwords*. Para isso, foi unido à lista de stopwords uma lista de abreviações e uma mais extensa lista de preposições por meio da função abaixo:

```

def merge_stopwords(arr1, arr2):
    merged = arr1.copy() # Cria uma cópia do primeiro array
    for element in arr2:

```

```

        if element not in merged:
            merged.append(element)
    return merged

```

### 8.3.7 Lematização

A função `lematizacao()` tem como objetivo realizar a lematização dos tokens, ou seja, converter as palavras para sua forma base ou lemma. O código utiliza o modelo pré-treinado do SpaCy para o idioma português (carregado anteriormente com o `spacy.load("pt_core_news_sm")`) para realizar a lematização.

```

def lematizacao(tokens):
    # Carregar o modelo pré-treinado do SpaCy para o idioma português
    nlp = spacy.load("pt_core_news_sm")
    lemmatized_tokens = []
    for sentence in tokens:
        lemma_list = []
        doc = nlp(" ".join(sentence)) # Unir as palavras da frase em uma
única string
        for token in doc:
            if token.lemma_ != '-PRON-':
                if token.pos_ == 'VERB':
                    palavra_lematizada = token.lemma_
                else:
                    palavra_lematizada = token.lemma_
                if palavra_lematizada:
                    lemma_list.append(palavra_lematizada)
        lemmatized_tokens.append(lemma_list)
    # Converter todas as palavras para minúsculas
    lemmatized_tokens_lower = []
    for sentence in lemmatized_tokens:
        sentence_lower = [palavra.lower() for palavra in sentence]
        lemmatized_tokens_lower.append(sentence_lower)
    return lemmatized_tokens_lower

```

### 8.3.8 Pipeline

No pipeline foi dividido cada uma das funções em células separadas e depois é executado todas na ordem correta. Essa etapa permite que as funções sejam executadas na ordem correta, garantindo a consistência e a precisão dos resultados, e

caso a ordem precise mudar, é mais simples fazer a alteração, essa organização torna o processo mais simples de entender e escalável.

Na parte de definição de funções, foi definida as funções que serão usadas no pipeline. As funções em questão são: `tokenizer()`; `demojize_tokens()`; `removendo_alfanumericos()`; `comentarios_normalizados()`; `remove_stopwords()`; `lematizacao()`, por fim, a função `pipeline()` executa cada uma das funções em ordem. Como as funções já foram apresentadas anteriormente, a seguir será mostrada a função:

```
def pipeline(comment):
    # Tokenização
    tokens = tokenizer(comment)
    # Tratamento de Emojis
    demojized = demojize_tokens(tokens, emoji_dict)
    # Remoção dos alfanuméricos
    no_alfanumericos = removendo_alfanumericos(tokens)
    # Normalização das abreviações
    normalizado = comentarios_normalizados(no_alfanumericos,
dicionario_girias)
    # Remoção das stopwords
    no_stopwords = remove_stopwords(normalizado)
    # Lematização
    tratados = lematizacao(no_stopwords)

    return tratados
```

Por fim, foram realizados alguns testes de função para garantir que o fluxo do pipeline estava operando adequadamente, para isso, foi criado um novo dataframe com uma coluna chamada “pós\_tratamento”, na qual está o resultado de todos os textos após passar pela função `pipeline()`.

```
df['pós_tratamento'] = df['texto'].apply(pipeline)
```

## 8.4 Conclusão

O pré-processamento dos dados é fundamental para garantir a qualidade e a confiabilidade das análises posteriores, contribuindo para um melhor entendimento dos dados e para a obtenção de resultados mais precisos e significativos.

## 9. Vetorização

### 9.1 Bag of Words

#### 9.1.1 Introdução

O modelo Bag of Words é uma das várias ferramentas de vetorização de frases e palavras, processo que é de suma importância para o desenvolvimento de um modelo PLN, visto que o modelo de machine learning só pode receber números como inputs.

#### 9.1.2 Método

Como última etapa de manipulação de dados antes do uso do modelo de Machine Learning para a classificação de resultados temos a vetorização dos comentários, processo que nesse caso, foi conduzido pelo modelo Bag of Words (BoW). O modelo BoW consiste na elaboração de uma matriz a partir de um vocabulário de todos os vocábulos presentes nos textos, então cada linha será um comentário que se deseja vetorizar. É importante notar que esse modelo é menos robusto, considerando apenas a frequência de palavras em cada frase e não os sentidos semânticos.

Para essa etapa, foi utilizada uma instância da classe `CountVectorizer()`, e seus métodos, da biblioteca `sklearn` (`scikit-learn`) a fim de que fosse gerado um vocabulário e as respectivas correspondências para cada comentário.

#### 9.1.3 Resultados

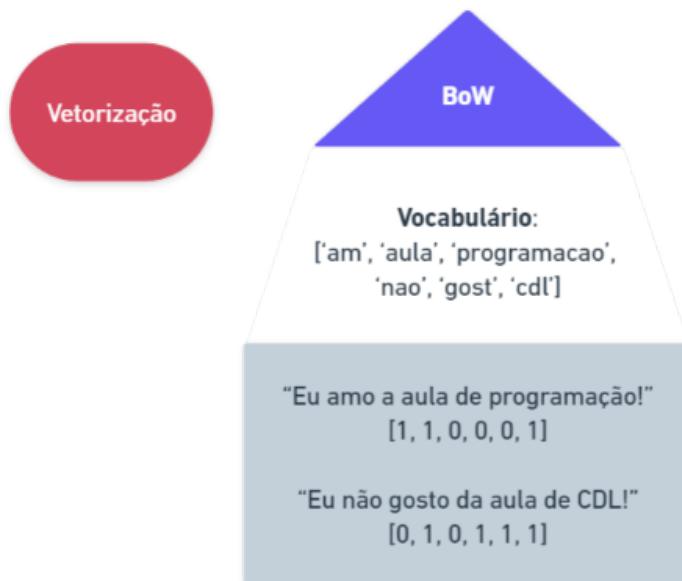


Figura 24: Demonstração do Bag of Words

Após o *corpus* dos textos terem passado pelo *pipeline*, chega o momento de analisar as repetições de acordo com cada comentário feito, por meio da técnica Bag of Words (BoW) utilizada em processamento de linguagem natural (PLN). Essa técnica é

utilizada para representar um texto como um conjunto de palavras desordenadas, ignorando a ordem e a estrutura gramatical das frases. Nesse modelo, cada palavra única do texto é transformada em uma *feature* (característica), e a frequência de cada palavra no texto é usada como um valor numérico para a *feature* correspondente.

Por exemplo, a frase "O gato preto pulou o muro" seria representada como um conjunto de palavras desordenadas: 'o', 'gato', 'preto', 'pulou', 'o', 'muro'. A frequência de cada palavra é contada, e o resultado é um vetor numérico que representa a frequência de cada palavra na frase. Assim, abaixo é possível visualizar o código necessário para realizar essa vetorização e o *output* dele:

```
def bow(frases):
    vectorizer = CountVectorizer()
    frases_concatenadas = [' '.join(tokens) for tokens in frases]
    bow_model = vectorizer.fit_transform(frases_concatenadas)
    dicionario = vectorizer.vocabulary_
    return bow_model, dicionario

bow_model, dicionario = bow(df['texto_tratado'].tolist())
```

	0	000	00000	0000000	0001	001	002	004	005	0050	...	tudo	uma	vai	valores	varias	volatilidade	120	2	20	30
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12188	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
12189	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
12190	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
12191	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
12192	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

12193 rows x 24331 columns

Figura 25: *Output* do código

Abaixo é demonstrado um exemplo resultante desta tabela, a qual possui um total de 12.193 linhas, que estão de acordo com cada comentário do csv disponibilizado pelo cliente, além de 24.331 colunas, que foram as palavras chaves selecionadas.

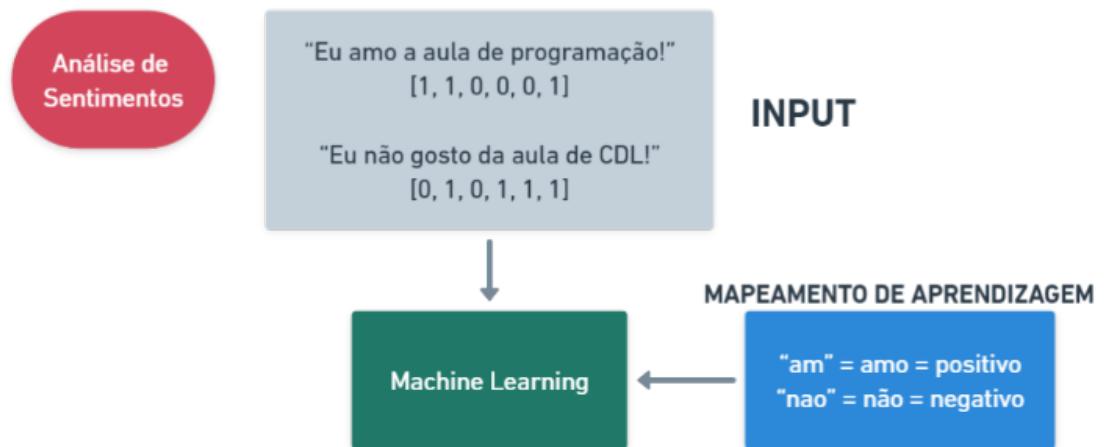
```
df['conf'].value_counts()
0      11795
1       396
2        2
Name: conf, dtype: int64
```

Neste exemplo, é possível perceber que o termo 'conf' se repete uma vez, em 396 comentários diferentes, e se repete duas vezes em 2 comentários diferentes.

Dessa forma, percebe-se como a função consegue selecionar palavras chaves que estão contidas nas diversas frases do dataframe.

#### 9.1.4 Conclusão

Com a aplicação do Modelo Bag of Words (BoW) é possível perceber a capacidade de seleção de palavras para a futura implementação na Machine Learning desenvolvida. O modelo é uma técnica simples e eficiente para representar textos em formato vetorial, o que permite utilizá-los em algoritmos de aprendizado de máquina. O objetivo do projeto é demonstrado a partir da imagem abaixo:



TEXTO	eu	am	nao	gost	aula	cdl	program	positivo	negativo	AVALIAÇÃO
"Eu amo a aula de programação!"	1	1	0	0	1	0	1	1	0	positivo
"Eu não gosto da aula de CDL!"	1	0	1	1	1	1	0	0	1	negativo

Figura 26: Demonstração do modelo pronto

Porém, é possível analisar que a necessidade uma renovação no tratamento dos dados e exclusão de determinadas palavras, já que foi percebido que havia uma alta diversidade de termos que estão exclusos e/ou outros que permanecerão nas frases e não deveriam permanecer. Abaixo há exemplo desta análise:

```
word_counts = df.sum()
top_words = word_counts.sort_values(ascending=False)
top_10 = top_words.head(10)
top_10

btgpactual      6489
invest          4014
btg             2822
tod             1783
banc            1771
```

```
sobr          1364  
melhor        1363  
cont          1332  
merc          1305  
financeir     1303  
dtype: int64
```

Além disso, foi feita uma plotagem de uma nuvem de palavras para ser mais intuitiva a visualização dos termos que serão necessários passar por um tratamento.



Figura 27: Nuvem de palavras

Assim, o próximo passo é um retratamento dos textos para ter melhor desenvolvimento e aplicação no momento de construção da Inteligência Artificial.

## 9.2 Word2Vec

### 9.2.1 Introdução

O modelo de vetorização Word2Vec, ao contrário do Bag Of Words, permite um entendimento das conexões semânticas das palavras, sendo assim, palavras podem ser analisadas a partir de sua similaridade com outras através da ocorrência contextual. Isso é possível porque o modelo relaciona as palavras a um espaço dimensional, representadas por um vetor denso, onde palavras com similaridade maior tendem a se agrupar no mesmo espaço. O modelo, através do uso de uma rede neural embutida, aprende por treinamento em *big corpus*, como o *Wikipédia*, a localização ideal para cada palavra.

### 9.2.2 Método

No projeto é utilizado dois métodos de vetorização com o Word2Vec: 1. o modelo pré-treinado de *Continuous Bag-of-Words* (CBOW) da biblioteca NILC e, posteriormente, 2. treino com o próprio corpus. O modelo pré-treinado, como dito anteriormente, não necessita de treinamento adicional, pois ele já dominou as relações linguísticas entre as palavras a partir do primeiro treinamento, além disso, o modelo usado é público e permite o uso quase instantâneo no corpus.

### 9.2.3 Resultados

Para ambos os métodos, é necessário realizar a soma dos vetores de palavras presentes nas frases, desse modo, cada frase terá um vetor. No método pré-treinado, é construído vetores de 50 dimensões, enquanto que, no modelo treinado no corpus da base de dados de comentários, é construído vetores de 100 dimensões.

	Frase	Vetor1	Vetor2	Vetor3	Vetor4	Vetor5	Vetor6	Vetor7	Vetor8	Vetor9	...	Vetor42	Vetor43	Vetor44	Vetor45	Vetor46	Vetor47	Vetor48	Vetor49	Vetor50
0	[alvarez, marsal, estar, conosco, sportfotanet, ...]	0.213634	-0.129877	0.241601	-0.075002	-0.015629	0.206194	0.072658	0.055472	0.061554	...	0.024361	-0.111328	0.157674	0.094309	-0.047458	0.157365	-0.033920	0.022211	0.182153
1	[blgpactual, with, makerepost, entender, impac...]	0.222697	-0.124886	0.213157	-0.059091	-0.010530	0.201566	0.071898	0.033920	0.059524	...	0.008988	-0.079109	0.159296	0.085387	-0.008607	0.158519	-0.022680	0.031107	0.189521
2	[minuto, touro, ouro]	0.265227	-0.068285	0.152235	-0.044329	-0.102729	0.141353	0.092800	0.113174	0.015783	...	0.078032	-0.202677	0.155750	0.062291	0.007038	0.134573	0.014635	0.034189	0.345674
3	[sim]	0.166258	-0.029796	0.204045	-0.297490	0.046077	0.140763	0.035251	-0.174491	0.211817	...	0.065839	-0.092451	0.308218	-0.034692	-0.032851	-0.028724	-0.068701	0.011158	0.258413
4	[querer, saber, banking, próprio, administro]	0.187512	-0.183612	0.300155	-0.052422	-0.034717	0.232278	0.058778	0.084289	0.068006	...	0.097538	-0.161461	0.196748	0.088577	-0.080884	0.167507	-0.049984	-0.000942	0.187811
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9202	[exceLENte, explicaçõE]	0.190917	-0.133475	0.241675	-0.053180	0.067256	0.201138	0.034109	-0.078718	-0.066131	...	-0.082151	0.016113	0.154861	0.068700	-0.004302	0.079717	-0.028388	-0.017448	0.188785
9203	[atender, telefone, amor, deus]	0.188641	-0.119377	0.199339	-0.105448	0.023176	0.178837	0.069476	-0.004494	0.034710	...	0.034035	-0.126673	0.165176	0.080313	-0.024160	0.118848	-0.003502	0.087053	0.215656

Figura 28: Word2Vec com modelo pré-treinado

	texto_tratado	sentimentoNumerico	Vetor0	Vetor1	Vetor2	Vetor3	Vetor4	Vetor5	Vetor6	Vetor7	...	Vetor90	Vetor91	Vetor92	Vetor93	Vetor94	Vetor95	Vetor96	Vetor97	Vetor98	Vetor99
0	[alvarez, marsal, estar, conosco, sportinmet, ...]	1	-0.034426	0.300972	-0.286190	-0.065209	-0.054679	0.096039	-0.256592	0.127671	...	-0.068513	-0.155436	0.140325	0.219084	-0.289465	-0.035990	0.078490	-0.070072	-0.003752	0.339329
1	[bigactual, with, makerepost, entender, impac...]	1	-0.037321	0.289959	-0.259288	-0.064339	-0.080668	0.098687	-0.287177	0.113840	...	-0.060072	-0.149105	0.130588	0.186867	-0.302065	-0.018805	0.059356	-0.054743	-0.009448	0.354135
2	[minuto, touro, ouro]	2	-0.193654	0.391434	-0.203922	0.001809	-0.143087	0.069520	-0.235220	0.141716	...	-0.066392	-0.108025	0.141333	0.167520	-0.338259	0.087925	0.089112	-0.092600	0.032975	0.329287
3	[sim]	1	0.082540	0.359721	-0.090063	0.209893	0.271785	-0.004673	-0.347171	0.258687	...	-0.064920	-0.291844	0.273709	0.119920	-0.276886	-0.050655	0.175334	-0.094184	-0.075605	0.116921
4	[querer, saber, banking, próprio, administro]	2	-0.076144	0.319210	0.289153	-0.018826	-0.025748	0.047773	-0.273074	0.146078	...	-0.103470	-0.161082	0.136264	0.172653	-0.259784	0.001959	0.127059	-0.129546	0.007025	0.357119
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9202	[excelente, explicação]	2	0.013271	0.233460	-0.269739	-0.228910	-0.098336	0.027902	-0.262189	0.145487	...	0.048463	-0.198648	0.078566	0.267573	-0.418458	-0.069492	0.134299	-0.233718	0.027100	0.381566
9203	[agenda, telefone, amor, deus]	2	-0.097030	0.357497	-0.289556	-0.076197	-0.054787	0.091044	-0.288496	0.154998	...	-0.089239	-0.201867	0.149533	0.174404	-0.306654	-0.032587	0.117704	-0.042513	0.026960	0.443921
9204	[saber, qual, grande, fis, mercado, selecione...]	2	-0.047819	0.247797	-0.269047	-0.119334	-0.083395	0.097093	-0.253981	0.148540	...	-0.031900	-0.186846	0.143561	0.254065	-0.343297	-0.009860	0.068257	-0.120914	0.022000	0.399099
9205	[erro, financeiro, eliminar, antes, ano, __, pa...]	1	-0.041982	0.288899	-0.303314	-0.088639	-0.096838	0.103386	-0.263547	0.119052	...	-0.067419	-0.155152	0.124977	0.252822	-0.333116	-0.019388	0.046955	-0.103713	0.002978	0.393447
9206	[porque, morning, call, aparecer, spotify, atu...]	0	-0.091172	0.372426	-0.301286	-0.091199	-0.111440	0.057613	-0.234171	0.124101	...	-0.070419	-0.142520	0.111241	0.168357	-0.301025	-0.012969	0.109775	-0.063298	0.005821	0.431900

Figura 29: Word2Vec treinado com o corpus

## 9.2.4 Conclusão

A conclusão obtida de qual será o melhor método de vetorização se dará na construção dos modelos de machine learning que será explicado na próxima sessão, pois assim poderemos comparar diretamente a acurácia e recall dos diferentes modelos.

## 9.3 TF - IDF

### 9.3.1 Introdução

O TF-IDF (Term Frequency-Inverse Document Frequency) é uma técnica que permite avaliar a importância relativa de um termo em um documento dentro de um conjunto de documentos. O modelo é composto por duas partes principais: a frequência do termo (TF) e a frequência inversa do documento (IDF). A frequência do termo mede quantas vezes um termo específico aparece em um documento, enquanto a frequência inversa do documento mede a raridade do termo em toda a coleção de documentos.

### 9.3.2 Método

O código que será apresentado abaixo utiliza o método `TfidfVectorizer()` da biblioteca `scikit-learn` (`sklearn`) para calcular o TF-IDF dos documentos presentes na coluna 'texto\_tratado' de um `dataframe` chamado '`df`'. E no final, o `dataframe` '`df_final`' conterá todas as colunas do `dataframe` original, além das colunas correspondentes às pontuações TF-IDF de cada termo nos documentos.

### 9.3.3 Resultados

A função `pd.read_csv()` é uma função da biblioteca `pandas` (`pd`) que permite ler dados de um arquivo CSV e retorná-los como um `dataframe`. Após a execução do código, ao imprimir o '`df`', será mostrada uma representação tabular dos dados contidos no arquivo.

```
df = pd.read_csv('caminho_arquivo')
df
```

A primeira linha cria um objeto `TfidfVectorizer()`, que é uma classe disponível na biblioteca `scikit-learn`, responsável por transformar textos em uma representação numérica usando o cálculo do TF-IDF. Em seguida, o método `fit_transform()` é aplicado aos comentários, que já passaram pelo pré - processamento, presente na coluna 'texto\_tratado' do 'df'. Isso transforma os documentos em uma matriz numérica esparsa, onde cada linha representa um documento e cada coluna representa um termo ponderado pelo TF-IDF.

```
tfidf_vectorizer = TfidfVectorizer()  
  
vetorizado = tfidf_vectorizer.fit_transform(df['texto_tratado'])
```

Após a vetorização, a variável 'feature\_names' armazena os termos que foram utilizados na vetorização, que serão colunas do *dataframe* resultante. A seguir, a matriz numérica esparsa resultante é convertida em um *dataframe* chamado 'df\_vetorizado', onde cada coluna corresponde a um termo e cada linha representa um documento. Os valores são preenchidos com as pontuações TF-IDF.

```
feature_names = tfidf_vectorizer.get_feature_names_out()  
  
df_vetorizado = pd.DataFrame(vetorizado.toarray(),  
columns=feature_names)
```

Por último, o 'df' é concatenado com o *dataframe* resultante da vetorização 'df\_vetorizado' ao longo do eixo das colunas (`axis=1`), utilizando a função `concat()` da biblioteca pandas. Isso adiciona as colunas com as pontuações TF-IDF ao *dataframe* original, criando assim o *dataframe* final, chamado de 'df\_final'.

```
df_final = pd.concat([df, df_vetorizado], axis=1)  
  
df_final
```

Para fins de teste, o método `value_counts()` é aplicado para que retorne uma contagem dos valores únicos presentes na coluna especificada. Ele conta quantas vezes cada valor aparece na coluna e retorna os resultados em ordem decrescente, com o valor mais frequente no topo. Somente algumas colunas foram testadas, e as colunas vão ser apresentadas abaixo.

```
df_final['ser'].value_counts()  
  
output:  
0.000000    8014
```

```
0.136855      1
0.089819      1
0.178045      1
0.162049      1
0.147501      1
0.176240      1
0.090406      1
0.073729      1
0.264882      1
0.436092      1
0.064391      1
0.145558      1
0.133492      1
0.405338      1
0.190045      1
0.169664      1
0.314675      1
0.081824      1
0.149980      1
0.066284      1
0.177696      1
0.143129      1
0.478244      1
0.164816      1
0.167999      1
0.120388      1
Name: ser, dtype: int64
```

```
df_final['aa'].value_counts()
```

```
output:
0.000000    8039
0.215284      1
Name: aa, dtype: int64
```

```
df_final['seus'].value_counts()
```

```
output:
0.000000    8039
0.131107      1
Name: seus, dtype: int64
```

### 9.3.4 Conclusão

O uso do TF-IDF em conjunto com técnicas de vetorização e manipulação de dados, como apresentado no código, é uma ferramenta valiosa para processamento de texto e análise de dados, fornecendo insights sobre a importância relativa dos termos em um conjunto de documentos e permitindo uma melhor compreensão e interpretação dos textos.

## 9.4 Comparação entre os modelos de vetorização

Para ser possível comparar os 3 modelos de vetorização desenvolvidos pelo grupo, foram adotados as seguintes métricas: Representação vetorial, tamanho de representação, semântica do contexto e flexibilidade. As características finais de cada modelo e as comparações entre eles estão apresentadas abaixo.

### 9.4.1 Representação Vetorial

A representação vetorial é uma técnica muito utilizada em PLN para facilitar o processamento e análise de texto. O objetivo é converter as informações textuais em representações numéricas, onde palavras e/ou documentos são representados por vetores em um espaço de alta dimensionalidade.

#### Modelos e suas características:

**BoW** cria uma matriz que representa a frequência de ocorrência de cada palavra em um documento ou corpus, então cada documento é representado como um vetor de tamanho fixo. Nesse caso, a dimensão corresponde a uma palavra e o valor representa a frequência.

**Word2Vec** mapeia palavras em vetores densos de tamanho fixo, também conhecidos como embeddings. Esses vetores capturam relações semânticas entre as palavras, permitindo operações matemáticas como soma e subtração.

**TF-IDF** atribui um valor numérico a cada palavra em um documento com base na frequência de termos (TF) e na frequência inversa do documento (IDF). O resultado é uma representação ponderada, onde palavras importantes recebem maior destaque.

A tabela 07 demonstra as características dos 3 modelos:

#	Modelo	Característica
1	BoW	Frequência de ocorrência
2	BoW	Vetor de tamanho fixo
3	Word2Vec	Vetores densos de tamanho fixo - Embedding
4	Word2Vec	Leva a semântica em consideração

5	TF-IDF	Frequência de termos (TF)
6	TF-IDF	Frequência inversa do documento (IDF)
7	TF-IDF	Resultado: representação ponderada

Tabela 07: Características de 3 modelos - Representação Vetorial

#### 9.4.2 Tamanho da Representação

O tamanho da representação refere-se ao número de dimensões ou características que compõem cada vetor na representação vetorial.

##### Modelos e suas características:

O tamanho da representação do **BoW** é determinado pelo tamanho do vocabulário utilizado. Quanto maior o vocabulário, maior será a dimensionalidade do vetor, por isso para vocabulário extensos, talvez não seja a melhor opção.

O tamanho da representação do **Word2Vec** é pré-definido e geralmente varia de 50 a 300 dimensões, independentemente do tamanho do vocabulário. Nesse projeto foi utilizado de 50 dimensões.

O tamanho da representação do **TF-IDF** é igual ao tamanho do vocabulário utilizado. Mesmo caso do BoW, então dependendo do tamanho do vocabulário, não é a melhor opção.

A tabela 08 demonstra as características dos 3 modelos:

#	Modelo	Característica	Projeto
1	BoW	Depende do vocabulário	Não se adapta tão bem por conta do tamanho do <i>dataset</i>
2	Word2Vec	Tamanho pré definido	Se adapta melhor por já ter um tamanho pré definido
3	TF-IDF	Depende do vocabulário	Não se adapta tão bem por conta do tamanho do <i>dataset</i>

Tabela 08: Características de 3 modelos - Tamanho da Representação

#### 9.4.3 Semântica e Contexto

Semântica e contexto são elementos de grande relevância no processamento de linguagem natural e na compreensão do significado das palavras e seu uso em diferentes contextos.

##### Modelos e suas características:

O **BoW** não captura a semântica ou o contexto das palavras, pois trata cada palavra como um elemento independente. O que para casos de análise de

sentimento, quando aplicado em algum modelo, o resultado não fica muito satisfatório por não levar o contexto em consideração.

O **Word2Vec** captura relações semânticas entre palavras, permitindo inferências como "rei - homem + mulher = rainha". Nessa vetorização, a análise de sentimento obtém um resultado muito satisfatório, quando aplicado em um modelo, já que o contexto é levado em consideração.

O **TF-IDF** não captura semântica ou contexto, pois é baseado em estatísticas de frequência. Segue o mesmo raciocínio do Bow.

A tabela 09 demonstra as características dos 3 modelos:

#	Modelo	Característica	Projeto
1	BoW	Não captura a semântica e contexto	Quando aplicado em um modelo, o resultado não é satisfatório
2	Word2Vec	Captura a semântica e contexto	Quando aplicado em um modelo, o resultado é satisfatório
3	TF-IDF	Não captura a semântica e contexto, somente frequência	Quando aplicado em um modelo, o resultado não é satisfatório, mas tem um resultado melhor que o BoW

Tabela 09: Características de 3 modelos - Semântica e Contexto

#### **9.4.4 Flexibilidade**

A flexibilidade refere-se à capacidade de adaptar ou ajustar uma técnica de representação vetorial.

##### Modelos e suas características:

O **BoW** é relativamente simples e fácil de implementar. É eficiente em termos de tempo de treinamento e adequado para tarefas básicas de classificação de texto. Apesar do tempo de treinamento ser baixo, que é um ponto positivo, esse projeto realiza tarefas de classificação de texto mais complexas, então o BoW pode não ser a melhor escolha.

O **Word2Vec** requer mais tempo de treinamento e uma quantidade maior de dados, mas é mais flexível em termos de capturar relações semânticas e pode ser usado para tarefas como agrupamento de palavras e busca por similaridade. Apesar do tempo de treinamento ser alto, a quantidade maior de dados é algo que o banco disponibilizou e as relações semânticas são importantes para essa classificação de texto.

O **TF-IDF** é simples de implementar e rápido de calcular. É adequado para tarefas de classificação de texto e recuperação de informações. Mesma situação do BoW, apesar de ter alguns pontos positivos, a classificação do projeto é mais complexa e precisa de um modelo mais complexo.

A tabela 10 demonstra as características dos 3 modelos:

#	Modelo	Característica	Projeto
1	BoW	Fácil de implementar	Ponto positivo para o projeto, dado o que seja um projeto acadêmico.
2	BoW	Tempo de treinamento baixo	Ponto positivo para o projeto.
3	BoW	Adequado para tarefas simples	Ponto negativo para o projeto, já que há uma certa dificuldade na análise de sentimento.
4	Word2Vec	Tempo maior de treinamento.	Ponto negativo para o projeto.
5	Word2Vec	Precisa de uma quantidade maior de dados.	Não é um problema, já que foi disponibilizado um <i>dataset</i> relativamente grande.
6	TF-IDF	Fácil de implementar	Ponto positivo para o projeto, dado o que seja um projeto acadêmico.
7	TF-IDF	Tempo de treinamento baixo	Ponto positivo para o projeto.

Tabela 10: Características de 3 modelos - Flexibilidade

#### 9.4.5 Uso de contexto local

O uso de contexto local refere-se à consideração do contexto imediato de uma palavra dentro de um espaço definido ao redor dessa palavra.

##### Modelos e suas características:

O **BoW** não considera a ordem das palavras em um documento, perdendo informações de contexto local. Ponto negativo para a classificação de sentimento, já que a ordem importa.

O **Word2Vec** leva em consideração o contexto local, pois palavras similares aparecem próximas umas das outras nos vetores de palavras. Ponto positivo para a classificação de sentimento.

O **TF-IDF** também não considera a ordem das palavras em um documento, assim como o BoW.

A tabela 11 demonstra as características dos 3 modelos:

#	Modelo	Característica	Projeto
1	BoW	Sem consideração da ordem e contexto local	Ponto negativo, pois a ordem importa.
2	Word2Vec	Leva em consideração a ordem e o contexto.	Ponto positivo para o projeto.
3	TF-IDF	Sem consideração da ordem e contexto local	Ponto negativo, pois a ordem importa.

Tabela 11: Características de 3 modelos - Uso de contexto local

## 10. Modelos

Abaixo serão apresentados os melhores modelos, o resto dos modelos que foram desenvolvidos estão no tópico Anexos deste documento.

### 10.1 Naive Bayes + Bow

#### 10.1.1 Introdução

Esse modelo junta o algoritmo Naive Bayes com um modelo de vetorização, já explicado, que é o BoW. Se espera que, com essa junção, os resultados sejam promissores nesta tarefa de classificação de texto. Todos os modelos que serão apresentados foram rodados com a base de dados com o pré-processamento da Sprint 3 e Sprint 4, por isso serão apresentados os resultados dos dois *dataframes*.

#### 10.1.2 Método

##### 10.1.2.1 Cross Validation

*Cross validation* é uma técnica usada para avaliar a capacidade de um modelo de generalizar para novos dados, que consiste em dividir o conjunto de dados em partes menores, treinar o modelo em uma parte e testá-lo em outra. Esse processo é repetido várias vezes e a média das métricas de avaliação é usada para avaliar o desempenho do modelo.

##### 10.1.2.2 Grid Search

*Grid search* é uma técnica de busca de hiperparâmetros usada para encontrar a melhor combinação de valores para um modelo de aprendizado de

máquina, que consiste em definir um conjunto de valores para cada hiperparâmetro e treinar e avaliar o modelo com todas as combinações possíveis. O conjunto de hiperparâmetros que produz a melhor métrica de avaliação é selecionado como a configuração final do modelo.

### 10.1.3 Resultados

#### 10.1.3.1 Naive Bayes

Primeiramente, o objeto `LabelEncoder()` é criado, a fim de transformar as classes de texto em números inteiros, já que o *Naive Bayes* somente trabalha com valores numéricos. O encoder é ajustado nos dados da coluna `sentimento` do `df`, e as classes são transformadas em números inteiros usando o método `fit_transform`, armazenando-os na variável `sentimento`. Em seguida, os dados são divididos em conjuntos de treino e teste usando a função `train_test_split`, os conjuntos de treino (`X_treino` e `y_treino`) e teste (`X_teste` e `y_teste`) são criados a partir dos dados do `bow_model` e dos rótulos transformados, com 20% dos dados destinados ao conjunto de teste e o restante para o conjunto de treino.

```
encoder = LabelEncoder()

sentimento = encoder.fit_transform(df['sentimento'])

X_treino, X_teste, y_treino, y_teste =
train_test_split(bow_model, sentimento, test_size=0.2,
random_state=42)
```

Após a divisão dos dados, o objeto do tipo `MultinomialNB()` é criado, e o modelo é treinado usando o conjunto de treino através do método `fit`, passando as matrizes de treino e os rótulos correspondentes. A próxima etapa é fazer a predição usando os dados de teste, por isso o método `predict` é aplicado ao modelo treinado usando os dados de teste, gerando previsões numéricas para as classes. Estas são decodificadas para obter as classes originais usando o método `inverse_transform` e são armazenadas na variável `predicao`. Por fim, é impresso o relatório de classificação.

```
modelo = MultinomialNB()
modelo.fit(X_treino, y_treino)

predicao_numerica = modelo.predict(X_teste)

predicao = encoder.inverse_transform(predicao_numerica)
```

```
print(classification_report(df['sentimento'].iloc[y_teste], predicao))
```

Abaixo é possível observar o código necessário para criar a matriz de confusão, onde esta é calculada usando a função `confusion_matrix(df['sentimento'].iloc[y_teste], predicao)`, que recebe como parâmetros os rótulos verdadeiros (`df['sentimento'].iloc[y_teste]`) e as previsões feitas pelo modelo (`predicao`). Após calcular a matriz de confusão, uma lista chamada `classes` é definida, contendo os nomes das classes presentes no problema, que serão utilizados para rotular as classes na matriz de confusão. Em seguida, é criada uma figura de plotagem com as dimensões especificadas usando `plt.figure(figsize=(8, 6))`. A função `sns.heatmap()` é chamada para criar o mapa de calor, os parâmetros são: `cm` é que é a matriz de confusão, `annot=True` e `fmt='g'` utilizados para exibir os valores da matriz nas células, `cmap='Blues'` que define a paleta de cores a ser utilizada no mapa de calor, e por último, os rótulos dos eixos `x` e `y` são definidos com base na lista de `classes` usando os parâmetros `xticklabels` e `yticklabels`.

```
cm = confusion_matrix(df['sentimento'].iloc[y_teste], predicao)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g',
            xticklabels=classes, yticklabels=classes)
plt.xlabel('Classe Predita')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()
```

### **Relatório de Classificação - Sprint 3:**

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	0.88	0.46	0.61	1230
2	0.63	0.76	0.69	612
accuracy			0.56	1842
macro avg	0.50	<b>0.41</b>	0.43	1842
weighted avg	0.79	0.56	0.63	1842

### **Matriz de Confusão - Sprint 3:**

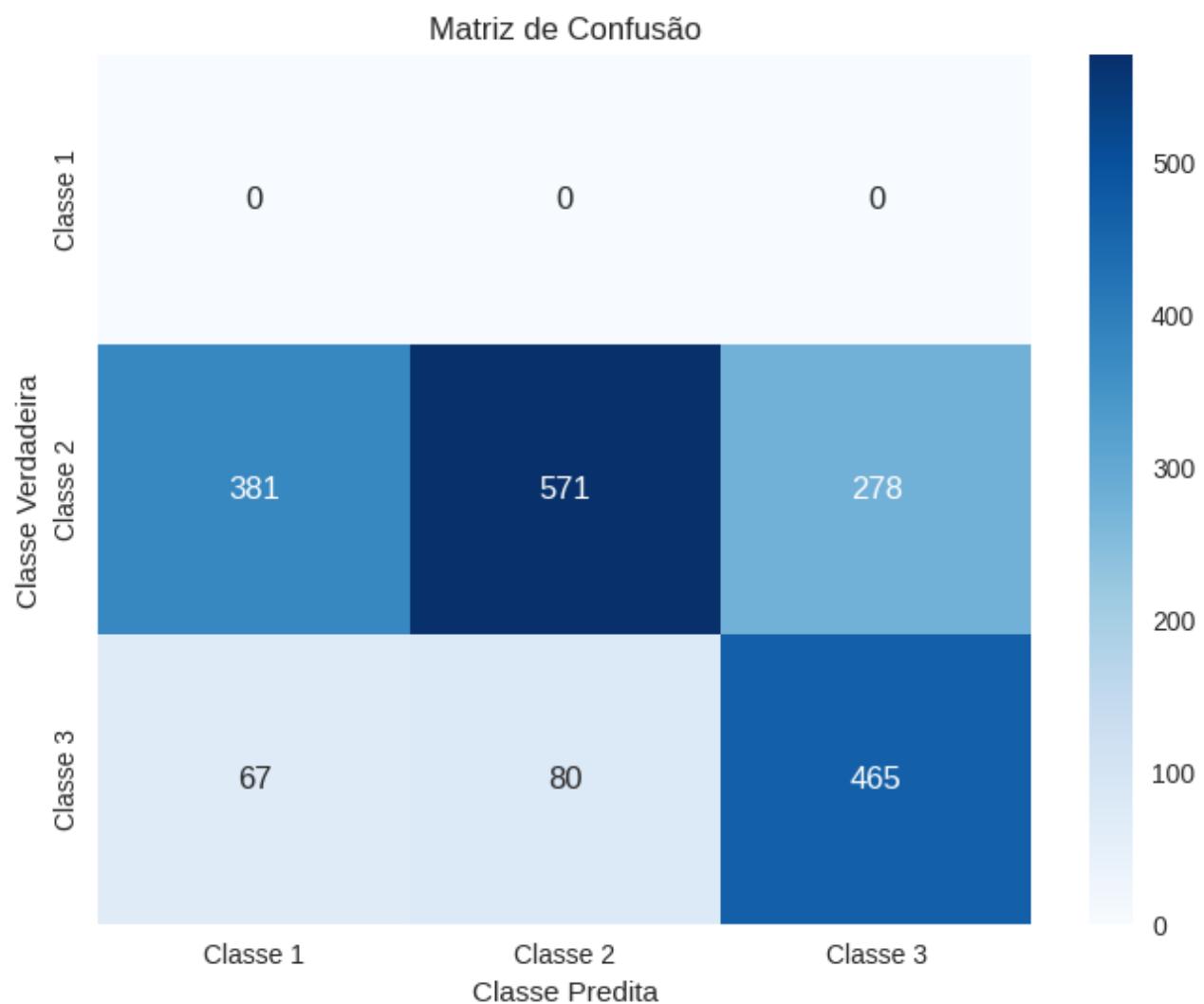


Figura 30: Matriz de confusão Naive Bayes - Sprint 3

#### **Relatório de Classificação - Sprint 4:**

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	0.85	0.30	0.45	957
2	0.66	0.79	0.72	651
accuracy			0.50	1608
macro avg	0.50	<b>0.36</b>	0.39	1608
weighted avg	0.77	0.50	0.56	1608

#### **Matriz de Confusão - Sprint 4:**

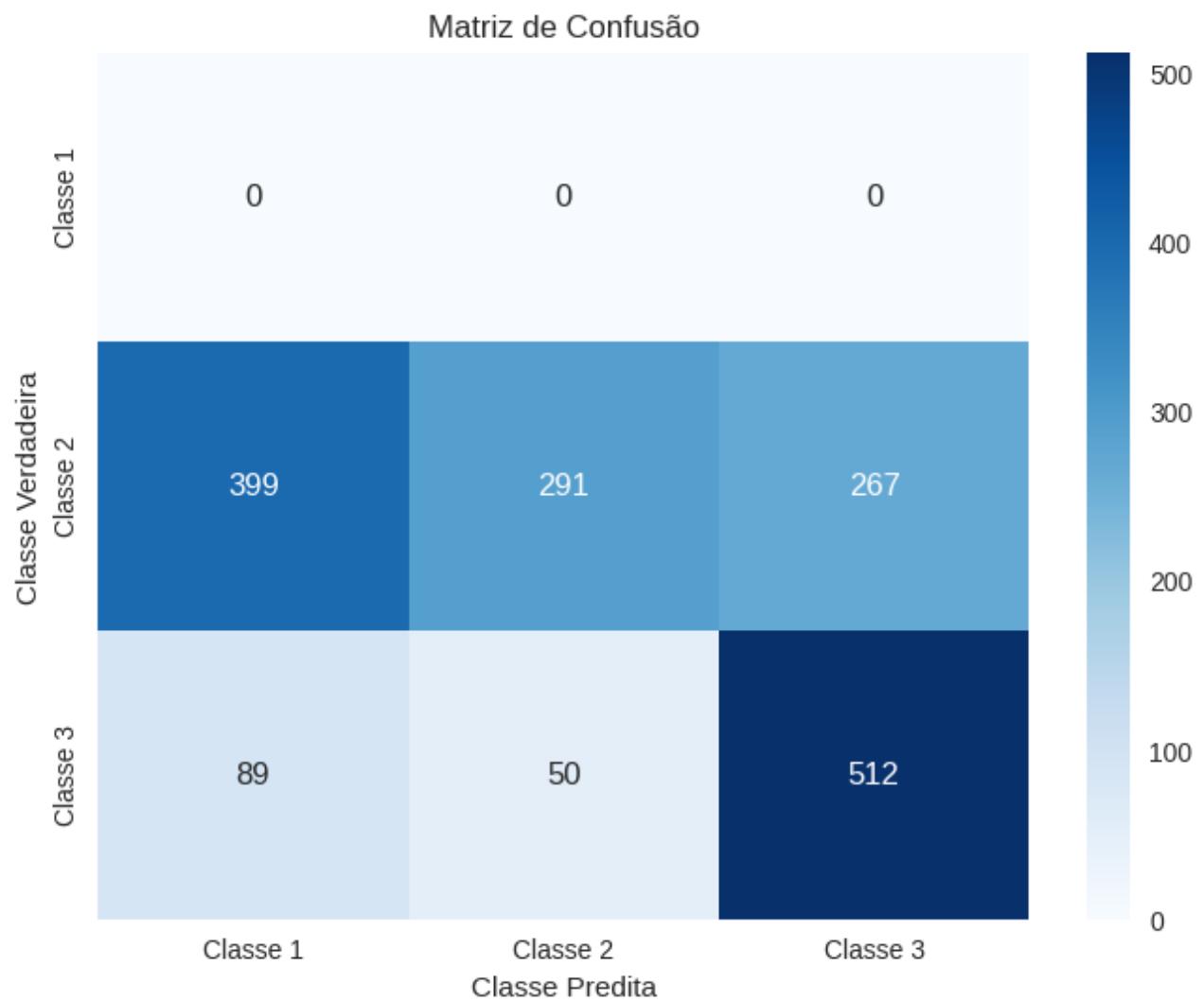


Figura 31: Matriz de confusão Naive Bayes - Sprint 4

#### 10.1.3.1 Naive Bayes - Cross Validation

Na primeira linha, o código utiliza a função `cross_val_score` para calcular as pontuações de *Cross Validation* do modelo, este é avaliado utilizando o conjunto de features `bow_model` e os rótulos de classe sentimento, o parâmetro `cv=7` especifica que a *cross validation* será realizada em 7 folds. A seguir, o código imprime a média das pontuações obtidas, utilizando a função `mean()` no objeto `scores`, que representa o desempenho médio do modelo em todos os folds.

```

scores = cross_val_score(modelo, bow_model, sentimento, cv=7)

print('Acurácia média:', scores.mean())

output:
Acurácia média: 0.6724311486588003 Sprint 3
Acurácia média: 0.6263681362502335 Sprint 4

```

Em seguida, o código utiliza a função cross\_val\_predict para fazer as previsões do modelo, os parâmetros chamados são os mesmos da função descrita no último parágrafo: modelo, bow\_model, sentimento, cv=7. Na segunda linha, é utilizado o objeto encoder para decodificar as classes preditas (predicoes) de volta aos seus valores originais. Por fim, o código imprime o relatório de classificação usando a função classification\_report.

```
predicoes = cross_val_predict(modelo, bow_model, sentimento,
cv=7)

predicao = encoder.inverse_transform(predicoes)

print('Relatório de Classificação:')
print(classification_report(df['sentimento'], predicao))
```

O código da Matriz de Confusão do Naive Bayes com Cross Validation é o mesmo do apresentado anteriormente.

### **Relatório de Classificação - Sprint 3:**

	precision	recall	f1-score	support
0	0.63	0.73	0.68	1974
1	0.78	0.58	0.67	4012
2	0.62	0.75	0.68	3221
accuracy			0.67	9207
macro avg	0.67	<b>0.69</b>	0.67	9207
weighted avg	0.69	0.67	0.67	9207

### **Matriz de Confusão - Sprint 3:**

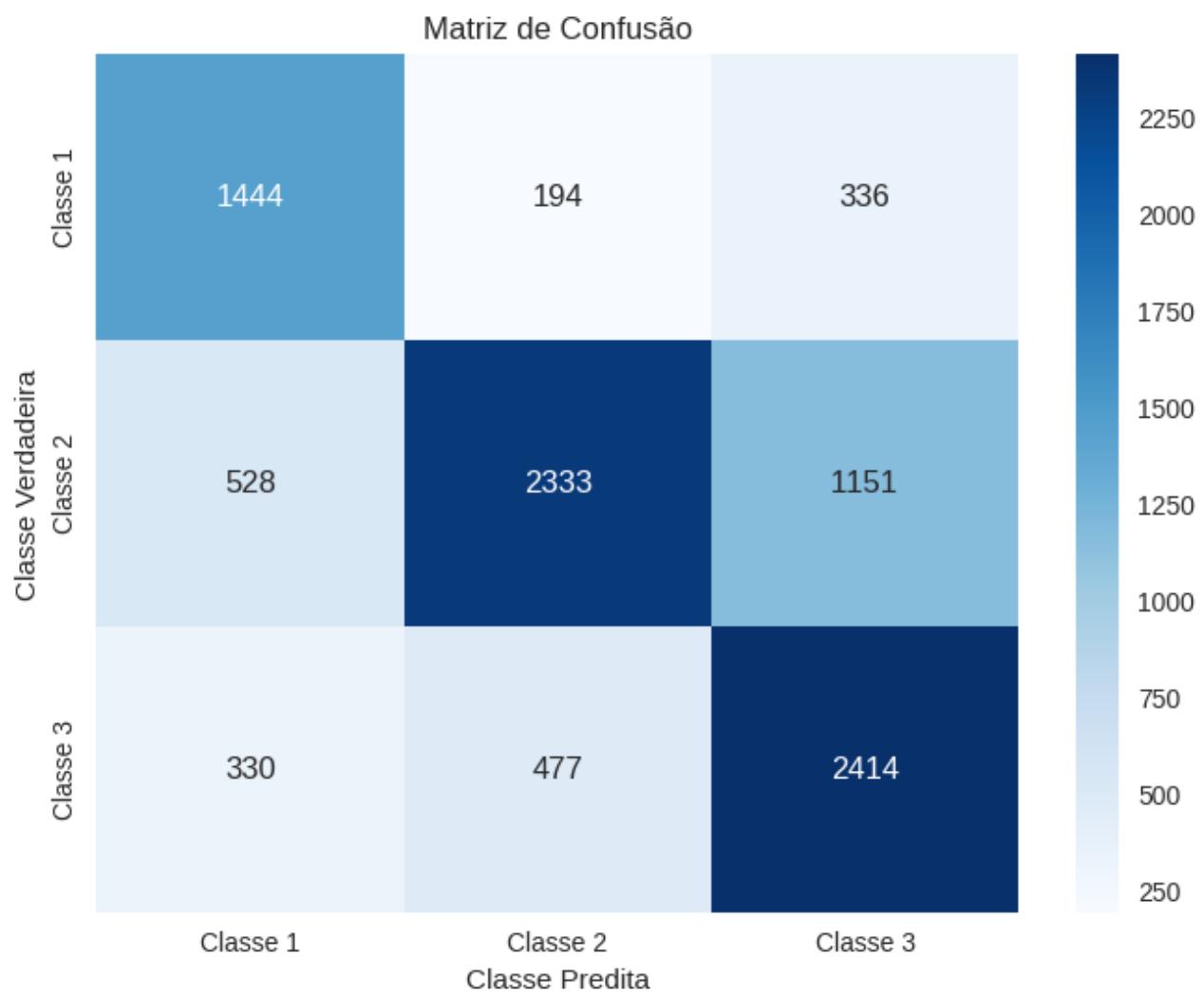


Figura 32: Matriz de confusão Naive Bayes Cross Validation - Sprint 3

#### **Relatório de Classificação - Sprint 4:**

	precision	recall	f1-score	support
0	0.58	0.75	0.65	1970
1	0.73	0.36	0.48	2918
2	0.62	0.79	0.70	3152
accuracy			0.63	8040
macro avg	0.64	0.64	0.61	8040
weighted avg	0.65	0.63	0.61	8040

#### **Matriz de Confusão - Sprint 4:**

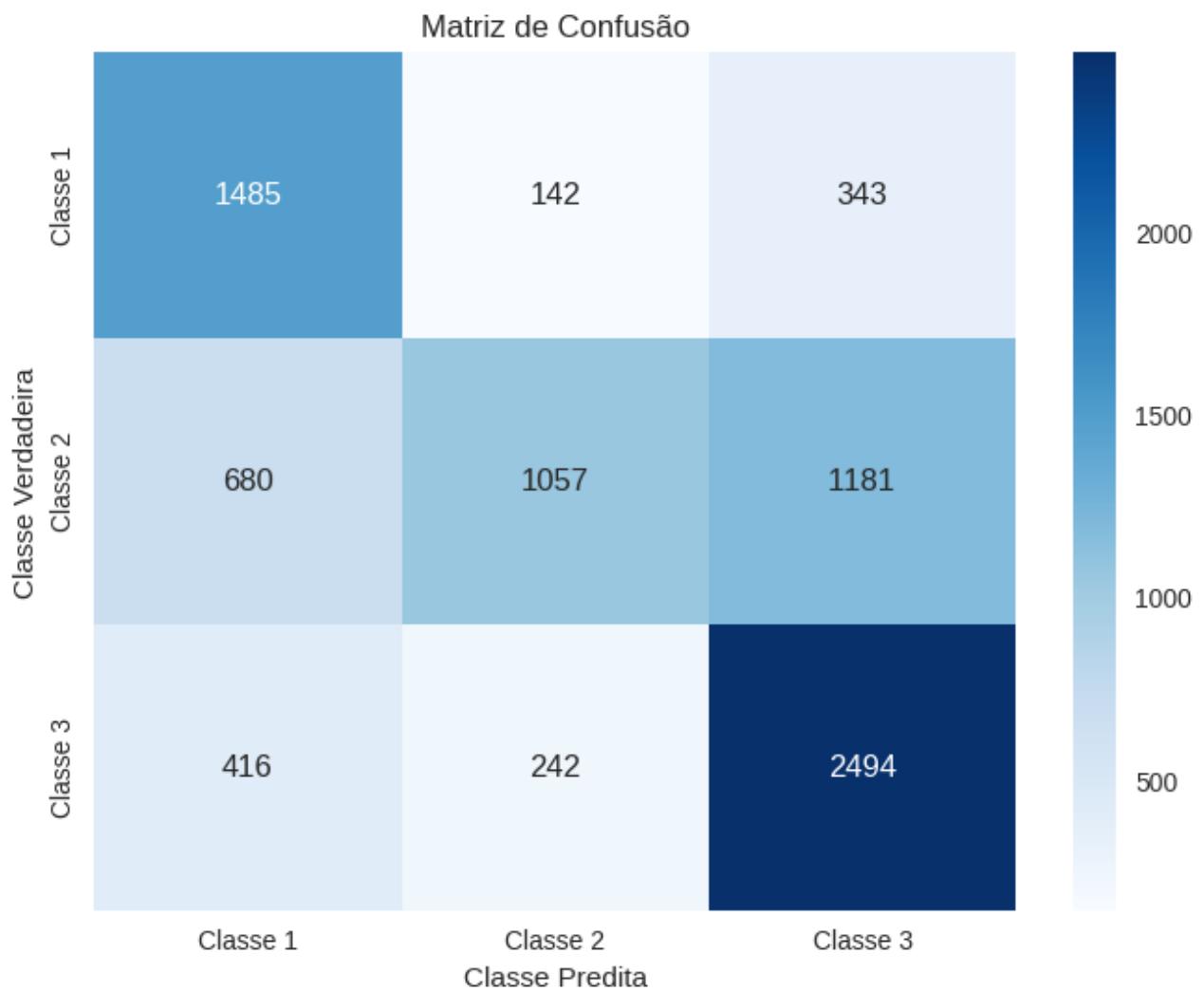


Figura 33: Matriz de confusão Naive Bayes Cross Validation - Sprint 4

#### 10.1.3.1 Naive Bayes - Grid Search

Na primeira linha, são definidos os valores a serem testados para os hiperparâmetros do modelo, e nesse caso, o hiperparâmetro `alpha` será testado com os valores `[0.1, 0.5, 1.0, 2.0, 5.0]`, e o hiperparâmetro `fit_prior` será testado com os valores `[True, False]`. Em seguida, na terceira linha, uma instância do modelo Naive Bayes Multinomial é criada utilizando a classe `MultinomialNB()`.

```
parametros = {'alpha': [0.1, 0.5, 1.0, 2.0, 5.0], 'fit_prior':
[True, False]}

modelo = MultinomialNB()
```

A seguir, é criada uma instância do objeto `GridSearchCV`, que é utilizado para realizar uma busca exaustiva dos melhores hiperparâmetros para o modelo, esse objeto recebe o modelo criado, os parâmetros a serem testados, o número de folds para a validação cruzada (`cv=5`) e a métrica de avaliação a ser utilizada

(scoring="accuracy"). Na próxima linha, o modelo é treinado através do método `fit()` do objeto `grid`, e os dados de treino (`X_treino` e `y_treino`) são utilizados para treinar o modelo e encontrar os melhores hiperparâmetros. Por último, os melhores hiperparâmetros são exibidos pelo atributo `best_params_` e a melhor acurácia com o atributo `best_score_`.

```
grid = GridSearchCV(modelo, parametros, cv=5,
                     scoring='accuracy')

grid.fit(X_treino, y_treino)

print('Melhores hiperparâmetros:', grid.best_params_)
print('Melhor acurácia:', grid.best_score_)

output:
Sprint 3:
Melhores hiperparâmetros: {'alpha': 0.5, 'fit_prior': True}
Melhor acurácia: 0.693550577053632
Sprint 4:
Melhores hiperparâmetros: {'alpha': 1.0, 'fit_prior': True}
Melhor acurácia: 0.6517396721129225
```

A seguir, um novo modelo é criado com os melhores hiperparâmetros encontrados. O `alpha` e o `fit_prior` são definidos de acordo com os valores ótimos encontrados pelo `GridSearchCV`, e esse novo modelo é treinado com os dados de treino. Após o treinamento, o modelo é utilizado para fazer a predição dos dados de teste (`X_teste`), que são retornadas como valores numéricos. Por meio do objeto `encoder`, as classes preditas numéricas são decodificadas.

```
modelo = MultinomialNB(alpha=grid.best_params_['alpha'],
                       fit_prior=grid.best_params_['fit_prior'])
modelo.fit(X_treino, y_treino)

predicao_numerica = modelo.predict(X_teste)

predicao = encoder.inverse_transform(predicao_numerica)
```

Por fim, a função `accuracy_score` é utilizada, e nela são passados dois argumentos: o primeiro argumento, `df['sentimento'].iloc[y_teste]`, refere-se às classes reais do conjunto de teste, e o segundo argumento é o `predicao`, que representa as classes preditas pelo modelo para o conjunto de teste. Por fim, o resultado da acurácia é exibido na tela utilizando a função `print`.

```
acuracia = accuracy_score(df['sentimento'].iloc[y_teste],  
predicao)  
print('Acurácia no conjunto de teste:', acuracia)
```

**Sprint 3:**

Acurácia no conjunto de teste: 0.5537459283387622

**Sprint 4:**

Acurácia no conjunto de teste: 0.4993781094527363

O próximo código calcula a revocação do modelo descrito. Primeiramente, é definida uma grade de valores para os hiperparâmetros a serem testados, que são os mesmos definidos anteriormente, em seguida, é criada uma instância do modelo MultinomialNB.

```
parametros = {'alpha': [0.1, 0.5, 1.0, 2.0, 5.0], 'fit_prior':  
[True, False]}  
  
modelo = MultinomialNB()
```

Após isso, é criada uma instância da métrica de avaliação 'recall', que é uma medida de desempenho que indica a proporção de instâncias positivas corretamente classificadas em relação ao total de instâncias positivas. Neste caso, a média 'macro' é utilizada, o que significa que o recall será calculado para cada classe individualmente e a média desses valores será obtida. Na próxima linha, é criado uma instância do objeto GridSearchCV, que realiza a busca exaustiva de hiperparâmetros através da validação cruzada. Ele recebe os três primeiros parâmetros iguais ao código anterior, e o último há uma mudança de *scoring*, de *accuracy* para *recall* (*scoring=recall*).

```
parametros = {'alpha': [0.1, 0.5, 1.0, 2.0, 5.0], 'fit_prior':  
[True, False]}  
  
modelo = MultinomialNB()  
  
recall = make_scorer(recall_score, average='macro')  
  
grid = GridSearchCV(modelo, parametros, cv=5, scoring=recall)
```

Por último, o modelo é treinado por meio do método *fit*, passando os dados de treinamento (*X\_treino*) e os rótulos correspondentes (*y\_treino*). E os resultados são exibidos pela função *print*.

```

grid.fit(X_treino, y_treino)

print('Melhores hiperparâmetros:', grid.best_params_)
print('Melhor revocação:', grid.best_score_)

output:
Sprint 3:
Melhores hiperparâmetros: {'alpha': 0.1, 'fit_prior': True}
Melhor revocação: 0.7103233123310607
Sprint 4:
Melhores hiperparâmetros: {'alpha': 0.5, 'fit_prior': True}
Melhor revocação: 0.6609166980005126

```

#### 10.1.4 Conclusão

Pode-se concluir que esse modelo teve resultados satisfatórios para o projeto, e que o grupo pode tirar diversos *insights*. Além disso, pode-se perceber que o pré processamento feito na Sprint 4 não obteve um melhor sucesso, comparado com a Sprint 3, nesse modelo.

### 10.2 Rede Neural (Sequência de palavras) - Word2Vec

#### 10.2.1 Introdução

Uma rede neural, também conhecida como rede neural artificial, é um modelo computacional inspirado no funcionamento do cérebro humano. Ela é composta por um conjunto interconectado de unidades de processamento, chamadas de neurônios artificiais ou nós, que trabalham em conjunto para resolver problemas complexos de forma eficiente. Para fins de comparação, foram desenvolvidos códigos que têm a base de dados diferentes.

#### 10.2.2 Método

O método de sequência de palavras geralmente é utilizado para frases que formam um significado, e a ordem das palavras é crucial para formar o sentido. Então, no caso do projeto, é interessante testar a utilização dessa abordagem.

#### 10.2.3 Resultado

O tópico será dividido da mesma forma que o notebook está dividido para facilitar a compreensão. Além disso, esse modelo foi testado com 3 tipos de bases de dados diferentes (base tratada, word2vec com cbow e word2vec com embedding layer) em dois momentos diferentes (sprint 3 e sprint 4).

### 10.2.3.1 Leitura da base de dados

Para a realização com a base tratada, o primeiro processo a ser realizado é a leitura do arquivo csv gerado no notebook do pré processamento, onde o arquivo já está com as etapas realizadas.

```
rede_neural_df = pd.read_csv("caminho do arquivo")
```

Para a realização dos modelos já vetorizados com o Word2Vec, é necessário somente referenciar a variável usada no notebook.

Word2Vec + CBOW: df\_vec

Word2Vec + Embedding Layer: df\_word2vec

### 10.2.3.2 Separação de treino e teste

A primeira linha do código é atribuído os valores das colunas "texto\_tratado" e "sentimento" do *dataframe* desejado às variáveis x e y, respectivamente. Essa fase significa que a primeira coluna, "texto\_tratado" ou x, contém o texto dos dados a serem classificados e a segunda coluna, "sentimento" ou y, representa a categoria desses dados.

```
x, y = rede_neural_df["texto_tratado"],  
       rede_neural_df["sentimento"]  
x, y = df_vec["texto_tratado"], df_vec["sentimento"]  
x, y = df_word2vec["texto_tratado"], df_word2vec["sentimento"]
```

obs: somente um desses códigos devem ser utilizados, a depender de qual base será utilizada no desenvolvimento.

As próximas linhas abaixo têm como objetivo realizar um pré-processamento novamente, feito pela rede neural, que contém: LabelEncoder, remoção de algumas palavras e Tokenização.

```
labelencoder = LabelEncoder()  
y = labelencoder.fit_transform(y)  
  
words = ["o", "ao", 'aos', 'os', 'a', 'as', 'e', 'um',  
         'uma', 'ele', 'ela', 'eles', 'elas', 'do', 'da', 'dos',  
         'das', 'de', 'no', 'na', 'nos', 'nas', 'pelo', 'pela',  
         'pelos', 'pelas', 'num', 'numa', 'nuns', 'numas', 'dum',  
         'duma', 'duns', 'dumas']
```

```

x_filter = []

for title in x:
    for word in words:
        title = title.replace(word, '')
    x_filter.append(title)

tokenizer = Tokenizer()
tokenizer.fit_on_texts(x_filter)

```

A próxima linha calcula o tamanho do vocabulário, por meio da criação da variável `vocab`. Além disso, o texto contido na lista `x_filter` é transformado em uma sequência numérica e logo depois é criado qual será o comprimento máximo dentro da sequência, por meio do `max_length`. Por último, as sequências de palavras em `x_filter` são ajustadas para ter o mesmo comprimento máximo através da função `pad_sequences` de uma biblioteca.

```

vocab = len(tokenizer.word_docs) + 1

x_filter = tokenizer.texts_to_sequences(x_filter)

max_length = max([len(z) for z in x_filter])
x_filter = pad_sequences(x_filter, maxlen=max_length,
padding='post')

```

A última etapa desse tópico é a divisão entre conjuntos de treinamento (`x_train` e `y_train`) e teste (`x_test` e `y_test`) usando a função `train_test_split` da biblioteca `sklearn.model_selection`. Nesse caso foram utilizados 33% dos dados para teste. Além disso, é *printado* o tamanho dos dados de entrada e saída.

```

x_train, x_test, y_train, y_test =
train_test_split(x_filter, y, test_size=0.33)

print("Tamanho de x:", len(x_filter))
print("Tamanho de y:", len(y))

output - sprint 3:
Tamanho de x: 9207
Tamanho de y: 9207

```

```
output - sprint 4:  
Tamanho de x: 8040  
Tamanho de y: 8040
```

### 10.2.3.3 Criação do modelo

A função `recall` implementada acima calcula a métrica de *recall* para avaliar o desempenho de um modelo de aprendizado de máquina em um problema de classificação binária. A função recebe dois parâmetros: `y_true` e `y_pred`, o primeiro representa as verdadeiras classes dos exemplos do conjunto de dados, enquanto o segundo representa as classes previstas pelo modelo. Em seguida, o número de `possible_positives` é calculado, isso é feito aplicando a função `K.clip` novamente para limitar os valores de `y_true` entre 0 e 1, convertendo-os em valores binários. Por último, o `recall` é calculado dividindo o número de verdadeiros positivos pelo número de positivos possíveis.

```
def recall(y_true, y_pred):  
    true_positives = K.sum(K.round(K.clip(y_true * y_pred,  
0, 1)))  
    possible_positives = K.sum(K.round(K.clip(y_true, 0,  
1)))  
    return true_positives / (possible_positives +  
K.epsilon())
```

As linhas abaixo definem como o modelo será estruturado, começando com a criação da variável `model` que define que o modelo será `Sequential()`, permitindo o empilhamento de camadas sequencialmente. A segunda linha, adiciona uma camada de *embedding*, que é responsável por transformar os números inteiros que representam as palavras em vetores densos de números reais. A terceira linha adiciona uma camada de *pooling global* máxima, que extrai o valor máximo de cada recurso da camada anterior e reduz a dimensão dos dados resultantes para um vetor unidimensional.

```
model = Sequential()  
model.add(Embedding(input_dim=vocab, output_dim=80,  
input_length=max_length, trainable = True))  
model.add(GlobalMaxPooling1D())
```

A próxima linha adiciona uma camada de *dropout*, que é uma técnica utilizada para prevenir o *overfitting*. A seguir, é adicionada uma camada densa que possui 3 unidades, correspondendo às 3 classes possíveis de sentimentos. A função de ativação softmax é aplicada para produzir probabilidades de pertencer a cada classe. E, por último, o modelo é compilado e o otimizador adam é usado para

ajustar os pesos da rede durante o treinamento. A métrica *recall*, definida pela função já descrita, é usada para avaliar o desempenho do modelo durante o treinamento.

```
model.add(Dropout(0.3))
model.add(Dense(units = 3, activation = 'softmax'))
model.compile(optimizer = 'adam', loss =
'sparse_categorical_crossentropy', metrics = [recall])
```

O objeto `ModelCheckpoint` é responsável por monitorar a acurácia do modelo durante o treinamento e salva apenas os melhores pesos em um arquivo `weight.best.hdf5`. Por último, o modelo é treinado usando o método `fit`, os dados de treinamento e de validação são fornecidos. O treinamento é realizado em lotes (`batch_size=32`) e por um total de 5 épocas.

```
mc          = ModelCheckpoint('weight.best.hdf5',
monitor='val_acc', save_best_only=True, mode='max')

model.fit(x_train, y_train, validation_data = (x_test,
y_test), batch_size = 32, epochs = 10, callbacks = [mc])
```

#### 10.2.3.3.1 Construção da rede neural com a base tratada - Sprint 3

Abaixo é possível observar o relatório de classificação do modelo com a base tratada da Sprint 3, onde o 0 é negativo, o 1 é neutro e o 2 é positivo. Após isso, é gerada uma matriz de confusão.

#### Relatório de Classificação:

	precision	recall	f1-score	support
0	0.70	0.68	0.69	662
1	0.77	0.76	0.77	1358
2	0.71	0.73	0.72	1019
accuracy			0.73	3039
macro avg	0.72	0.72	0.72	3039
weighted avg	0.73	0.73	0.73	3039

#### Código da matriz de confusão:

```
cm = confusion_matrix(y_test, y_pred_classes)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot
```

```

        =True, cmap='Blues', fmt='g', xticklabels=classes,
        yticklabels=classes)
plt.xlabel('Classe Predita')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()

```

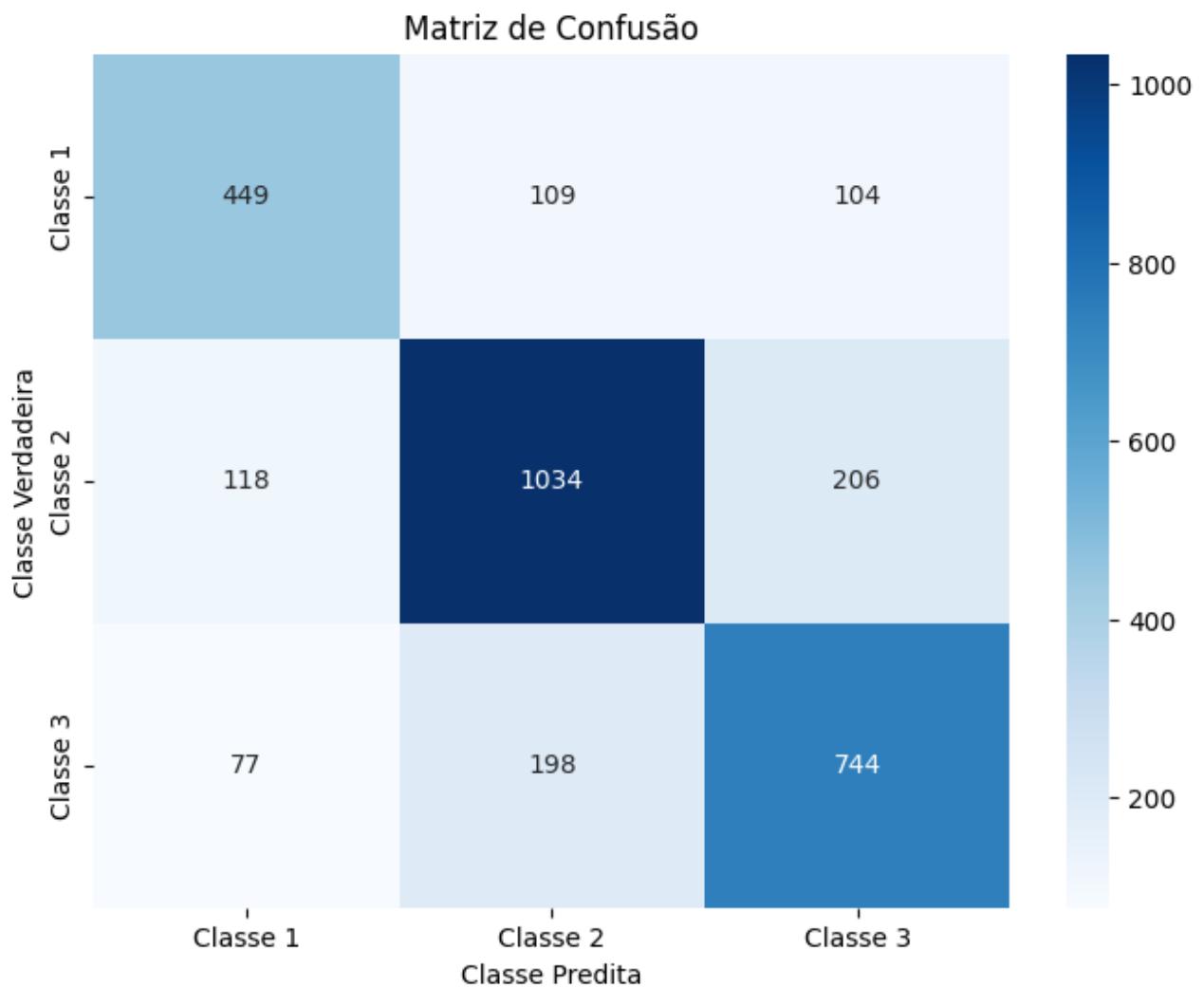


Figura 34: Matriz de Confusão - base tratada Sprint 3

#### **10.2.3.3.2 Construção da rede neural com Word2Vec + CBoW - Sprint 3**

Abaixo é possível observar o relatório de classificação do modelo com o Word2Vec + CBoW da Sprint 3, onde o 0 é negativo, o 1 é neutro e o 2 é positivo. Após isso, é gerada uma matriz de confusão.

#### **Relatório de Classificação:**

precision	recall	f1-score	support
-----------	--------	----------	---------

0	0.70	0.70	0.70	633
---	------	------	------	-----

1	0.79	0.74	0.77	1308
2	0.71	0.77	0.74	1098
accuracy			0.74	3039
macro avg	0.73	0.74	0.73	3039
weighted avg	0.74	0.74	0.74	3039

### Código da matriz de confusão:

```

cm = confusion_matrix(y_test, y_pred_classes)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot
=True, cmap='Blues', fmt='g', xticklabels=classes,
yticklabels=classes)
plt.xlabel('Classe Preditá')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()

```

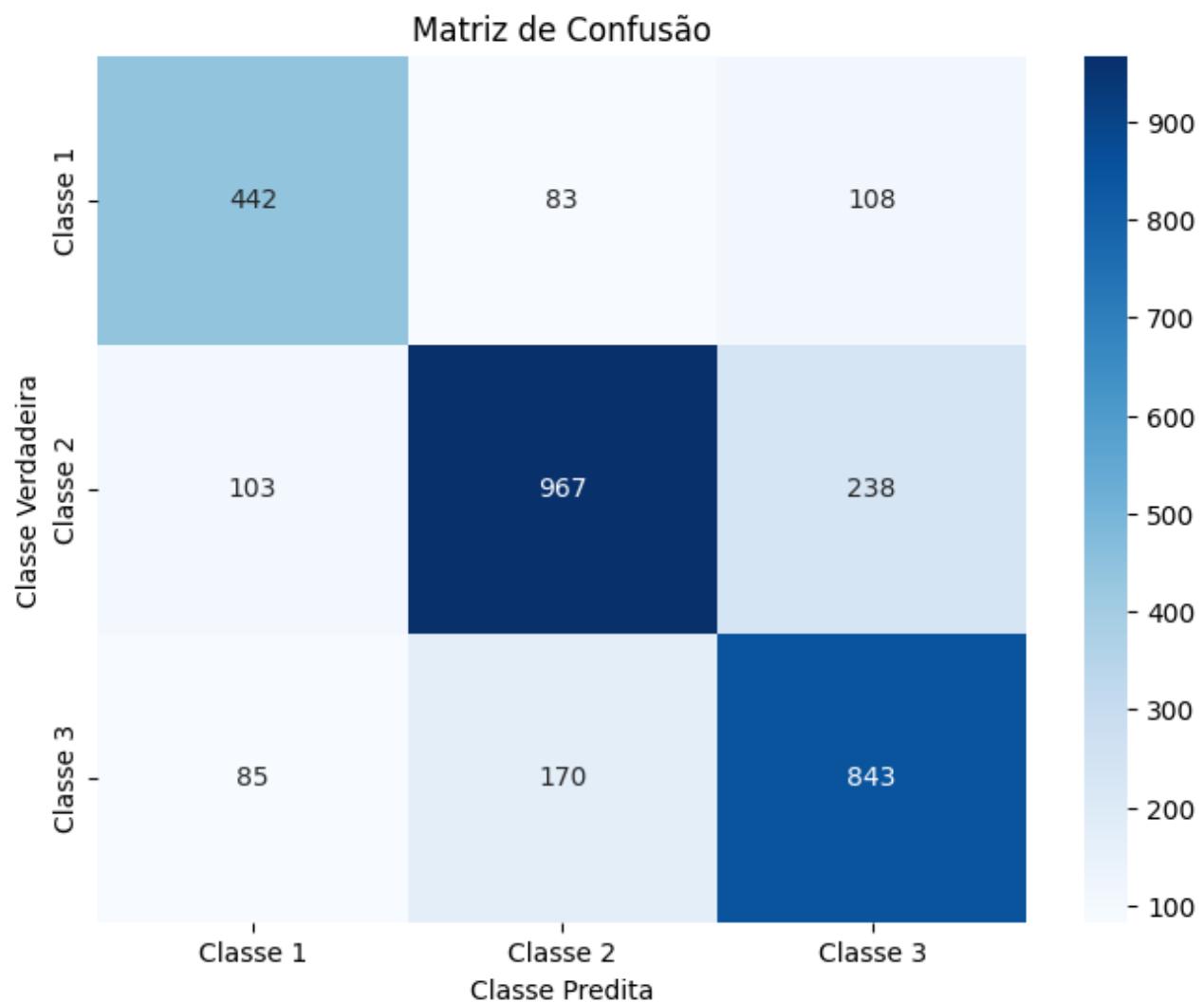


Figura 35: Matriz de Confusão - Word2Vec + CBoW Sprint 3

#### **10.2.3.3.3 Construção da rede neural com Word2Vec + Embedding Layer - Sprint 3**

Abaixo é possível observar o relatório de classificação do modelo com o Word2Vec + Embedding Layer da Sprint 3, onde o 0 é negativo, o 1 é neutro e o 2 é positivo. Após isso, é gerada uma matriz de confusão.

#### **Relatório de Classificação:**

	precision	recall	f1-score	support
0	0.67	0.69	0.68	632
1	0.76	0.75	0.76	1321
2	0.72	0.72	0.72	1086
accuracy			0.73	3039
macro avg	0.72	0.72	0.72	3039
weighted avg	0.73	0.73	0.73	3039

### Código da matriz de confusão:

```
cm = confusion_matrix(y_test, y_pred_classes)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=classes,
            yticklabels=classes)
plt.xlabel('Classe Predita')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()
```

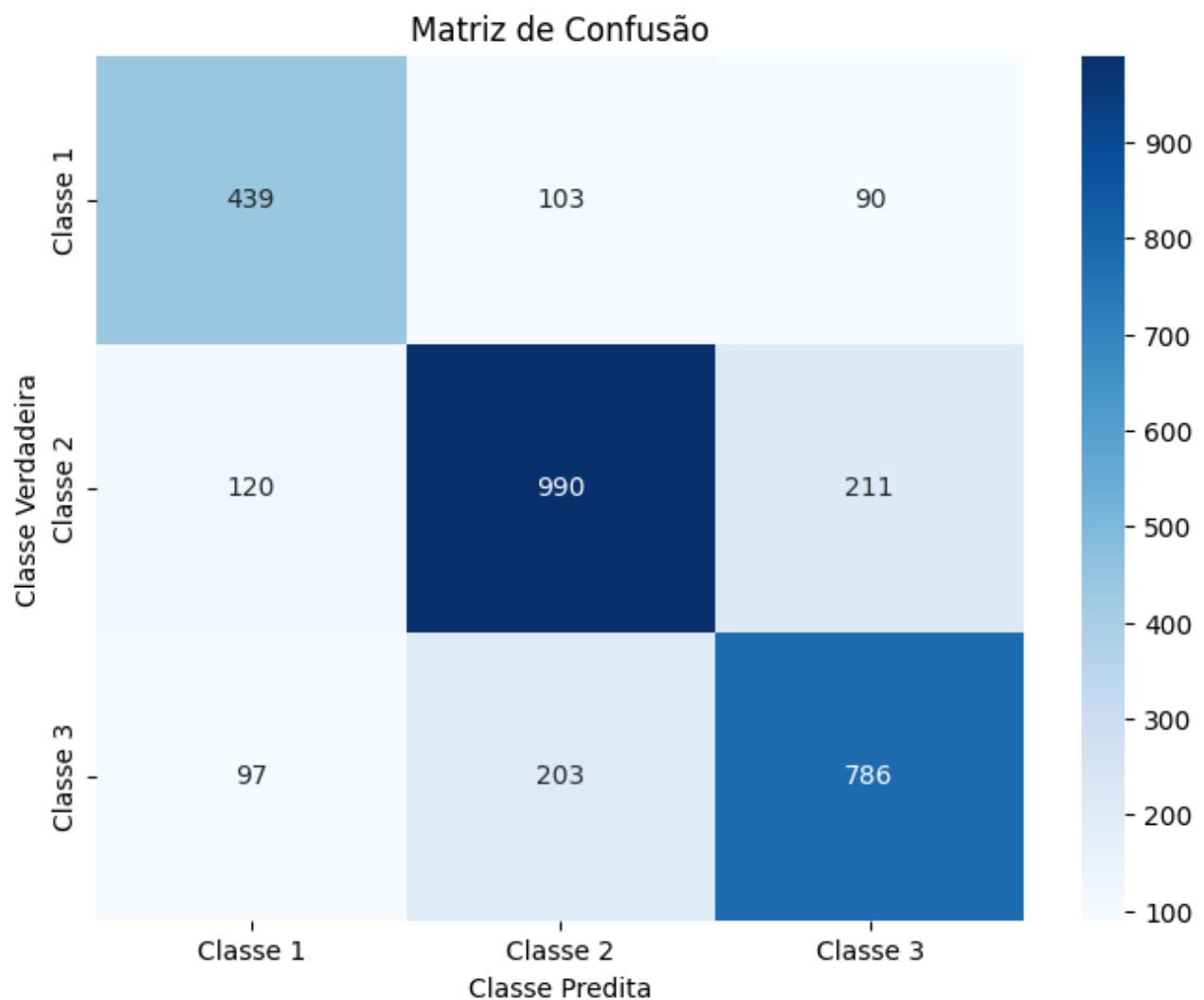


Figura 36: Matriz de Confusão - Word2Vec + Embedding Layer Sprint 3

#### **9.2.3.3.4 Construção da rede neural com a base tratada - Sprint 4**

Abaixo é possível observar o relatório de classificação do modelo com a base tratada da Sprint 4, onde o 0 é negativo, o 1 é neutro e o 2 é positivo. Após isso, é gerada uma matriz de confusão.

#### **Relatório de Classificação:**

	precision	recall	f1-score	support
0	0.67	0.66	0.67	624
1	0.72	0.64	0.68	990
2	0.69	0.78	0.73	1040
accuracy			0.70	2654
macro avg	0.70	0.69	0.69	2654
weighted avg	0.70	0.70	0.70	2654

#### **Código da matriz de confusão:**

```
cm = confusion_matrix(y_test, y_pred_classes)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot
=True, cmap='Blues', fmt='g', xticklabels=classes,
yticklabels=classes)
plt.xlabel('Classe Predita')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()
```

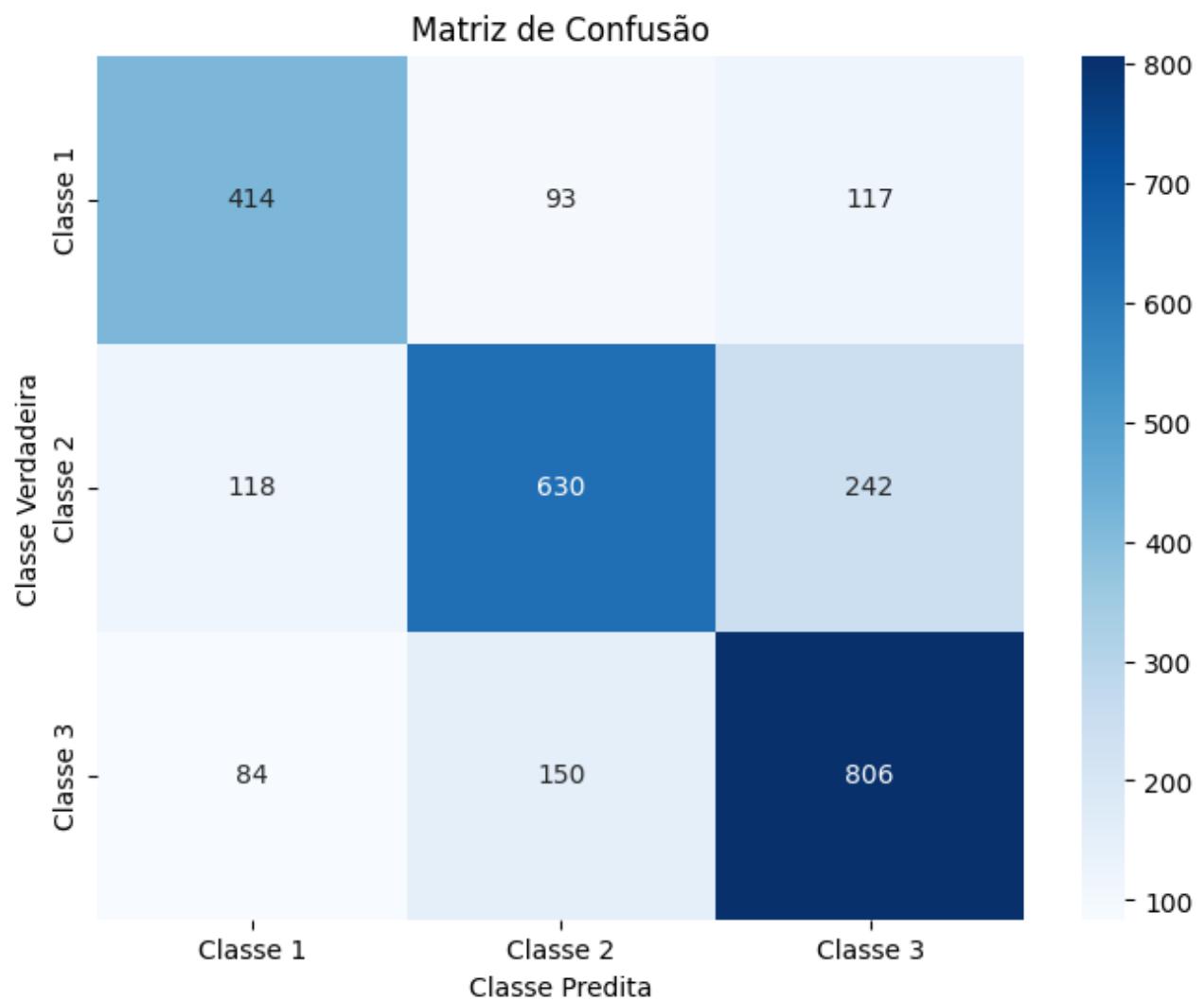


Figura 37: Matriz de Confusão - Base tratada Sprint 4

#### **10.2.3.3.5 Construção da rede neural com Word2Vec + CBoW - Sprint 4**

Abaixo é possível observar o relatório de classificação do modelo com o Word2Vec + CBoW da Sprint 4, onde o 0 é negativo, o 1 é neutro e o 2 é positivo. Após isso, é gerada uma matriz de confusão.

#### **Relatório de Classificação:**

	precision	recall	f1-score	support
0	0.66	0.63	0.65	655
1	0.71	0.64	0.68	969
2	0.69	0.77	0.73	1030
accuracy			0.69	2654
macro avg	0.69	0.68	0.68	2654
weighted avg	0.69	0.69	0.69	2654

#### **Código da matriz de confusão:**

```

cm = confusion_matrix(y_test, y_pred_classes)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=classes,
            yticklabels=classes)
plt.xlabel('Classe Preditá')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()

```

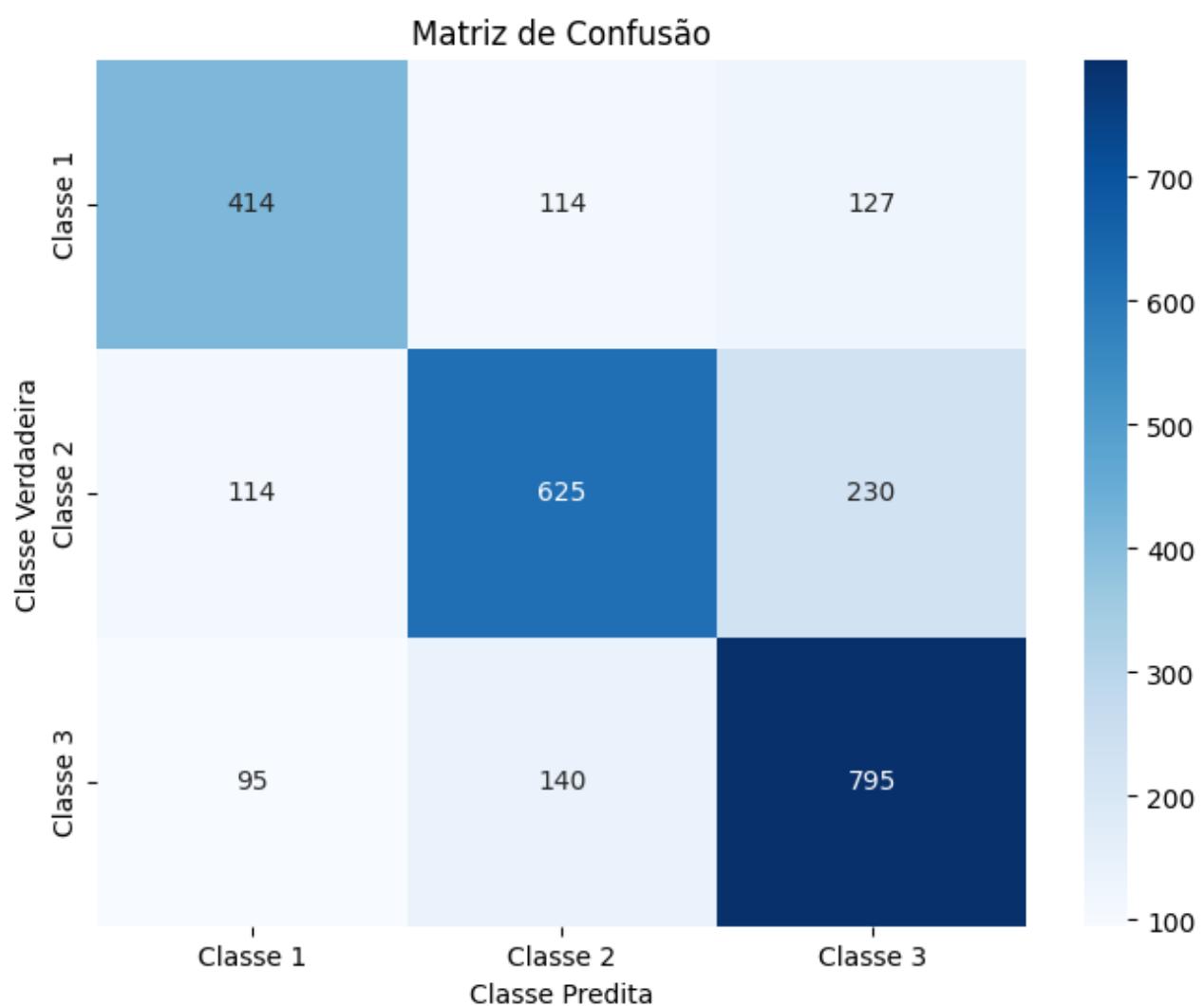


Figura 38: Matriz de Confusão - Word2Vec + CBoW Sprint 4

#### **10.2.3.3.6 Construção da rede neural com Word2Vec + Embedding Layer - Sprint 4**

Abaixo é possível observar o relatório de classificação do modelo com o Word2Vec + Embedding Layer da Sprint 4, onde o 0 é negativo, o 1 é neutro e o 2 é positivo. Após isso, é gerada uma matriz de confusão.

### **Relatório de Classificação:**

	precision	recall	f1-score	support
0	0.68	0.64	0.66	676
1	0.68	0.69	0.69	980
2	0.70	0.72	0.71	998
accuracy			0.69	2654
macro avg	0.69	0.68	0.69	2654
weighted avg	0.69	0.69	0.69	2654

### **Código da matriz de confusão:**

```
cm = confusion_matrix(y_test, y_pred_classes)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot
=True, cmap='Blues', fmt='g', xticklabels=classes,
yticklabels=classes)
plt.xlabel('Classe Preditá')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()
```

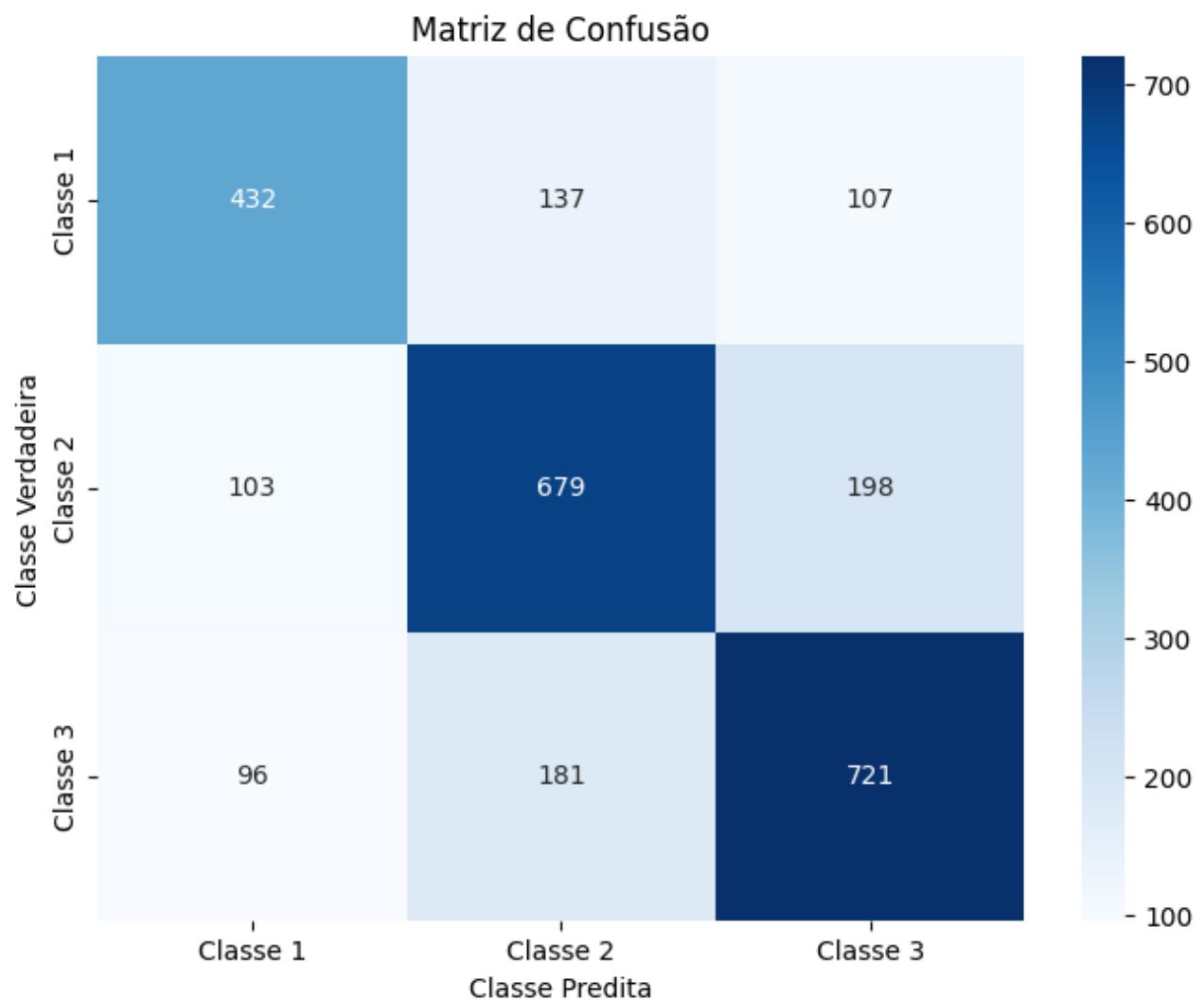


Figura 39: Matriz de Confusão - Word2Vec + Embedding Layer Sprint 4

#### 10.2.3.4 Exportação com a biblioteca pickle

A primeira linha abre o arquivo, em modo escrita, que será utilizado para armazenar o modelo de rede neural, nesse caso o desenvolvedor pode escolher o nome do arquivo. A segunda linha utiliza a função `pickle.dump()`, da biblioteca `pickle`, para salvar o modelo no arquivo aberto anteriormente. A terceira linha abre o arquivo, em modo leitura, para que a função `pickle.load()` carregue o conteúdo na variável, convertendo os bytes do arquivo novamente em um objeto modelo de rede neural utilizável. Todos os modelos foram salvos por meio do código abaixo.

```

with open('nome_escolhido.pkl', 'wb') as arquivo:
    pickle.dump(model, arquivo)
with open('nome_escolhido.pkl', 'rb') as arquivo:
    nome_escolhido = pickle.load(arquivo)

```

## 10.2.4 Conclusão

Os códigos apresentados mostram o processo de construção e treinamento de uma rede neural para classificação de texto, além de fornecer uma maneira de salvar e carregar o modelo treinado para uso posterior, com a utilização da biblioteca *pickle*.

# 10.3 Random Forest - Word2vec

## 10.3.1 Introdução

Nesta seção, discutiremos o uso da técnica de Random Forest em combinação com o modelo Word2Vec. O Word2Vec é um modelo de aprendizado de máquina que representa palavras como vetores numéricos, capturando relações semânticas e contextuais entre elas.

O Random Forest é um algoritmo de aprendizado supervisionado baseado em *ensemble*, que combina múltiplas árvores de decisão para tomar decisões finais. Vamos explorar como o uso do Word2Vec em conjunto com a Random Forest pode melhorar a precisão e o desempenho dos modelos de classificação.

## 10.3.2 Método

### 10.3.2.1 Cross validation

*Cross validation* utilizado para avaliar a capacidade de um modelo de generalizar para novos dados, que consiste em dividir o conjunto de dados em partes menores, treinar o modelo em uma parte e testá-lo em outra. Esse processo é repetido várias vezes e a média das métricas de avaliação é usada para avaliar o desempenho do modelo.

## 10.3.3 Resultados

O primeiro processo que é realizado é a verificação e eliminação de valores nulos no *dataframe*, como mostra a primeira linha do código abaixo. A segunda linha o *target* é atribuída à variável *target*, que contém a coluna 'sentimentoNumerico'. As features são atribuídas à variável *features*, que contém todas as colunas a partir da coluna 2 até a coluna 102.

```
df_word2vec = df_word2vec.dropna()  
target = df_word2vec['sentimentoNumerico']  
features = df_word2vec.iloc[:, 2:102]
```

Em seguida, os dados são divididos em conjuntos de treino e teste usando a função *train\_test\_split*, os conjuntos de treino (*X\_treino* e *y\_treino*) e teste (*X\_teste* e *y\_teste*) são criados. A função recebe 4 parâmetros: *features*, *target*, *test\_size=0.2* e *random\_state=42*.

```
X_train, X_test, y_train, y_test = train_test_split(features,  
target, test_size=0.2, random_state=42)
```

Primeiramente, uma instância do modelo *Random Forest* é criada utilizando a classe `RandomForestClassifier` e atribuída à variável `model_rf`, representando um modelo de classificação baseado nesta técnica. Em seguida, o modelo é treinado utilizando os dados de treino fornecidos, que são compostos pelos recursos (`X_train`) e as classes correspondentes (`y_train`). O método `fit` é chamado na instância do modelo, que ajusta o modelo aos dados de treino, permitindo que ele aprenda os padrões presentes nos dados.

```
model_rf = RandomForestClassifier(n_estimators=100,  
random_state=42)  
  
model_rf.fit(X_train, y_train)
```

O código apresentado abaixo realiza a validação cruzada com 5 *folds* usando a função `cross_val_score`. Essa função divide os dados de treinamento `X_train` e `y_train` em 5 partes e realiza 5 iterações de treinamento e avaliação. O modelo `model_rf` é utilizado para treinar e avaliar o desempenho em cada fold. Os parâmetros são:

- `model_rf` é o modelo de classificador de Random Forest que será avaliado;
- `X_train` é a matriz de recursos de treinamento;
- `y_train` é o vetor de classes ou rótulos correspondentes aos dados de treinamento;
- `cv=5` define o número de *folds* utilizadas na validação cruzada, que neste caso é 5.

```
cv_scores = cross_val_score(model_rf, X_train, y_train, cv=5)  
  
print("Scores de validação cruzada:", cv_scores)  
print("Média dos scores:", np.mean(cv_scores))  
  
output:  
Scores de validação cruzada: [0.62864902 0.65444671 0.66259335  
0.63543788 0.6496945]  
Média dos scores: 0.6461642905634759
```

O código utiliza o classificador treinado (`model_rf`) para realizar previsões nas amostras de teste (`X_test`). O método `predict` é utilizado para prever as classes das amostras de teste com base no modelo treinado. Por último, o método `print` é

utilizado para gerar um relatório de classificação para avaliar o desempenho do modelo nas previsões, com métricas de recall, acurácia e f1-score.

```
y_pred = model_rf.predict(X_test)

classification_report = classification_report(y_test, y_pred)
print(classification_report)
```

output - **SPRINT 3**

	precision	recall	f1-score	support
0	0.56	0.54	0.55	386
1	0.75	0.70	0.73	844
2	0.60	0.67	0.63	612
accuracy			0.66	1842
macro avg	0.64	<b>0.64</b>	0.64	1842
weighted avg	0.66	0.66	0.66	1842

output - **SPRINT 4**

	precision	recall	f1-score	support
0	0.55	0.54	0.55	360
1	0.66	0.57	0.61	597
2	0.62	0.70	0.65	651
accuracy			0.61	1608
macro avg	0.61	<b>0.60</b>	0.60	1608
weighted avg	0.62	0.61	0.61	1608

Abaixo é possível observar o código necessário para criar a matriz de confusão, onde esta é calculada usando a função `confusion_matrix(y_test, y_pred)`. Após calcular a matriz de confusão, uma lista chamada `classes` é definida, contendo os nomes das classes presentes no problema, que serão utilizados para rotular as classes na matriz de confusão. Em seguida, é criada uma figura de plotagem com as dimensões especificadas usando `plt.figure(figsize=(8, 6))`. A função `sns.heatmap()` é chamada para criar o mapa de calor, os parâmetros são: `cm` é que é a matriz de confusão, `annot=True` e `fmt='g'` utilizados para exibir os valores da matriz nas células, `cmap='Blues'` que define a paleta de cores a ser utilizada no mapa

de calor, e por último, os rótulos dos eixos x e y são definidos com base na lista de classes usando os parâmetros `xticklabels` e `yticklabels`.

```
matriz_confusao = confusion_matrix(y_test, y_pred)

classes = ['Classe 1', 'Classe 2', 'Classe 3']

plt.figure(figsize=(8, 6))
sns.heatmap(matriz_confusao, annot=True, cmap='Blues', fmt='g',
            xticklabels=classes, yticklabels=classes)
plt.xlabel('Classe Predita')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()
```

### Sprint 3:

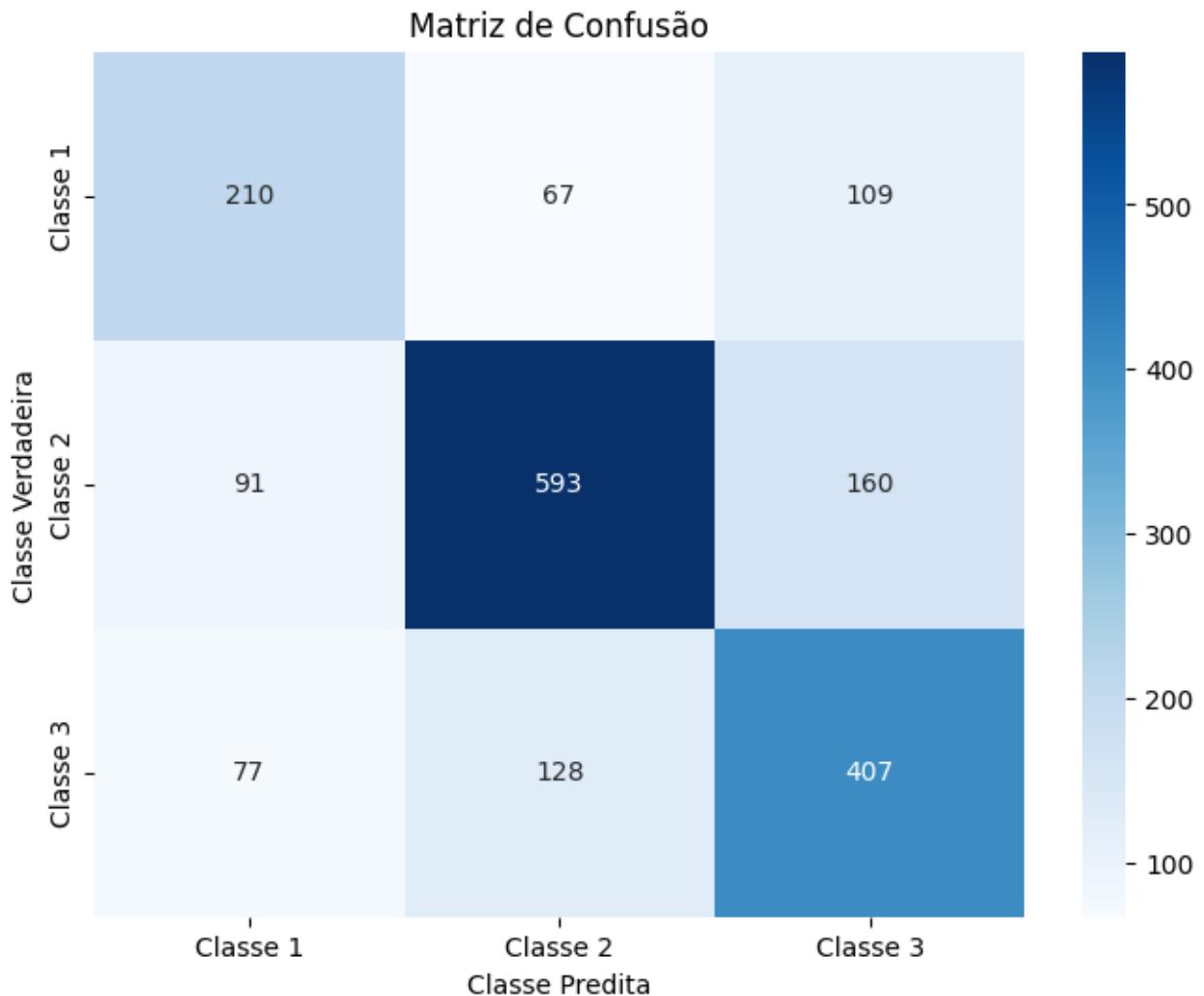


Figura 40: Matriz de Confusão Random Forest Sprint 3

Pode-se concluir, que de acordo com a matriz de confusão acima, o modelo teve um alto acerto sobre a classe dos negativos (210), porém, ainda teve um erro de (67)

sendo classificado como neutro quando na verdade é negativo e (109) sendo classificado como positivo quando é negativo. Logo, estamos em busca do aumento de acertos da classe negativa e diminuição desses erros, em que o modelo está classificando erroneamente um total de (176) que pertencem a classe do negativo.

#### Sprint 4:

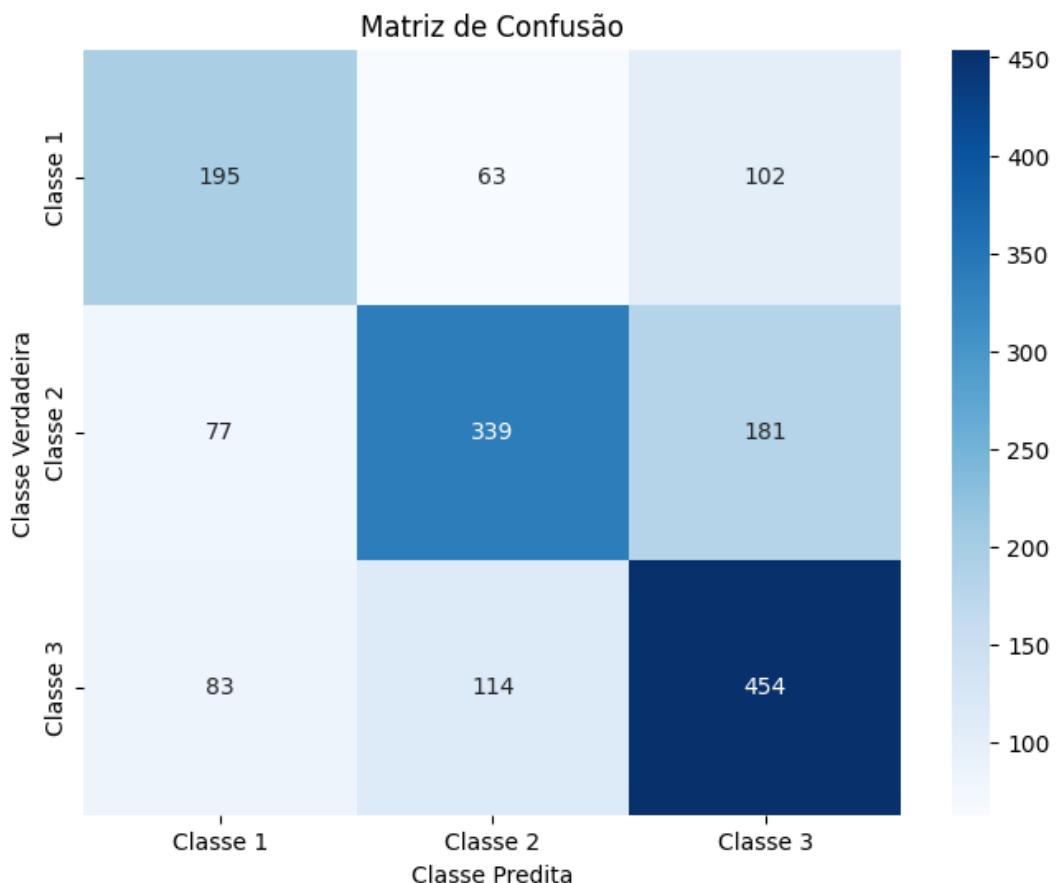


Figura 41: Matriz de Confusão Random Forest Sprint 3

Pode-se concluir, que de acordo com a matriz de confusão acima, o modelo teve um alto acerto sobre a classe dos negativos (195), porém, ainda teve um erro de (63) sendo classificado como neutro quando na verdade é negativo e (102) sendo classificado como positivo quando é negativo. Logo, estamos em busca do aumento de acertos da classe negativa e diminuição desses erros, em que o modelo está classificando erroneamente um total de (165) que pertencem a classe do negativo.

#### 10.3.4 Conclusão

Em suma, o *Random Forest* é um modelo promissor que apresentou resultados pouco satisfatórios com esse modelo de vetorização e no final o modelo desenvolvido foi exportado com a biblioteca pickle.

## 10.4 XGboost - Bag of Words e Word2Vec

### 10.4.1 Introdução

Nesta seção, é explicado o uso do algoritmo XGBoost em combinação com dois processos de vetorização diferentes: Bag of Words e Word2Vec. Como falado em tópicos anteriores, o modelo Bag of Words é uma técnica de representação de texto que transforma cada documento em um vetor numérico, considerando a frequência das palavras no documento. Já o Word2Vec é um modelo de linguagem que mapeia palavras em vetores de alta dimensionalidade, preservando as relações semânticas entre elas.

### 10.4.2 Método

O XGBoost, que significa Extreme Gradient Boosting, é um algoritmo de aprendizado de máquina que se baseia no método de boosting para criar um modelo preditivo mais forte. A técnica de boosting combina várias árvores de decisão fracas para melhorar a precisão do modelo final. Assim, é possível analisar como foi o desenvolvimento do código e o resultado final da combinação do algoritmo XGboost com os processos de vetorização BoW e Word2Vec.

#### 10.4.2.1 Cross Validation

*Cross validation* é uma técnica usada para avaliar a capacidade de um modelo de generalizar para novos dados, que consiste em dividir o conjunto de dados em partes menores, treinar o modelo em uma parte e testá-lo em outra. Esse processo é repetido várias vezes e a média das métricas de avaliação é usada para avaliar o desempenho do modelo.

#### 10.4.2.2 Grid Search

*Grid search* é uma técnica de busca de hiperparâmetros usada para encontrar a melhor combinação de valores para um modelo de aprendizado de máquina, que consiste em definir um conjunto de valores para cada hiperparâmetro e treinar e avaliar o modelo com todas as combinações possíveis. O conjunto de hiperparâmetros que produz a melhor métrica de avaliação é selecionado como a configuração final do modelo.

### 10.4.3 Resultados

Na primeira linha o modelo de vetorização é carregado. O desenvolvedor deve escolher um deles e aplicá-lo em todo o código demonstrado aqui. Em seguida, os dados são divididos em conjuntos de treino e teste usando a função `train_test_split`, os conjuntos de treino (`X_train` e `y_train`) e teste (`X_test` e `y_test`) são criados. A função recebe 4 parâmetros: `features`, `target`, `test_size=0.2` e `random_state=42`.

```
X_train, X_test, y_train, y_test = train_test_split(features,  
target, test_size=0.2, random_state=42)
```

Primeiramente, uma instância do modelo *XGBoost* é criada utilizando a classe `XGBClassifier` e atribuída à variável `modelo_xgb`, representando um modelo de classificação baseado nesta técnica. Em seguida, o modelo é treinado utilizando os dados de treino fornecidos, que são compostos pelos recursos (`X_treino`) e as classes correspondentes (`y_treino`). O método `fit` é chamado na instância do modelo, que ajusta o modelo aos dados de treino, permitindo que ele aprenda os padrões presentes nos dados.

```
modelo_xgb = xgb.XGBClassifier()  
  
modelo_xgb.fit(X_treino, y_treino)
```

O código utiliza o classificador treinado (`modelo_xgb`) para realizar previsões nas amostras de teste (`X_teste`). O método `predict` é utilizado para prever as classes das amostras de teste com base no modelo treinado. Por último, o método `encoder.inverse_transform` é utilizado para decodificar as classes preditas.

```
predicao_numerica_xgb = modelo_xgb.predict(X_teste)  
  
predicao_xgb = encoder.inverse_transform(predicao_numerica_xgb)  
  
print(classification_report(sentimento, predicao_xgb))  
  
output:  


|              | precision | recall      | f1-score | support |
|--------------|-----------|-------------|----------|---------|
| 0            | 0.67      | 0.57        | 0.61     | 1970    |
| 1            | 0.60      | 0.69        | 0.64     | 2918    |
| 2            | 0.70      | 0.67        | 0.68     | 3152    |
| accuracy     |           |             | 0.65     | 8040    |
| macro avg    | 0.66      | <b>0.64</b> | 0.65     | 8040    |
| weighted avg | 0.66      | 0.65        | 0.65     | 8040    |


```

Abaixo é possível observar o código necessário para criar a matriz de confusão, onde esta é calculada usando a função `confusion_matrix(y_teste, predicao_numerica_xgb)`. Após calcular a matriz de confusão, uma lista chamada `classes` é definida, contendo os nomes das classes presentes no problema, que serão

utilizados para rotular as classes na matriz de confusão. Em seguida, é criada uma figura de plotagem com as dimensões especificadas usando `plt.figure(figsize=(8, 6))`. A função `sns.heatmap()` é chamada para criar o mapa de calor, os parâmetros são: `cm` é que é a matriz de confusão, `annot=True` e `fmt='g'` utilizados para exibir os valores da matriz nas células, `cmap='Blues'` que define a paleta de cores a ser utilizada no mapa de calor, e por último, os rótulos dos eixos x e y são definidos com base na lista de classes usando os parâmetros `xticklabels` e `yticklabels`.

```
cm = confusion_matrix(y_teste, predicao_numerica_xgb)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot= True, cmap='Blues', fmt='g', xticklabels=classes,
            yticklabels=classes)
plt.xlabel('Classe Preditiva')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()
```

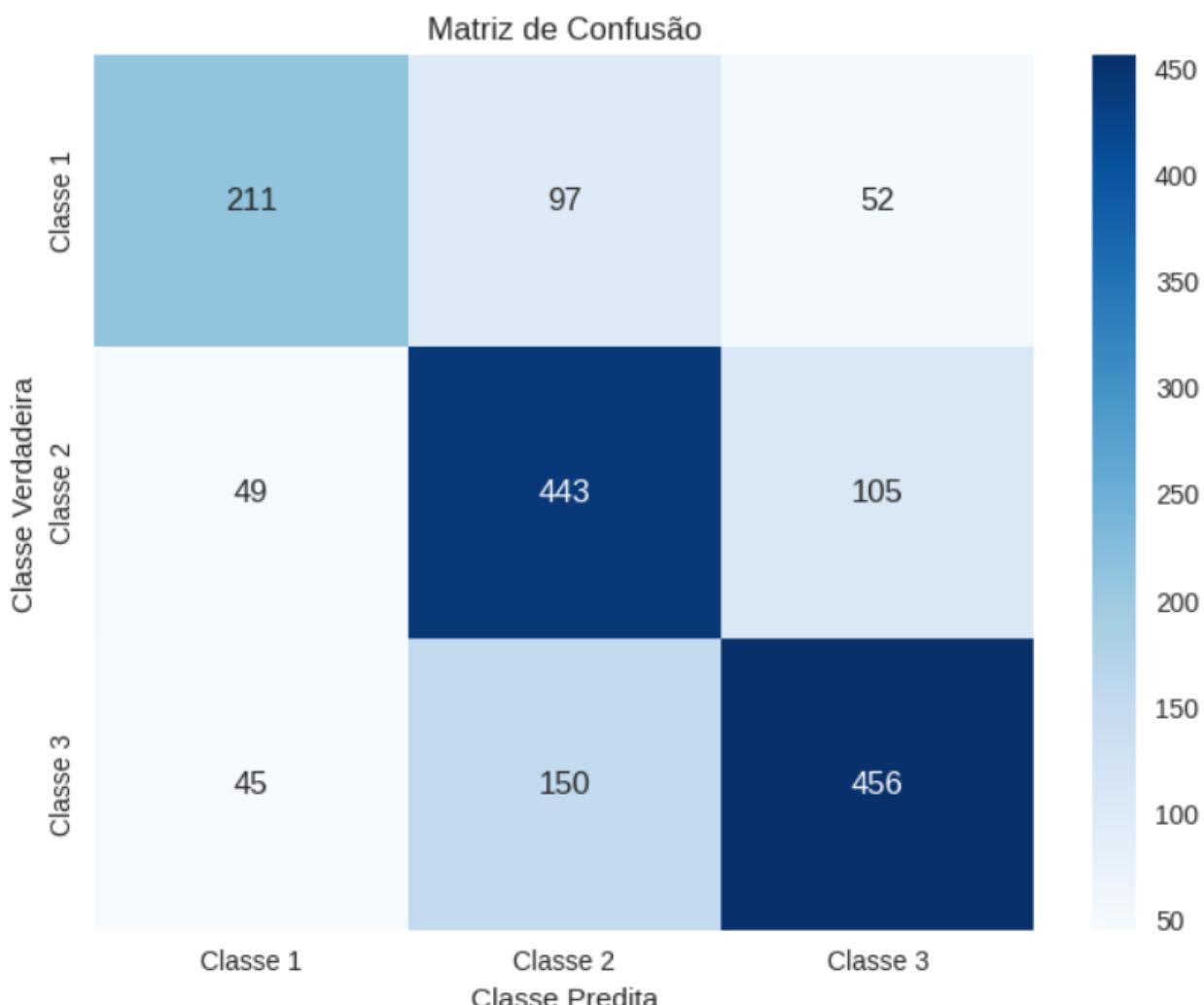


Figura 42: Matriz de Confusão XGBoost

#### 10.4.3.1 Cross Validation

Em seguida, o código utiliza a função `cross_val_predict` para fazer as previsões do modelo, os parâmetros chamados são os mesmos da função descrita no último parágrafo: `modelo`, `bow_model`, `sentimento`, `cv=7`. Na segunda linha, é utilizado o objeto `encoder` para decodificar as classes preditas (`predicoes_cv`) de volta aos seus valores originais. Por fim, o código imprime o relatório de classificação usando a função `classification_report`.

```

predicoes_cv = cross_val_predict(modelo_xgb, bow_model,
                                 sentimento, cv=7)

predicoes_cv_decodificadas =
encoder.inverse_transform(predicoes_cv)

print(classification_report(sentimento,
                           predicoes_cv_decodificadas))

output: (BoW)
          precision    recall   f1-score   support
          0       0.69      0.59      0.63      360
          1       0.64      0.74      0.69      597
          2       0.74      0.70      0.72      651

          accuracy                           0.69      1608
         macro avg       0.69      0.68      0.68      1608
    weighted avg       0.69      0.69      0.69      1608

output: (Word2Vec)
          precision    recall   f1-score   support
          0       0.49      0.50      0.50      360
          1       0.62      0.54      0.58      597
          2       0.60      0.66      0.63      651

          accuracy                           0.58      1608
         macro avg       0.57      0.57      0.57      1608
    weighted avg       0.58      0.58      0.58      1608

```

Pode-se concluir que o modelo com BoW acerta com um recall de 59% os comentários negativos, 74% os comentários neutros e 70% os comentários positivos.

Portanto, o recall geral obtido com esse modelo foi de 68%. Já o Word2Vec tem um recall de 50% os comentários negativos, 54% os comentários neutros e 66% os comentários positivos. Portanto, o recall geral obtido com esse modelo foi de 57%.

#### 10.4.3.1 Grid Search

Na primeira linha, são definidos os valores a serem testados para os hiperparâmetros do modelo, e nesse caso, o hiperparâmetro 'learning\_rate' será testado com os valores [0.1, 0.01], 'n\_estimators' será testado com os valores [100, 200], 'max\_depth' será testado com os valores [3, 5], 'subsample' será testado com os valores [0.6, 0.8] e o hiperparâmetro 'colsample\_bytree' será testado com os valores [0.6, 0.8]. Em seguida, na terceira linha, uma instância do modelo *XGBoost* é criada utilizando a classe XGBClassifier().

```
parametros = {
    'learning_rate': [0.1, 0.01],
    'n_estimators': [100, 200],
    'max_depth': [3, 5],
    'subsample': [0.6, 0.8],
    'colsample_bytree': [0.6, 0.8]
}

modelo_xgb = xgb.XGBClassifier()
```

A seguir, é criada uma instância do objeto GridSearchCV, que é utilizado para realizar uma busca exaustiva dos melhores hiperparâmetros para o modelo, esse objeto recebe o modelo criado, os parâmetros a serem testados, o número de folds para a validação cruzada (cv=5) e a métrica de avaliação a ser utilizada (scoring="accuracy"). Na próxima linha, o modelo é treinado através do método fit() do objeto grid, e os dados de treino (X\_treino e y\_treino) são utilizados para treinar o modelo e encontrar os melhores hiperparâmetros. Por último, os melhores hiperparâmetros são exibidos pelo atributo grid.best\_params\_.

```
grid = GridSearchCV(modelo_xgb, parametros, cv=5,
scoring='accuracy', n_jobs=-1)

grid.fit(X_treino, y_treino)

print('Melhores hiperparâmetros:', grid.best_params_)

output:
```

```
Melhores hiperparâmetros: {'colsample_bytree': 0.8,
'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 200,
'subsample': 0.6}
```

Por último, um novo modelo é criado, somente com os melhores hiperparâmetros, chamado de `modelo_xgb`, e o mesmo processo de fit é realizado. A predição desse modelo é realizada por meio do método `predict` e a decodificação das classes preditas com o método `encoder`. Para finalizar, é impresso o relatório de classificação.

```
modelo_xgb = xgb.XGBClassifier(**grid.best_params_)
modelo_xgb.fit(X_treino, y_treino)

predicao_numerica_xgb = modelo_xgb.predict(X_teste)

predicao_xgb = encoder.inverse_transform(predicao_numerica_xgb)

print(classification_report(y_teste, predicao_xgb))
```

output:

	precision	recall	f1-score	support
0	0.68	0.58	0.63	360
1	0.62	0.75	0.68	597
2	0.74	0.66	0.70	651
accuracy			0.68	1608
macro avg	0.68	<b>0.66</b>	0.67	1608
weighted avg	0.68	0.68	0.68	1608

Pode-se concluir que o modelo acerta com um recall de 58% os comentários negativos, 75% os comentários neutros e 66% os comentários positivos. Portanto, o recall geral obtido com esse modelo foi de 66%.

#### 10.4.4 Conclusão

A partir dos valores apresentados acima, é possível concluir que o modelo não apresenta resultados satisfatórios com o processo BoW e Word2Vec.

## 10.5 Naive Bayes + TF-IDF

### 10.5.1 Introdução

Esse modelo junta o algoritmo Naive Bayes com um modelo de vetorização, já explicado, que é o TF-IDF. Se espera que, com essa junção, os resultados sejam promissores nesta tarefa de classificação de texto. O modelo que será apresentado foi rodado com a base de dados com o pré-processamento da Sprint 4.

### 10.5.2 Método

Na abordagem que combina Naive Bayes e o modelo de vetorização TF-IDF, a coluna *target* será definida e as features também serão definidas. Além disso, é definido que 20% dos dados serão usados como teste, e os outros 80% para treino (*test\_size=0.2*). Por último, os dados são divididos aleatoriamente por meio do parâmetro *random\_state=42*.

### 10.5.3 Resultados

A linha `target = df_final['sentimento']` indica que a coluna "sentimento" é selecionada do *dataframe* `df_final` e a atribui à variável `target`.

```
target = df_final['sentimento']
```

```
target
```

A primeira parte da linha, `df_final.iloc`, o método `iloc` é chamado e ele é um indexer do pandas que permite a seleção baseada em índices posicionais (números inteiros) em vez de rótulos. Isso significa que é possível usar números inteiros para selecionar linhas e colunas do *dataframe*. A segunda parte da linha, `[:, 3:df_final.shape[1]]`, o ":" indica que será selecionado todas as linhas do *dataframe*. Em seguida, o `3:df_final.shape[1]` seleciona as colunas, a partir da terceira coluna até a última coluna do *dataframe*, e o `df_final.shape[1]` retorna o número de colunas no *dataframe*.

```
feature = df_final.iloc[:, 3:df_final.shape[1]]
```

```
feature
```

O código apresentado abaixo realiza a divisão dos conjuntos de dados `feature` e `target` em conjuntos de treinamento e teste, utilizando a função `train_test_split` da biblioteca scikit-learn. Abaixo será analisado cada parte do código:

- `train_test_split`: é uma função da biblioteca scikit-learn que permite dividir um conjunto de dados em conjuntos de treinamento e teste;

- feature: é o conjunto de dados contendo as variáveis independentes (ou recursos) que serão usadas para treinar o modelo, já definido anteriormente;
- target: é o conjunto de dados contendo a variável alvo (ou rótulo) que o modelo está tentando prever, já definido anteriormente;
- test\_size=0.2: define a proporção do conjunto de dados que será alocada para o conjunto de teste. Neste caso, 20% do conjunto de dados será usado para teste, enquanto 80% será usado para treinamento;
- random\_state=42: é o valor da semente aleatória que é usada para garantir que a divisão do conjunto de dados em treinamento e teste seja a mesma em diferentes execuções, isso garante que os resultados sejam reproduzíveis;
- X\_train, X\_test, y\_train, y\_test: essas variáveis recebem os resultados da função train\_test\_split. X\_train e X\_test são os conjuntos de treinamento e teste das variáveis independentes, respectivamente, enquanto y\_train e y\_test são os conjuntos de treinamento e teste da variável alvo, respectivamente.

```
X_train, X_test, y_train, y_test = train_test_split(feature,
                                                    target, test_size=0.2, random_state=42)
```

O código apresentado abaixo utiliza o classificador Gaussian Naive Bayes (GaussianNB) da biblioteca scikit-learn para realizar uma classificação de dados. Abaixo será explicado cada linha do código:

- clf = GaussianNB(): Cria uma instância do classificador Gaussian Naive Bayes e atribui-a à variável clf. Esse classificador é um algoritmo de aprendizado supervisionado que utiliza o teorema de Bayes com a suposição de independência condicional entre os recursos para realizar classificação;
- clf = clf.fit(X\_train, y\_train.values.ravel()): Treina o classificador aos conjuntos de treinamento X\_train e y\_train. A função fit é usada para treinar o modelo, então ela encontra os parâmetros que são os melhores e se ajustam aos dados de treinamento. O método .values.ravel() é utilizado para converter o *dataframe* y\_train em um array unidimensional, que é esperado pelo classificador;

- `Y_pred = clf.predict(X_test)`: Utiliza o classificador treinado (`clf`) para realizar previsões nas amostras de teste (`X_test`). O método `predict` é utilizado para prever as classes das amostras de teste com base no modelo treinado;
- `print(classification_report(y_test, Y_pred))`: Gera um relatório de classificação para avaliar o desempenho do modelo nas previsões, com métricas de *recall*, acurácia e f1-score.

```

clf = GaussianNB()

clf = clf.fit(X_train,y_train.values.ravel())

Y_pred = clf.predict(X_test)

print(classification_report(y_test, Y_pred))

```

output:

	precision	recall	f1-score	support
0	0.35	0.75	0.48	360
1	0.64	0.32	0.43	597
2	0.63	0.52	0.57	651
accuracy			0.50	1608
macro avg	0.54	<b>0.53</b>	0.49	1608
weighted avg	0.57	0.50	0.50	1608

Abaixo é possível observar o código necessário para criar a matriz de confusão, onde esta é calculada usando a função `confusion_matrix(y_test, Y_pred)`. Após calcular a matriz de confusão, uma lista chamada `classes` é definida, contendo os nomes das classes presentes no problema, que serão utilizados para rotular as classes na matriz de confusão. Em seguida, é criada uma figura de plotagem com as dimensões especificadas usando `plt.figure(figsize=(8, 6))`. A função `sns.heatmap()` é chamada para criar o mapa de calor, os parâmetros são: `cm` é que é a matriz de confusão, `annot=True` e `fmt='g'` utilizados para exibir os valores da matriz nas células, `cmap='Blues'` que define a paleta de cores a ser utilizada no mapa de calor, e por último, os rótulos dos eixos x e y são definidos com base na lista de classes usando os parâmetros `xticklabels` e `yticklabels`.

```

cm = confusion_matrix(y_test, Y_pred)

```

```

classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g',
            xticklabels=classes, yticklabels=classes)
plt.xlabel('Classe Predita')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()

```

**Matriz de Confusão - Sprint 4:**

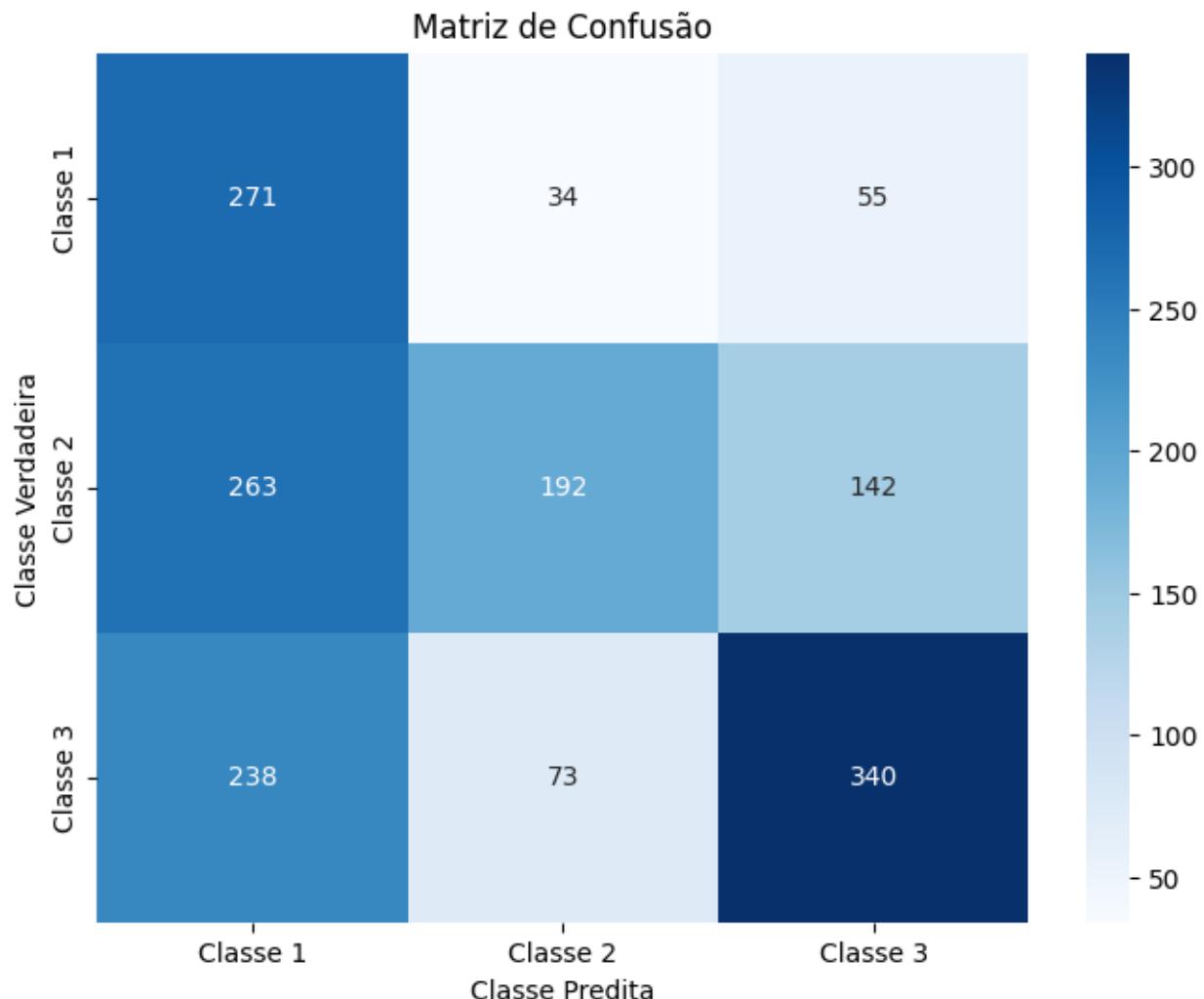


Figura 43: Matriz de confusão Naive Bayes - Sprint 4

#### 10.5.4 Conclusão

Em suma, o *Naive Bayes* é um modelo muito promissor que apresentou resultados muito satisfatórios e no final todos os modelos desenvolvidos foram exportados com a biblioteca pickle.

## 10.6 Random Forest + TF-IDF

### 10.6.1 Introdução

Foi criado um modelo de *Random Forest* juntamente do processo de vetorização *TF-IDF*. Esse modelo é um algoritmo de aprendizado de máquina que combina várias árvores de decisão para formar um modelo mais preciso e robusto, criando várias árvores de decisão independentes, onde cada árvore é treinada em uma amostra aleatória do conjunto de dados e um subconjunto aleatório das características.

### 10.6.2 Método

Na abordagem que combina Random Forest e o modelo de vetorização TF-IDF, a coluna *target* será definida e as features também serão definidas. Além disso, é definido que 20% dos dados serão usados como teste, e os outros 80% para treino (*test\_size=0.2*). Por último, os dados são divididos aleatoriamente por meio do parâmetro *random\_state=42*.

### 10.6.3 Resultados

A linha `target = df_final['sentimento']` indica que a coluna "sentimento" é selecionada do *dataframe* `df_final` e a atribui à variável `target`.

```
target = df_final['sentimento']
```

```
target
```

A primeira parte da linha, `df_final.iloc`, o método `iloc` é chamado e ele é um indexer do pandas que permite a seleção baseada em índices posicionais (números inteiros) em vez de rótulos. Isso significa que é possível usar números inteiros para selecionar linhas e colunas do *dataframe*. A segunda parte da linha, `[:, 3:df_final.shape[1]]`, o ":" indica que será selecionado todas as linhas do *dataframe*. Em seguida, o `3:df_final.shape[1]` seleciona as colunas, a partir da terceira coluna até a última coluna do *dataframe*, e o `df_final.shape[1]` retorna o número de colunas no *dataframe*.

```
feature = df_final.iloc[:, 3:df_final.shape[1]]
```

```
feature
```

O código apresentado abaixo realiza a divisão dos conjuntos de dados `feature` e `target` em conjuntos de treinamento e teste, utilizando a função `train_test_split` da biblioteca scikit-learn. Abaixo será analisado cada parte do código:

- `train_test_split`: é uma função da biblioteca scikit-learn que permite dividir um conjunto de dados em conjuntos de treinamento e teste;
- `feature`: é o conjunto de dados contendo as variáveis independentes (ou recursos) que serão usadas para treinar o modelo, já definido anteriormente;
- `target`: é o conjunto de dados contendo a variável alvo (ou rótulo) que o modelo está tentando prever, já definido anteriormente;
- `test_size=0.2`: define a proporção do conjunto de dados que será alocada para o conjunto de teste. Neste caso, 20% do conjunto de dados será usado para teste, enquanto 80% será usado para treinamento;
- `random_state=42`: é o valor da semente aleatória que é usada para garantir que a divisão do conjunto de dados em treinamento e teste seja a mesma em diferentes execuções, isso garante que os resultados sejam reproduzíveis;
- `X_train, X_test, y_train, y_test`: essas variáveis recebem os resultados da função `train_test_split`. `X_train` e `X_test` são os conjuntos de treinamento e teste das variáveis independentes, respectivamente, enquanto `y_train` e `y_test` são os conjuntos de treinamento e teste da variável alvo, respectivamente.

```
X_train, X_test, y_train, y_test = train_test_split(feature,
                                                    target, test_size=0.2, random_state=42)
```

O código apresentado abaixo utiliza o classificador Gaussian Naive Bayes (`GaussianNB`) da biblioteca scikit-learn para realizar uma classificação de dados. Abaixo será explicado cada linha do código:

- `model_rf = RandomForestClassifier(n_estimators=100, random_state=42)`: Cria um objeto de classificador de Random Forest chamado `model_rf`, que é um algoritmo de aprendizado de máquina que combina várias árvores de decisão para tomar uma decisão final. O parâmetro `n_estimators=100` define o número de árvores como, nesse caso é 100. O parâmetro `random_state=42` define uma semente aleatória para garantir a reprodutibilidade dos resultados.
- `model_rf.fit(X_train, y_train)`: Treina o modelo `model_rf` utilizando os dados de treinamento `X_train` e `y_train`. O primeiro é uma matriz de recursos que contém os dados de entrada para o treinamento do modelo,

enquanto o segundo é um vetor que contém as classes ou rótulos correspondentes aos dados de treinamento. O método `fit` ajusta o modelo aos dados de treinamento, permitindo que o modelo aprenda a relação entre os recursos e as classes.

```
model_rf = RandomForestClassifier(n_estimators=100,  
random_state=42)  
  
model_rf.fit(X_train, y_train)
```

O código apresentado abaixo realiza a validação cruzada com 5 *folds* usando a função `cross_val_score`. Essa função divide os dados de treinamento `X_train` e `y_train` em 5 partes e realiza 5 iterações de treinamento e avaliação. O modelo `model_rf` é utilizado para treinar e avaliar o desempenho em cada fold. Os parâmetros são:

- `model_rf` é o modelo de classificador de Random Forest que será avaliado;
- `X_train` é a matriz de recursos de treinamento;
- `y_train` é o vetor de classes ou rótulos correspondentes aos dados de treinamento;
- `cv=5` define o número de *folds* utilizadas na validação cruzada, que neste caso é 5.

```
cv_scores = cross_val_score(model_rf, X_train, y_train, cv=5)  
  
# Imprimir os scores de validação cruzada  
print("Scores de validação cruzada:", cv_scores)  
print("Média dos scores:", np.mean(cv_scores))  
  
output:  
Scores de validação cruzada: [0.69308469 0.70862471 0.68273717  
0.67573872 0.68429238]  
Média dos scores: 0.6888955350852706
```

O código utiliza o classificador treinado (`model_rf`) para realizar previsões nas amostras de teste (`X_test`). O método `predict` é utilizado para prever as classes das amostras de teste com base no modelo treinado. Por último, o método `print` é utilizado para gerar um relatório de classificação para avaliar o desempenho do modelo nas previsões, com métricas de recall, acurácia e f1-score.

```
y_pred = model_rf.predict(X_test)  
  
# Gerar o relatório de classificação  
classification_report = classification_report(y_test, y_pred)
```

```
print(classification_report)
```

output:

	precision	recall	f1-score	support
0	0.70	0.58	0.64	360
1	0.66	0.72	0.69	597
2	0.73	0.73	0.73	651
accuracy			0.70	1608
macro avg	0.70	<b>0.68</b>	0.69	1608
weighted avg	0.70	0.70	0.69	1608

Abaixo é possível observar o código necessário para criar a matriz de confusão, onde esta é calculada usando a função `confusion_matrix(y_test, Y_pred)`. Após calcular a matriz de confusão, uma lista chamada `classes` é definida, contendo os nomes das classes presentes no problema, que serão utilizados para rotular as classes na matriz de confusão. Em seguida, é criada uma figura de plotagem com as dimensões especificadas usando `plt.figure(figsize=(8, 6))`. A função `sns.heatmap()` é chamada para criar o mapa de calor, os parâmetros são: `cm` é que é a matriz de confusão, `annot=True` e `fmt='g'` utilizados para exibir os valores da matriz nas células, `cmap='Blues'` que define a paleta de cores a ser utilizada no mapa de calor, e por último, os rótulos dos eixos x e y são definidos com base na lista de classes usando os parâmetros `xlabel` e `ylabel`.

```
matriz_confusao = confusion_matrix(y_test, y_pred)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(matriz_confusao, annot=True, cmap='Blues', fmt='g',
            xlabel='Classe Predita',
            ylabel='Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()
```

#### Matriz de Confusão - Sprint 4:

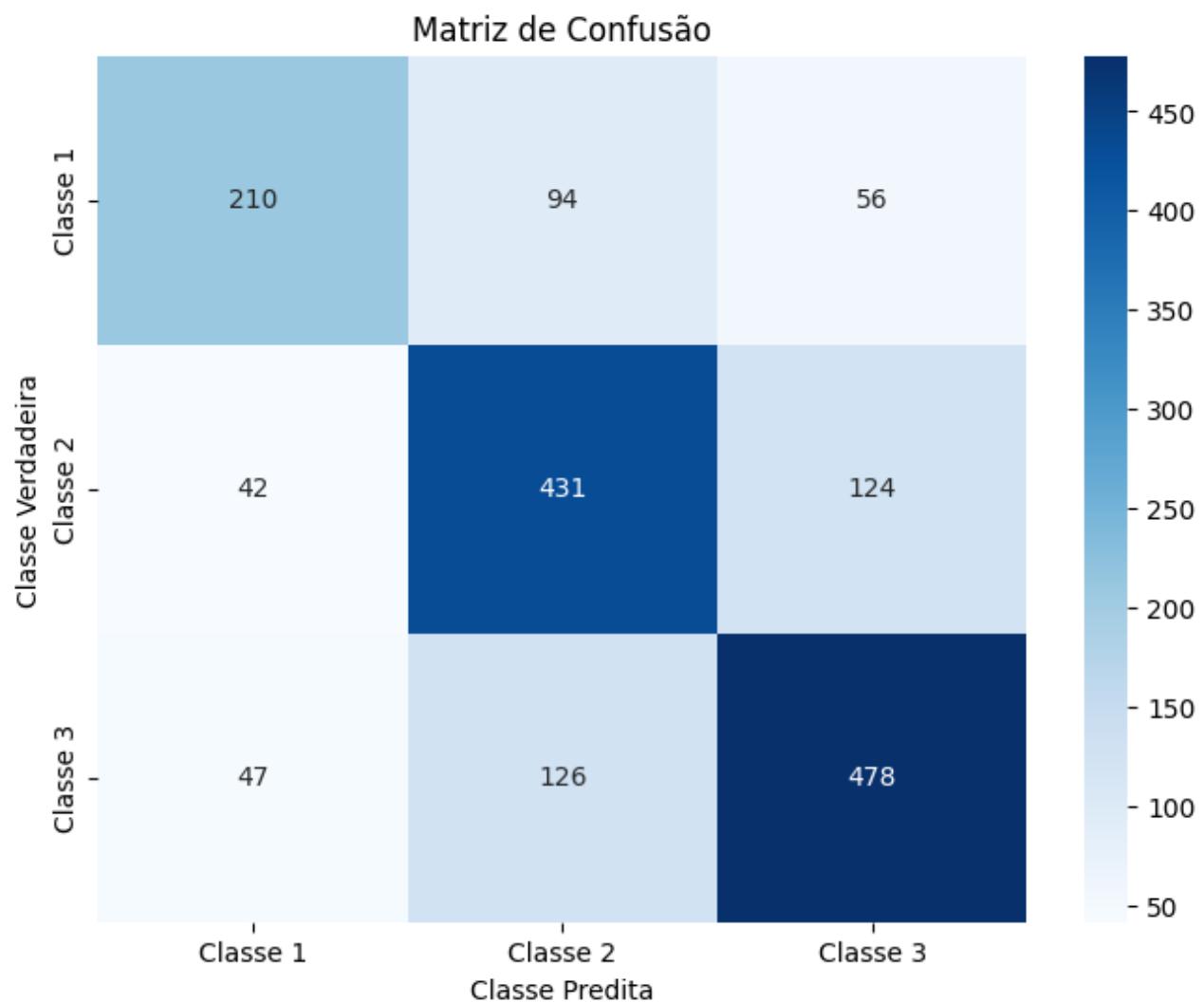


Figura 44: Matriz de confusão Random Forest - Sprint 4

#### 10.6.4 Conclusão

Em suma, o *Random Forest* é um modelo promissor que apresentou resultados melhores com esse modelo de vetorização e no final o modelo desenvolvido foi exportado com a biblioteca pickle.

### 10.7 Modelo com novas features Random Forest + TF-IDF

#### 10.7.1 Introdução

Nesta seção, descreveremos o desenvolvimento de um novo modelo utilizando a técnica Random Forest em conjunto com a abordagem TF-IDF (Term Frequency-Inverse Document Frequency) para a evolução do projeto. Essa nova abordagem visa melhorar os resultados de um modelo anterior, levando em conta uma base de dados que contém novas features relevantes.

## 10.7.2 Método

O método adotado para a construção deste modelo envolve a combinação de duas técnicas principais: Random Forest e TF-IDF. O Random Forest é um algoritmo de aprendizado de máquina baseado em árvores de decisão, que é conhecido por sua capacidade de lidar com problemas de classificação e regressão em conjuntos de dados complexos. O TF-IDF, por sua vez, é uma técnica amplamente utilizada para processamento de texto, que visa atribuir uma pontuação a cada palavra em um documento com base na sua importância relativa no contexto do documento e do corpus geral.

## 10.7.3 Resultados

A linha `target = df_final['sentimento']` indica que a coluna "sentimento" é selecionada do *dataframe* `df_final` e a atribui à variável `target`. Já a segunda linha, `df_final.iloc`, o método `iloc` é chamado e ele é um indexer do pandas que permite a seleção baseada em índices posicionais (números inteiros) em vez de rótulos. Isso significa que é possível usar números inteiros para selecionar linhas e colunas do *dataframe*. A segunda parte da linha, `[ :, 3:df_final.shape[1]]`, o ":" indica que será selecionado todas as linhas do *dataframe*. Em seguida, o `3:df_final.shape[1]` seleciona as colunas, a partir da terceira coluna até a última coluna do *dataframe*, e o `df_final.shape[1]` retorna o número de colunas no *dataframe*.

```
target = df_final['sentimento']
feature = df_final.iloc[:,3:df_final.shape[1]]
```

Em seguida, os dados são divididos em conjuntos de treino e teste usando a função `train_test_split`, os conjuntos de treino (`X_train` e `y_train`) e teste (`X_test` e `y_test`) são criados. A função recebe 4 parâmetros: `feature`, `target`, `test_size=0.2` e `random_state=42`.

```
X_train, X_test, y_train, y_test = train_test_split(feature,
                                                    target, test_size=0.2, random_state=42)
```

Primeiramente, uma instância do modelo *Random Forest* é criada utilizando a classe `RandomForestClassifier` e atribuída à variável `model_rf`, representando um modelo de classificação baseado nesta técnica. Em seguida, o modelo é treinado utilizando os dados de treino fornecidos, que são compostos pelos recursos (`X_train`) e as classes correspondentes (`y_train`). O método `fit` é chamado na instância do modelo, que ajusta o modelo aos dados de treino, permitindo que ele aprenda os padrões presentes nos dados.

```

model_rf = RandomForestClassifier(n_estimators=100,
random_state=42)

model_rf.fit(X_train, y_train)

```

O código apresentado abaixo realiza a validação cruzada com 5 *folds* usando a função `cross_val_score`. Essa função divide os dados de treinamento `X_train` e `y_train` em 5 partes e realiza 5 iterações de treinamento e avaliação. O modelo `model_rf` é utilizado para treinar e avaliar o desempenho em cada fold. Os parâmetros são:

- `model_rf` é o modelo de classificador de Random Forest que será avaliado;
- `X_train` é a matriz de recursos de treinamento;
- `y_train` é o vetor de classes ou rótulos correspondentes aos dados de treinamento;
- `cv=5` define o número de *folds* utilizadas na validação cruzada, que neste caso é 5.

```

cv_scores = cross_val_score(model_rf, X_train, y_train, cv=5)

print("Scores de validação cruzada:", cv_scores)
print("Média dos scores:", np.mean(cv_scores))

output:
Scores de validação cruzada: [0.72858122 0.72515422 0.71124829
0.70781893]
Média dos scores: 0.7160272881720855

```

O código utiliza o classificador treinado (`model_rf`) para realizar previsões nas amostras de teste (`X_test`). O método `predict` é utilizado para prever as classes das amostras de teste com base no modelo treinado. Por último, o método `print` é utilizado para gerar um relatório de classificação para avaliar o desempenho do modelo nas previsões, com métricas de recall, acurácia e f1-score.

```

y_pred = model_rf.predict(X_test)

classification_report = classification_report(y_test, y_pred)
print(classification_report)

output: (Base de dados novas features - Sprint 5)

```

	precision	recall	f1-score	support
0	0.72	0.62	0.66	396
1	0.72	0.79	0.76	809
2	0.71	0.69	0.70	619

accuracy			0.72	1824
macro avg	0.72	<b>0.70</b>	0.71	1824
weighted avg	0.72	0.72	0.72	1824

Pode-se concluir que o modelo acerta com um recall de 62% os comentários negativos, 79% os comentários neutros e 69% os comentários positivos.

Portanto, o recall geral obtido com esse modelo foi de 70%.

Abaixo é possível observar o código necessário para criar a matriz de confusão, onde esta é calculada usando a função `confusion_matrix(y_test, y_pred)`. Após calcular a matriz de confusão, uma lista chamada `classes` é definida, contendo os nomes das classes presentes no problema, que serão utilizados para rotular as classes na matriz de confusão. Em seguida, é criada uma figura de plotagem com as dimensões especificadas usando `plt.figure(figsize=(8, 6))`. A função `sns.heatmap()` é chamada para criar o mapa de calor, os parâmetros são: `cm` é que é a matriz de confusão, `annot=True` e `fmt='g'` utilizados para exibir os valores da matriz nas células, `cmap='Blues'` que define a paleta de cores a ser utilizada no mapa de calor, e por último, os rótulos dos eixos x e y são definidos com base na lista de classes usando os parâmetros `xlabel` e `ylabel`.

```

matriz_confusao = confusion_matrix(y_test, y_pred)

classes = ['Classe 1', 'Classe 2', 'Classe 3']

plt.figure(figsize=(8, 6))
sns.heatmap(matriz_confusao, annot=True, cmap='Blues', fmt='g',
            xlabel='Classe Preditiva',
            ylabel='Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()

```

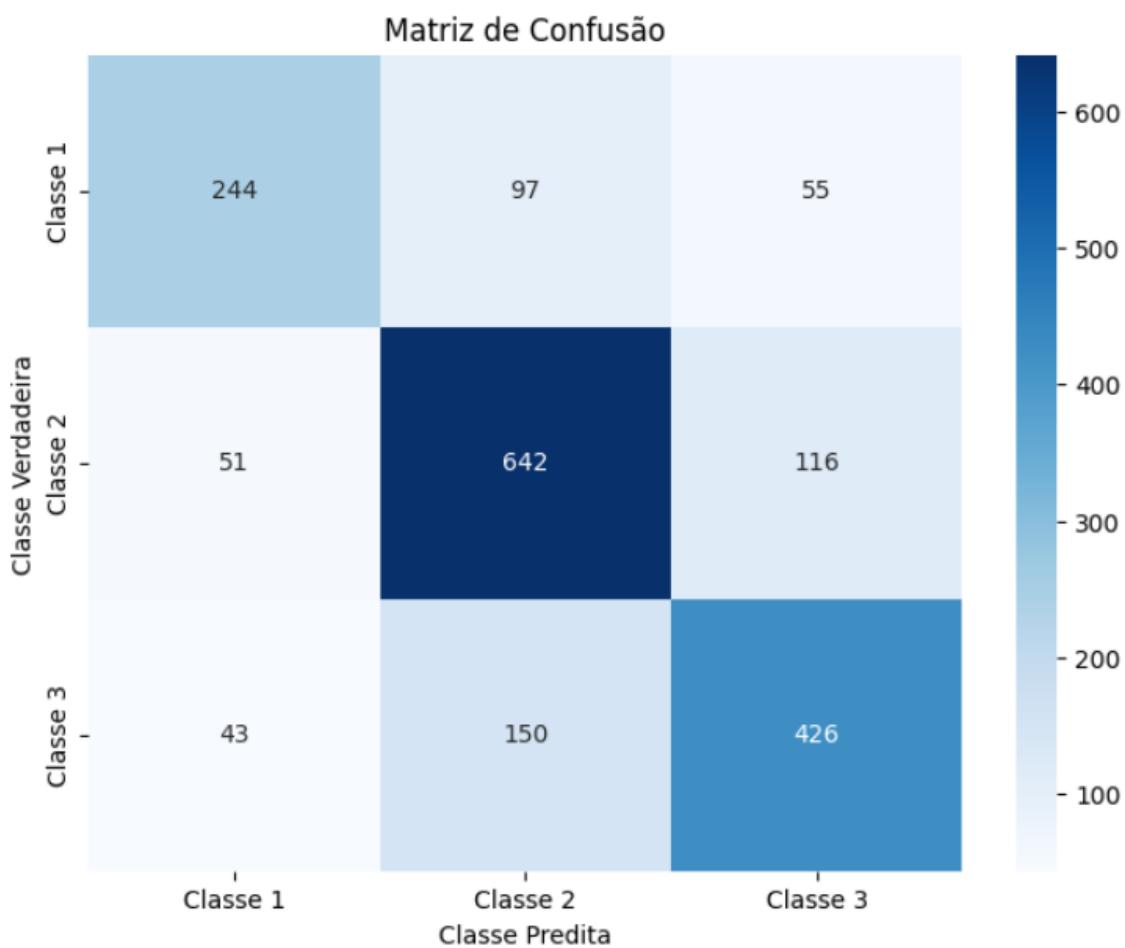


Figura 45: Matriz de confusão Random Forest - Sprint 5

Pode-se concluir, que de acordo com a matriz de confusão acima, o modelo teve um alto acerto sobre a classe dos negativos (244), porém, ainda teve um erro de (97) sendo classificado como neutro quando na verdade é negativo e (55) sendo classificado como positivo quando é negativo. Logo, estamos em busca do aumento de acertos da classe negativa e diminuição desses erros, em que o modelo está classificando erroneamente um total de (152) que pertencem a classe do negativo.

#### **10.7.4 Conclusão**

Em suma, o *Random Forest* é um modelo promissor que apresentou resultados muito satisfatórios com esse modelo de vetorização e no final o modelo desenvolvido foi exportado com a biblioteca pickle.

## **11. Comparação entre os modelos**

### **11.1 Introdução**

Essa comparação é feita com os modelos desenvolvidos durante a Sprint 4, que foram realizados com base com pré-processamento de duas formas diferentes: sprint 3

e sprint 4. Por isso, os tópicos abaixo serão divididos com os nomes dos modelos e será comparado o resultado de recall ou acurácia, a depender do modelo.

## 11.2 Método

A comparação abaixo será feita por meio de uma tabela, onde todos os resultados serão comparados. O modelo que obteve os melhores resultados foi o Random Forest + Word2Vec aplicado na base de dados com as features criadas no tópico 7.

## 11.3 Tabela de comparação

Abaixo é apresentado uma tabela mostrando características, resultados, prós, contras e conclusão de cada modelo descrito no documento. A coluna de características apresenta todas as particularidades do modelo e qual é a base ou modelo de vetorização que foi utilizado para o desenvolvimento do mesmo, onde serão classificados como prós e contras posteriormente.

A próxima coluna apresenta o relatório de classificação, contendo a precisão e *recall* da classe 0 (negativo) e geral, e também a acurácia do modelo. Por último é apresentada uma conclusão que expõe se o resultado foi satisfatório, se foi satisfatório mas não é a melhor opção e se não foi satisfatório.

#	Modelo	Características	Resultados	Prós	Contras	Conclusão
1	Naive Bayes com BoW	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Bag of Words</li> <li>- Considera a palavra isoladamente</li> <li>- Utiliza o Teorema de Bayes</li> <li>- Rápido e eficiente em termos de treinamento + classificação</li> </ul>	<p><b>SPRINT 3</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 50%</li> <li>- Classe 0: 0%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 41%</li> <li>- Classe 0: 0%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 56%</li> </ul> <p><b>SPRINT 4</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 50%</li> <li>- Classe 0: 0%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 36%</li> <li>- Classe 0: 0%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 50%</li> </ul>	<ul style="list-style-type: none"> <li>- Rápido e eficiente em termos de treinamento + classificação</li> </ul>	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Bag of Words</li> <li>- Considera a palavra isoladamente</li> </ul>	<ul style="list-style-type: none"> <li>- Tem valores da Classe 0 não satisfatórios (precisão e recall) - Sprint 3 e 4</li> </ul>
2	Naive Bayes com BoW - Cross Validation	<ul style="list-style-type: none"> <li>- Mesmas características do Naive Bayes</li> <li>- Sem divisão única dos dados</li> <li>- Hiperparâmetros</li> <li>- Detecção de overfitting</li> </ul>	<p><b>SPRINT 3</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 67%</li> <li>- Classe 0: 63%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 69%</li> <li>- Classe 0: 73%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 67%</li> </ul> <p><b>SPRINT 4</b></p> <p>Precisão:</p>	<ul style="list-style-type: none"> <li>- Sem divisão única dos dados</li> <li>- Hiperparâmetros</li> <li>- Detecção de overfitting</li> <li>- Rápido e eficiente em termos de treinamento + classificação</li> </ul>	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Bag of Words</li> <li>- Considera a palavra isoladamente</li> </ul>	<ul style="list-style-type: none"> <li>- Apresentou valores satisfatórios porém com um menor desempenho em relação ao modelo Modelo com novas features -</li> </ul>

			<ul style="list-style-type: none"> <li>- Geral: 58%</li> <li>- Classe 0: 64%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 64%</li> <li>- Classe 0: 75%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 63%</li> </ul>			Random Forest + Word2Vec (16)
<b>3</b>	Naive Bayes com BoW - Grid Search	<ul style="list-style-type: none"> <li>- Mesmas características do Naive Bayes</li> <li>- Combinação dos hiperparâmetros</li> </ul>	<p><b>SPRINT 3</b></p> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 71%</li> </ul> <p><b>SPRINT 4</b></p> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 66%</li> </ul>	<ul style="list-style-type: none"> <li>- Combinação dos hiperparâmetros</li> <li>- Rápido e eficiente em termos de treinamento + classificação</li> </ul>	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Bag of Words</li> <li>- Considera a palavra isoladamente</li> </ul>	Não temos informações da Classe 0 isoladamente
<b>4</b>	Random Forest com BoW	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Bag of Words</li> <li>- Usa diversas árvores de decisão</li> </ul>	<p><b>SPRINT 3</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 70%</li> <li>- Classe 0: 69%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 67%</li> <li>- Classe 0: 50%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 70%</li> </ul> <p><b>SPRINT 4</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 72%</li> <li>- Classe 0: 68%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 50%</li> <li>- Classe 0: 65%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 67%</li> </ul>	<ul style="list-style-type: none"> <li>- Usa diversas árvores de decisão</li> </ul>	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Bag of Words</li> </ul>	Apresentou valores satisfatórios porém com um menor desempenho em relação ao Modelo com novas features - Random Forest + Word2Vec (16)

5	Random Forest com BoW - Cross Validation	<ul style="list-style-type: none"> <li>- Mesmas características do Random Forest</li> <li>- Sem divisão única dos dados</li> <li>- Hiperparâmetros</li> <li>- Detecção de overfitting</li> </ul>	<p><b>SPRINT 3</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 71%</li> <li>- Classe 0: 70%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 50%</li> <li>- Classe 0: 67%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 70%</li> </ul> <p><b>SPRINT 4</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 74%</li> <li>- Classe 0: 70%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 51%</li> <li>- Classe 0: 66%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 69%</li> </ul>	<ul style="list-style-type: none"> <li>- Usa diversas árvores de decisão</li> <li>- Sem divisão única dos dados</li> <li>- Hiperparâmetros</li> <li>- Detecção de overfitting</li> </ul>	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Bag of Words</li> </ul>	<p>Apresentou valores satisfatórios porém com um menor desempenho em relação ao Modelo com novas features - Random Forest + Word2Vec (16)</p>
6	Random Forest com BoW - Grid Search	<ul style="list-style-type: none"> <li>- Mesmas características do Random Forest</li> <li>- Combinação dos hiperparâmetros</li> </ul>	<p><b>SPRINT 3</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 69%</li> <li>- Classe 0: 70%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 50%</li> <li>- Classe 0: 67%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 70%</li> </ul> <p><b>SPRINT 4</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 70%</li> <li>- Classe 0: 74%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 66%</li> </ul>	<ul style="list-style-type: none"> <li>- Usa diversas árvores de decisão</li> <li>- Combinação dos hiperparâmetros</li> </ul>	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Bag of Words</li> </ul>	<p>Apresentou valores satisfatórios porém com um menor desempenho em relação ao Modelo com novas features - Random Forest + Word2Vec (16)</p>

			<ul style="list-style-type: none"> <li>- Classe 0: 51%</li> <li>Acurácia:</li> <li>- Geral: 69%</li> </ul>			
7	Rede Neural - Sequência de palavra	<ul style="list-style-type: none"> <li>- Base: base pré processada</li> <li>- Aprende os padrões e relações entre palavras</li> <li>- Treinamento com grande volumes de dados</li> </ul>	<p><b>SPRINT 3</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 72%</li> <li>- Classe 0: 70%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 72%</li> <li>- Classe 0: 68%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 73%</li> </ul> <p><b>SPRINT 4</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 70%</li> <li>- Classe 0: 67%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 69%</li> <li>- Classe 0: 66%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 70%</li> </ul>	<ul style="list-style-type: none"> <li>- Aprende os padrões e relações entre palavras</li> <li>- Treinamento com grande volumes de dados</li> </ul>	/	<p>Apresentou valores satisfatórios porém com um menor desempenho em relação ao Modelo com novas features - Random Forest + Word2Vec (16)</p>
8	Rede Neural - Sequência de palavra - Word2Vec + CBoW	<ul style="list-style-type: none"> <li>- Base: Word2Vec com CBoW</li> <li>- Aprende os padrões e relações entre palavras</li> <li>- Treinamento com grande volumes de dados</li> </ul>	<p><b>SPRINT 3</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 73%</li> <li>- Classe 0: 70%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 74%</li> <li>- Classe 0: 70%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 74%</li> </ul> <p><b>SPRINT 4</b></p> <p>Precisão:</p>	<ul style="list-style-type: none"> <li>- Aprende os padrões e relações entre palavras</li> <li>- Treinamento com grande volumes de dados</li> </ul>	/	<p>Apresentou valores satisfatórios porém com um menor desempenho em relação ao modelo Modelo com novas features -</p>

			<ul style="list-style-type: none"> <li>- Geral: 69%</li> <li>- Classe 0: 66%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 68%</li> <li>- Classe 0: 63%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 69%</li> </ul>			<p>Random Forest + Word2Vec (16) - Sprint 3</p> <p>Apresentou valores satisfatórios porém com um menor desempenho em relação ao Modelo com novas features - Random Forest + Word2Vec (16) - Sprint 4</p>
<b>9</b>	Rede Neural - Sequência de palavra - Word2Vec + Embedding Layer	<ul style="list-style-type: none"> <li>- Base: Word2Vec com Embedding Layer</li> <li>- Aprende os padrões e relações entre palavras</li> <li>- Treinamento com grande volumes de dados</li> </ul>	<p><b>SPRINT 3</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 72%</li> <li>- Classe 0: 67%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 72%</li> <li>- Classe 0: 69%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 73%</li> </ul> <p><b>SPRINT 4</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 69%</li> <li>- Classe 0: 68%</li> </ul> <p>Recall:</p>	<ul style="list-style-type: none"> <li>- Aprende os padrões e relações entre palavras</li> <li>- Treinamento com grande volumes de dados</li> </ul>	/	<p>Apresentou valores satisfatórios porém com um menor desempenho em relação ao Modelo com novas features - Random Forest + Word2Vec (16)</p>

			<ul style="list-style-type: none"> <li>- Geral: 68%</li> <li>- Classe 0: 64%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 69%</li> </ul>			
<b>10</b>	Rede Neural sem embedding - Word2Vec	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Word2Vec</li> <li>- Aprendizado end-to-end</li> </ul>	<p><b>SPRINT 3</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 59%</li> <li>- Classe 0: 53%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 59%</li> <li>- Classe 0: 54%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 61%</li> </ul> <p><b>SPRINT 4</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 53%</li> <li>- Classe 0: 40%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 54%</li> <li>- Classe 0: 62%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 53%</li> </ul>	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Word2Vec</li> <li>- Aprendizado end-to-end</li> </ul>	/	Não apresentou valores satisfatórios
<b>11</b>	Random Forest	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Word2Vec</li> <li>- Usa diversas árvores de decisão</li> </ul>	<p><b>SPRINT 3</b></p> <p>Precisão:</p> <ul style="list-style-type: none"> <li>- Geral: 64%</li> <li>- Classe 0: 56%</li> </ul> <p>Recall:</p> <ul style="list-style-type: none"> <li>- Geral: 64%</li> <li>- Classe 0: 54%</li> </ul> <p>Acurácia:</p> <ul style="list-style-type: none"> <li>- Geral: 66%</li> </ul> <p><b>SPRINT 4</b></p>	<ul style="list-style-type: none"> <li>- Modelo de vetorização: Word2Vec</li> <li>- Usa diversas árvores de decisão</li> </ul>	/	Apresentou valores satisfatórios porém com um menor desempenho em relação ao Modelo com novas features -

			Precisão: - Geral: 61% - Classe 0: 55% Recall: - Geral: 60% - Classe 0: 54% Acurácia: - Geral: 61%			Random Forest + Word2Vec (16)
12	XGBoost + BoW	- Modelo de vetorização: BoW - Algoritmo de Gradient Boosting - Técnicas para evitar overfitting	<b>SPRINT 4</b> Precisão: - Geral: 69% - Classe 0: 69% Recall: - Geral: 68% - Classe 0: 59% Acurácia: - Geral: 69%	- Algoritmo de Gradient Boosting - Técnicas para evitar overfitting	- Modelo de vetorização: BoW	Apresentou valores satisfatórios porém com um menor desempenho em relação ao Modelo com novas features - Random Forest + Word2Vec (16)
13	XGBoost + Word2Vec	- Modelo de vetorização: Word2Vec - Algoritmo de Gradient Boosting - Técnicas para evitar overfitting	<b>SPRINT 3</b> Precisão: - Geral: 63% - Classe 0: 55% Recall: - Geral: 63% - Classe 0: 49% Acurácia: - Geral: 65% <b>SPRINT 4</b>	- Algoritmo de Gradient Boosting - Técnicas para evitar overfitting	- Modelo de vetorização: Word2Vec	Não apresentou valores satisfatórios

			Precisão: - Geral: 57% - Classe 0: 49% Recall: - Geral: 57% - Classe 0: 50% Acurácia: - Geral: 58%			
14	Modelo com Novas Features - Random Forest + TF-IDF	- Modelo de vetorização: TF-IDF - Random Forest é um método de aprendizado que combina várias árvores de decisão para fazer previsões. - Base de dados com novas features que são relevantes para o projeto.	<b>SPRINT 5</b> Precisão: - Geral: 72% - Classe 0: 72% Recall: - Geral: 70% - Classe 0: 62% Acurácia: - Geral: 72%	- Precisão: o Random Forest geralmente produz resultados precisos, pois combina as previsões de várias árvores de decisão. - A maioria das probabilidades tendem a reduzir o viés e a variância, originada em um modelo mais estável e confiável.	- Dificuldade de interpretação: devido à combinação de várias árvores de decisão. - A interpretação dos resultados do Random Forest pode ser mais complexa do que com modelos mais simples.	Apresentou valores satisfatórios porém com um menor desempenho em relação ao Modelo com novas features - Random Forest + Word2Vec (16)
15	Modelo com novas features - Rede Neural +	- Modelo de vetorização: TF-IDF - A rede neural é uma abordagem avançada de aprendizado de	<b>SPRINT 5</b> Precisão: - Geral: 72% - Classe 0: 71% Recall:	- Capacidade de aprendizado: As redes neurais têm a capacidade de aprender com	- Necessidade de dados de treinamento: As redes neurais	Apresentou valores satisfatórios porém com um menor

	TF-IDF	máquina amplamente utilizada em problemas de classificação. - Base de dados enriquecida com novas features relevantes.	- Geral: 71% - Classe 0: 62% Acurácia: - Geral: 73%	dados e melhorar seu desempenho ao longo do tempo. - Podem ser treinadas em conjuntos de dados grandes e complexos para identificar padrões e realizar tarefas específicas com alta precisão.	geralmente exigem grandes dados de treinamento para alcançar um bom desempenho. - Coletar, rotular e preparar conjuntos de dados extensos pode ser um processo demorado e caro.	desempenho em relação ao modelo Modelo com novas features - Random Forest + Word2Vec (16)
16	Modelo com novas features - Random Forest + Word2Vec	- Modelo de vetorização: Word2Vec - Usa diversas árvores de decisão	<b>SPRINT 5</b> Precisão: - Geral: 77% - Classe 0: 79% Recall: - Geral: 77% - Classe 0: 80% Acurácia: - Geral: 77%	- Modelo de vetorização: Word2Vec - Usa diversas árvores de decisão	/	Apresentou resultados satisfatórios e é o melhor.

## 11.5 Conclusão

Vale ressaltar que é importante considerar outros aspectos além do recall ao avaliar a adequação de um modelo para uma tarefa específica. A simplicidade e a eficiência computacional são fatores relevantes a serem considerados. Os modelos acima foram comparados entre si, pois eles trabalham de formas diferentes quando se tem base diferente.

## 12. Arquitetura macro

A arquitetura macro é uma estrutura abrangente que define a organização geral e a distribuição funcional dos componentes de um sistema complexo. Essa arquitetura fornece uma visão ampla e geral do sistema, identificando os principais subsistemas ou módulos e as interações entre eles.

O primeiro *Step* está com cor diferente pois ele é externo, que é a obtenção da fonte de dados, que nesse caso é um arquivo CSV, por isso o que será mandado para o próximo, é o arquivo. O *Step* 2, que recebe como input o CSV (1), é o processo de exploração de dados, onde é *plotado* diversos gráficos, de 3 tipos: pizza, barra e linha, além disso é realizado alguns processos de estatística. O output é o mesmo csv que o processo recebeu, e ele manda os metadados para o próximo.

O terceiro *Step* recebe o CSV (1) e realiza o pré processamento, que consiste em duas partes: 1. Pipeline, onde é realizado a criação das features, tokenização, remoção de stopwords, tratamento de emojis, remoção de alfanuméricos, tratamento abreviações e a lematização; 2. modelos de vetorização: BoW, Word2Vec e TF-IDF. O output desse processo é o CSV (2).

O quarto *Step* recebe o CSV (2) e realiza o treinamento de diversos modelos, aplicando os 3 modelos de vetorização, além disso as métricas são impressas e é utilizado o balanceamento de métricas. O output será a IA, o modelo que foi desenvolvido e escolhido. O quinto *Step* é o serviço de API, que realiza os processos do *step* 3 e 4 e gera o dashboard, próximo *step*, que recebe a API como input e retorna um front end com espaço para inserir o arquivo, o resultado e gráficos.



Figura 46: Arquitetura Macro da Solução

## 13. Diagrama

O diagrama de implantação é um dos diagramas da Linguagem de Modelagem Unificada (UML) que tem como objetivo visualizar e descrever a distribuição física dos elementos de um sistema em tempo de execução. Ele representa como os componentes de software, hardware e de rede são implantados e interagem entre si em um ambiente de produção.

A máquina Virtual, que nesse caso é o Google Collaboratory, recebe o CSV do Instagram, Pré-processamento, Vetorização, Classificação do modelo e a escolha do modelo. O csv vem do Servidor externo, do Instagram, que são os dados dos comentários de um ou mais posts. Esses dados também são enviados para o computador do usuário, onde o usuário pode colocar na aplicação WEB, essa aplicação é gerada pela API. O servidor local tem a API, que realiza todos os processos citados na máquina virtual, além disso ela recebe o modelo escolhido para poder treinar com outros dados e manda essas informações para aplicação WEB.

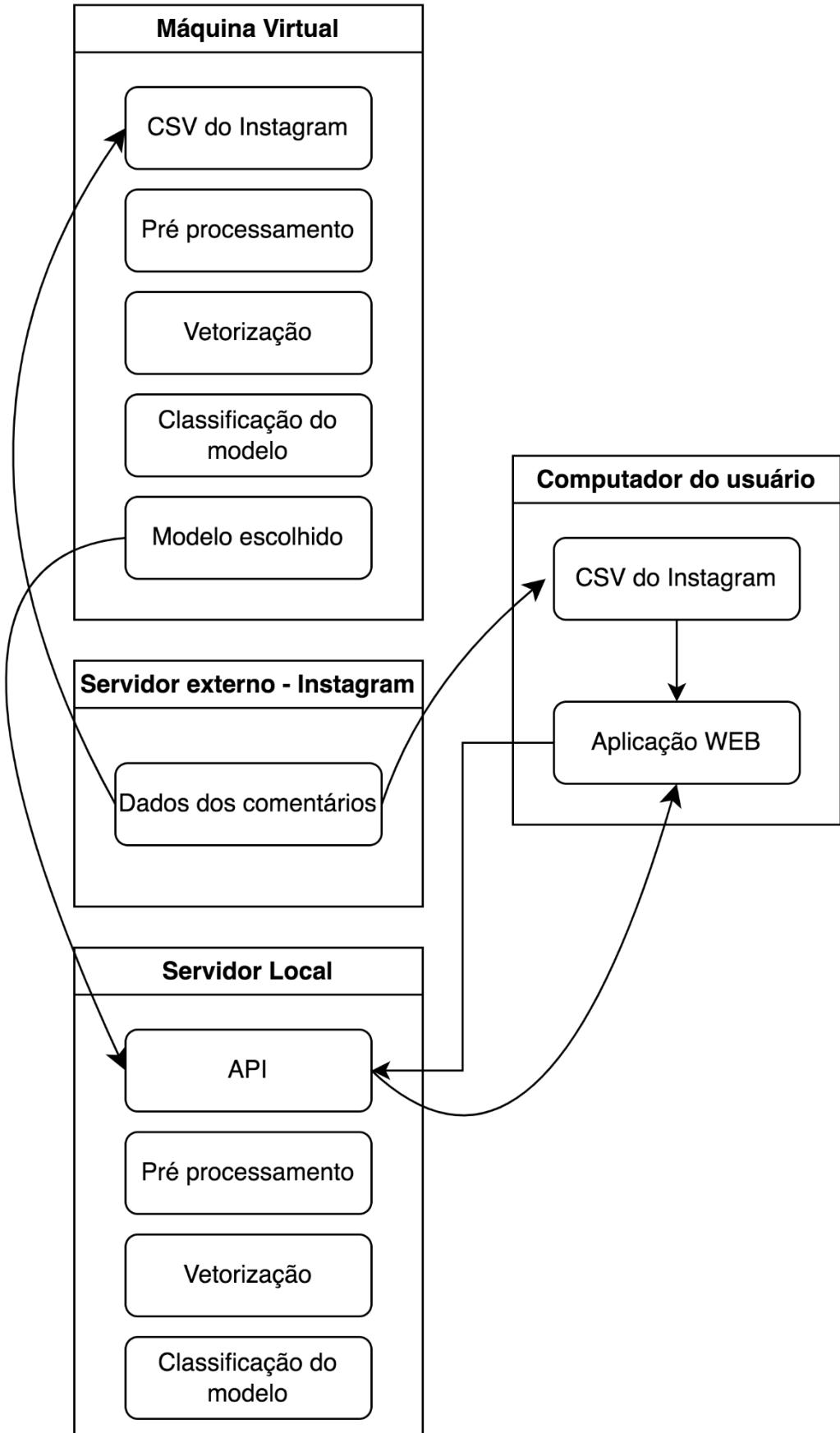


Figura 47: Diagrama de implantação UML

## **Referências:**

Utilize esta seção para anexar materiais extras que julgar necessário.

Fontes para a seção 1.3.3:

<http://www.swge.inf.br/ANALIS/CBCM2017/PDF/CBCM2017-0039.PDF>

Naive Bayes: [https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)

BoW: [https://scikit-learn.org/stable/modules/feature\\_extraction.html#text-feature-extraction](https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction)

## **Anexos:**

# **Word2Vec com CBOW**

## **Introdução**

O Word2Vec é um modelo de aprendizado de representação de palavras, que captura eficientemente as relações semânticas entre palavras com base em seu contexto. Nesse caso é utilizado esse modelo com o CBOW (Continuous Bag-of-Words), que já é um modelo treinado.

## **Método**

Nesse caso, como é um modelo pré-treinado, o único processo que foi necessário foi o Label Encoding, onde é transformado a coluna ‘sentimento’ em valores numéricos: -1, 0 e 1. O output é uma tabela com: 1 coluna com a frase do *post*, 50 colunas de vetores e 1 coluna de sentimento.

## **Resultados**

A primeira etapa da realização do Word2Vec com CBOW é ler o arquivo do modelo já treinado.

```
cbow = 'caminho_do_arquivo'  
model_cbow = KeyedVectors.load_word2vec_format(cbow)
```

Para fins acadêmicos, o modelo acima foi testado com duas palavras para que se pudesse visualizar os vetores delas e provar que funciona. Abaixo está o output encontrado.

```
wordvec_test = model_cbow['projeto']  
wordvec_test
```

```

        output: array([-0.074174, -0.152088,  0.086627, -0.224567,
0.362562,  0.130683, -0.089179, -0.086973,  0.309501,  0.004112,
-0.308202,  0.351789, -0.477863,  0.050276,  0.213283,  0.159895,
-0.285545, -0.08832 , -0.015449,  0.014816, -0.613861,  0.502556,
0.021688,  0.369492,  0.280691,  0.016868,  0.105584, -0.180754,
-0.078456,  0.148032,  0.36293 , -0.011634,  0.412191, -0.009049,
0.010404,  0.131242, -0.032483, -0.133067, -0.063802,  0.434015,
-0.214768, -0.072132,  0.045601, -0.368866,  0.502808,  0.048293,
-0.254894,  0.142581, -0.075066,  0.015646], dtype=float32)

wordvec_test = model_cbow[ 'banco' ]
wordvec_test

array([ 1.81041e-01,  1.07700e-01, -1.04667e-01,  2.43361e-01,
       6.06380e-02,  3.92829e-01, -3.33944e-01, -3.81778e-01,
      1.42200e-01,  8.59360e-02, -1.16615e-01,  3.95722e-01,
     -6.12684e-01, -7.68980e-02,  3.34396e-01,  8.11270e-02,
     -5.17700e-02, -3.21950e-01, -6.91509e-01, -3.31210e-01,
     -5.43213e-01,  6.09881e-01,  2.43700e-01,  3.73240e-02,
      1.16518e-01,  1.78859e-01, -3.78839e-01,  1.27430e-01,
      1.94497e-01,  7.32000e-04,  3.14395e-01, -2.04550e-01,
      5.34431e-01, -5.55100e-03,  3.52343e-01, -4.92000e-02,
     -1.38384e-01,  2.31630e-02, -3.40013e-01,  5.00201e-01,
     -1.14170e-02, -1.29925e-01, -6.12800e-03, -1.80481e-01,
      1.99391e-01,  1.37645e-01, -7.66434e-01, -2.26784e-01,
     -6.16110e-02,  9.05920e-02], dtype=float32)

```

A função abaixo, chamada de `vetorizando()` recebe um modelo de vetores de palavras treinado e um `dataFrame` (`df`) contendo um texto já pré tratado. Ela verifica se a palavra na frase está no modelo de vetores, caso tenha, o vetor é adicionado a uma lista. A seguir, a função calcula o vetor médio dessas palavras encontradas na sentença, caso a palavra não seja encontrada, é criada uma lista de 100 elementos "None". A função armazena a sentença original e os primeiros 50 elementos do vetor médio, além de criar o `df_vec`, uma nova base de dados.

O `dataframe` original `df` é modificado adicionando uma coluna `sentimentoNumerico` que transforma as categorias de sentimento: "NEGATIVE", "POSITIVE" e "NEUTRAL", para valores numéricos: -1, 1 e 0, respectivamente. Em seguida, a função adiciona a coluna `sentimento` a `df_vec` com base em `df[ 'sentimentoNumerico' ]`. Em seguida, ela remove quaisquer linhas que contenham valores ausentes no `df_vec` e retorna o `dataframe`.

```
def create_sentence_vector(model, df):
```

```

sentence_table = []
for sentence in df['texto_tratado']:
    word_vectors = [model[word] for word in sentence if word in model]
    if len(word_vectors) > 0:
        sentence_vector = sum(word_vectors) / len(word_vectors)
    else:
        sentence_vector = [None] * 100
    sentence_table.append((sentence, *sentence_vector[:50]))
column_labels = ['Frase']
for i in range(50):
    column_labels.append(f'Vetor{i+1}')
df_vec = pd.DataFrame(sentence_table, columns=column_labels)
df["sentimentoNumerico"] = df["sentimento"].replace({'NEGATIVE': -1,
'POSITIVE': 1, 'NEUTRAL': 0})
df_vec.set_index(df["sentimentoNumerico"].index, inplace=True)
df_vec['sentimento'] = df["sentimentoNumerico"]
df_vec = df_vec.dropna()

return df_vec

```

O código abaixo testa a função definida e tem como output a imagem abaixo.

```

df_vec = create_sentence_vector(model_cbow, df)
df_vec

```

	Frase	Vetor1	Vetor2	Vetor3	Vetor4	Vetor5	Vetor6	Vetor7	Vetor8	Vetor9	...	Vetor42	Vetor43	Vetor44	Vetor45	Vetor46	Vetor47	Vetor48	Vetor49	Vetor50	sentimento
0	[alvarez, marsal, estar, conosco, sportinmet,...	0.216308	-0.126807	0.240276	-0.070348	-0.016067	0.209098	0.073239	0.057978	0.061847	...	0.021205	-0.104079	0.155380	0.091749	-0.043675	0.158847	-0.037001	0.020596	0.184101	1
1	[btgpactual, with, makerepost, entender, impac...	0.225879	-0.123276	0.215921	-0.052563	-0.007740	0.207449	0.073616	0.037889	0.062654	...	0.003267	-0.071278	0.156911	0.084502	-0.004404	0.159728	-0.029227	0.029953	0.190338	1
2	[minuto, touro, ouro]	0.265227	-0.068285	0.152235	-0.044329	-0.102729	0.141363	0.092800	0.113174	0.015783	...	0.078032	-0.202677	0.155750	0.062291	0.007038	0.134673	0.014635	0.034189	0.345674	2
3	[sim]	0.166258	-0.029796	0.204045	-0.297490	0.046077	0.140783	0.035251	-0.174491	0.211817	...	0.068639	-0.092451	0.308218	-0.034692	-0.032851	-0.028724	-0.068701	0.011158	0.256413	1
4	[querer, saber, btg, banking, próprio, blg, ad...]	0.192599	-0.164825	0.279021	-0.053466	-0.041365	0.211765	0.074588	0.092593	0.094428	...	0.088143	-0.130285	0.164846	0.077747	-0.068860	0.181872	-0.078526	-0.003717	0.197040	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9472	[excelente, explicação]	0.190917	-0.133475	0.241675	-0.053180	0.067256	0.201138	0.034109	-0.078718	-0.066131	...	-0.082151	0.016113	0.154881	0.068700	-0.004302	0.079717	-0.028388	-0.017448	0.188785	2
9473	[atender, telefone, amor, deus]	0.188641	-0.119377	0.199339	-0.105448	0.023176	0.178837	0.069476	-0.004494	0.034710	...	0.034035	-0.126673	0.165176	0.080313	-0.024160	0.118848	-0.005052	0.067053	0.215656	2
9474	[saber, qual, 10, grande, fisi, mercado, selec...]	0.213684	-0.135475	0.219169	-0.072920	-0.011112	0.203863	0.063960	0.038996	0.069897	...	0.034963	-0.097084	0.175341	0.088866	-0.047218	0.153991	-0.030393	0.021887	0.179608	2
9475	[erro, financeiro, eliminar, antes, 30, ano, ...]	0.212846	-0.112742	0.221387	-0.078302	-0.032960	0.218673	0.071489	0.038917	0.037444	...	0.022441	-0.094323	0.150688	0.081716	-0.028744	0.145769	-0.029518	0.024455	0.191446	1
9476	[porque, morning, call, aparecer, spotify, atu...]	0.225402	-0.137580	0.274048	-0.044282	0.008441	0.249057	0.043162	0.060058	0.049930	...	0.007607	-0.097939	0.163142	0.100946	-0.034086	0.136629	-0.024904	0.030681	0.170319	0

Figura 48: Output do Word2Vec + CBOW

## Conclusão

Em conclusão, o método Word2Vec, em particular com o CBOW, é uma técnica eficaz para a representação de palavras em espaços vetoriais contínuos. O processo de vetorização de um texto usando o CBOW envolve a criação de vetores de palavras e o cálculo de vetores médios para as sentenças.

# Naive Bayes + Word2Vec com CBOW

## Introdução

O Naive Bayes é um classificador probabilístico amplamente utilizado em problemas de aprendizado de máquina. Ele se baseia no teorema de Bayes para estimar a probabilidade condicional das classes com base em evidências fornecidas pelas características dos dados. Junto a isso, o Word2Vec com CBOW é um algoritmo de aprendizado de representação de palavras que visa capturar as relações semânticas e sintáticas entre as palavras em um corpus de texto.

## Método

Na abordagem que combina Naive Bayes e Word2Vec com CBOW, o Word2Vec é primeiro treinado em um corpus de texto já treinado para aprender as representações vetoriais das palavras. Em seguida, essas representações vetoriais são utilizadas como características no modelo Naive Bayes. Por isso, o *dataframe* referenciado neste tópico será o gerado no tópico anterior.

## Resultados

No exemplo a seguir, é considerado um conjunto de dados representado por um *dataframe* df\_vec. A coluna sentimento representa os valores numéricos dos sentimentos associados a cada texto, enquanto as colunas restantes, de 1 a 50, contêm as representações vetoriais das palavras. O código abaixo define a variável-alvo que será usada no treinamento do modelo de classificação.

```
target = df_vec[ 'sentimento' ]
```

Para selecionar os recursos relevantes para o modelo de classificação, é utilizado o código abaixo, que seleciona todas as linhas do *dataframe* e as colunas de 1 a 50, que correspondem às representações vetoriais das palavras. A figura a seguir demonstra o *output*.

```
feature = df_vec.iloc[:,1:50]  
feature
```

	Vetor1	Vetor2	Vetor3	Vetor4	Vetor5	Vetor6	Vetor7	Vetor8	Vetor9	Vetor10	...	Vetor40	Vetor41	Vetor42	Vetor43	Vetor44	Vetor45	Vetor46	Vetor47	Vetor48	Vetor49
0	0.216308	-0.126807	0.240276	-0.070348	-0.016067	0.209095	0.073239	0.057978	0.061847	0.172586	...	0.076437	-0.272009	0.021205	-0.104079	0.155380	0.091749	-0.043675	0.158847	-0.037001	0.020596
1	0.225879	-0.123278	0.215921	-0.052553	-0.007740	0.207449	0.073616	0.037889	0.062654	0.167859	...	0.092294	-0.304136	0.003267	-0.071278	0.158911	0.084502	-0.004404	0.159728	-0.029227	0.029953
2	0.265227	-0.068285	0.152235	-0.044329	-0.102729	0.141363	0.092800	0.113174	0.015783	0.202198	...	-0.008447	-0.193025	0.078032	-0.202677	0.155750	0.062291	0.007038	0.134573	0.014635	0.034189
3	0.166258	-0.029798	0.204045	-0.297490	0.046077	0.140763	0.035251	-0.174491	0.211817	0.288314	...	0.183434	-0.415105	0.065839	-0.092451	0.308218	-0.034692	-0.032851	-0.028724	-0.068701	0.011158
4	0.192599	-0.164825	0.279021	-0.053466	-0.041365	0.211765	0.074588	0.092593	0.094428	0.177284	...	0.048799	-0.253769	0.088143	-0.130285	0.164846	0.077747	-0.068860	0.181872	-0.076526	-0.003717
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9472	0.190917	-0.133475	0.241675	-0.053180	0.067256	0.201138	0.034109	-0.078718	-0.066131	0.187608	...	0.014565	-0.321192	-0.082151	0.016113	0.154861	0.068700	-0.004302	0.079717	-0.028388	-0.017448
9473	0.188641	-0.119377	0.199339	-0.105448	0.023176	0.178837	0.069476	-0.004494	0.034710	0.150081	...	0.071114	-0.194863	0.034035	-0.126673	0.165176	0.080313	-0.024160	0.118848	-0.003502	0.087053
9474	0.213684	-0.135475	0.219169	-0.072920	-0.014112	0.203863	0.063960	0.038996	0.069897	0.170876	...	0.081039	-0.298570	0.034963	-0.097084	0.175341	0.088866	-0.047218	0.153991	-0.030393	0.021887
9475	0.212646	-0.112742	0.221387	-0.078302	-0.032960	0.218673	0.071489	0.038917	0.037444	0.146022	...	0.065854	-0.252245	0.022441	-0.094323	0.150688	0.081716	-0.028744	0.145769	-0.029518	0.024455
9476	0.225402	-0.137580	0.274046	-0.044282	0.008441	0.249057	0.043162	0.080058	0.049930	0.108385	...	0.085501	-0.235024	0.007607	-0.097939	0.163142	0.100946	-0.034086	0.138629	-0.024904	0.030681

Figura 49: Output Naive Bayes + Word2Vec com CBOW

No código abaixo, é realizado um particionamento dos dados em conjuntos de treinamento e teste usando a função `train_test_split`, ela divide um conjunto de dados em subconjuntos para fins de treinamento e avaliação de modelos de aprendizado de máquina.

```
X_train, X_test, y_train, y_test = train_test_split(feature,  
target, test_size=0.2, random_state=42)
```

A primeira linha cria um objeto do tipo `GaussianNB`, que é o classificador Naive Bayes Gaussiano. Na segunda linha o modelo é treinado usando o método `fit()`, o conjunto de treinamento `X_train`, já definido, é fornecido como as características e `y_train.values.ravel()` como a variável-alvo. O método `values.ravel()` é utilizado para converter `y_train` em um array unidimensional, necessário para o treinamento do modelo. A função `predict()` é usada para fazer previsões com base nos dados de teste e as previsões resultantes são armazenadas na variável `Y_pred`. Por fim, a função `classification_report` é utilizada para imprimir o relatório de classificação, comparando as previsões `Y_pred` com as verdadeiras classes do conjunto de teste `y_test` e exibe métricas como precisão, recall, F1-score e suporte para cada classe, imagem presente após o código abaixo.

```
clf = GaussianNB()  
clf = clf.fit(X_train,y_train.values.ravel())  
Y_pred = clf.predict(X_test)  
print(classification_report(y_test, Y_pred))
```

### Relatório de classificação:

	precision	recall	f1-score	support
0	0.29	0.69	0.41	417
1	0.74	0.45	0.56	855
2	0.37	0.24	0.29	624
accuracy			0.43	1896
macro avg	0.47	<b>0.46</b>	0.42	1896
weighted avg	0.52	0.43	0.44	1896

A função `accuracy_score(y_test, Y_pred)` abaixo é usada para comparar as previsões feitas pelo modelo (`Y_pred`) com as verdadeiras classes do conjunto de teste (`y_test`), definindo à variável `acc_score`. Para exibir o resultado de forma mais legível, a porcentagem da precisão é formatada com a função `format()` do Python. A expressão

"{:.2%}" .format(acc\_score) indica que a formatação com 2 casas decimais e o símbolo de porcentagem.

```
acc_score = accuracy_score(y_test, Y_pred)
format_output = "{:.2%}".format(acc_score)
print("Precisão final de :",format_output)

output: Precisão final de : 43.35%
```

## Conclusão

O modelo Word2Vec com CBOW é capaz de capturar informações contextuais e semânticas das palavras, fornecendo representações vetoriais que preservam relações entre as palavras. Essas representações vetoriais são úteis para entender a semelhança e a estrutura do texto, enriquecendo a qualidade das características utilizadas pelo Naive Bayes.

# Word2Vec com o corpus

## Introdução

Uma das principais técnicas usadas no Word2Vec é a camada de embedding. Essa camada é responsável por mapear palavras individuais para vetores de números reais em um espaço de alta dimensão. Cada palavra é representada por um vetor denso, onde as dimensões desse vetor capturam informações contextuais e semânticas sobre a palavra.

## Método

Nesse caso o Word2Vec é treinado diretamente no corpus do *dataframe*, ao invés de já ter um modelo pré treinado. Para realizar esse processo, é utilizado a biblioteca gensim.models, importando o Word2Vec.

## Resultados

O código abaixo se inicia com a importação da classe Word2Vec da biblioteca Gensim, que é usada para treinar o modelo. Em seguida, a função `train_word2vec()` recebe dois parâmetros: `df`, que é o *dataframe*, e `column_name`, que é o nome da coluna que contém as frases *tokenizadas*.

A variável `sentences` é inicializada com uma lista das frases *tokenizadas* presentes na coluna especificada. A função `tolist()` é usada para converter os valores da coluna em uma lista.

Em seguida, o modelo é treinado usando a função Word2Vec do Gensim. O parâmetro `sentences` é passado para representar o corpus de treinamento, já o parâmetro `min_count=1` indica que todas as palavras devem ter pelo menos uma ocorrência para serem consideradas no treinamento. E por fim, o modelo é retornado.

```

from gensim.models import Word2Vec

def train_word2vec(df, column_name):
    sentences = df[column_name].tolist()
    model = Word2Vec(sentences, min_count=1)
    return model

```

A primeira função abaixo recebe dois parâmetros: `model`, que é o modelo Word2Vec treinado, e `sentence`, que é uma lista de palavras *tokenizadas* representando uma frase. A seguir, é inicializada uma lista vazia chamada `vectors` para armazenar os vetores de palavras.

Em seguida, ocorre um loop sobre cada palavra na lista `sentence`. A propriedade `model.wv` verifica se a palavra está presente no vocabulário do modelo Word2Vec, e caso esteja presente, o seu vetor é obtido usando `model.wv[word]` e adicionado à lista `vectors`.

Depois de iterar todas as palavras da frase, é feita uma verificação se a lista `vectors` contém algum vetor, e se houver, eles são somados usando `np.sum(vectors, axis=0)` para obter um vetor que representa a frase como um todo. Esse vetor é então normalizado dividindo-o pelo número de palavras na frase (`len(sentence)`) para obter a média dos vetores. Porém, caso a lista `vectors` esteja vazia, é retornado um vetor de zeros com o mesmo tamanho dos vetores do modelo (`np.zeros(model.vector_size)`).

```

def get_word_vectors(model, sentence):
    vectors = []
    for word in sentence:
        if word in model.wv:
            vectors.append(model.wv[word]) # Append na lista de
    vetores
    if vectors:
        return np.sum(vectors, axis=0)/len(sentence) # Soma dos
    vetores para cada frase
    else:
        return np.zeros(model.vector_size)

```

A segunda função recebe três parâmetros: `df`, que é um dataframe contendo os dados, `column_name`, que é o nome da coluna, e `model`, que é o modelo Word2Vec treinado. Primeiramente, a função converte as frases *tokenizadas* da coluna especificada em uma lista chamada `sentences`, e em seguida, ocorre um loop sobre cada frase na lista. Para cada frase, a função `get_word_vectors` é chamada para obter o vetor representativo da frase.

Os vetores resultantes para cada frase são armazenados na lista `vectors`, que é construída utilizando uma compreensão de lista, onde cada elemento da lista é o vetor. Após iterar por todas as frases, é criado um novo *dataframe* chamado `df_vectors`, onde

cada coluna representa uma dimensão do vetor. O número de colunas é determinado pelo tamanho dos vetores do modelo (`model.vector_size`). O `df_vectors` é então concatenado ao dataframe original `df` usando a função `pd.concat`, resultando no dataframe final `df_word2vec`.

```
def create_word2vec_dataframe(df, column_name, model):
    sentences = df[column_name].tolist()
    vectors = [get_word_vectors(model, sentence) for sentence in sentences]
    df_vectors = pd.DataFrame(vectors, columns=[f"Vetor{i}" for i in range(model.vector_size)])
    df_word2vec = pd.concat([df, df_vectors], axis=1)
    return df_word2vec
```

Por fim, as funções são testadas no *dataframe* original.

## Conclusão

O código apresentado ilustra como treinar um modelo Word2Vec usando a biblioteca Gensim e como criar um dataframe com vetores de palavras para frases *tokenizadas* usando esse modelo. E o Word2Vec com a camada de embedding é uma abordagem poderosa para aprender representações vetoriais de palavras em tarefas de processamento de linguagem natural.

# Naive Bayes + Word2Vec com o corpus

## Introdução

Da mesma forma que o Naive Bayes funciona com o Word2Vec + CBOW é o jeito que funciona com esse modelo, o Word2Vec com o corpus. A grande diferença é que o segundo não utiliza um modelo já pré treinado, o que pode ou não melhorar o resultado final.

## Método

Na abordagem que combina Naive Bayes e Word2Vec com o corpus, o Word2Vec é treinado com o corpus do *dataset*. Em seguida, essas representações vetoriais são utilizadas como características no modelo Naive Bayes.

## Resultados

A primeira etapa a ser feita é a separação entre teste e treino, como mostra os códigos abaixo. O primeiro, cria a variável `target`, que armazena a coluna chamada `sentimentoNumerico` do `df_word2vec`, onde cada valor nessa coluna representa o sentimento atribuído. Essa variável `target` será usada posteriormente como o objetivo de indicar qual sentimento é esperado para cada instância do corpus.

```
target = df_word2vec['sentimentoNumerico']
```

O segundo, cria a variável `feature`, que contém um recorte do `df_word2vec`. Mais especificamente, todas as linhas e todas as colunas da posição 2 até a posição 101 são selecionadas. Cada valor dessa variável `feature` representa um componente do vetor Word2Vec associado a uma palavra específica do texto.

```
feature = df_word2vec.iloc[:, 2:102]
```

	Vetor0	Vetor1	Vetor2	Vetor3	Vetor4	Vetor5	Vetor6	Vetor7	Vetor8	Vetor9	...	Vetor90	Vetor91	Vetor92	Vetor93	Vetor94	Vetor95	Vetor96	Vetor97	Vetor98	Vetor99
0	0.428724	0.432991	-0.069373	-0.390625	0.167993	0.150017	-0.125445	-0.307210	-0.142480	0.231476	...	0.158695	-0.021530	0.064255	0.200677	-0.343568	-0.150278	0.230582	0.138340	0.116164	0.015622
1	0.403599	0.396830	-0.090693	-0.378203	0.168807	0.165952	-0.137153	-0.299038	-0.139476	0.258859	...	0.149624	-0.021719	0.076423	0.185731	-0.343840	-0.103155	0.225854	0.109737	0.125461	0.038858
2	0.411561	0.379824	-0.028566	-0.463731	0.139106	0.132234	-0.182196	-0.331240	-0.126577	0.281796	...	0.159725	0.066124	0.105851	0.169309	-0.307872	0.018729	0.325074	0.177072	0.128487	-0.057827
3	0.043417	0.729222	-0.035982	-0.106024	0.147785	0.231467	-0.147459	-0.076083	-0.280814	0.387266	...	-0.018889	-0.232910	0.265774	0.039684	-0.324337	0.001886	0.156238	-0.021586	0.092202	0.265736
4	0.330063	0.426375	-0.119240	-0.384473	0.195382	0.153858	-0.135111	-0.289961	-0.199326	0.239902	...	0.185977	-0.036598	0.019594	0.171955	-0.361425	-0.090451	0.166882	0.058767	0.070553	0.079802
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9472	0.381040	0.407306	-0.065901	-0.504870	0.080631	0.252352	-0.074826	-0.285107	-0.200422	0.303381	...	0.134118	0.041042	0.041759	0.281209	-0.357967	-0.192821	0.302168	0.096228	0.154049	0.025645
9473	0.395917	0.530222	-0.039731	-0.400390	0.122958	0.180124	-0.127471	-0.352738	-0.170759	0.313089	...	0.110479	0.000207	0.061072	0.156445	-0.315394	-0.072328	0.232020	0.139051	0.095105	0.036066
9474	0.428404	0.398813	-0.022072	-0.430962	0.147732	0.159678	-0.106857	-0.310689	-0.122832	0.232796	...	0.149765	-0.004145	0.061801	0.212190	-0.354424	-0.174809	0.281553	0.152908	0.108495	0.006989
9475	0.472402	0.341054	-0.040685	-0.382366	0.148118	0.116719	-0.172834	-0.373485	-0.039884	0.235572	...	0.251312	0.017855	0.042332	0.199385	-0.370954	-0.160366	0.176536	0.147924	0.046004	-0.067451
9476	0.426157	0.487541	-0.017883	-0.453064	0.102162	0.114602	-0.153672	-0.347261	-0.136811	0.277927	...	0.149623	0.020280	0.048169	0.159178	-0.410129	-0.087800	0.274279	0.128100	0.093122	0.044570

Figura 50: Output feature

Os códigos a seguir são os mesmos utilizados anteriormente para a separação de treino e teste no Naive Bayes e avaliação do modelo, e a seguir há uma imagem com os resultados obtidos.

```
X_train, X_test, y_train, y_test = train_test_split(feature,
target, test_size=0.2, random_state=42)
```

```
clf = GaussianNB()
clf = clf.fit(X_train,y_train.values.ravel())
Y_pred = clf.predict(X_test)
print(classification_report(y_test, Y_pred))
```

### Relatório de classificação:

	precision	recall	f1-score	support
0	0.30	0.77	0.43	417
1	0.77	0.45	0.57	855
2	0.34	0.17	0.22	624
accuracy			0.43	1896
macro avg	0.47	0.46	0.41	1896
weighted avg	0.52	0.43	0.42	1896

## Conclusão

Em conclusão, o código apresentado realiza a preparação dos dados e a divisão do conjunto de características e rótulos para aplicação do método Naive Bayes com o Word2Vec. O objetivo desse código é realizar a classificação ou análise de texto com base nos vetores Word2Vec gerados a partir do corpus.

## Rede Neural sem embedding - Word2Vec

### Introdução

Uma rede neural é um modelo computacional inspirado no funcionamento do cérebro humano. É composta por unidades chamadas neurônios, que estão organizadas em camadas interconectadas. Cada neurônio recebe entradas ponderadas, realiza um cálculo e produz uma saída.

Através de um processo de treinamento, uma rede neural pode aprender a mapear entradas para saídas, permitindo realizar tarefas como classificação, regressão, processamento de linguagem natural, entre outras.

Neste contexto, estamos discutindo uma Rede Neural sem a utilização de embedding, mas com Word2vec. O Word2Vec é um algoritmo para representação de palavras como vetores densos, capturando as relações semânticas e contextuais entre elas.

### 10.8.2 Método

#### 10.8.2.1 Cross Validation

O método faz o uso de validação cruzada (cross-validation) para avaliar o desempenho da rede neural. A validação cruzada é uma técnica utilizada para avaliar a capacidade de generalização de um modelo. Consiste em dividir o conjunto de dados em várias partes chamadas de "folds". O modelo é treinado em uma combinação de folds e testado nos folds restantes. Esse processo é repetido várias vezes, alterando os folds de treinamento e teste, e os resultados são combinados para obter uma estimativa mais robusta do desempenho do modelo.

#### 10.8.2.2 Construção do modelo

##### 1) Definição do modelo:

O modelo possui três camadas: uma camada de entrada com 100 neurônios, uma camada oculta com 32 neurônios e uma camada de saída com 3 neurônios para classificação nas categorias negative, neutral e positive.

```
# Criando o modelo da rede neural
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(100,))) # Camada de
# Entrada com 100 neurônios
model.add(Dense(32, activation='relu')) # Camada oculta com 32
# Neurônios
```

```
model.add(Dense(3, activation='softmax'))      # Camada de saída com 3 neurônios para classificação (NEGATIVE, NEUTRAL, POSITIVE)
```

## 2) Compilação do modelo:

Configura-se o otimizador como 'adam', que é um método de otimização popular, e a função de perda como 'sparse\_categorical\_crossentropy', que é apropriada para problemas de classificação com várias categorias.

```
# Compilando o modelo
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

## 3) Treinamento do modelo:

O modelo é treinado utilizando o conjunto de treinamento ( $X_{train}$  e  $y_{train}$ ). O treinamento ocorre durante 70 épocas e a validação dos dados de teste é feita utilizando o conjunto de testes ( $X_{test}$  e  $y_{test}$ ).

```
# Treinamento do modelo
history = model.fit(X_train, y_train, epochs=60, validation_data=(X_test,
y_test))
```

## 4) Avaliação do modelo: Base de dados sprint 3

A função `evaluate` é usada para calcular a perda (loss) e a acurácia (accuracy) do modelo nos dados de teste, e em seguida esses valores são exibidos.

```
| # Avaliação do modelo
| loss, accuracy = model.evaluate(X_test, y_test)
| print('Loss:', loss)
| print('Accuracy:', accuracy)
|
58/58 [=====] - 0s 2ms/step - loss: 0.8403 - accuracy: 0.6064
Loss: 0.8403391242027283
Accuracy: 0.6064060926437378
```

Figura 51: Avaliação do modelo Sprint 3

No entanto, a métrica que estamos levando em consideração no contexto do nosso projeto atual, é a métrica recall, uma vez que essa métrica dá mais ênfase aos Falsos Negativos, que é quando o modelo classifica erroneamente como positivo, quando na verdade é negativo.

Sendo assim, estamos utilizando essa métrica para ter uma melhor precisão do nosso modelo, e aumentar principalmente o acerto de comentários negativos, uma vez que o foco do projeto é justamente esse, não deixar passar batido nenhum comentário que seja negativo. Logo, é necessário que o modelo obtenha bons resultados na métrica recall.

### 4.1) Avaliação do modelo: Base de dados sprint 4

```

# Avaliação do modelo
loss, accuracy = model.evaluate(X_test, y_test)
print('Loss:', loss)
print('Accuracy:', accuracy)

51/51 [=====] - 0s 1ms/step - loss: 0.9665 - accuracy: 0.5417
Loss: 0.9665026068687439
Accuracy: 0.5416666865348816

```

Figura 52: Avaliação do modelo Sprint 4

### 5) Precisão do modelo:

O código abaixo faz previsões usando o modelo treinado com o conjunto de teste (`X_test`).

Assim sendo, gera-se um relatório de classificação usando a função `classification_report` do scikit-learn, comparando as classes verdadeiras (`y_test`) com as classes preditas (`y_pred_classes`).

```

# Previsões do modelo
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)

# Gerar o relatório de classificação
report = classification_report(y_test, y_pred_classes)

# Imprimir a tabela de classificação
print(report)

```

## 10.8.3 Resultados

### 10.8.3.1 Rede Neural - Precisão (Base de dados sprint 3)

Portanto, levando em conta toda a explicação acima e a definição da métrica escolhida, os resultados obtidos com esse modelo, foi o seguinte:

	precision	recall	f1-score	support
0	0.53	0.54	0.53	386
1	0.69	0.65	0.67	844
2	0.56	0.59	0.57	612
accuracy			0.61	1842
macro avg	0.59	<b>0.59</b>	0.59	1842
weighted avg	0.61	0.61	0.61	1842

Levando em consideração que:

0 - negativo

1- neutro

2 - positivo

Pode-se concluir que o modelo acerta com um recall de 54% os comentários negativos, 65% os comentários neutros e 59% os comentários positivos. Portanto, o recall geral obtido com esse modelo foi de 59%.

#### 10.8.3.2 Rede Neural - Precisão (Base de dados sprint 4)

	precision	recall	f1-score	support
0	0.40	0.62	0.49	360
1	0.62	0.47	0.54	597
2	0.58	0.54	0.56	651
accuracy			0.53	1608
macro avg	0.53	<b>0.54</b>	0.53	1608
weighted avg	0.56	0.53	0.53	1608

Levando em consideração que:

0 - negativo

1- neutro

2 - positivo

Pode-se concluir que o modelo acerta com um recall de 62% os comentários negativos, 47% os comentários neutros e 54% os comentários positivos. Portanto, o recall geral obtido com esse modelo foi de 54%.

#### 10.8.3.3 Rede Neural - Matriz de confusão

A matriz de confusão,também conhecida como tabela de contingência, é utilizada para visualizar o desempenho de um modelo de classificação. É muito útil quando se trabalha com problemas de classificação em machine learning.

A matriz de confusão apresenta uma tabela com duas dimensões: as classes reais (verdadeiras) e as classes previstas (estimadas) pelo modelo. Cada célula da matriz representa o número de instâncias que foram classificadas de determinada forma pelo modelo.

Código da matriz de confusão:

```
cm = confusion_matrix(y_test, y_pred_classes)
```

```
# Definir as classes
classes = ['Classe 1', 'Classe 2', 'Classe 3']
```

```
# Plotar a matriz de confusão
```

```

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=classes,
yticklabels=classes)
plt.xlabel('Classe Predita')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()

```

### Sprint 3:

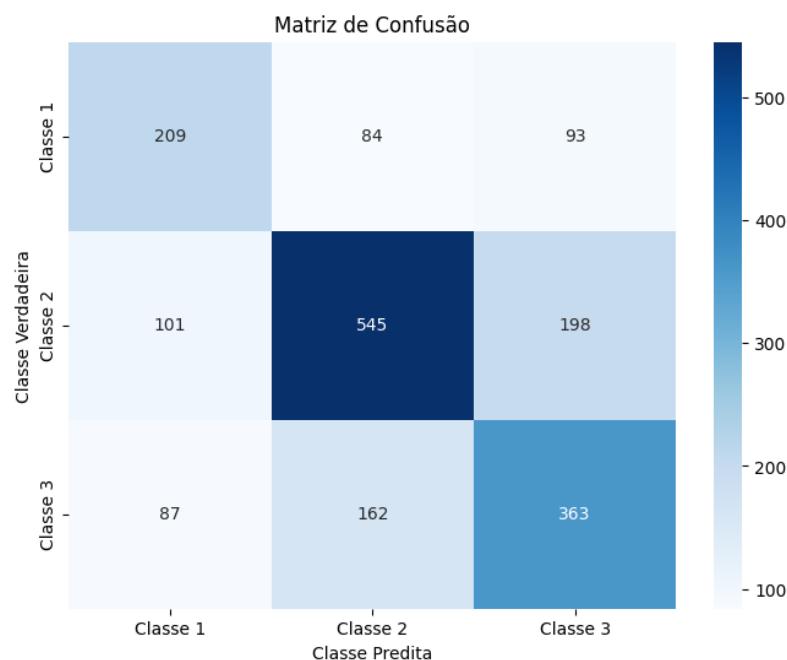


Figura 53: Matriz de confusão - Rede Neural Sprint 3

Levando em consideração que:

Classe 1 - negativos

Classe 2 - neutros

Classe 3 - positivos

Pode-se concluir, que de acordo com a matriz de confusão acima, o modelo teve um alto acerto sobre a classe dos negativos (**209**), porém, ainda teve um erro de (84) sendo classificado como neutro quando na verdade é negativo e (93) sendo classificado como positivo quando é negativo. Logo, estamos em busca do aumento de acertos da classe negativa e diminuição desses erros, em que o modelo está classificando erroneamente um total de (177) que pertencem a classe do negativo.

### Sprint 4:

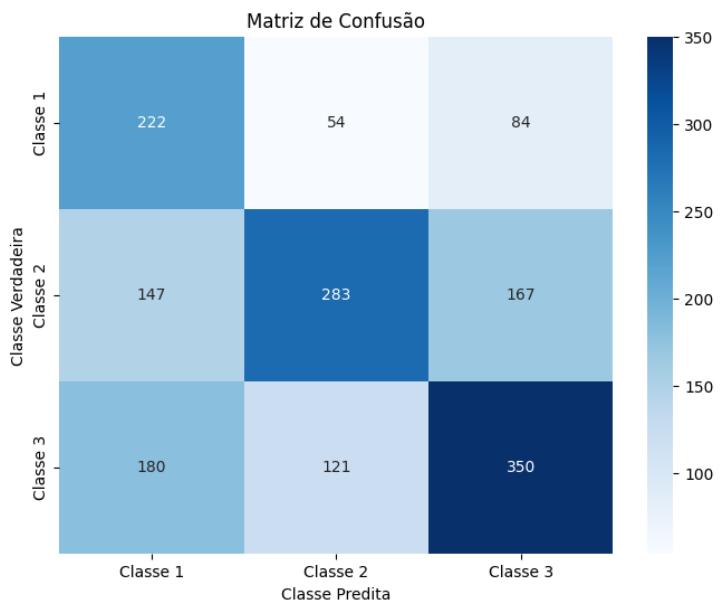


Figura 54: Matriz de confusão - Rede Neural Sprint 4

Levando em consideração que:

Classe 1 - negativos

Classe 2 - neutros

Classe 3 - positivos

Pode-se concluir, que de acordo com a matriz de confusão acima, o modelo teve um alto acerto sobre a classe dos negativos (222), porém, ainda teve um erro de (54) sendo classificado como neutro quando na verdade é negativo e (84) sendo classificado como positivo quando é negativo. Logo, estamos em busca do aumento de acertos da classe negativa e diminuição desses erros, em que o modelo está classificando erroneamente um total de (138) que pertencem a classe do negativo.

#### 10.8.4 Conclusão

Os códigos apresentados acima demonstram todo o processo de construção e treinamento de uma rede neural para classificação de texto, com todos os passos necessários para a construção do modelo de uma rede neural para machine learning, com relatório de classificação, precisão e matriz de confusão.

Além disso, é importante ressaltar que esse modelo de rede neural foi criado um utilizando a base de dados antiga já tratada e o outro utilizando a base atual, com um novo tratamento. Sendo assim a comparação ficou a seguinte:

Rede neural - Word2vec - Sprint 3:

- recall = 59%
- classe 0 (negativo) recall = 54%
- classe 1 (neutro) recall = 65%
- classe 2 (positivo) = 59%

- Matriz de confusão
  - Acertos dos negativos - 209
  - Erros dos negativos - 177

Rede neural - Word2vec - Sprint 4:

- recall = 54%
- classe 0 (negativo) recall = 62%
- classe 1 (neutro) recall = 47%
- classe 2 (positivo) = 54%
- Matriz de confusão
  - Acertos dos negativos - 222
  - Erros dos negativos - 138

## Random Forest + BoW

### Introdução

Foi criado um modelo de *Random Forest* juntamente do processo de vetorização *Bag of Words*. Esse modelo é um algoritmo de aprendizado de máquina que combina várias árvores de decisão para formar um modelo mais preciso e robusto, criando várias árvores de decisão independentes, onde cada árvore é treinada em uma amostra aleatória do conjunto de dados e um subconjunto aleatório das características. Para fins de comparação, foram desenvolvidos códigos que têm a base de dados diferentes.

### Método

#### Cross validation

*Cross validation* é uma técnica usada para avaliar a capacidade de um modelo de generalizar para novos dados, que consiste em dividir o conjunto de dados em partes menores, treinar o modelo em uma parte e testá-lo em outra. Esse processo é repetido várias vezes e a média das métricas de avaliação é usada para avaliar o desempenho do modelo.

#### Grid search

*Grid search* é uma técnica de busca de hiperparâmetros usada para encontrar a melhor combinação de valores para um modelo de aprendizado de máquina, que consiste em definir um conjunto de valores para cada hiperparâmetro e treinar e avaliar o modelo com todas as combinações possíveis. O conjunto de hiperparâmetros que produz a melhor métrica de avaliação é selecionado como a configuração final do modelo.

## Resultado

### Random Forest

Primeiramente, uma instância do modelo *Random Forest* é criada utilizando a classe `RandomForestClassifier` e atribuída à variável `rfc`, representando um modelo

de classificação baseado nesta técnica. Em seguida, o modelo é treinado utilizando os dados de treino fornecidos, que são compostos pelos recursos ( $X_{treino}$ ) e as classes correspondentes ( $y_{treino}$ ). O método `fit` é chamado na instância do modelo, que ajusta o modelo aos dados de treino, permitindo que ele aprenda os padrões presentes nos dados.

```
rfc = RandomForestClassifier()  
  
rfc.fit(X_treino, y_treino)
```

Após o treinamento, o modelo é utilizado para fazer previsões nos dados de teste, utilizando o método `predict` do modelo treinado e fornecendo os recursos de teste ( $X_{teste}$ ), são geradas as previsões para as classes. Para avaliar o desempenho do modelo, é calculada a acurácia, que é uma métrica que mede a proporção de exemplos corretamente classificados em relação ao total de exemplos. A função `accuracy_score` é utilizada para calcular a acurácia, recebendo como parâmetros as classes verdadeiras ( $y_{teste}$ ) e as classes previstas ( $y_{pred}$ ) pelo modelo. A seguir, a função `np.unique(y_pred)` é utilizada para obter os valores únicos das previsões feitas pelo modelo e, em seguida, a função `len()` é aplicada para obter o número de valores únicos. Essa linha verifica se o número de classes de saída únicas é igual a 1. Caso a condição seja verdadeira, o modelo tem apenas uma classe de saída possível, onde será exibida uma mensagem na tela usando a função `print()`. Caso contrário, o código chama a função `classification_report(y_teste, y_pred)`, que recebe os rótulos verdadeiros e as previsões feitas pelo modelo como parâmetros.

```
y_pred = rfc.predict(X_teste)  
  
acuracia = accuracy_score(y_teste, y_pred)  
  
if len(np.unique(y_pred)) == 1:  
    print("O modelo tem apenas uma classe de saída possível.")  
else:  
    classification = classification_report(y_teste, y_pred)  
    print("\nRelatório de Classificação:")  
    print(classification)
```

Abaixo é possível observar o código necessário para criar a matriz de confusão, onde esta é calculada usando a função `confusion_matrix(y_teste, y_pred)`, que recebe como parâmetros os rótulos verdadeiros ( $y_{teste}$ ) e as previsões feitas pelo modelo ( $y_{pred}$ ). Após calcular a matriz de confusão, uma lista chamada `classes` é definida, contendo os nomes das classes presentes no problema, que serão

utilizados para rotular as classes na matriz de confusão. Em seguida, é criada uma figura de plotagem com as dimensões especificadas usando `plt.figure(figsize=(8, 6))`. A função `sns.heatmap()` é chamada para criar o mapa de calor, os parâmetros são: `cm` é que é a matriz de confusão, `annot=True` e `fmt='g'` utilizados para exibir os valores da matriz nas células, `cmap='Blues'` que define a paleta de cores a ser utilizada no mapa de calor, e por último, os rótulos dos eixos `x` e `y` são definidos com base na lista de classes usando os parâmetros `xticklabels` e `yticklabels`.

```
cm = confusion_matrix(y_teste, y_pred)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot
=True, cmap='Blues', fmt='g', xticklabels=classes,
yticklabels=classes)
plt.xlabel('Classe Predita')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()
```

### Relatório de Classificação - Sprint 3:

	precision	recall	f1-score	support
0	0.69	0.50	0.58	386
1	0.76	0.74	0.75	844
2	0.64	0.77	0.70	612
accuracy			0.70	1842
macro avg	0.70	<b>0.67</b>	0.68	1842
weighted avg	0.71	0.70	0.70	1842

### Matriz de Confusão - Sprint 3:

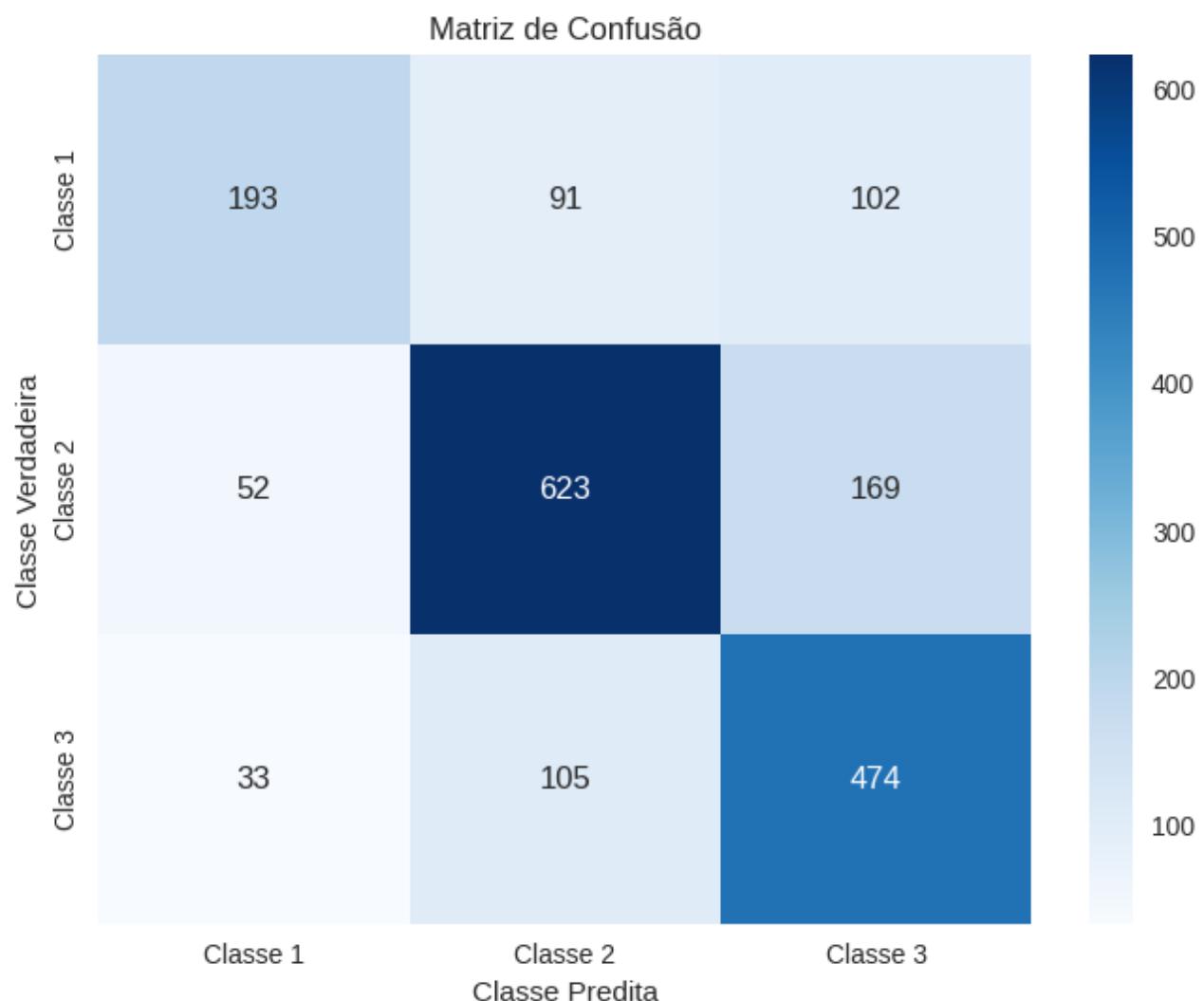


Figura 55: Matriz de Confusão - Random Forest Sprint 3

#### Relatório de Classificação - Sprint 4:

	precision	recall	f1-score	support
0	0.72	0.50	0.59	360
1	0.65	0.68	0.66	597
2	0.68	0.76	0.72	651
accuracy			0.67	1608
macro avg	0.68	<b>0.65</b>	0.66	1608
weighted avg	0.68	0.67	0.67	1608

#### Matriz de Confusão - Sprint 4:

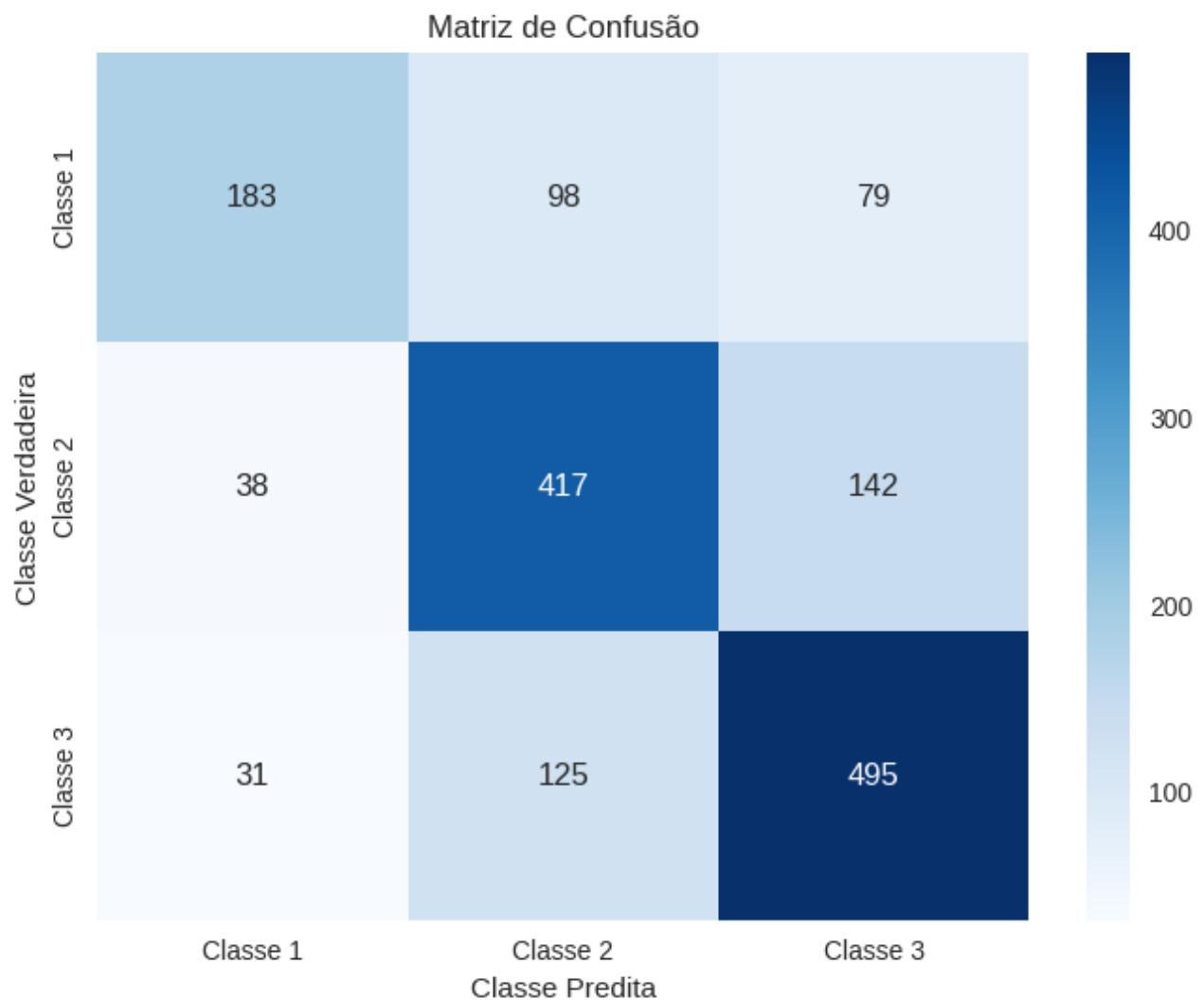


Figura 56: Matriz de Confusão - Random Forest Sprint 4

### Random Forest - Validação Cruzada

No próximo código, é realizada uma validação cruzada usando o método `cross_val_score`, que avalia o desempenho do modelo aplicando-o a múltiplos conjuntos de treinamento e teste. Nesse caso, o modelo *Random Forest* é usado como o estimador a ser avaliado, onde os dados de entrada `bow_model` e os rótulos de classe sentimento são passados como parâmetros. A validação cruzada é realizada com 5 *folds*, ou seja, os dados são divididos em 5 partes iguais e o modelo é treinado e testado em cada combinação dessas partes. As pontuações de validação cruzada são armazenadas na variável `scores`. O bloco de `if` já foi explicado anteriormente.

```

rfc = RandomForestClassifier()

scores = cross_val_score(rfc, bow_model, sentimento, cv=5)

print('Pontuações de validação cruzada:', scores)

```

```

print('Média da validação cruzada:', scores.mean())

if len(np.unique(y_pred)) == 1:
    print("O modelo tem apenas uma classe de saída possível.")
else:
    classification = classification_report(y_teste, y_pred)
    print("\nRelatório de Classificação:")
    print(classification)

```

O código da Matriz de Confusão do Random Forest com *Cross Validation* é o mesmo do apresentado anteriormente.

### **Relatório de Classificação - Sprint 3:**

	precision	recall	f1-score	support
0	0.71	0.50	0.58	386
1	0.77	0.75	0.76	844
2	0.63	0.77	0.69	612
accuracy			0.70	1842
macro avg	0.70	<b>0.67</b>	0.68	1842
weighted avg	0.71	0.70	0.70	1842

### **Matriz de Confusão - Sprint 3:**

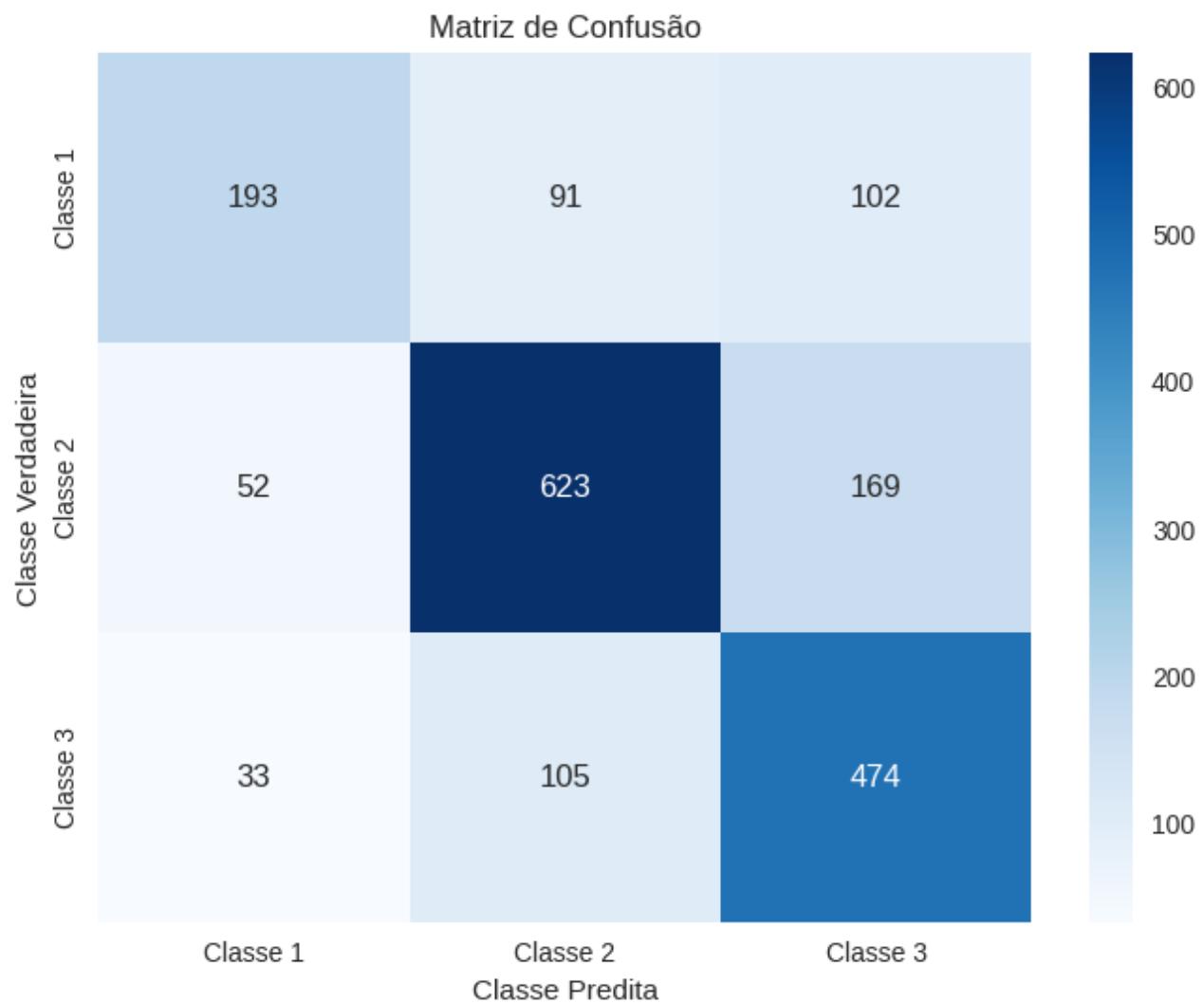


Figura 57: Matriz de Confusão - Random Forest com validação Sprint 3

#### **Relatório de Classificação - Sprint 4:**

	precision	recall	f1-score	support
0	0.74	0.51	0.60	360
1	0.65	0.70	0.68	597
2	0.70	0.77	0.73	651
accuracy			0.69	1608
macro avg	0.70	<b>0.66</b>	0.67	1608
weighted avg	0.69	0.69	0.68	1608

#### **Matriz de Confusão - Sprint 4:**

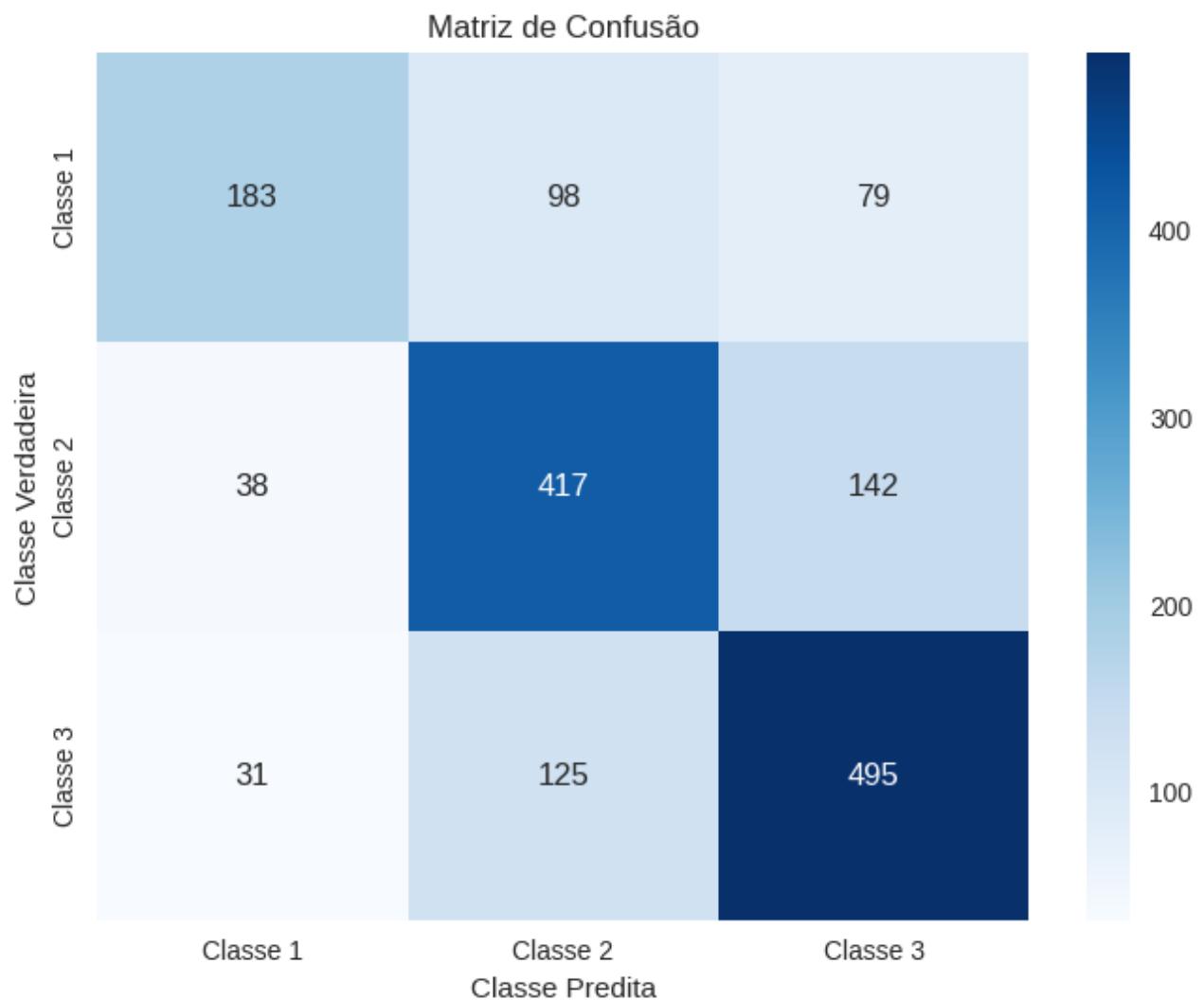


Figura 58: Matriz de Confusão - Random Forest com validação Sprint 4

### Random Forest - Grid Search

O código abaixo realiza uma busca em grade com validação cruzada para encontrar os melhores hiperparâmetros para um modelo *Random Forest*. Inicialmente, é definida uma grade de valores para os hiperparâmetros a serem testados, que são definidos no dicionário `parametros`. Nesse caso, são considerados três hiperparâmetros: `n_estimators` (número de estimadores), `max_depth` (profundidade máxima da árvore) e `min_samples_split` (número mínimo de amostras para dividir um nó interno). Diferentes valores são fornecidos para cada hiperparâmetro, permitindo que diferentes combinações sejam testadas durante a busca em grade.

```
parametros = {'n_estimators': [100, 200, 300],
              'max_depth': [None, 10, 20],
              'min_samples_split': [2, 5, 10]}
```

Em seguida, é criada uma instância do modelo *Random Forest* utilizando a classe `RandomForestClassifier` e atribuída à variável "rfc". Após a criação do modelo, é criada

uma instância do objeto GridSearchCV, que é responsável por realizar a busca em grade com validação cruzada, este recebe três parâmetros principais: o modelo rfc como estimador, parametros e o número de folds da validação cruzada, definido como 5 através do parâmetro cv=5. Além disso, o parâmetro n\_jobs=-1 indica que a busca em grade pode ser executada em paralelo, utilizando todos os núcleos de CPU disponíveis. Por último, o método fit no objeto, usa os dados de entrada bow\_model e os rótulos de classe sentimento como parâmetros. O GridSearchCV avalia todas as combinações possíveis dos hiperparâmetros especificados usando a validação cruzada e retorna o melhor modelo encontrado.

```

rfc = RandomForestClassifier()

grid = GridSearchCV(rfc, parametros, cv=5, n_jobs=-1)

grid.fit(bow_model, sentimento)

print('Melhores hiperparâmetros:', grid.best_params_)
print('Melhor pontuação:', grid.best_score_)

if len(np.unique(y_pred)) == 1:
    print("O modelo tem apenas uma classe de saída possível.")
else:
    classification = classification_report(y_teste, y_pred)
    print("\nRelatório de Classificação:")
    print(classification)

```

O código da Matriz de Confusão do Naive Bayes com *Grid Search* é o mesmo do apresentado anteriormente.

### **Relatório de Classificação - Sprint 3:**

	precision	recall	f1-score	support
0	0.69	0.50	0.58	386
1	0.76	0.74	0.75	844
2	0.64	0.77	0.70	612
accuracy			0.70	1842
macro avg	0.70	<b>0.67</b>	0.68	1842
weighted avg	0.71	0.70	0.70	1842

### **Matriz de Confusão - Sprint 3:**



Figura 59: Matriz de Confusão - Random Forest com Grid Search Sprint 3

#### Relatório de Classificação - Sprint 4:

	precision	recall	f1-score	support
0	0.74	0.51	0.60	360
1	0.65	0.70	0.68	597
2	0.70	0.77	0.73	651
accuracy			0.69	1608
macro avg	0.70	<b>0.66</b>	0.67	1608
weighted avg	0.69	0.69	0.68	1608

#### Matriz de Confusão - Sprint 4:

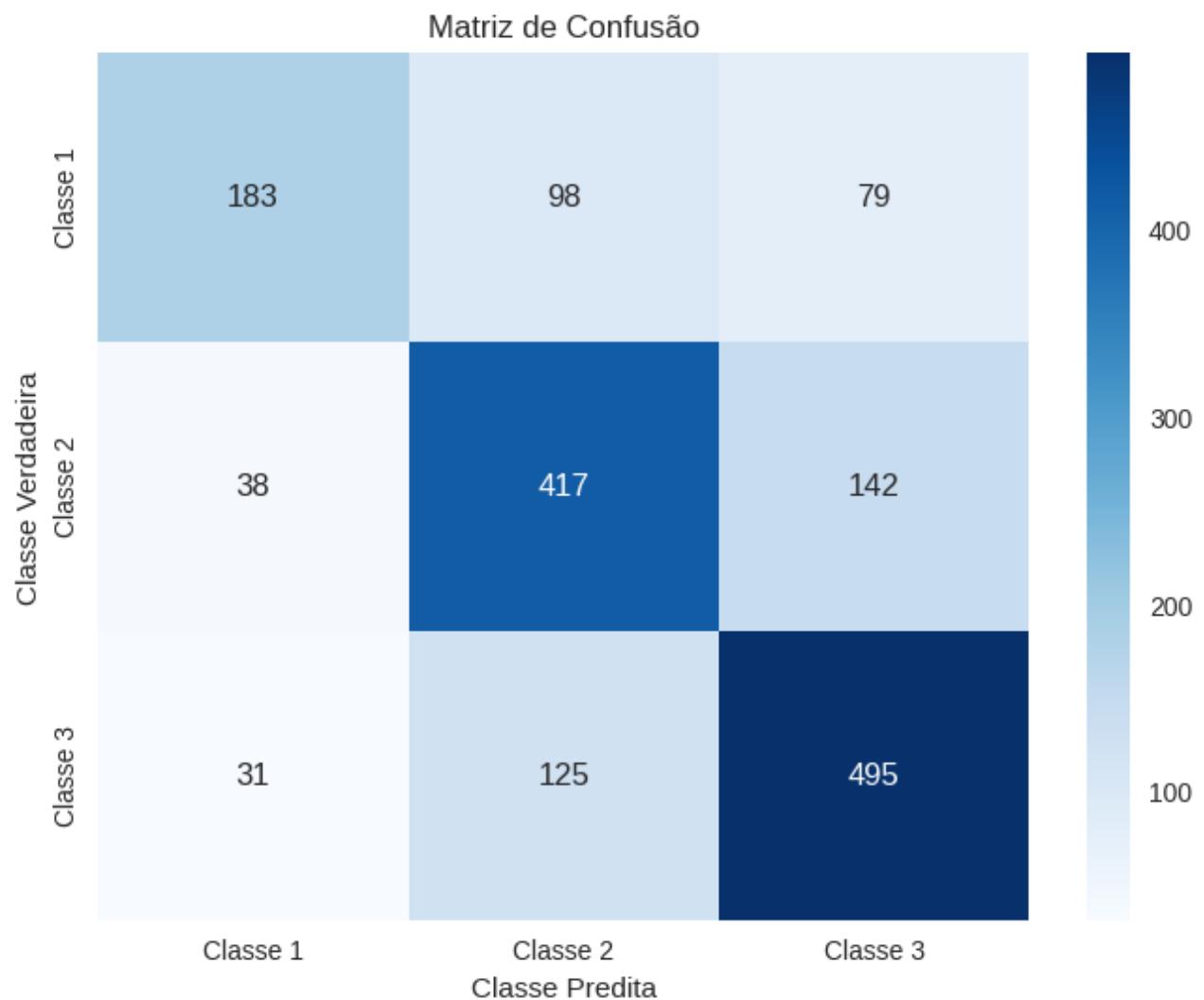


Figura 60: Matriz de Confusão - Random Forest com Grid Search Sprint 4

Todos os modelos apresentados acima foram exportados com a biblioteca pickle, código apresentado abaixo. A primeira linha abre o arquivo, em modo escrita, que será utilizado para armazenar o modelo de rede neural, nesse caso o desenvolvedor pode escolher o nome do arquivo. A segunda linha utiliza a função `pickle.dump()`, da biblioteca pickle, para salvar o modelo no arquivo aberto anteriormente. A terceira linha abre o arquivo, em modo leitura, para que a função `pickle.load()` carrega o conteúdo na variável, convertendo os bytes do arquivo novamente em um objeto modelo de rede neural utilizável.

```

with open('nome_escolhido.pkl', 'wb') as arquivo:
    pickle.dump(model, arquivo)
with open('nome_escolhido.pkl', 'rb') as arquivo:
    nome_escolhido = pickle.load(arquivo)

```

## Conclusão

Em suma, o *Random Forest* é um modelo muito promissor que apresentou resultados muito satisfatórios e no final todos os modelos desenvolvidos foram exportados com a biblioteca pickle.

## Decision Tree

### Introdução

A Árvore de Decisão é um algoritmo popular de aprendizado de máquina utilizado para resolver problemas de classificação e regressão. É uma técnica de modelagem que usa uma estrutura em forma de árvore para tomar decisões com base em regras condicionais.

Uma árvore de decisão é construída a partir de um conjunto de dados de treinamento, onde cada instância é representada por um conjunto de características (também conhecido como atributos). O objetivo do algoritmo é dividir o conjunto de dados em subconjuntos cada vez mais puros ou homogêneos, com base nas características fornecidas, até que a classificação final seja alcançada.

### Método

#### Aplicação de hiperparâmetros

- 1) **class\_weight='balanced'**: esse hiperparâmetro define o peso das classes durante o treinamento. A opção 'balanced' ajusta automaticamente os pesos inversamente proporcionais às frequências das classes. Isso é útil quando temos um conjunto de dados desequilibrado, onde uma classe é muito mais frequente do que as outras.
- 2) **max\_depth=10**: hiperparâmetro que define a profundidade máxima da árvore. A profundidade máxima da árvore é limitada a 10, o que impede que ela cresça muito e ajuda a evitar o overfitting.
- 3) **min\_samples\_split=10**: hiperparâmetro que define o número mínimo de amostras necessárias para dividir um nó interno da árvore. Definido como 10, o que significa que um nó só será dividido se houver pelo menos 10 amostras nele.
- 4) **min\_samples\_leaf=3**: é o número mínimo de amostras necessárias em uma folha. Estamos definindo-o como 3, o que garante que cada folha tenha pelo menos 3 amostras.
- 5) **max\_features='sqrt'**: a implementação do algoritmo de Árvore de Decisão selecionará aleatoriamente a raiz quadrada do número total de recursos a serem usados em cada divisão. Isso pode aumentar a variabilidade das árvores construídas e melhorar o desempenho geral do modelo.

## Construção do modelo

### 1) Definição do modelo de árvore de decisão:

```
# Construir o modelo de Árvore de Decisão com hiperparâmetros ajustados
model_dt = DecisionTreeClassifier(class_weight='balanced', max_depth=10,
min_samples_split=8, min_samples_leaf=3, max_features='sqrt')
```

No código acima, estamos construindo o modelo de Árvore de Decisão com hiperparâmetros ajustados.

### 2) Treinando o modelo:

```
model_dt.fit(X_train, y_train)
```

A linha `model_dt.fit(X_train, y_train)` treina o modelo de Árvore de Decisão com os dados de treinamento `X_train` e os rótulos correspondentes `y_train`. O modelo aprenderá a mapear os recursos para as classes com base nos hiperparâmetros fornecidos.

### 3) Previsões do modelo:

```
# Fazendo previsões no conjunto de teste
y_pred_dt = model_dt.predict(X_test)
```

A linha `y_pred_dt = model_dt.predict(X_test)` aplica o modelo treinado `model_dt` aos dados de teste `X_test` para fazer previsões das classes. As previsões são armazenadas na variável `y_pred_dt`.

## Resultados

### Decision tree - Precisão (Base de dados sprint 3)

#### Avaliação do modelo:

```
# Gerar o relatório de classificação
classification_report_dt = classification_report(y_test, y_pred_dt)

# Imprimir o relatório de classificação
print("Relatório de Classificação:")
print(classification_report_dt)
```

#### Relatório de Classificação:

	precision	recall	f1-score	support
0	0.42	0.56	0.48	386
1	0.75	0.51	0.61	844
2	0.49	0.59	0.53	612
accuracy			0.55	1842
macro avg	0.55	<b>0.56</b>	0.54	1842
weighted avg	0.59	0.55	0.56	1842

Levando em consideração que:

- 0 - negativo
- 1- neutro
- 2 - positivo

Pode-se concluir que o modelo acerta com um recall de 56% os comentários negativos, 51% os comentários neutros e 59% os comentários positivos.  
Portanto, o recall geral obtido com esse modelo foi de 56%.

#### Decision tree - Precisão (Base de dados sprint 4)

Relatório de Classificação:

	precision	recall	f1-score	support
0	0.42	0.53	0.47	360
1	0.59	0.48	0.53	597
2	0.57	0.59	0.58	651
accuracy			0.54	1608
macro avg	0.53	<b>0.53</b>	0.53	1608
weighted avg	0.55	0.54	0.54	1608

Levando em consideração que:

- 0 - negativo
- 1- neutro
- 2 - positivo

Pode-se concluir que o modelo acerta com um recall de 53% os comentários negativos, 48% os comentários neutros e 59% os comentários positivos.  
Portanto, o recall geral obtido com esse modelo foi de 53%.

#### Decision tree - Matriz de confusão

Código da matriz de confusão:

```
# Gerar a matriz de confusão
```

```

cm_decisiontree = confusion_matrix(y_test, y_pred_dt)

# Definir as classes
classes = np.unique(target)

# Plotar a matriz de confusão
plt.figure(figsize=(8, 6))
sns.heatmap(cm_decisiontree, annot=True, cmap='Blues', fmt='g',
xticklabels=classes, yticklabels=classes)
plt.xlabel('Classe Preditiva')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()

```

### Sprint 3:

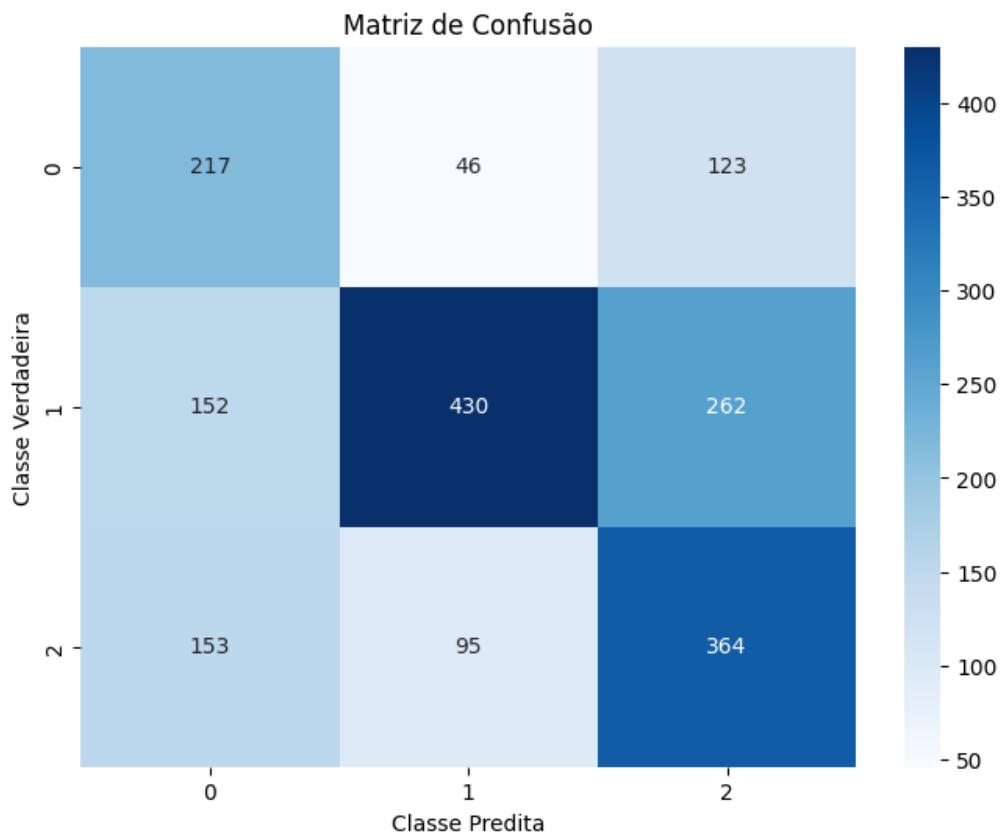


Figura 61: Decision tree - Matriz de confusão Sprint 3

Levando em consideração que:

Classe 1 - negativos

Classe 2 - neutros

Classe 3 - positivos

Pode-se concluir, que de acordo com a matriz de confusão acima, o modelo teve um alto acerto sobre a classe dos negativos (**217**), porém, ainda teve um erro de (**46**) sendo classificado como neutro quando na verdade é negativo e (**123**) sendo classificado

como positivo quando é negativo. Logo, estamos em busca do aumento de acertos da classe negativa e diminuição desses erros, em que o modelo está classificando erroneamente um total de (169) que pertencem a classe do negativo.

#### Sprint 4:

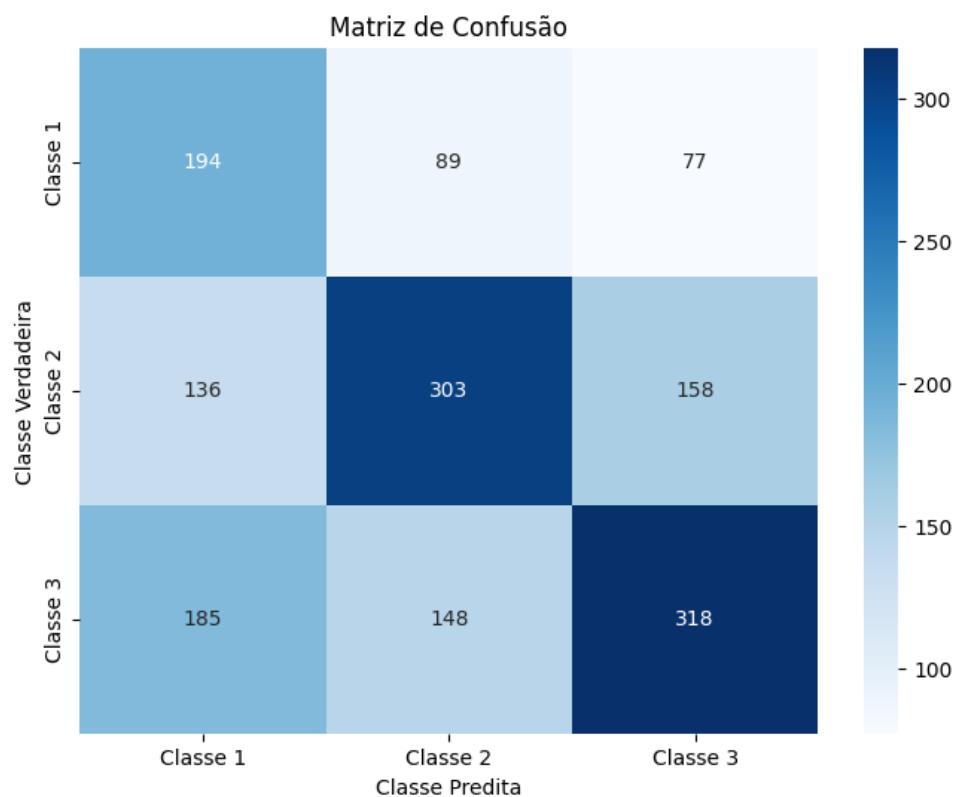


Figura 62: Decision tree - Matriz de confusão Sprint 4

Levando em consideração que:

Classe 1 - negativos

Classe 2 - neutros

Classe 3 - positivos

Pode-se concluir, que de acordo com a matriz de confusão acima, o modelo teve um alto acerto sobre a classe dos negativos (**194**), porém, ainda teve um erro de (89) sendo classificado como neutro quando na verdade é negativo e (77) sendo classificado como positivo quando é negativo. Logo, estamos em busca do aumento de acertos da classe negativa e diminuição desses erros, em que o modelo está classificando erroneamente um total de (166) que pertencem a classe do negativo.

## Conclusão

Decision Tree - Word2vec - Sprint 3:

- recall = 56%
- classe 0 (negativo) recall = 56%

- classe 1 (neutro) recall = 51%
- classe 2 (positivo) = 59%
- Matriz de confusão
  - Acertos dos negativos - 217
  - Erros dos negativos - 169

Decision Tree - Word2vec - Sprint 4:

- recall = 53%
- classe 0 (negativo) recall = 53%
- classe 1 (neutro) recall = 48%
- classe 2 (positivo) = 59%
- Matriz de confusão
  - Acertos dos negativos - 194
  - Erros dos negativos - 166

## Modelo com novas features Rede Neural + TF-IDF

### Introdução

Nesta seção, apresentaremos o desenvolvimento de um modelo de classificação aprimorado que combina uma Rede Neural com a técnica TF-IDF para processamento de texto. O objetivo principal desse modelo é melhorar os resultados obtidos por um modelo anterior, levando em consideração uma base de dados enriquecida com novas features relevantes.

### Método

Para construir esse modelo, utilizamos uma Rede Neural, uma abordagem avançada de aprendizado de máquina amplamente utilizada em problemas de classificação.

Além disso, aplicamos a técnica TF-IDF para processamento de texto, que permite atribuir uma pontuação a cada palavra com base em sua importância relativa no documento e no corpus geral.

### Construção do modelo

#### 1- Definição do modelo:

```
import tensorflow.keras.backend as K

def recall(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    return true_positives / (possible_positives + K.epsilon())

model_rede_neural_df = Sequential()
```

```

model_rede_neural_df.add(Embedding(input_dim=vocab,           output_dim=80,
                                    input_length=max_length, trainable=True))
model_rede_neural_df.add(GlobalMaxPooling1D())
model_rede_neural_df.add(Dropout(0.3))
model_rede_neural_df.add(Dense(units=3, activation='softmax'))

model_rede_neural_df.compile(optimizer='adam',
                             loss='sparse_categorical_crossentropy', metrics=[recall])

mc      = ModelCheckpoint('weight.best.hdf5',      monitor='val_acc',
                           save_best_only=True, mode='max')

model_rede_neural_df.fit(x_train,     y_train,     validation_data=(x_test,
y_test), batch_size=32, epochs=4, callbacks=[mc])

print(model_rede_neural_df.evaluate(x_test, y_test))

```

O código apresentado acima é um exemplo de implementação de uma Rede Neural utilizando a biblioteca TensorFlow e a função de perda "sparse\_categorical\_crossentropy" para um problema de classificação. A métrica de avaliação utilizada é o recall, que mede a proporção de exemplos corretamente identificados para cada classe.

A arquitetura do modelo consiste em uma camada de Embedding para representação vetorial das palavras, seguida por uma camada de GlobalMaxPooling1D para extrair os recursos mais relevantes do texto. Em seguida, é aplicada uma camada de Dropout para evitar o overfitting e, finalmente, uma camada densa com ativação softmax para a classificação em três classes.

O modelo é compilado com o otimizador 'adam' e a função de perda 'sparse\_categorical\_crossentropy'. Além disso, é definida a métrica de recall para avaliar o desempenho do modelo durante o treinamento.

O modelo é treinado utilizando os dados de treinamento (x\_train e y\_train) e validado com os dados de teste (x\_test e y\_test) em lotes de tamanho 32, durante 4 épocas. O callback ModelCheckpoint garante que os pesos do modelo com a melhor acurácia na validação sejam salvos.

Por fim, é exibido o resultado da avaliação do modelo nos dados de teste, utilizando a função evaluate, que retorna a perda e as métricas definidas durante a compilação do modelo.

## Resultados

**Rede Neural - TF-IDF - Precisão (Base de dados novas features - Sprint 5)**

**Avaliação do modelo:**

```

from sklearn.metrics import classification_report, confusion_matrix

y_pred_probs = model_rede_neural_df.predict(x_test)
y_pred_classes = np.argmax(y_pred_probs, axis=1)

classification = classification_report(y_test, y_pred_classes)

```

```
print("\nRelatório de Classificação:")
print(classification)
```

## Relatório de classificação:

	precision	recall	f1-score	support
0	0.71	0.62	0.66	655
1	0.73	0.78	0.76	1321
2	0.73	0.72	0.72	1033
accuracy			0.73	3009
macro avg	0.72	<b>0.71</b>	0.71	3009
weighted avg	0.73	0.73	0.72	3009

Levando em consideração que:

0 - negativo

1- neutro

2 - positivo

Pode-se concluir que o modelo acerta com um recall de 62% os comentários negativos, 78% os comentários neutros e 72% os comentários positivos.

Portanto, o recall geral obtido com esse modelo foi de 71%.

## Random Forest - TF-IDF - Matriz de confusão

### Código da matriz de confusão:

```
cm = confusion_matrix(y_test, y_pred_classes)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=classes,
            yticklabels=classes)
plt.xlabel('Classe Predita')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()
```

### Matriz de confusão:

## Sprint 5

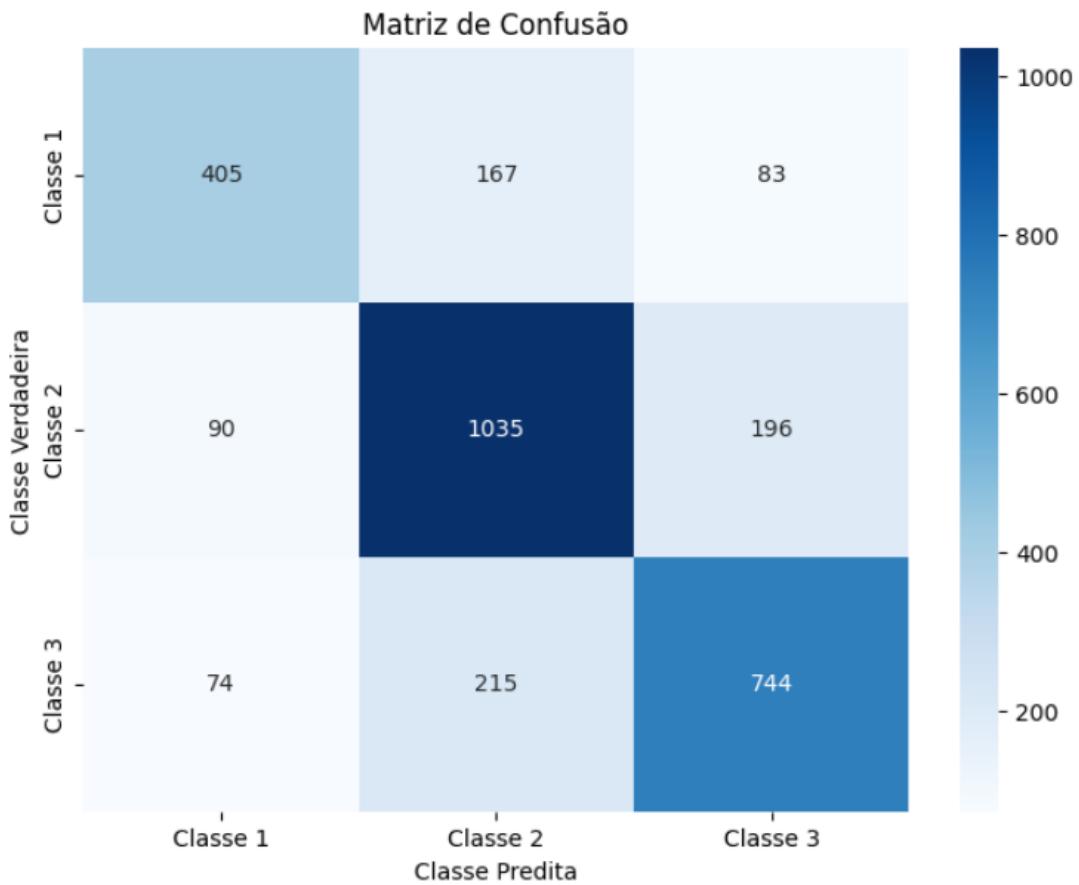


Figura 63: Random Forest - TF-IDF - Matriz de confusão Sprint 5

Levando em consideração que:

Classe 1 - negativos

Classe 2 - neutros

Classe 3 - positivos

Pode-se concluir, que de acordo com a matriz de confusão acima, o modelo teve um alto acerto sobre a classe dos negativos (405), porém, ainda teve um erro de (167) sendo classificado como neutro quando na verdade é negativo e (83) sendo classificado como positivo quando é negativo. Logo, estamos em busca do aumento de acertos da classe negativa e diminuição desses erros, em que o modelo está classificando erroneamente um total de (250) que pertencem a classe do negativo.

## Conclusão

Rede Neural - TF-IDF- Sprint 5 - novas features

- recall = 71%
- classe 0 (negativo) recall = 62%
- classe 1 (neutro) recall = 78%
- classe 2 (positivo) = 72%
- Matriz de confusão
  - Acertos dos negativos - 405

- Erros dos negativos - 250

## Modelo com novas features Naive Bayes + TF-IDF

### Introdução

O modelo proposto neste estudo utiliza o algoritmo de classificação Naive Bayes em conjunto com a técnica TF-IDF (Term Frequency-Inverse Document Frequency) para extrair características relevantes dos dados. O objetivo é melhorar o desempenho da classificação em relação a um modelo anterior. Neste contexto, foram realizados experimentos e análises para avaliar a eficácia do modelo proposto.

### Método

Para construir o modelo com as novas features, foram utilizados os algoritmos Naive Bayes e TF-IDF. O Naive Bayes é um algoritmo de aprendizado supervisionado baseado no teorema de Bayes, que assume independência condicional entre as variáveis preditoras.

Já o TF-IDF é uma técnica de processamento de linguagem natural que atribui um peso às palavras em um documento, levando em consideração tanto a frequência da palavra no documento quanto a sua frequência inversa em todo o conjunto de documentos.

#### Construção do modelo

##### 1- Definição do modelo:

1.1) Separando as features e o target:

```
target = df_final['sentimento']
feature = df_final.iloc[:,3:df_final.shape[1]]
```

1.2) Dividindo os dados em treinamento e teste:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(feature, target,
test_size=0.2, random_state=42)
```

##### 2 - Construindo o modelo:

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.naive_bayes import GaussianNB

clf = GaussianNB()

clf = clf.fit(X_train,y_train.values.ravel())
```

```

Y_pred = clf.predict(X_test)

print(classification_report(y_test, Y_pred))

```

## Resultados

### Naive Bayes - TF-IDF - Precisão (Base de dados novas features - Sprint 5)

#### Relatório de classificação:

	precision	recall	f1-score	support
0	0.31	0.75	0.44	396
1	0.62	0.24	0.35	809
2	0.58	0.51	0.54	619
accuracy			0.44	1824
macro avg	0.50	<b>0.50</b>	0.44	1824
weighted avg	0.54	0.44	0.43	1824

Levando em consideração que:

- 0 - negativo
- 1- neutro
- 2 - positivo

Pode-se concluir que o modelo acerta com um recall de 75% os comentários negativos, 24% os comentários neutros e 51% os comentários positivos. Portanto, o recall geral obtido com esse modelo foi de 50%.

#### Naive Bayes - TF-IDF - Matriz de confusão

#### Código da matriz de confusão:

```

import matplotlib.pyplot as plt
import seaborn as sns
# Gerar a matriz de confusão
matriz_naivebayes = confusion_matrix(y_test, Y_pred)

# Definir as classes
classes = ['Classe 1', 'Classe 2', 'Classe 3']

# Plotar a matriz de confusão
plt.figure(figsize=(8, 6))

```

```

sns.heatmap(matriz_naivebayes, annot=True, cmap='Blues', fmt='g',
xticklabels=classes, yticklabels=classes)
plt.xlabel('Classe Predita')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()

```

## Sprint 5

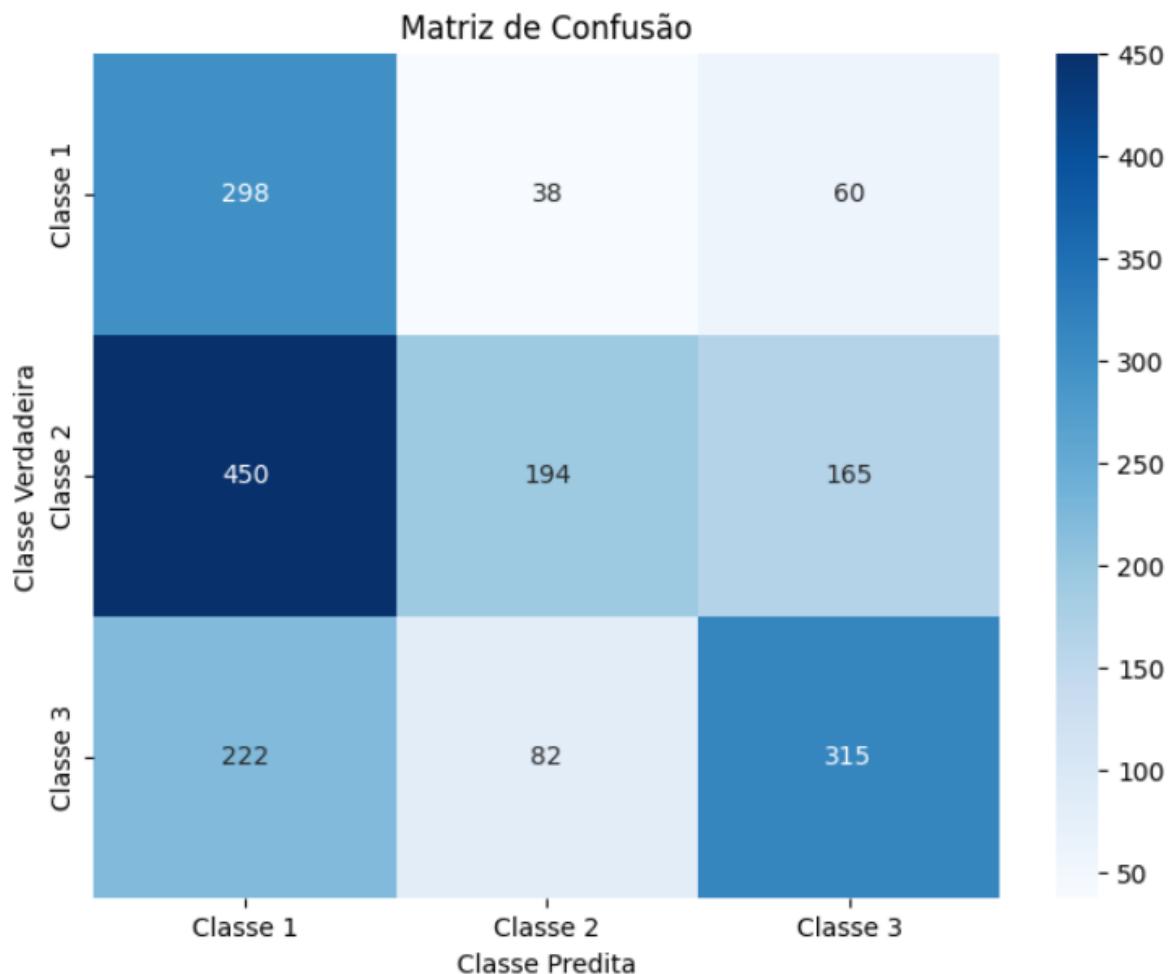


Figura 64: Naive Bayes - TF-IDF - Matriz de confusão Sprint 5

Levando em consideração que:

Classe 1 - negativos

Classe 2 - neutros

Classe 3 - positivos

Pode-se concluir, que de acordo com a matriz de confusão acima, o modelo teve um alto acerto sobre a classe dos negativos (298), porém, ainda teve um erro de (38) sendo classificado como neutro quando na verdade é negativo e (60) sendo classificado como positivo quando é negativo. Logo, estamos em busca do aumento de acertos da

classe negativa e diminuição desses erros, em que o modelo está classificando erroneamente um total de (98) que pertencem a classe do negativo.

## Conclusão

Naive Bayes - TF-IDF- Sprint 5 - novas features

- recall = 50%
- classe 0 (negativo) recall = 75%
- classe 1 (neutro) recall = 24%
- classe 2 (positivo) = 51%
- Matriz de confusão
  - Acertos dos negativos - 298
  - Erros dos negativos - 98

Logo, pode-se concluir que foi um dos modelos com os resultados mais baixos e com grande dispersão, uma vez que o recall da classe 0 (negativo) obteve 75%, porém em contrapartida o recall da classe 1 (positivo) obteve um baixo recall, sendo 24%. Portanto, foi um modelo que não foi satisfatório.

## Modelo com novas features Rede neural - sequência de palavras + Word2Vec

### Introdução

Nesta seção, apresentaremos o desenvolvimento de um modelo de classificação aprimorado que combina a utilização de uma rede neural sequencial em conjunto com a técnica Word2Vec para criar um modelo com novas features. O objetivo é melhorar a performance da classificação em relação a um modelo anterior. Neste contexto, foram realizados experimentos e análises para avaliar a eficácia do modelo proposto

### Método

Para construir o modelo com as novas features, utilizamos uma rede neural sequencial, que é uma arquitetura de rede neural amplamente utilizada para tarefas de processamento de linguagem natural. Além disso, aplicamos a técnica Word2Vec para representar as palavras como vetores numéricos densos, capturando informações semânticas e contextuais das palavras.

### Construção do modelo

#### 1- Definição do modelo:

1.1) Separando as features e o target:

```
target = df_final['sentimento']
```

```
feature = df_final.iloc[:,3:df_final.shape[1]]
```

### 1.2) Dividindo os dados em treinamento e teste:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(feature, target,
test_size=0.2, random_state=42)
```

## 2 - Construindo o modelo:

```
def recall(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    return true_positives / (possible_positives + K.epsilon())
model_rede_neural_df = Sequential()
model_rede_neural_df.add(Embedding(input_dim=vocab, output_dim=80,
input_length=max_length, trainable=True))
model_rede_neural_df.add(GlobalMaxPooling1D())
model_rede_neural_df.add(Dropout(0.3))
model_rede_neural_df.add(Dense(units=3, activation='softmax'))
model_rede_neural_df.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=[recall])
mc = ModelCheckpoint('weight.best.hdf5', monitor='val_acc',
save_best_only=True, mode='max')
model_rede_neural_df.fit(x_train, y_train, validation_data=(x_test, y_test),
batch_size=32, epochs=10, callbacks=[mc])
print(model_rede_neural_df.evaluate(x_test, y_test))
```

O código acima apresenta a implementação de uma rede neural sequencial utilizando Keras. O modelo inclui uma camada de embedding, global max pooling e dropout, seguida por uma camada densa com ativação softmax. A função de perda é sparse categorical crossentropy e a métrica utilizada é recall. O treinamento é realizado em lotes de tamanho 32 durante 10 épocas, com o uso do callback ModelCheckpoint para salvar os melhores pesos. Por fim, o modelo é avaliado utilizando o método evaluate.

## Resultados

### Rede neural - sequencia de palavras + Word2Vec- Precisão (Base de dados novas features - Sprint 5)

#### Relatório de classificação:

	precision	recall	f1-score	support
0	0.68	0.69	0.68	648
1	0.77	0.72	0.74	1291
2	0.71	0.76	0.73	1070
accuracy			0.73	3009

macro avg	0.72	<b>0.72</b>	0.72	3009
weighted avg	0.73	0.73	0.73	3009

Levando em consideração que:

0 - negativo

1- neutro

2 - positivo

Pode-se concluir que o modelo acerta com um recall de 69% os comentários negativos, 72% os comentários neutros e 76% os comentários positivos. Portanto, o recall geral obtido com esse modelo foi de 72%.

### Código da matriz de confusão:

```
cm1 = confusion_matrix(y_test, y_pred_classes)
classes = ['Classe 1', 'Classe 2', 'Classe 3']
plt.figure(figsize=(8, 6))
sns.heatmap(cm1, annot=True, cmap='Blues', fmt='g', xticklabels=classes, yticklabels=classes)
plt.xlabel('Classe Preditiva')
plt.ylabel('Classe Verdadeira')
plt.title('Matriz de Confusão')
plt.show()
```

### Sprint 5

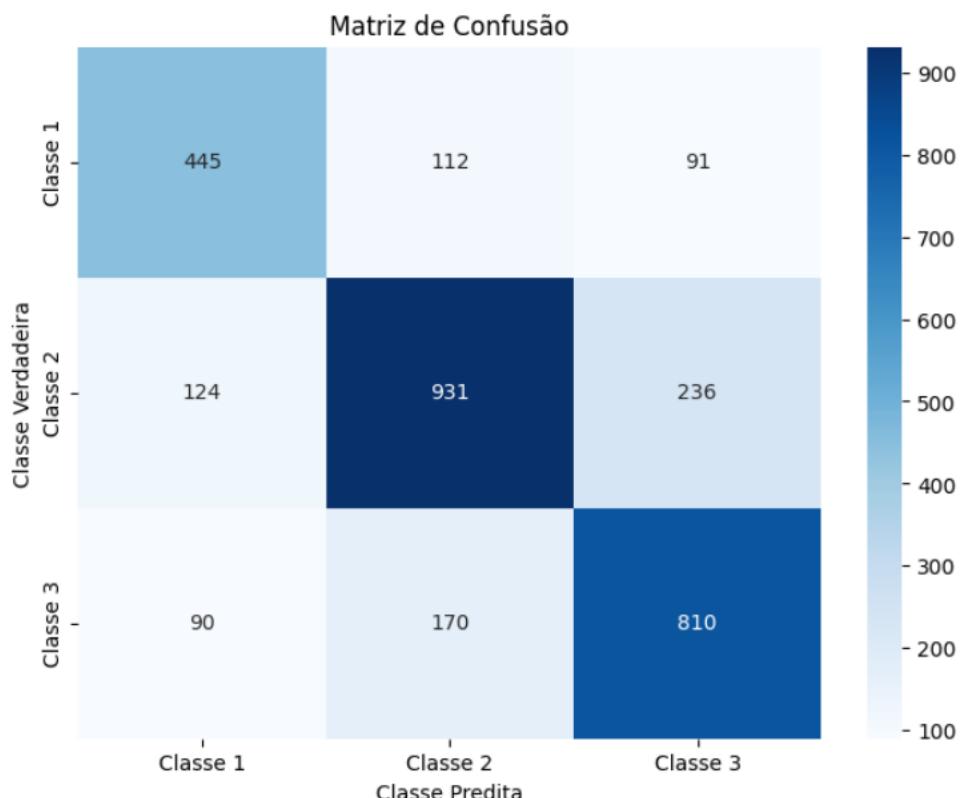


Figura 65: Rede Neural - TF-IDF - Matriz de confusão Sprint 5

Levando em consideração que:

Classe 1 - negativos

Classe 2 - neutros

Classe 3 - positivos

Pode-se concluir, que de acordo com a matriz de confusão acima, o modelo teve um alto acerto sobre a classe dos negativos (445), porém, ainda teve um erro de (112) sendo classificado como neutro quando na verdade é negativo e (91) sendo classificado como positivo quando é negativo. Logo, estamos em busca do aumento de acertos da classe negativa e diminuição desses erros, em que o modelo está classificando erroneamente um total de (203) que pertencem a classe do negativo.

## Conclusão

Rede Neural- sequência de palavras + Word2Vec- Sprint 5 - novas features

- recall = 72%
- classe 0 (negativo) recall = 69%
- classe 1 (neutro) recall = 72%
- classe 2 (positivo) = 71%
- Matriz de confusão
  - Acertos dos negativos - 445
  - Erros dos negativos - 203

Logo, pode-se concluir que foi um dos modelos com bons resultados, uma vez que o recall da classe 0 (negativo) obteve 69%, enquanto o recall da classe 2 (positivo) obteve um recall de 71%. Portanto, foi um modelo que satisfez a expectativa.

