CHAT BTG

BTG PACTUAL

inteli
instituto
de tecnologia
e liderança

Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
‹08/05/23›	‹João Tourinho› ‹Lucas Pereira›	.1. 1	Criação do Documento; Atualização do Análise do Negócio e UX;
‹13/06/23›	‹Vinícius Fernandes› ‹Thainá Lima›	1.2	Passagem dos dados do GitHub para o documento; Revisão do documento.
‹23/06/23›	‹Vinícius Fernandes› ‹Thainá Lima› ‹João Tourinho› ‹Lucas Pereira›	1.3	Revisão do documento.

Sumário

1. Introdução 4
2. Objetivos e Justificativa
   2.1. Objetivos
   2.2. Justificativa
3. Análise do Negócio
   3.1. Matriz de avaliação de valor Oceano Azul
   3.2. Matriz de Riscos
   3.3. Canvas Proposta de Valor
   3.4. Análise Financeira
4. Análise de Experiência do Usuário
   4.1. Personas
   4.2. User Stories
5. Solução Proposta 6
   5.1. Solução 6
   5.2. Arquitetura Proposta 6
   5.3. Diagrama Macro da Solução 6
   5.4. Descrição da Solução 6
   5.5. Algoritmos e Equações 6
6. Desenvolvimento e Resultados 7
   6.1. Nome do Módulo 7
      6.1.1. Descrição 7
      6.1.2. Tecnologia adotada 7
      6.1.3. User Stories 7
      6.1.4. Prototipação 7
      6.1.5. Diagramas 7
   6.2. Avaliação 7
7. Conclusões e Recomendações 8
8. Referências 9
Anexos 10
   ANEXO I – Sprint 1 10
   ANEXO II – Sprint 2 10
   ANEXO III – Sprint 3 10

1. Introdução

É nítido que as redes sociais têm se tornado cada vez mais presentes no dia a dia das pessoas, e com sua crescente popularidade, as empresas precisam se adaptar a esse ambiente. Com este crescimento assíduo, as oportunidades dentro deste contexto são muito amplas. Navegando nesse mar de oportunidades, muitas empresas têm tido um olhar mais aprofundado em relação às métricas e engajamento nas redes sociais, sendo assim, um ponto de atenção muito pertinente às empresas no cenário atual. Isso também é evidenciado na estratégia dessas empresas, e o investimento: de acordo com a Pesquisa Anual de CMO da Deloitte, o marketing representará aproximadamente 13,6% do orçamento total de uma empresa em 2023. Isso representa um aumento de 3,9% em relação aos dois anos anteriores. Anteriormente, as empresas precisavam se limitar a pesquisas de mercado e estudos de opinião para compreender as necessidades e desejos dos clientes, com as redes sociais, esse "embate" ocorre em tempo real. Nesse contexto, o banco BTG Pactual reconheceu a importância dessa tendência e decidiu adotar uma abordagem inovadora para analisar e classificar, automaticamente, os comentários dos usuários sobre suas campanhas.

O BTG Pactual, um dos principais bancos de investimento da América Latina, compreendeu o impacto das redes sociais em seus processos e reconheceu a importância de ouvir seus clientes para aprimorar seus produtos, serviços e a percepção geral do banco. Em parceria com a Inteli, o BTG vai implementar o Processamento de Linguagem Natural (PLN) para criar um modelo de análise de sentimentos e detecção de palavras-chave nos comentários das redes sociais, especificamente na rede social Instagram.

Com o modelo em pleno funcionamento e rodando junto com outras iniciativas das respectivas áreas envolvidas, o BTG Pactual poderá compreender as opiniões dos usuários e obter feedbacks valiosos, que ajudarão na construção de um banco mais inclusivo e próximo aos seus clientes. Consequentemente, esses insights permitirão que o banco aprimore os resultados de suas campanhas e melhore a qualidade de seus serviços e produtos para atender de maneira mais eficiente às necessidades dos clientes.

1.1. Parceiro de Negócios

O parceiro deste módulo, é o banco BTG Pactual. O Banco BTG Pactual é uma instituição financeira brasileira, reconhecida como uma das principais instituições de investimento do mundo, principalmente na América Latina. Fundado em 1983, o BTG Pactual tem sua sede na cidade de São Paulo e é conhecido por sua expertise em serviços financeiros, gestão de investimentos e assessoria financeira. Da mesma maneira, o banco atua em diversos segmentos do mercado, oferecendo serviços de wealth management (gestão de patrimônio), banco de investimento, asset management (gestão de ativos), mercado de capitais, private equity e serviços de corretagem. Com uma ampla gama de soluções financeiras, o BTG Pactual atende tanto clientes individuais quanto corporativos, incluindo empresas de médio e grande porte.

No contexto do projeto, o BTG Pactual reconhece a importância de compreender as opiniões e necessidades de seus clientes expressos nas redes sociais. Sendo assim, o projeto visa suprir a necessidade de entender as expectativas, preocupações, satisfação e insatisfação dos clientes em relação aos produtos, serviços e experiências fornecidos pelo banco. Com a análise dos comentários na rede social, o BTG Pactual busca extrair insights valiosos para aprimorar seu relacionamento com os clientes, identificar áreas de melhoria e tomar decisões estratégicas baseadas nas percepções e feedbacks dos usuários, adaptando suas ofertas com o intuito proporcionar uma experiência cada vez mais satisfatória.

Por fim, o projeto se justifica pela busca contínua em compreender as expectativas e sentimentos dos clientes do banco BTG Pactual, com o objetivo de aprimorar seus produtos, serviços e estratégias, garantindo uma relação sólida e satisfatória com seu público.

1.2. Definição do Problema

1.2.1. Problema

Diante do que foi apresentado como objetivo no TAPI, o principal problema ou necessidade de negócio identificado pelo Banco BTG Pactual é compreender as opiniões, sentimentos e necessidades dos clientes expressos nas redes sociais, em especial, para fins de prova de conceito, na plataforma Instagram. O banco reconhece o impacto das redes sociais

em seus processos e entende que ouvir seus clientes é fundamental para aprimorar seus produtos, serviços e a percepção geral da instituição.

Dessa forma, o BTG Pactual identificou a necessidade de implementar algum tipo de serviço que realizasse a classificação de acordo com o sentimento nos comentários. Neste sentido, pretendemos entregar um modelo de análise de sentimentos que faça a detecção de palavras-chave nos comentários das redes sociais, utilizando o Processamento de Linguagem Natural (PLN).

Com esse modelo em pleno funcionamento, o banco poderá obter insights valiosos a partir dos feedbacks dos usuários, permitindo uma compreensão mais aprofundada das opiniões dos clientes.

2. Objetivos e Justificativa

2.1. Objetivos

O objetivo do projeto é a construção de um modelo de PLN para aplicação nos comentários da rede social do Instagram do parceiro BTG Pactual.

Este projeto visa suprir a necessidade de entender o cliente, o que ele deseja e o que não deseja em relação ao banco e suas campanhas. A análise de sentimento permite ao banco compreender melhor o cliente, adaptando-se e oferecendo uma experiência cada vez mais satisfatória.

Ao analisar os comentários na rede social, o BTG Pactual busca extrair insights valiosos para aprimorar seu relacionamento com os clientes, identificar áreas de melhoria e tomar decisões estratégicas baseadas nas percepções e feedbacks dos usuários.

As estratégias de marketing do banco podem também ser otimizadas com base nos resultados obtidos com a solução. Por fim, a solução permitirá que o banco tome decisões estratégicas bem informadas e melhore seu relacionamento com os clientes.

2.2. Justificativa

A proposta de solução do nosso grupo é focar em diminuir os falsos positivos da forma mais eficiente possível no modelo utilizado. Esta é a ideia primária na solução, uma vez que deixar de alertar sobre o feedback negativo em uma campanha seria muito mais nocivo ao cliente do que fazer um falso alerta a empresa (o que seria um falso negativo). Desta forma, ter foco neste objetivo é uma forma de alcançar um resultado melhor para a situação em questão.

Além disso, pretendemos testar diversos modelos diferentes, analisando e comparando o resultado de cada um em relação aos outros. Modelos como o Cat Boost serão utilizados (além do Naive Bayes) e os resultados serão comparados a fim de chegar a um modelo ideal.

Desta forma, o projeto será validado com modelos além dos mais utilizados, diferenciando-o e trazendo mais potencial à solução.

Por fim, também pretendemos, a fim de diferenciar de outras e otimizar nossa solução, trabalhar minuciosamente no pré-processamento tendo em vista os resultados obtidos com os modelos utilizados. A entrega final irá trazer análises e insights gerados a partir do modelo mais eficiente.

3. Análise do Negócio

3.1. Matriz de avaliação de valor Oceano Azul

A Matriz Oceano Azul é a ferramenta que ajuda as empresas a identificar novas oportunidades de crescimento dentro do mercado e quais fatores as diferenciam da concorrência, criando valor para seus clientes.

Para esta análise, foram setadas outras Inteligências Artificiais, baseadas em Processamento de Linguagem Natural (PLN), como **Open AI, Amazon Comprehend, Microsoft Azure Cognitive Services, Google Cloud Natural Language, IBM Watson.**

A OpenAI é uma plataforma de inteligência artificial que oferece uma ampla gama de ferramentas e recursos para empresas e indivíduos. A plataforma utiliza modelos de linguagem natural avançados, como o GPT-3, para criar soluções personalizadas para problemas específicos de negócios. A Amazon Comprehend é um serviço de análise de texto fornecido pela AWS que permite a identificação de insights e informações úteis a partir de dados textuais, em grande escala. A tecnologia usa técnicas de aprendizado de máquina para extrair informações úteis a partir de dados textuais, como sentimentos, emoções, tópicos, entidades, relações e muito mais. O serviço é capaz de analisar grandes quantidades de texto, em várias línguas, e produzir informações significativas para apoiar a tomada de decisões de negócios, como por exemplo conjuntos de dados de texto, incluindo documentos, mensagens de texto, posts de redes sociais e outras formas de comunicação escrita, para identificar tendências.

O Microsoft Azure Cognitive Services é um conjunto de serviços cognitivos com alocação em nuvem que permite que desenvolvedores agreguem recursos de inteligência artificial (IA) a seus aplicativos sem a necessidade de experiência em aprendizado de máquina ou análise de dados. Esses serviços são construídos com base em algoritmos de aprendizado de máquina, visão computacional, reconhecimento de fala, processamento de linguagem natural e outras tecnologias. Consequentemente, os serviços cognitivos da Azure são altamente escaláveis e personalizáveis, permitindo que as empresas criem aplicativos sob medida para suas necessidades específicas, como por exemplo para realizar tarefas como

reconhecimento de voz, análise de sentimentos, detecção de imagens, tradução de idiomas e muito mais. Além disso, os serviços podem ser facilmente integrados a outras tecnologias da Microsoft, como o Microsoft Power BI, Microsoft Dynamics 365 e Microsoft Office 365.

A Google Cloud Natural Language é uma ferramenta de processamento de linguagem natural (PLN) criada pela Google que permite aos usuários extrair informações valiosas a partir de textos não estruturados, incluindo e-mails, documentos, artigos e outras fontes de dados. Com a técnica de aprendizado de máquina e processamento de linguagem natural, a Google Cloud Natural Language pode identificar entidades ou indivíduos mencionados em um texto, como nomes de pessoas, locais, organizações e datas, além de extrair sentimentos e emoções expressos pelo autor. Além disso, há possibilidade de agregação com o Google Cloud Natural Language, que permite às empresas analisarem grandes volumes de dados para obter insights úteis que podem ajudar a orientar decisões importantes de negócios.

O IBM Watson é um sistema de PLN, desenvolvido pela IBM. A ferramenta utiliza da análise de dados e aprendizado de máquina para processar informações e fornecer insights relevantes para seus usuários. O mesmo é utilizado em vários setores, incluindo saúde, finanças, educação e manufatura. O IBM Watson é uma plataforma de computação cognitiva e inteligência artificial desenvolvida pela empresa americana IBM. Ele utiliza tecnologias avançadas de processamento de linguagem natural, machine learning, análise de dados e outras técnicas de inteligência artificial para fornecer insights e soluções para uma ampla variedade de aplicações.

Eliminar → Assertividade, Escalabilidade, Robustez, confiabilidade, adaptabilidade, Assertividade, Tecnologia

A ação de eliminar, dentro da Matriz Oceano Azul, está relacionada à enumeração de fatores que podem ser retirados ou melhorados, ao analisar os atributos do negócio. Portanto, analisa-se a distribuição das ferramentas, em termos de escalabilidade, assertividade, acessibilidade, robustez e outras seções atribuídas à proposta do Chat-BTG, em comparação às outras soluções que utilizam Processamento de Linguagem Natural. Dado que a projeto consiste no desenvolvimento de um MVP, atualmente, o chat-BTG é uma iniciativa sem afunilamento em sua complexidade, por isso, sua escalabilidade é um fator a ser pensado nos

próximos passos do projeto, principalmente com ênfase na robustez e maior assertividade da tecnologia. Já os outros modelos, por já estarem no mercado e ter uma infraestrutura posicionada, tendo em vista que serviços de cloud e outros "web services" fornecem agregação em tempo real, com dados estruturados e já disponibilizados no próprio serviço, conseguem maior escalabilidade e confiabilidade, conforme as tecnologias e funcionalidades contidas em seus serviços.

Aumentar → Personalização

A personalização, nessa análise, é um fator de destaque no mercado, em comparação aos concorrentes. Destaca-se que o modelo de Processamento de Linguagem Natural é desenvolvido conforme a base de dados fornecido pelo cliente, voltado aos comentários feitos nos posts da conta BTG Pactual no Instagram. Ressalta-se que a visualização dos dados também será pensada a fim de facilitar o processo de tomada das decisões nos times de marketing e produto, como vantagem competitiva do banco.

Criar → Custo do Setup

Com a visão de ampliar, agregar e integrar diversas ferramentas, pode-se citar alguns fatores determinantes na análise financeira para o escopo do projeto. Neste sentido, é importante ressaltar que o projeto "Chat-BTG" ainda é limitado ao escopo mínimo de requisitos de uma rede social específica, dentro de dados levantados a partir de uma base de dados pré-definida, sem atualização em tempo real. Consequentemente, a níveis de complexidade baixos, possuímos um custo muito interessante com base no benchmark do mercado. Sendo uma solução nativa, o custo de setup da solução é muito baixo, e outros custos de "web services" são mitigados com o processo interno realizado.

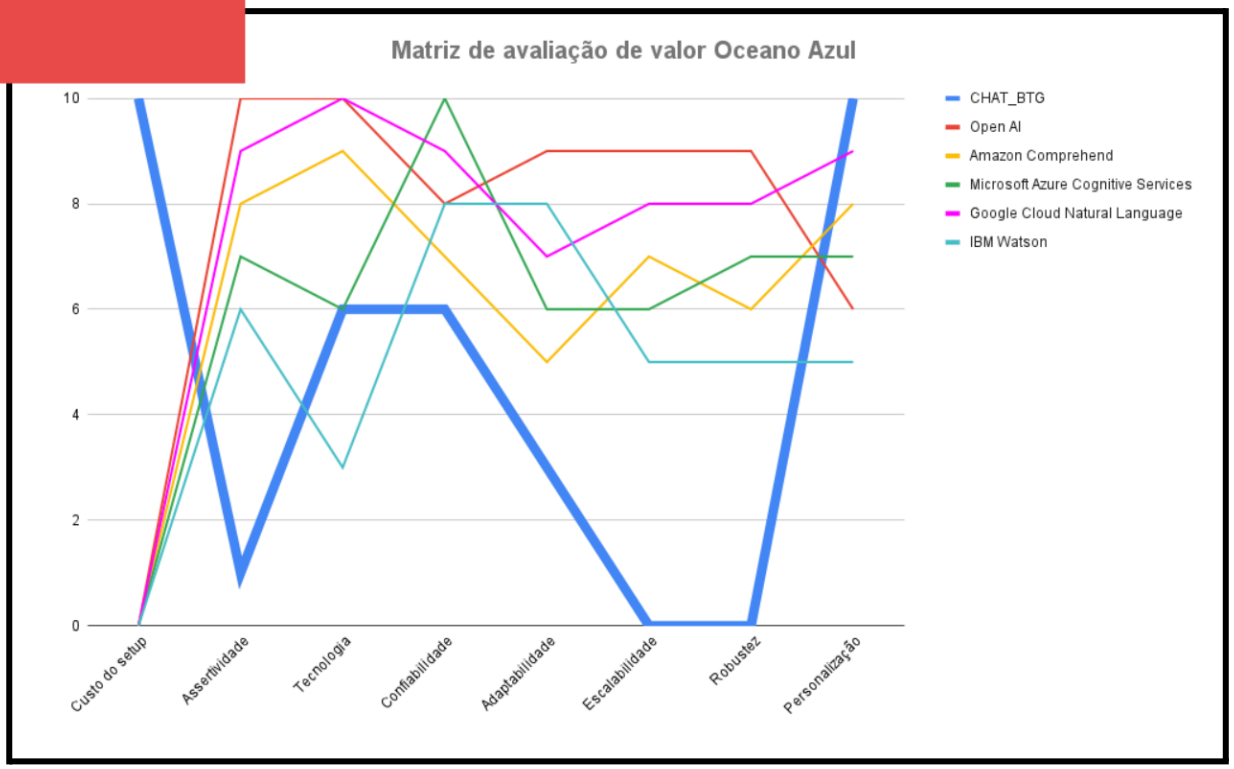


Imagem 1: Matriz de avaliação de valor Oceano Azul
Fonte: Autores

3.2. Matriz de riscos

A Matriz de Riscos consiste em uma ferramenta de gestão de riscos e oportunidades, a fim de identificar, avaliar e priorizar os fatores associados ao projeto, por grau de impacto e probabilidade de ocorrência.

	Ameaças					Oportunidades				
90%	-	-	-	-	-	-	-	-	-	-
70%	-	-	-	**Ameaça 001:** Resultados do modelo não serem corespondentes com o tratamento de falsos positivos previsto	-	-	**Oportunidade 002:** Reusabilidade do código, para atender possível escalabilidade do projeto	-	-	-
50%	-	-	**Ameaça 005:** Baixa precisão do modelo PLN desenvolvido, na identificação imprecisa do contexto dos comentários e palavras de duplo sentido	**Ameaça 003:** Assimetria de repetição das palavras e predição de sentimento, com base no serviço ou produto oferecido	**Ameaça 002:** falhas no processo de tratamento de dados, desqualificando os resultados	**Oportunidade 001:** Resultados do modelo contribuirem com a efetividade de futuras campanhas			-	-
30%	-	-	-	-	**Ameaça 004:** Uso não assertivo da métrica de classificação	-	-	**Oportunidade 003:** Considerar emojis na análise do comentários	-	-
10%	-	-	-	-	-	-	-	-	-	-
	Muito baixo	**Baixo**	**Moderado**	**Alto**	**Muito alto**	**Muito alto**	**Alto**	**Moderado**	**Baixo**	**Muito Baixo**
	Impacto									

Imagem 2: Matriz de Riscos
Fonte: Autores

Portanto, entende-se então, que o desenvolvimento de um Plano de Ação é de indubitável importância, dado que, o documento descreve as atividades que precisam ser realizadas para mitigar riscos e alcançar objetivos estipulados. O Plano define o que será feito, quem fará, como, quando e com quais recursos.

No que diz respeito aos riscos evidenciados na Matriz acima, ressalta-se o plano de ação desenvolvido:

Ameaça 01: Thainá - Identificar quais são as expectativas do cliente e manter o foco nelas durante a produção do modelo, além de revisar o modelo após cada mudança, comparando-o com as expectativas definidas para o projeto.

Ameaça 02: Vinicius - Identificar as causas da falha, e implementar medidas de solução para monitorar os testes no tratamento dos dados. Revisar a metodologia de pré-processamento de dados.

Ameaça 03: Kathy e Lucas - Verificar por quais possíveis motivos existe um erro na coleta das palavras chaves, e identificar padrões de possíveis palavras que estejam causando esses erros, buscando informações em relação a campanha de marketing. Além disso, caso o problema não for na coleta das palavras, ajustaremos o algoritmo e procuraremos possíveis falhas e tentaremos treinar o modelo com mais dados e exemplos para melhorar a precisão, realizando testes regulares.

Ameaça 04: Henri e Rodrigo - Avaliar se a métrica de classificação dos comentários é suficientemente complexa para a correta classificação dos comentários, conferir com testes se não há algum caso que produza resultados incorretos/indesejados.

Ameaça 05: João - Aplicar o 'word embedding' após a realização de testes, definindo as adaptações necessárias, validando-as constantemente e utilizando diferentes frases para testar a identificação de contexto no modelo.

3.3. Canvas Proposta de Valor

O Canvas Proposta de Valor é uma ferramenta de negócios que auxilia no entendimento e criação do posicionamento do projeto (como um produtos que será desenvolvido), com base na criação de ganho que o cliente realmente valoriza e precisa.

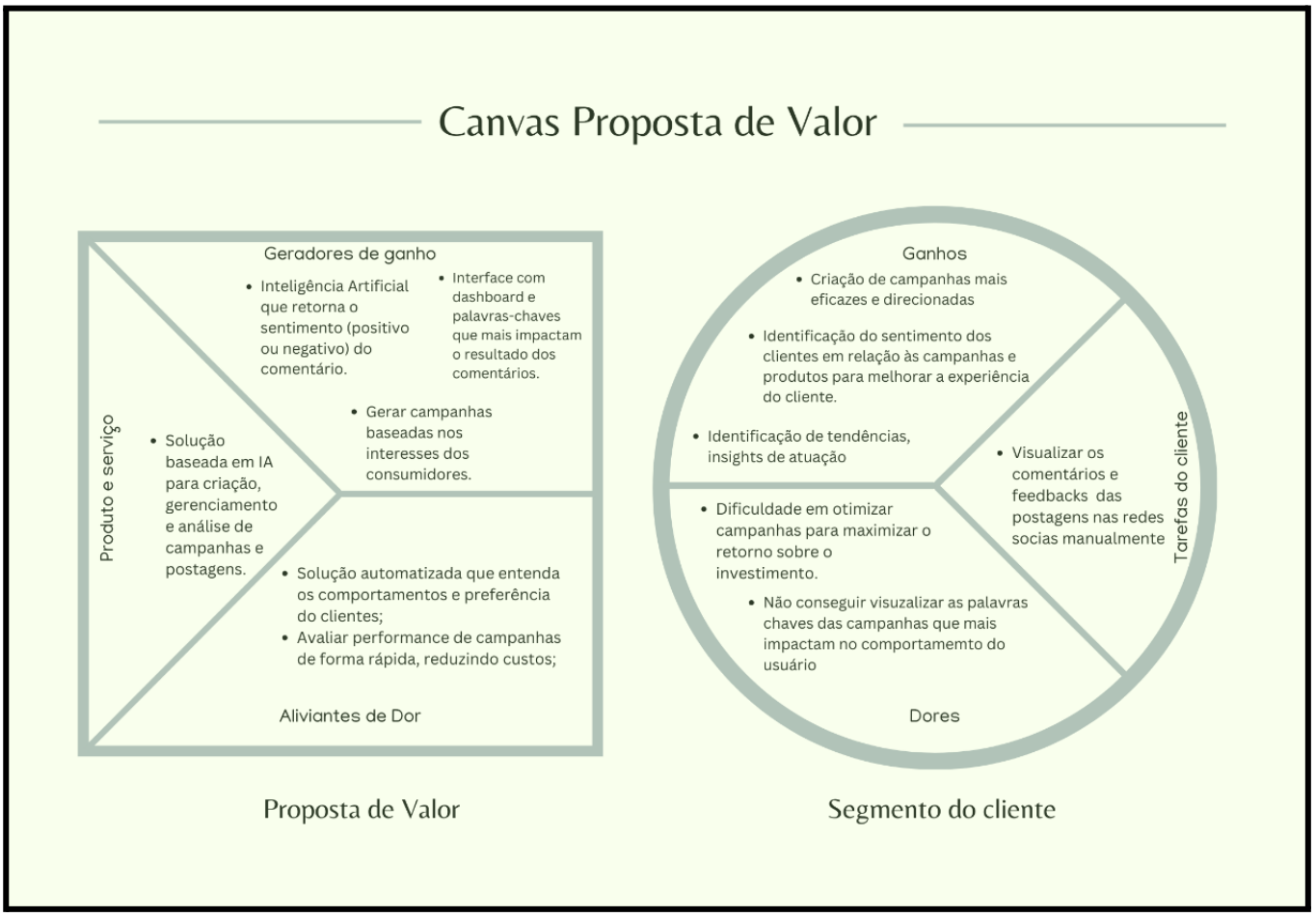


Imagem 3: Canvas Proposta de Valor
Fonte: Autores

3.4. Análise Financeira

A análise financeira corresponde à uma avaliação dos aspectos econômicos e financeiros que permeiam o projeto, com o objetivo de estimar a viabilidade e escalabilidade do mesmo.

Conforme o site de notícias Reuters, o BTG Pactual (BTG Pactual S.A.) é um banco de investimentos brasileiro especializado em capital de investimento e capital de risco. O BTG se configura como uma empresa de capital aberto com cerca de 200 sócios (constituído por funcionários internos). Mesmo sua sede sendo no Rio de Janeiro, Brasil, sua atuação ocorre em escala global, alcançando EUA, Chile, México, Reino Unido, Portugal, Argentina, Colômbia e Peru. Além das já citadas, o banco atua em áreas como 'Corporate Lending (Empréstimos e Financiamentos)', Sales and Trading, Asset Management, Wealth Management e Ativos Florestais. Apesar da queda das receitas anuais totais (de 2022 para 2023) e também das ações nos últimos 6 meses, o banco BTG continua sendo uma das empresas mais relevantes e consolidadas do Brasil.

Custos em relação ao projeto: Os custos estimados pelo cliente foram de R$250.000. Não foram informadas projeções de receita pelo cliente (projeto interno).

Em paralelo, foi feito uma projeção dos custos do projeto, contabilizando os gastos com funcionários e serviços de infraestrutura da AWS, tendo como média o valor de R$ 167.784.

Profissionais Necessários Para o Desenvolvimento da Solução:

- Engenheiro em Machine Learning
- Desenvolvedor de Software
- Cientista de Dados - Opcional (Incomum no Brasil)
- Especialista em linguística Computacional

- Salário Médio para Engenheiro em Machine Learning: R$ 8.900 /mês (Glassdoor
- Salário Médio para Desenvolver Pleno: R$ 10.200 /mês (Glassdoor)
- Salário Médio para Cientista de Dados: R$ 8.710 /mês (Glassdoor)

- Gasto Mensal com os três profissionais: R$ 27.810
- Gasto em dois meses: R$ 55.620
- Gasto em três meses: R$ 83.430

Custos da infraestrutura em AWS:

Sem o Amazon Comprehend:

- Custo Inicial - R$12.312,18
- Custo Mensal - R$34.341,58
- Gastos em dois meses: R$ 80.995,34
- Gastos em três meses: R$ 115.336,92

Com o Amazon Comprehend:

- Custo Inicial - R$12.312,18
- Custo Mensal - R$34.404,18
- Gastos em dois meses: R$ 81.120,54
- Gastos em três meses: R$ 115.524,72

Contabilização total: Considerando os custos dos funcionários, somado ao da AWS, o valor do projeto pode variar entre R$136.615 e R$ 198.954, com uma média de R$ 167.784.

4. Análise de Experiência do Usuário

4.1. Personas

A Persona é a representação fictícia do cliente ideal para o projeto, ela tem com o objetivo compreender os seus comportamentos e necessidades. Nesse momento, entende-se que as personas configuram-se em representantes dos times de marketing, produto e automações.

4.1.1 Analista de Marketing

4.1.2 Gestor de Produtos

Renata Andrade

PERFIL

- Idade: 31
- Ocupação: Gestora de Produtos no banco BTG;
- Graduada em Administração de Empresas, com especialização em Gestão de Produtos.
- Atua no setor de produtos há 5 anos.

COMPORTAMENTO

- Ela está sempre atualizada com as novas tendências e tecnologias do mercado;
- É observadora e está sempre atenta aos dados e informações das campanhas, com o objetivo de tomar decisões assertivas.
- É uma pessoa bastante dedicada e comprometida com o trabalho, sempre buscando superar suas metas e objetivos.

NECESSIDADES

- Solução que acompanhe a repercussão dos produtos do banco nas redes sociais.
- Busca desenvolver estratégias eficazes para melhorar o desempenho dos produtos no mercado, levando em consideração o crescimento do Banco BTG.
- Busca oferecer soluções que atendam as demandas dos clientes.
- Avaliar a performance de campanhas de marketing;
- Gerar campanhas baseadas nos interesses dos consumidores;

4.1.3 Gestor de Produtos

Thiago de Costa Kadeberg

PERFIL

- Idade: 25;
- Ocupação: Gerente de Tech Lead Automation do banco BTG;
- Graduado em Engenharia da Computação;
- Atua no setor de Automation há 4 anos.

COMPORTAMENTO

- Sempre está buscando aprimorar seus conhecimentos de inteligência artificial;
- Thiago sempre utiliza dados para provar seus argumentos.
- É uma pessoa que sempre procura alternativas para resolver problemas do seu dia-a-dia.

NECESSIDADES

- Gerenciar efetivamente projetos de automação
- Identificar e avaliar ferramentas de automação dentro da área de Marketing
- Liderar e motivar sua equipe de automação, fornecendo direção clara e apoio. Isso inclui treinamento e desenvolvimento de habilidades para a equipe e gerenciamento de conflitos internos para garantir o sucesso do projeto.

4.2. User Stories

A User Stories são representações simples e claras dos requisitos e funcionalidades de um software, escritas do ponto de vista do usuário final. Essas histórias ajudam a manter o foco nas necessidades dos usuários e a priorizar as funcionalidades mais importantes para o projeto. Portanto, a seguir, existem duas User Story por persona (Marketing, Produto e Automações).

Testes de Aceitação	C01: Ao selecionar uma determinada campanha, retornar os sentimentos relacionados a ela. Aceitou → Correto, os sentimentos devem ser retornados. Recusou → Errado, um feedback de erro deve ser retornado. C02: Ao selecionar uma determinada campanha, retornar os valores referentes a ela. Aceitou → Correto = Retorna a descrição da campanha Recusou → Errado = o nome inserido da campanha não existe ou foi digitado incorretamente.
	dashboard que deve exibir os sentimentos dos usuários em relação às campanhas do Banco BTG de forma clara; * condição: Possuir os nomes das campanhas relacionadas aos posts e gerar um gráfico com as informações específicas dos sentimentos gerados pela campanha específica; * pós-condição: Retornar o sentimento majoritário dos comentários sobre a campanha. 2) A interface deve permitir a filtragem dos dados por campanha, podendo selecionar uma campanha em específico; * condição: Colocar o nome da campanha. * pós-condição: retornar o sentimento da campanha.

Título	Análise de Palavras Chaves
Número	2.0
Nome	**Juliano Fonseca**
Personas	**Analista de Marketing no Banco BTG**
História	Como analista de Marketing, gostaria de analisar as palavras chaves que mais impactam o resultado do sentimento e sua frequência para poder entender melhor o comportamento do consumidor
Critérios de aceitação	1) Será disponibilizado um dashboard que exibirá as palavras com maior grau de importância em relação ao resultado dos sentimentos produzidos pelas campanhas e comentários dos usuários. * condição: Possuir os nomes das campanhas relacionadas aos posts e gerar a interface (gráfico); * pós-condição: Retornar as palavras com maior importância no resultado do sentimento e a quantidade de vezes que ela foi usada.

Testes de Aceitação	C01: Ao selecionar uma determinada campanha, retornar os sentimentos relacionados a ela. Aceitou → Correto, os sentimentos devem ser retornados. Recusou → Errado, um feedback de erro deve ser retornado. C02: Ao selecionar uma determinada campanha, retornar os valores referentes a ela. Aceitou → Correto = Retorna a descrição da campanha Recusou → Errado = o nome inserido da campanha não existe ou foi digitado incorretamente.

Título	Verifica os Perfis mais ativos
Número	3.0
Nome	**Juliano Fonseca**
Personas	**Analista de Marketing no Banco BTG**
História	Como analista de Marketing, quero encontrar os perfis mais ativos para entender os interesses dos clientes
Critérios de aceitação	1) Será disponibilizado um dashboard que exibirá os usuários mais frequentes e sua quantidade de posts. * condição: Possuir os nomes dos usuários relacionados com seus posts; * pós-condição: Retorna um gráfico com o nome dos usuários e sua quantidade de posts.
Testes de Aceitação	C01: Retornar um dashboard exibindo os usuários mais frequentes e a quantidade de posts de cada um. Aceitou → Correto = retorna o gráfico Recusou →Errado = retorna um feedback dizendo que não foi possível executar o processo

Título	Proporção Geral dos Sentimentos
Número	4.0
Nome	**Renata Andrade**
Personas	**Gestora de produtos no Banco BTG**
História	Como gestora de produtos, gostaria de visualizar a proporção dos sentimentos de todas as campanhas do banco BTG para que seja possível identificar a efetividade do marketing.
Critérios de aceitação	1) Será disponibilizado um dashboard que deve exibir a porcentagem entre todas as campanhas. * condição: Possuir o sentimento relacionado a todas as campanhas; * pós-condição: Retorna um gráfico com a porcentagem de cada sentimento.
Testes de Aceitação	C01: Ao selecionar uma determinada campanha, retornar a porcentagem em relação a todas as campanhas. Aceitou → Correto = retorna o gráfico. Recusou →Errado = retorna um feedback dizendo que não foi possível executar o processo

Título	Selecionar o Pré-processamento
Número	5.0
Nome	**Thiago de Costa**
Personas	**Tech Lead de Automation**
História	Como Tech Lead de Automation, gostaria de ter a capacidade de fazer ajustes técnicos no modelo, como a seleção dos melhores parâmetros de treinamento e a utilização de técnicas de processamento de linguagem natural

Critérios de Aceitação	1) Será disponibilizado um dashboard que deve exibir os sentimentos dos usuários em relação às campanhas do Banco BTG de forma clara; * condição: Possuir os nomes das campanhas relacionadas aos posts e gerar um gráfico com as informações específicas dos sentimentos gerados pela campanha específica; * pós-condição: Retornar o sentimento majoritário dos comentários sobre a campanha. 2) A interface deve permitir a filtragem dos dados por campanha, podendo selecionar uma campanha em específico; * condição: Colocar o nome da campanha. * pós-condição: retornar o sentimento da campanha.
Testes de Aceitação	C01: Ao selecionar uma determinada campanha, retornar os sentimentos relacionados a ela. Aceitou → Correto, os sentimentos devem ser retornados. Recusou → Errado, um feedback de erro deve ser retornado. C02: Ao selecionar uma determinada campanha, retornar os valores referentes a ela. Aceitou → Correto = Retorna a descrição da campanha Recusou → Errado = o nome inserido da campanha não existe ou foi digitado incorretamente.

Título	Métricas de qualidade do Modelo
Número	6.0
Nome	**Thiago de Costa**
Personas	**Tech Lead de Automation**
História	Como Tech Lead de Automation, gostaria de ter uma funcionalidade que me permita avaliar o desempenho do modelo em tempo real, por meio de métricas de qualidade como precisão, recall e f1-score.
Critérios de aceitação	1) Será disponibilizado um dashboard que deve exibir os valores de precisão, recall e f1-score. * condição: Possuir uma métrica avaliando cada campanha; * pós-condição: Retorna um dashboard mostrando os resultados da métrica.
Testes de Aceitação	C01: Ao selecionar uma determinada campanha, retornar os valores para cada métrica em relação a ela. Aceitou → Correto = retorna o dashboard. Recusou → Errado = retorna um feedback dizendo que não foi possível executar o processo

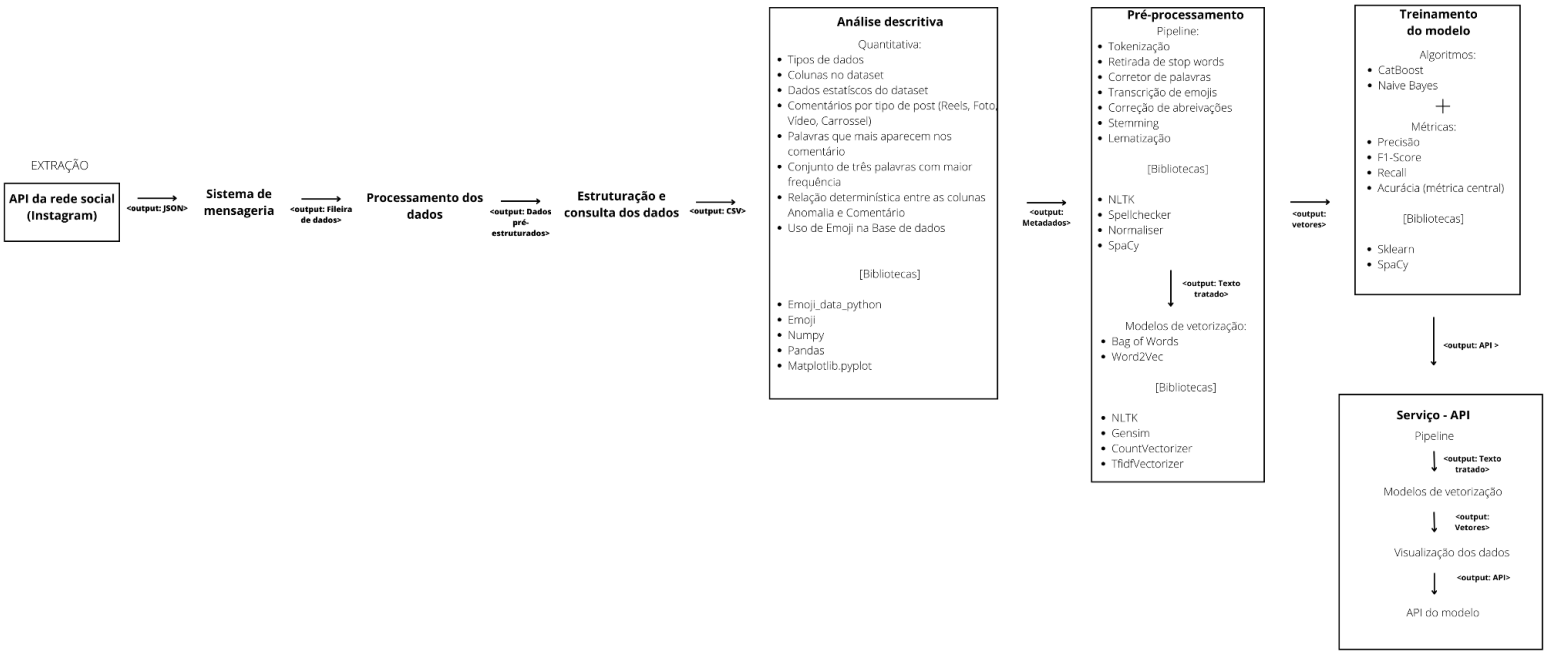
5. Solução Proposta

5.1. Solução

A solução proposta corresponde a um modelo de Processamento de Linguagem Natural, no qual, comentários deixados no Instagram do BTG Pactual são analisados e classificados conforme o sentimento que eles expressam em três categorias, sendo elas: positivo, negativo e neutro.

5.2. Arquitetura Proposta

A arquitetura apresentada a seguir demonstra os steps requeridos ao projeto.



Primordialmente, evidencia-se a extração e transporte dos dados. Em seguida destaca-se o processo de análise descritiva, tendo como output metadados para o pré-processamento. Já no pré-processamento, estrutura-se um pipeline contendo técnicas de processamento dos dados, derivando assim, uma saída condizente aos modelos de vetorização utilizados (Bag of Words e Word2Vec). A partir dos vetores, é possível fazer o treinamento dos

algoritmos e avaliação a partir das métricas. Por fim, com a finalidade de oferecer esse serviço de forma facilitada, ressalta-se o desenvolvimento de uma API, que alimente ferramentas de visualização.

Entende-se a construção da solução dividida nos seguintes blocos:

1- Extração, transporte e processamento:

- Utilizando a API da rede social Instagram, é possível extrair dados de comentários em postagens públicas. Esses dados são retornados em formato JSON, que é uma forma de representação de dados estruturados que facilita a análise e manipulação.
- Depois de extrair os dados da API do Instagram, eles são processados e estruturados em uma fileira de dados, que pode ser um arquivo CSV, por exemplo. Isso facilita a manipulação e análise dos dados em etapas posteriores.
- Em seguida, os dados são enviados para um sistema de mensageria, que pode ser um serviço de mensagens instantâneas(RabbitMQ), por exemplo. Isso permite que os dados sejam acessados por usuários autorizados em tempo real.
- Depois de serem recebidos pelo sistema de mensageria, os dados passam por um processo de processamento, no qual podem ser manipulados e transformados em uma estrutura mais adequada para análise.
- Por fim, os dados são pré-estruturados, o que significa que são organizados em uma estrutura que facilita a consulta e análise. Isso pode incluir a organização dos dados em tabelas ou a atribuição de tags ou categorias específicas para cada dado. Com a pré-estruturação, é possível realizar consultas mais eficientes e respostas mais rápidas a perguntas específicas sobre os dados.

2- Análise descritiva:

- Tipos de dados: identificação de quais são os tipos de dados presentes no dataset. Por exemplo, os dados podem incluir informações sobre o autor do comentário, a data e hora em que o comentário foi feito, o tipo de post (Reels, Foto, Vídeo, Carrossel) e o próprio comentário. Cada um desses tipos de dados pode ser tratado de forma diferente na análise.

- Colunas no dataset: listar todas as colunas no dataset e entender o que cada uma delas representa.
- Dados estatísticos do dataset: é possível calcular estatísticas básicas dos dados, como média, mediana, desvio padrão e intervalos de confiança. Usado para a entender a distribuição dos dados e identificar possíveis outliers.

- Comentários por tipo de post: análise da distribuição dos comentários por tipo de post (Reels, Foto, Vídeo, Carrossel). Usado para entender qual tipo de postagem gera mais engajamento e interação com os usuários.
- Palavras que mais aparecem nos comentários: é possível identificar as palavras mais frequentes nos comentários. Isso pode ajudar a entender os principais temas e assuntos abordados pelos usuários e possíveis stop-words. Essa análise pode ser feita utilizando técnicas de processamento de linguagem natural, como a tokenização e a contagem de frequência de palavras.
- Conjunto de três palavras com maior frequência (trigramas): além das palavras individuais, é possível identificar os conjuntos de três palavras mais frequentes nos comentários. Isso pode ajudar a entender quais as expressões mais comuns utilizadas pelos usuários.
- Relação determinística entre as colunas Anomalia e Comentário: Análise se há alguma relação determinística entre a presença de anomalias em uma postagem e a natureza dos comentários feitos pelos usuários.
- Uso de emojis na base de dados: é possível identificar o uso de emojis nos comentários e analisar quais são os emojis mais frequentes. Isso pode ajudar a entender a emoção e o sentimento dos usuários em relação às postagens. Sendo realizado uma substituição por palavras de seus respectivos significados.

3- Pré-processamento do Dataset

No pré-processamento do dataset, são realizadas etapas de limpeza e preparação dos dados antes de serem usados em análises ou modelos de machine learning. Isso envolve a aplicação de técnicas e bibliotecas para transformar os dados brutos em um formato adequado para análise.

Neste caso específico, foram utilizadas as seguintes bibliotecas adicionais para realizar o pré-processamento:

- Normaliser: A biblioteca "Normaliser" é uma ferramenta utilizada para normalizar e padronizar textos. Ela oferece recursos para lidar com a normalização de caracteres, remoção de caracteres especiais, conversão de letras maiúsculas para minúsculas, entre outros. Essa biblioteca pode ser útil para garantir que os textos do dataset estejam em um formato consistente e pronto para serem processados.
- SpaCy: A biblioteca "SpaCy" é uma biblioteca de processamento de linguagem natural (NLP) em Python. Ela fornece uma variedade de recursos para realizar tarefas de processamento de texto, como tokenização, lematização, reconhecimento de entidades nomeadas, análise de dependência, entre outros. O SpaCy é amplamente utilizado em tarefas de NLP e pode ser aplicado no pré-processamento de dados para extrair informações relevantes e realizar análises mais avançadas.

Para utilizar essas bibliotecas, é necessário instalá-las previamente no ambiente Python em que o código está sendo executado. Você pode instalar as bibliotecas usando os seguintes comandos:

```
!pip install Normaliser
```

```
!pip install spacy
```

Esses comandos utilizam o gerenciador de pacotes pip para instalar as bibliotecas "Normaliser" e "SpaCy" em seu ambiente Python. Após a instalação, você pode importar essas bibliotecas em seu código e utilizar suas funcionalidades para realizar as etapas necessárias de pré-processamento do dataset.

Pipeline

No pré-processamento do dataset, são realizadas etapas de limpeza e preparação dos dados antes de serem usados em análises ou modelos de machine learning. Isso envolve a aplicação de técnicas e bibliotecas para transformar os dados brutos em um formato adequado para análise. Neste caso, o pré-processamento do dataset foi realizado utilizando um pipeline que incluiu as seguintes etapas:

- Tokenização: A tokenização é o processo de dividir o texto em unidades menores, chamadas de tokens. Os tokens podem ser palavras individuais, pontuações, números, ou outras unidades significativas. A biblioteca "SpaCy" foi utilizada para realizar a tokenização dos textos, dividindo-os em tokens que podem ser processados individualmente.
- Retirada de stopwords: Stop words são palavras comuns que geralmente não contribuem muito para o significado do texto, como "a", "o", "para", etc. A remoção de stop words é uma etapa comum no pré-processamento de textos para reduzir o ruído e melhorar a eficiência da análise. A biblioteca "SpaCy" foi utilizada para remover as stop words dos textos do dataset.
- Corretor de palavras: Um corretor de palavras é utilizado para identificar e corrigir erros ortográficos em palavras. A biblioteca "pyspellchecker" foi utilizada como corretor de palavras, verificando e corrigindo erros ortográficos nas palavras do texto.
- Transcrição de emojis: Emojis são símbolos usados para expressar emoções, sentimentos ou ideias em mensagens de texto. A biblioteca "emoji" foi utilizada para realizar a transcrição dos emojis presentes nos textos, convertendo-os em uma representação textual adequada.
- Correção de abreviações: Abreviações e formas reduzidas de palavras são comuns em mensagens de texto, mas podem dificultar a compreensão e análise dos textos. Foi utilizado um método para corrigir abreviações, substituindo-as por suas formas completas para facilitar a interpretação dos textos.
- Stemming: O stemming é um processo de redução de palavras à sua forma base ou radical. Ele remove os sufixos das palavras, mantendo apenas o núcleo significativo. O stemming pode ajudar a reduzir a dimensionalidade dos dados e agrupar palavras relacionadas. No pré-processamento, foi aplicado o stemming nas palavras do texto.
- Lematização: A lematização é um processo semelhante ao stemming, mas em vez de simplesmente remover sufixos, ele retorna a forma base da palavra com base em seu significado léxico. A lematização considera a parte gramatical da palavra e busca o "lemma" correspondente. O objetivo é obter uma representação mais precisa das palavras no texto. A biblioteca "SpaCy" foi utilizada para realizar a lematização das palavras.

Após a aplicação dessas etapas de pré-processamento, o texto tratado estará pronto para análises posteriores, como modelagem de tópicos, classificação de sentimentos, entre outros.

Modelos de vetorização:

Após as etapas de pré-processamento mencionadas anteriormente, foram utilizados dois modelos de vetorização para representar os textos de forma numérica. Esses modelos são:

- Bag of Words (Saco de Palavras): O modelo Bag of Words é uma abordagem comum na vetorização de textos. Ele cria uma representação numérica dos textos, considerando a frequência de ocorrência de cada - palavra em um documento ou em todo o corpus. Cada documento é representado por um vetor em que cada elemento corresponde a uma palavra e seu valor é a contagem de ocorrências dessa palavra no documento. Essa representação é adequada para muitas tarefas de processamento de texto, como classificação de documentos e análise de sentimentos.
- Word2Vec: O Word2Vec é um modelo de vetorização baseado em redes neurais que captura as relações semânticas entre as palavras. Ele mapeia palavras para vetores densos de valores reais, de modo que palavras semanticamente similares fiquem próximas no espaço vetorial. Essa técnica permite representar palavras como vetores contínuos e capturar relações de significado entre elas, como analogias e similaridades. O modelo Word2Vec é amplamente utilizado para tarefas de processamento de linguagem natural, como análise de sentimento, recomendação de palavras e tradução automática.
- Após a aplicação desses modelos de vetorização, o texto tratado é representado em forma numérica, pronta para ser utilizada como entrada para algoritmos de aprendizado de máquina ou análises estatísticas. O output dessas etapas de vetorização será uma matriz numérica em que cada documento é representado por um vetor de características correspondentes às palavras ou conceitos relevantes extraídos do texto.

Bibliotecas para os modelos de vetorização

Para o modelo acima, foram utilizadas as seguintes bibliotecas e técnicas para realizar o pré-processamento do texto:

- NLTK (Natural Language Toolkit): O NLTK é uma biblioteca popular em Python para processamento de linguagem natural. Ele oferece uma ampla gama de recursos e ferramentas para tarefas como tokenização, lematização, stemming, análise sintática,

entre outros. A biblioteca NLTK foi utilizada para realizar algumas etapas de pré-processamento, como tokenização e stemming.

- Gensim: O Gensim é uma biblioteca de processamento de linguagem natural em Python que oferece ferramentas para modelagem de tópicos, indexação e similaridade de documentos. Neste contexto, o Gensim foi utilizado para aplicar o modelo Word2Vec mencionado anteriormente, que permite a vetorização baseada em palavras.

- CountVectorizer: O CountVectorizer é uma classe da biblioteca scikit-learn em Python que permite a vetorização de textos usando a abordagem Bag of Words. Ele transforma o texto em uma matriz numérica em que cada documento é representado por um vetor que contém a contagem de ocorrência de cada palavra. O CountVectorizer foi utilizado como um modelo de vetorização baseado em frequência de palavras.
- TfidfVectorizer: O TfidfVectorizer também é uma classe da biblioteca scikit-learn em Python que realiza a vetorização de textos usando a abordagem TF-IDF (Term Frequency-Inverse Document Frequency). Essa abordagem leva em consideração a frequência de uma palavra em um documento específico e sua frequência inversa em todo o corpus. Isso permite atribuir maior importância a palavras menos frequentes e reduzir a importância de palavras muito comuns. O TfidfVectorizer foi utilizado como um modelo de vetorização baseado no esquema TF-IDF.
- Após a aplicação dessas técnicas e bibliotecas de pré-processamento, o texto tratado é transformado em uma representação numérica adequada para análise posterior, como classificação de documentos, agrupamento de tópicos ou outros modelos de aprendizado de máquina. O output dessas etapas de vetorização será uma matriz

numérica em que cada documento é representado por um vetor de características correspondentes às palavras relevantes extraídas do texto.

4- Treinamento:

6- Serviço - API:

5.3. Diagrama Macro da Solução

Apresente o diagrama macro da solução. Aqui deve-se apresentar o diagrama macro da solução. O diagrama deve representar as jornadas do usuário.

Destaque:

** Front End*

** Back End*

** Serviços (API)*

5.4. Descrição da Solução

Descreva o passo-a-passo dos módulos ao leitor (como os módulos se encaixam logicamente).

Nesse ponto, o ideal é que o cliente/stakeholder consiga imaginar o funcionamento do MVP em relação aos processos de negócio.

5.5. Algoritmos

Na área de Processamento de Linguagem Natural (NLP), os algoritmos desempenham um papel fundamental no desenvolvimento de modelos preditivos e na extração de informações valiosas a partir de dados textuais. Nesta seção, exploraremos os algoritmos que fizeram parte da modelagem e testes aplicados ao projeto, sendo eles: Naive Bayes, Regressão Logística, CatBoost, e Rede Neural.

5.5.1 Algoritmos utilizados

1. CatBoost:

O CatBoost é um algoritmo de aprendizado de máquina desenvolvido pela Yandex. Ele é particularmente adequado para lidar com dados categóricos, como variáveis com valores discretos. O CatBoost utiliza a técnica de "gradient boosting" para construir um modelo preditivo. Ele é capaz de tratar automaticamente variáveis categóricas, evitando a necessidade de codificação manual dessas variáveis. Além disso, o CatBoost inclui recursos como tratamento automático de valores ausentes e implementações eficientes para lidar com grandes conjuntos de dados.

2. Naive Bayes:

O Naive Bayes é um algoritmo de aprendizado de máquina probabilístico baseado no teorema de Bayes. Ele é amplamente utilizado em problemas de classificação de texto, como filtragem de spam, análise de sentimento e categorização de documentos. O Naive Bayes assume a independência entre as características, o que simplifica o cálculo das probabilidades condicionais. Embora essa suposição simplificadora nem sempre seja verdadeira na prática, o Naive Bayes é conhecido por sua simplicidade, eficiência computacional e bom desempenho em muitos casos.

3. Regressão Logística:

A Regressão Logística, é um modelo estatístico usado para classificação binária ou multiclasse. Este algoritmo é frequentemente empregado para identificação de tópicos e detecção de padrões específicos. É um algoritmo simples de entender e interpretar, fornecendo coeficientes que indicam a importância das características na classificação. Ele modela a relação entre variáveis independentes e a probabilidade de uma resposta pertencer a uma determinada classe. Sua escolha foi feita especialmente pela sua capacidade de lidar com dados categóricos (como é o caso da base de dados trabalhada).

4. Rede Neural:

As redes neurais são modelos de aprendizado profundo que se inspiram no funcionamento do cérebro humano. Elas podem aprender representações complexas dos dados textuais e lidar com situações no corpus, um pouco mais desafiadoras, como tradução automática, geração de texto (NLG) e resumo automático. No projeto, utilizamos com o modelo BERT e ELMo, ambos são baseados em redes neurais.

6. Desenvolvimento e Resultados

6.1. Módulo 6

6.1.1. Descrição

Para realizar as análises descritivas e o modelo NLP (Linguagem de Processamento Natural), foi necessário o recebimento de uma base de dados com as respectivas classificações dos sentimentos dos comentários. Com isso, o uso de tecnologias possibilitou o pré-processamento, vetorização e implementação de modelos de inteligência artificial. Portanto, para que seja possível uma melhor visualização e escalabilidade, foi criada uma API do modelo escolhido.

6.1.2. Tecnologia adotada

No projeto, foi adotada uma ampla gama de tecnologias com a intenção de criar um sistema robusto e eficiente. A linguagem de programação escolhida para o desenvolvimento foi o Python, devido à sua ampla disponibilidade de bibliotecas e facilidade de uso para o processamento de linguagem natural. Para o processamento de linguagem natural, foram utilizadas diversas bibliotecas especializadas, incluindo Spacy, NLTK, Textblob, String, BeautifulSoup, Emoji e entre outros. As bibliotecas principais na parte de pré-processamento, foram o Spacy e a NLTK realizando tarefas como tokenização, stemming, lematização e análise gramatical, fornecendo uma base sólida para a compreensão maior do corpus.

Além disso, foram empregadas bibliotecas como Pandas e Numpy para o gerenciamento eficiente de dados, permitindo a manipulação, compreensão e análise dos dados textuais de

maneira mais descritiva. Essas bibliotecas desempenharam um papel crucial em todas as etapas, principalmente, no pré-processamento dos dados e na estruturação dos conjuntos de dados, tendo em vista que possuem “chamadas” que facilitam a conversão e manipulação dos dataframes.

Para a implementação dos algoritmos, foram utilizadas as bibliotecas Scikit Learn, Gensim, TensorFlow, Torch. Essas bibliotecas oferecem uma ampla variedade de algoritmos de aprendizado de máquina e redes neurais, permitindo a construção de modelos de classificação e análise de sentimentos precisos e eficazes.

A visualização dos resultados e a criação de gráficos interativos foram realizadas por meio das bibliotecas Plotly, Matplotlib e Wordcloud (essa com o objetivo um pouco mais simples para compreender). As seguintes bibliotecas forneceram recursos interessantes para a representação visual dos dados, como a Curva ROC, Curva de Aprendizado, Histogramas, facilitando a interpretação dos resultados e a apresentação das informações de forma clara e intuitiva.

Ao adotar essas tecnologias, fomos capazes de extrair insights valiosos dos textos, permitindo uma compreensão mais profunda das opiniões e percepções em relação aos produtos e serviços oferecidos pelo banco. A combinação de linguagem Python e bibliotecas especializadas forneceu uma base sólida para o desenvolvimento de uma solução eficiente e escalável, capacitando o banco a tomar decisões estratégicas.

6.1.3. Modelagem

Nesta seção serão expostos os modelos de vetorização e resultados aplicados nos algoritmos durante a sprint 4.

1. Elmo

O Elmo (Embeddings from Language Models) é um modelo de vetorização pré-treinado que captura representações contextuais de palavras e frases. Tal utilização é fundamentada nas características estruturais do modelo. Evidencia-se a possibilidade de identificar palavras iguais ou semelhantes em contextos diferentes, agrupá-los e diferenciá-los. O modelo se destaca dado que possui capacidade de generalização, dado que, foi treinado em grandes quantidades de texto. Ao aplicá-lo, a fim de obter melhores resultados, utiliza-se o csv do texto

pré-processado. Em seguida, aplica-se no modelo de vetorização ELMo. É de indubitável importância ressaltar que para a utilização do ELMo os arquivos de pré-treinamento, em português precisam ser importados.

“

weight_file = "/content/drive/MyDrive/Colab Notebooks/SI-MOD6/entrega_spt4/elmo_pt_weights_dgx1 (1).hdf5

“

Já a linha a seguir, define o local do arquivo JSON de opções para o modelo ELMo em português. O arquivo JSON contém a configuração e os hiperparâmetros do modelo.

options_file = 'https://s3-us-west-2.amazonaws.com/allennlp/models/elmo/contributed/pt/brwac/options.json'

As frases de input são convertidas em identificadores de caracteres e, em seguida, é calculado os embeddings correspondentes. Os resultantes são armazenados em um array. Após a vetorização dos dados, os mesmos são direcionados para a rede neural simples. Na rede neural, seta-se 2 dimensões dos embeddings redimensionados, uma vez que, a dimensionalidade do input, sem tratamento, é incompatível com o suporte da rede. Na mesma, são passados os vetores resultantes do ELMo e os targets.

Defini-se, então, a arquitetura da rede neural em duas camadas densas. A primeira camada possui 2 unidades com ativação sigmoidal, e a segunda camada possui 1 unidade com ativação sigmoidal.

Destaca-se a utilização do otimizador “Adam” devido a frequente utilização em conjuntos de dados enxutos e com redes profundas. Além disso, o otimizador atualiza os pesos da rede mais rapidamente.

A métrica escolhida para avaliar o resultado do modelo é recall, visto que, dentre todas as classificações positivas como valor esperado, quantas foram assertivas. Entretanto,

ocorreram alguns desajustes durante o arranjo da rede neural. Portanto, nesse momento, a fim de avaliar a performance geral, utiliza-se a acurácia.

Os resultados emitidos na rede neural demonstram 43.17% de acurácia. A porcentagem apresentada, em consideração ao que é considerado ideal ou avaliativo para os modelos, não é satisfatória para aplicação, em comparação à modelos que obtiveram resultados mais assertivos.

Em síntese, destaca-se que esse número pode ser melhorado em decorrência da separação dos dados de treino e teste, diferenciação no número de epochs e variação nas dimensões.

2. BERT

O BERT (Bidirectional Encoder Representations from Transformers) é um modelo de linguagem baseado em Transformers, arquitetura que é baseada em mecanismos de "atenção" e relevância. Isto permite que o modelo capture relações de dependência entre palavras de maneira eficiente e paralela.

Neste caso, o que utilizamos é uma versão pré-treinada, com o uso do "preenchimento de máscara" (masked language modeling), com a intenção de uma futura implementação da funcionalidade de predição da "próxima sentença" (next sentence prediction). Uma das principais características do BERT é sua capacidade de gerar representações contextualizadas de palavras. O modelo leva em consideração o contexto em que uma palavra aparece, o que ajuda a capturar relações e significados mais precisos. Consequentemente, essas representações contextualizadas são obtidas através do pré-treinamento bidirecional e podem ser usadas como entrada para o algoritmo.

Algoritmo: Rede Neural

Como algoritmo, utilizamos a Rede Neural na arquitetura Transformers, para realizar as etapas de predição.

Etapas de implementação

Preparação dos dados (exemplo com Label Encoder realizado):

```
label_encoder = LabelEncoder()

df_lemma['sentimento_3'] =
label_encoder.fit_transform(df_lemma['sentimento'])
```

Pré-treinamento do modelo BERT:

```
model = BertModel.from_pretrained('neuralmind/bert-base-portuguese-cased')
```

Esta etapa envolve também a definição dos parâmetros, a alimentação dos dados no modelo, atribuição dos tensores e da famosa "attention_mask".

Avaliação e ajuste:

Nesta etapa realizamos a avaliação do modelo, com base nos dados de treino e teste que foram previamente divididos, e realizamos a iteração do modelo sobre o conjunto de treinamento, para assim, em seguida, atribuir as predições às categorias (e targets) definidos. Por fim, realizamos a impressão das métricas de classificação.

Após diversas implementações do modelo, e devido a sua complexidade nas etapas de entrada da Rede Neural, obtivemos resultados medianos com o modelo BERT. Consequentemente, com a primeira implementação, na métrica de maior relevância para o projeto, o "Recall" obtivemos 72% (0.72). Em termos de assertividade, em comparação aos outros modelos foi o nosso terceiro maior.

No entanto, devido ao seu alto nível de complexidade, obtivemos resultados com muitos sinais de overfitting e do enviesamento na entrada dos tensores, o que ocorreu após tentativa de resolução de problemas no código.

Devido a isso, tratamos o modelo BERT como um "nice to have", mas não como principal modelo a ser utilizado. Mesmo assim, consideramos que na escala de modelos com algoritmos de Rede Neural, é com certeza o que mais gera escalabilidade e por isso há o interesse em sua implementação.

Próximos passos

Fine-tuning para tarefas específicas: Depois do pré-treinamento, o BERT pode ser ajustado (fine-tuned) para tarefas específicas, como classificação de texto, extração de

informações e resposta a perguntas. Ao ajustar o BERT para uma tarefa específica, as camadas de classificação são adicionadas ao modelo e o modelo é treinado em um conjunto de dados rotulados para aprender a tarefa específica.

Attention Mask: Na tarefa de preenchimento de máscara, palavras são mascaradas aleatoriamente e o modelo é treinado para prever as palavras mascaradas com base no contexto das palavras vizinhas e sua relevância.

É importante citar que o modelo BERT tem sido amplamente utilizado em uma variedade de tarefas de processamento de linguagem natural e estabeleceu um novo parâmetro em muitas delas. Isso foi um dos fatores determinantes para a escolha e utilização deste modelo em nosso grupo. Ele se destaca pela sua capacidade de capturar informações contextuais em textos e fornecer representações de alta qualidade que podem ser usadas em várias aplicações de NLP.

3. Doc2Vec

O modelo Doc2Vec é uma técnica de aprendizado de máquina utilizada para representar documentos em formato vetorial. Ele é uma extensão do modelo Word2Vec, que é usado para representar palavras em vetores. O Doc2Vec permite que documentos inteiros sejam representados como vetores contínuos de valores numéricos, capturando o contexto semântico dos documentos.

O objetivo do Doc2Vec é gerar representações vetoriais para documentos que preservem a semântica e a similaridade entre eles. Essas representações vetoriais podem ser usadas em várias tarefas de processamento de linguagem natural, como classificação de documentos, recomendação de conteúdo, agrupamento de documentos semelhantes e recuperação de informações.

Algoritmos utilizados: Naive Bayes, Regressão Logística e Rede Neural

No processamento do modelo, realizamos alguns tipos de targeamento, que são próprios para treinamento do modelo, assim como a definição de parâmetros e outros recursos de vetorização. Abaixo segue o exemplo de targeamento:

tagged_data = [TaggedDocument(words=word_tokenize(text.lower()), tags=[str(i)]) for i, text in enumerate(dados['texto'])]

Após estas definições realizamos a vetorização dos tokens com base nas definições da própria biblioteca gensim, e após impressão das iterações destes vetores, podemos aplicar nos algoritmos.

** É importante ressaltar que a métrica de avaliação central foi o Recall **.

Na primeira implementação, com Regressão Logística, tivemos a porcentagem de 58% (0.58), em relação às classificações, assim como na segunda implementação em Random Search, que obtivemos, também, o recall em 58%. Já na terceira implementação utilizando Naive Bayes, conseguimos o bom resultado de 82% (0.82) em recall.

Por fim, implementamos a Rede Neural, apenas para satisfazer os termos acadêmicos, e obtivemos a acurácia de 73% (0.73).

No caso específico, o modelo Doc2Vec foi escolhido como modelo final devido às suas vantagens e desempenho em relação aos outros métodos de representação de documentos. Além disso, levamos em conta a sua aplicabilidade, tendo em vista que o Doc2Vec é capaz de capturar relações semânticas complexas entre palavras e documentos, gerando vetores que preservam a semelhança semântica entre textos.

Também foi evidenciado, nos testes que realizamos, que o modelo Doc2Vec possui uma implementação mais simples, eficiente e bem estabelecida na biblioteca Gensim, o que facilita sua utilização e treinamento. E em termos de assertividade, com a métrica de prioridade (Recall), em relação aos outros modelos, o modelo Doc2vec foi muito bem.

4. GloVe

O Global Vectors for Word Representation, também conhecido como GloVe, é um modelo de vetorização de palavras desenvolvido com o intuito de identificar as relações sintáticas e semânticas em um conjunto de um texto. Esse modelo, utiliza estatísticas de co-ocorrência global de palavras para desenvolver representações vetoriais. O seu processo de treinamento,

envolve a construção de uma matriz que registra a frequência da ocorrência das palavras a partir da utilização da 'função de perda' (loss function) com o intuito de maximizar a probabilidade de co-ocorrência de pares de palavras.

Algoritmo: Regressão Logística

O código está trabalhando com a leitura de um arquivo CSV, contendo os dados lematizados que já passaram pelo Pré-Processamento. Depois disso, como mostra o código abaixo, foi realizado o carregamento do modelo spaCy com vetores GloVe e feito um teste com a palavra 'amor' para calcular seus vetores.

```
import spacy
# Carregamento do modelo com a utilização de vetores GloVe
nlp = spacy.load('en_core_web_sm')


# Vetor da palavra teste (amor)
word_vector = nlp('amor')[0].vector
print("Vetor de 'amor':", word_vector)
```

Após isso, foi realizado a utilização da classe CountVectorizer do sklearn.feature_extraction.text para vetorizar os dados contidos na coluna "texto" doDataframe desenvolvido. Assim, a vetroização foi aplicada aos dados de teste (x_test) e treinamento (x_train) e armazenados em duas variáveis com o parâmetro 'random_state' definido como 42. Depois disso, foi realizado o treinamento do modelo de regressão logística utilizando os dados de treinamento. Dessa forma, a acurácia do modelo é calculada a partir do método score com X_test e y_test.

```
X = df_lemma['texto']
y = df_lemma['sentimento']


vectorizer = CountVectorizer()
X_counts = vectorizer.fit_transform(X)
tfidf_transformer = TfidfTransformer()
X_tfidf = tfidf_transformer.fit_transform(X_counts)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y,
test_size=0.2, random_state=42)
```

```
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
accuracy = model.score(X_test, y_test)
print("Acurácia:", accuracy)
```

Com base nos resultados obtidos é possível fazer as seguintes análises do algoritmo de Regressão Logística. A acurácia obtida para o modelo foi de 0.75, o que indica uma boa significância e uma taxa de acerto razoavelmente alta. Além disso foi também calculado os valores de precisã para cada uma das classes desenvolvidas. O valor obtido para as classe NEGATIVE, NEUTRAL e POSITIVE foram: 0,77; 0,71; e 0,81 respectivamente. Valores relativamente bons. Além disso, foi obtido também os valores de recall para cada uma das classes (NEGATIVE = 0,54; NEUTRAL = 0,85 e POSITIVE = 0,75. Por fim, foi então calculado o F1-score, que é uma medida que combina a precisão e o recall em uma única métrica. Para a classe NEGATIVE, o F1-score é de 0,64, para a classe NEUTRAL é de 0,77 e para a classe POSITIVE é de 0,78.que o modelo tem uma taxa de acerto razoavelmente alta.

Algoritmo: Modelo Naive Bayes

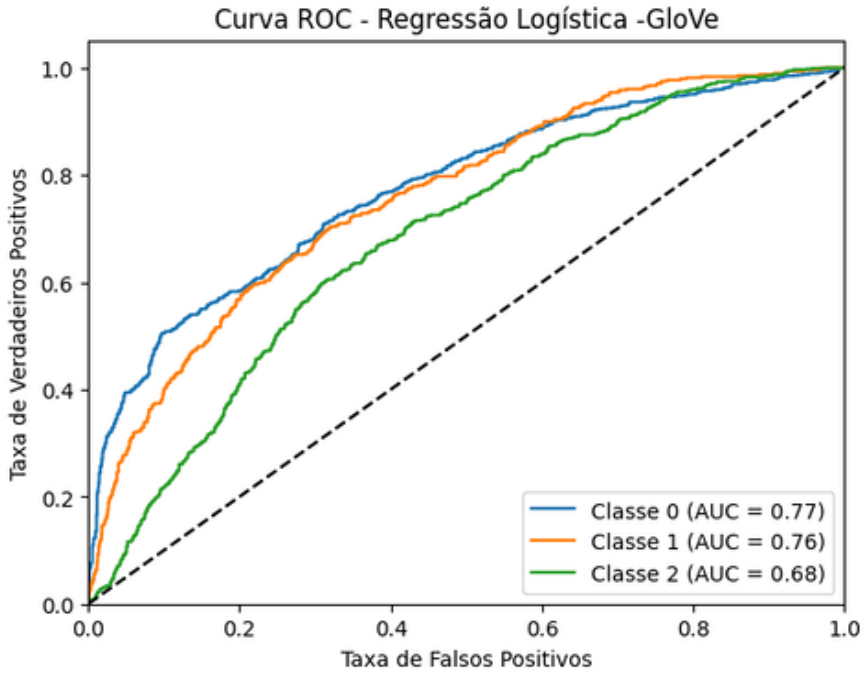
Primeiramente foi mapeado os rótulos 'POSITIVE', 'NEUTRAL' e 'NEGATIVE' para os valores numéricos 3, 1 e 2, respectivamente, e o resultado foi armazenado em uma variável.

```
sentimento_mapping = {'POSITIVE': 3, 'NEUTRAL': 1, 'NEGATIVE': 2}
y_mapped = df['sentimento'].map(sentimento_mapping)
```

Depois disso, foi realizado o treinamento do modelo Naive Bayes Gaussiano, porém não foi obtido um valor de acurácia tão satisfatório, equivalente a 0.57. Isso indicou que o modelo tem uma taxa de acerto um pouco mais baixa com o valor obtido previamente com o algoritmo da regressão logística.

```
model = GaussianNB()
model.fit(X_train, y_train)

# Predição e cálculo da acurácia
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

Por fim, foi realizado um gráfico de plotagem da curva ROC, que pode ser visto abaixo.



Em resumo, os resultados obtidos indicam que o modelo de regressão logística teve um desempenho melhor em comparação com o modelo de Naive Bayes. A regressão logística obteve uma acurácia mais alta e um melhor equilíbrio entre precisão e recall. Além disso, o gráfico de curva ROC mostra que o modelo de regressão logística possui uma capacidade satisfatória de distinguir entre as diferentes classes. No entanto, é importante lembrar que esse modelo obteve valores bons, porém não foram os melhores diante dos outros modelos desenvolvidos.

5. FastText

O FastText é uma biblioteca gratuita e de código aberto do Facebook AI Research (FAIR) para aprender embeddings e classificações de palavras. Este modelo permite a criação de um

algoritmo de aprendizado para a obtenção de representações vetoriais de palavras, avaliando esses modelos.

Primeiramente, os valores da coluna alvo foram transformados em valores numéricos:

```
df_2['sentimento'] = df_2['sentimento'].map({'NEUTRAL': 0, 'POSITIVE': 1, 'NEGATIVE': -1})
```

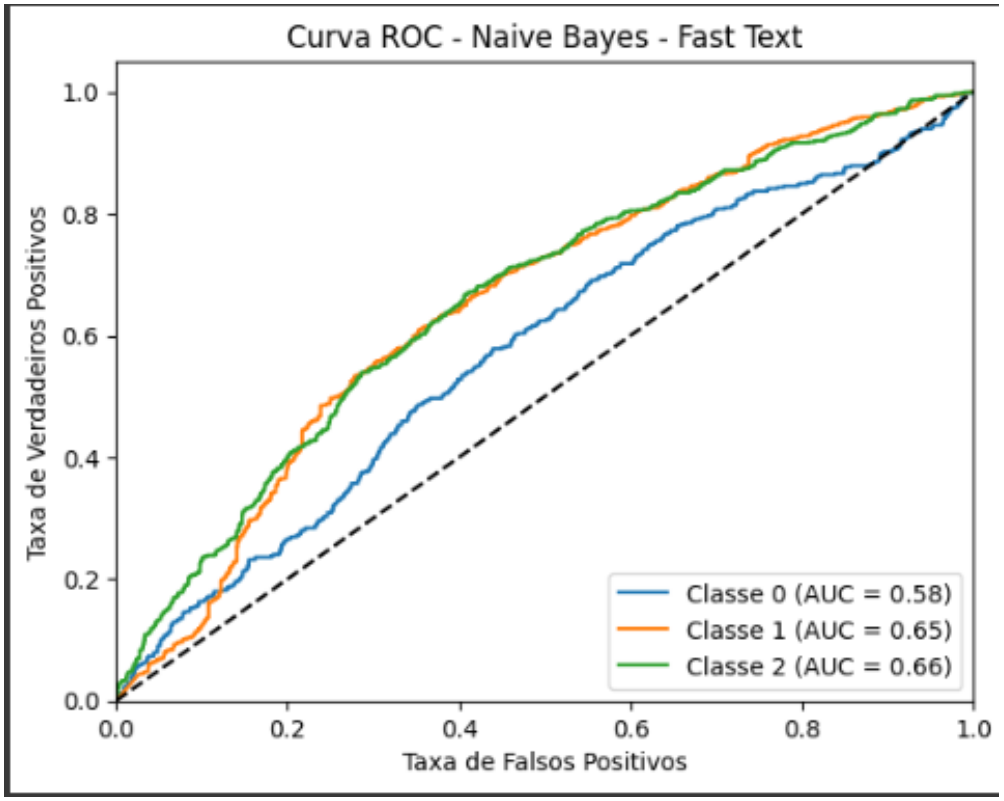
Foi feito o teste com o modelo possuindo 50 e 100 dimensões mas, como a diferença não foi significante, foi utilizado o de 50. O modelo de word embeddings pré treinadas foi instalado do repositório de word embeddings do NILC. Para aplicação no projeto, foi utilizada a base de dados com os dados já lematizados e tratados. O modelo foi carregado e aplicado em uma vetorização da coluna 'tokens', assim como é feito com o modelo Word2Vec.

```
# Função para vetorizar um token
def vetorizar_token(token):
  vetor = np.zeros(model.vector_size) # incializa vetor de zeros com a mesma dimensão
  if token in model: # verifica se a palavra está no word2vec treinado
    vetor = model[token] # adiciona o valor do vetor
  return vetor

# Função para vetorizar uma frase
def vetorizar_frase(frase):
  vetores_tokens = [vetorizar_token(token) for token in frase] # verifica cada token da lista
  return np.sum(vetores_tokens, axis=0) # retorna a soma dos vetores

# Aplicar a função 'vetorizar_frase' a todas as frases
df_2['vetores'] = df_2['tokens'].apply(vetorizar_frase)
```

Após isso, com os vetores criados, foram aplicados o modelo Naive Bayes (que retornou uma acurácia de 0.31) e a Regressão Logística (com esta última obtendo um melhor resultado, com acurácia de 0.56). Estes resultados foram um pouco inferiores em relação aos testes realizados em outros modelos com os mesmos algoritmos (Regressão logísitica e Naive Bayes).

Sendo assim, o modelo não foi considerado com grande relevância para a escolha do modelo final.

Com base nos resultados e processamentos, construímos diferentes gráficos: o da curva ROC, o da curva de aprendizado e o da curva de validação. Abaixo estão os gráficos da curva ROC obtidos com cada modelo:



6. TF-IDF

O TF-IDF (Term Frequency-Inverse Document Frequency) é uma medida estatística que permite avaliar a importância de uma palavra em um documento. Essa técnica foi escolhida por sua capacidade de destacar palavras-chave relevantes e reduzir o peso de palavras comuns, ajudando a identificar a relevância de um termo em relação ao contexto específico de um documento. Suas vantagens incluem a capacidade de lidar com grandes volumes de texto de maneira eficiente, reduzindo essa influência de palavras comuns e destacando exatamente os

termos-chave que fornecem insights relevantes. O TF-IDF também é uma técnica simples de implementar e interpretar.

Algoritmo: Regressão Logística

A Regressão Logística é um algoritmo de aprendizado de máquina que é amplamente utilizado para tarefas de classificação binária. Ele modela a relação entre variáveis independentes e a probabilidade de uma resposta pertencer a uma determinada classe. Sua escolha foi feita especialmente pela sua capacidade de lidar com dados categóricos (como é o caso da base de dados trabalhada).

A combinação de TF-IDF com Regressão Logística aproveita as vantagens de ambos os métodos. O TF-IDF fornece uma representação ponderada das palavras em um documento, enquanto a Regressão Logística modela a relação entre essas palavras e a probabilidade de classificação. Essa combinação é comum em processos de análise de sentimentos como esse.

A base de dados utilizada já passou pelos primeiros processos de PLN (remoção de stop words, substituição de gírias e, em destaque, a lematização). Essa é a mesma base de dados lematizada que foi desenvolvida na Sprint 3 e utilizada no modelo anterior. Contudo, apenas as colunas das frases e suas qualificações estão sendo utilizadas.

```
#dividi os dados em conjunto de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(dados["texto"], dados["sentimento"], test_size=0.2, random_state=42)
```

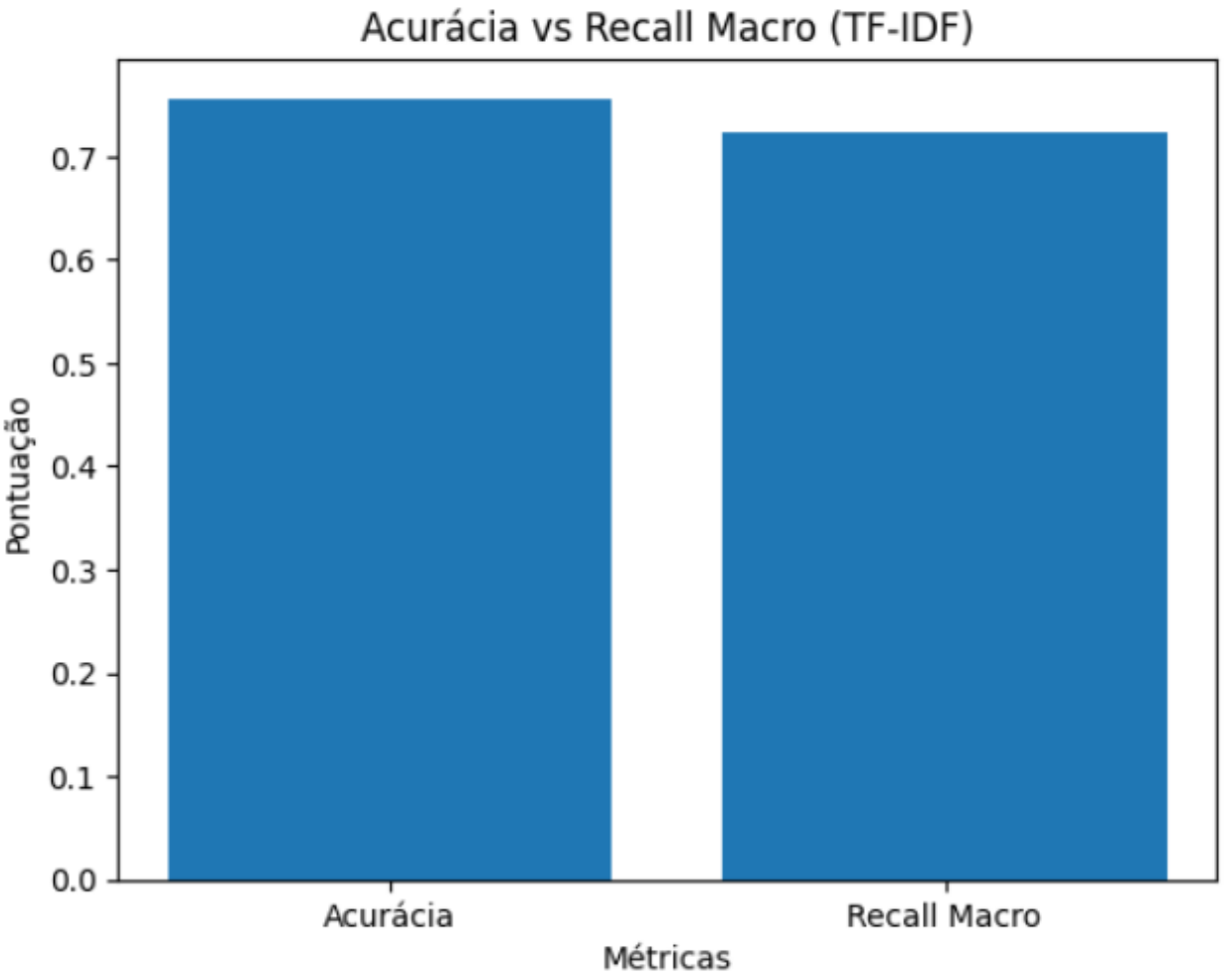
Após a importação da base de dados no modelo, foi feita uma divisão dos dados em conjunto de treinamento e teste, com o test size de 0.2 e random state de 42.

```
#cria pipeline com TF-IDF e modelo de classificação
pipeline = Pipeline([
  ("tfidf", TfidfVectorizer()),
  ("model", LogisticRegression(max_iter=1000))  # número máximo de iterações
])
```

Como já indicado, o modelo criado utiliza TF-IDF com Regressão Logística para a análise de sentimentos. Para a regressão Logística o número de iterações foi configurado para mil a fim de conseguir um modelo que identifica mais correlações entre as palavras.

Para avaliar os resultados obtidos foi utilizado principalmente os parâmetros de Recall e Acurácia:

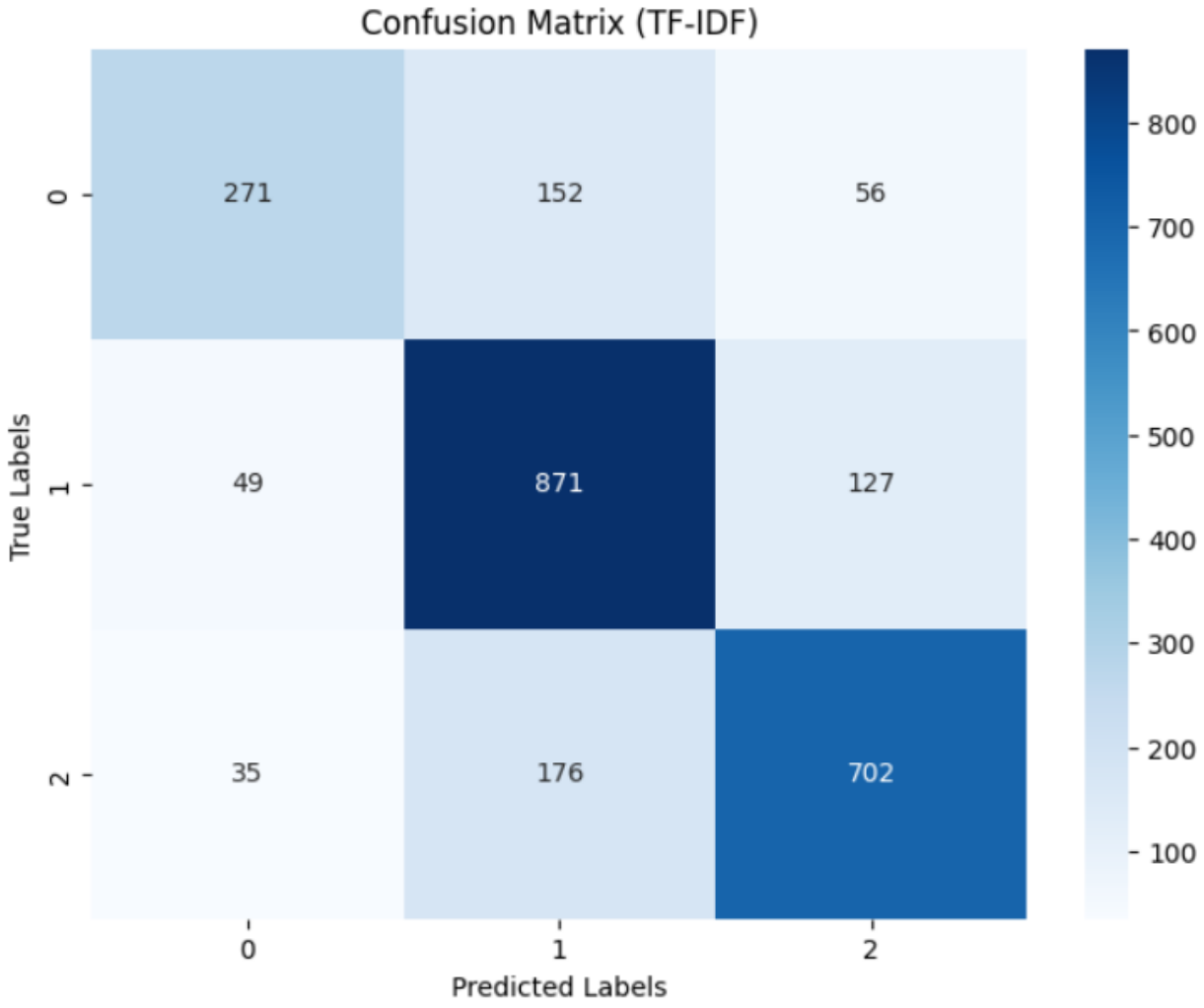
```
Acurácia: 0.7560475604756047
Recall (macro): 0.7221854765326053
```



Com base nos resultados obtidos, podemos concluir que o modelo apresenta um desempenho equilibrado e consistente, com valores de acurácia e recall macro próximos, na faixa de 0,7. Isso indica que o modelo é capaz de fazer previsões precisas na maioria das

amostras de teste, classificando corretamente tanto as instâncias positivas quanto as negativas.

Além disso, a similaridade entre a acurácia e o recall macro sugere que o modelo não está enviesado para uma classe específica e está tratando ambas as classes de forma equilibrada.



A partir da matriz de confusão é possível perceber que o modelo tem mais dificuldades ao avaliar comentários neutros e uma facilidade para avaliar positivos e, em segundo lugar, negativos. Os resultados nessa instância foram satisfatórios.

Por fim, o modelo obteve bom resultados em avaliações de recall, acurácia e matriz de confusão, contudo, não foram os melhores diante dos outros modelos desenvolvidos.

7. Word2vec

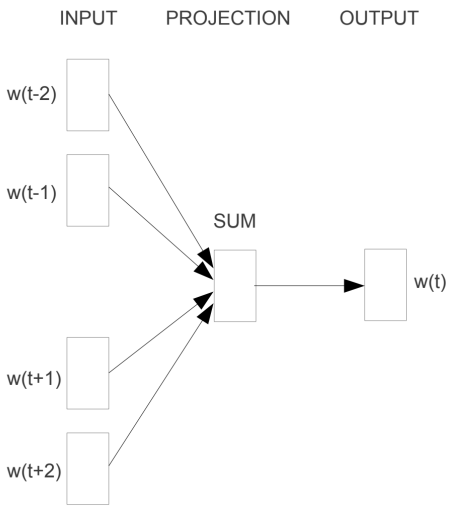
O modelo Word2Vec é uma técnica de PLN que permite representar palavras como vetores numéricos em um espaço de várias dimensões. Este processo consiste em capturar relações entre as palavras com base nos seus contextos. Como resultado final, é possível ter uma representação matemática da similaridade entre as palavras disponibilizadas para o treinamento.

Ele, também, permite capturar nuances e contextos que podem influenciar o sentimento de um texto, que são dados pelo nível de similaridade e proximidade entre as palavras. Além disso, o word2vec pode facilitar a extração de características relevantes para a classificação, reduzindo a dimensionalidade e a esparsidade dos dados textuais. Estes fatores, são determinantes na justificativa da sua utilização no projeto, assim como sua importância.

O modelo word2vec pode ser combinado com outros modelos de aprendizado de máquina com facilidade, para obter melhores resultados de classificação. O word2vec também pode ser usado para gerar incorporações de frases ou documentos inteiros, usando técnicas como média ou soma dos vetores das palavras (neste caso utilizamos a soma, com adição de uma coluna em um novo dataframe). Portanto, o word2vec é um modelo muito útil para a construção e desenvolvimento de nossa análise, pois permite representar as palavras de forma mais rica e eficiente, capturando aspectos semânticos e sintáticos que afetaram na classificação de determinado corpus.

Arquitetura do Modelo Word2Vec: CBOW

O modelo Word2Vec possui duas arquiteturas principais: CBOW (Continuos Bag-Of-Words) e Skip-Gram. Nossa equipe optou por utilizar o modelo CBOW pois computacionalmente ele é mais eficiente. Esse tipo de arquitetura recebe as palavras circundantes e tenta prever a palavra central. Ambos os modelos (CBOW e Skip-Gram) são treinados para maximizar a probabilidade de previsão correta das palavras.



CBOW

Construção do Modelo Word2Vec

Para a construção do modelo Word2Vec, a equipe fez uma nova limpeza e pré-processamento de dados, só que agora, na segunda base disponibilizada:

- Substituição de Emojis: nas frases, substituímos os emojis por palavras. Esse processo melhora e abrange a base de dados trabalhada. Segue exemplo abaixo:

```
pipeline('Eu gosto de investir nesse banco 🚀 ')
```

```
['gostar', 'investir', 'em esse', 'banco', 'foguete']
```

- Substituição de Abreviações: substituímos abreviações por suas formas originais. Segue visualização do exemplo abaixo:

```python
def substituir_abreviacoes(frase):
    dicionario_girias = {
    'vc': 'voce',
    'vcs':'voce',
    'Vc': 'voce',
    'pq': 'porque',
    'tbm': 'tambem',
    'q': 'que',
    'td': 'tudo',
    'blz': 'beleza',
    'flw': 'falou',
    'kd': 'cade',
    'to': 'estou',
    'mt': 'muito',
    'cmg': 'comigo',
```

- Lematização: é o processo de transformação de palavras para sua forma base (derivação inversa). Para esse método de pré-processamento utilizamos a bibliotecas spaCy, como é evidenciado no exemplo abaixo:

	autor	texto	sentimento	tokens_lema
0	v8_capital	Confira os resultados dos nossos fundos no mês...	NEUTRAL	[confira, o, resultado, de o, nosso, fundo, me...
1	winthegame_of	A Alvarez & Marsal estará conosco no Sportainm...	NEUTRAL	[Alvarez, Marsal, estara, conosco, sportainmet...
2	marta_bego	#Repost btgpactual with make_repost • • • Entend...	NEUTRAL	[repost, btg, With, makerepost, entenda, o, im...
3	lmviapiana	Minuto touro de ouro	POSITIVE	[minuto, touro, ouro]
4	vanilson_dos	@ricktolledo Sim	NEUTRAL	[Ricktolledo, sim]

- Modelo word2Vec e Características: já na construção do Modelo Word2Vec em si, configuramos seus parâmetros da seguinte forma: 150 vetores de dimensionalidade, 5 janelas de contexto, contagem mínima de palavras para 1 e 4 threads para treinamento paralelo:

```python
# Crie o modelo Word2Vec
modelo_w2v = Word2Vec(tokens, vector_size=150, window=5, min_count=1, workers=4)

# O parâmetro 'size' define a dimensionalidade dos vetores de palavra
# O parâmetro 'window' define o tamanho máximo da janela de contexto
# O parâmetro 'min_count' define a contagem mínima de ocorrências para uma palavra ser considerada
# O parâmetro 'workers' define o número de threads para treinamento paralelo (aumenta a velocidade)
```

Definições e construção do modelo Word2Vec

- Vetorização para Word2Vec: a vetorização consiste em transformar dados textuais em representações numéricas. É um processo crucial para a construção do modelo Word2Vec, só assim será possível organizar a distribuição das palavras em um plano. Todos os tokens são vetorizados e suas somas em uma frase também são contabilizados.

```
# Função para vetorizar um token
def vetorizar_token(token):
    vetor = np.zeros(modelo_w2v.vector_size) #incializa vetor de zeros com a mesma dimensão
    if token in modelo_w2v.wv: #verifica se a palavra está no word2vec treinado
        vetor = modelo_w2v.wv[token] #adiciona o valor do vetor
    return vetor

# Função para vetorizar uma frase
def vetorizar_frase(frase):
    vetores_tokens = [vetorizar_token(token) for token in frase] # verifica cada token da lista
    return np.sum(vetores_tokens, axis=0) # retorna a soma dos vetores

# Aplicar a função 'vetorizar_frase' a todas as frases
df_texto['vetores'] = df_texto['tokens_lema'].apply(vetorizar_frase)
df_texto
```

Função para o processo de Vetorização

- Output e tabela pós Word2Vec: ao final do processo de Word2Vec, configuramos a tabela que será utilizada para algoritmos de aprendizado. Um dos passos é a transformação da coluna de sentimentos para valores numéricos. O segundo passo é exatamente a atribuição dos tokens às 150 colunas definidas.

```
df_texto['tokens_lema'] = np.array(df_texto['tokens_lema'])
df_texto['sentimento'] = df['sentimento'].map({'NEUTRAL': 0, 'POSITIVE': 1, 'NEGATIVE': -1})
df_texto
```

Mapeamento da Coluna Sentimentos

```
sentence_table = []
#Pecorre cada token e faz seu vetor correspondente
for sentence in df_texto['tokens_lema']:
    word_vectors = [modelo_w2v.wv[word] for word in sentence if word in modelo_w2v.wv]
    if len(word_vectors) > 0:
        sentence_vector = sum(word_vectors) / len(word_vectors)
    else:
        sentence_vector = [None] * 150 # Cria uma lista de 150 elementos None
    sentence_table.append((sentence, *sentence_vector[:150]))  # Adiciona apenas os primeiros 150 elementos do vetor

column_labels = ['tokens_lema']

#looping para adicionar a coluna vetor com seu número correspondente
for i in range(150):
    column_labels.append(f'vec{i+1}')

# Cria um novo DataFrame com os vetores de sentença
df_vec = pd.DataFrame(sentence_table, columns=column_labels)
# Define o índice do DataFrame df_vec como o mesmo índice de df_texto['sentimento']
df_vec.set_index(df_texto["sentimento"].index, inplace=True)
# Adiciona a coluna 'sentimento' ao DataFrame df_vec a partir de df_texto['sentimento']
df_vec['sentimento'] = df_texto["sentimento"]
```

função para distribuição dos tokens nos vetores

	tokens_lema	vec1	vec2	vec150	sentimento
0	['confira', 'o', 'resultado', 'de o', 'nosso',...	0.262776	-0.351657	-0.315743	0
1	['Alvarez', 'Marsal', 'estara', 'conosco', 'sp...	0.029775	-0.371140	-0.509014	0
2	['repost', 'btg', 'With', 'makerepost', 'enten...	0.222636	-0.364943	-0.431190	0
3	['minuto', 'touro', 'ouro']	0.019667	0.067234	-0.002506	1
4	['Ricktolledo', 'sim']	0.304922	-0.007242	-0.183279	0
...	...	...	...	...	...
12169	['um', 'noite', 'encontro', 'muito', 'conhecim...	0.385639	-0.163126	-0.428794	0
12170	['erro', 'financeiro', 'eliminar', 'antes', 'd...	0.501189	-0.143322	-0.398472	0
12171	['estar', 'muito', 'grato', 'todo', 'esforco',...	0.574576	0.287958	-0.245109	1
12172	['dorsodamaocomdedoindicadorapontandoparaadire...	0.304480	0.013746	-0.196458	0
12173	['btg', 'Morning', 'call', 'nao', 'este', 'mai...	0.503737	0.093182	-0.268835	-1

Acima podemos visualizar a tabela final pós Word2Vec

Algoritmos de Aprendizado:

O objetivo do nosso projeto é exatamente fazer a classificação de frases com o intuito de conferir o desempenho de campanhas de marketing, assim, apenas o modelo Word2Vec não é o suficiente, pois mesmo organizando a similaridade de palavras, ele não consegue fazer a classificação de sentimentos. A solução é utilizar algoritmos de aprendizado supervisionado para fazer esse tipo de classificação. Neste sentido, nesta Sprint, testamos alguns algoritmos, mas optamos pela utilização do Naive Bayes e do Catboost.

Naive Bayes:

O Naive Bayes foi o primeiro algoritmo testado pelo grupo, ele se baseia em uma teoria matemática de probabilidades condicionais (teorema de Bayes). O algoritmo se destaca por sua eficiência e simplicidade. A biblioteca utilizada para esse método foi o sklearn (GaussianNB). A principal intenção do grupo, era de usar o algoritmo para realizar o cálculo da probabilidade condicional de cada palavra ou "n-grama" ocorrer em cada classe, com o intuito de estimar a probabilidade do texto pertencer a uma classificação de sentimento específico.

```
from sklearn.model_selection import train_test_split
dropar = df_vec[['tokens_lema', 'sentimento']]
X = df_vec.drop(columns = dropar)
y = df_vec['sentimento']
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Naive Bayes
classifier = GaussianNB()

# Treino de classificação
classifier.fit(X_train, y_train)

# Predição do teste
y_pred = classifier.predict(X_test)

# Validando modelo
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print classification report
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```

Construção do modelo Naive Bayes aplicado

CatBoost:

O catboots é outro algorítimo de classificação, se destacando principalmente com dados com características categóricas e dados desbalanceados. Esse algorítimo se baseia em conhecimentos matemáticos de gradiente (gradient boosting). É importante ressaltar que o Catboost é muito usado na definição de características categóricas como palavras ou frases, sem a necessidade de codificá-las numericamente, o que pode reduzir a complexidade e o tempo de processamento. Com base nestes fatores, e mediante o uso prévio de alguns membros de nosso grupo, decidimos optar pela sua utilização nesta Sprint.

```python
from sklearn.model_selection import GridSearchCV
from catboost import CatBoostClassifier

# Crie uma instância do modelo CatBoost
classifier = CatBoostClassifier()

# Defina um dicionário de hiperparâmetros a serem ajustados
param_grid = {
    'learning_rate': [0.01, 0.1, 1.0]
}

# Crie uma instância de GridSearchCV
grid_search = GridSearchCV(classifier, param_grid=param_grid, cv=5)

# Treine o modelo com a busca em grade
grid_search.fit(X_train, y_train)

# Obtenha a melhor combinação de hiperparâmetros encontrada pela busca em grade
best_params = grid_search.best_params_

# Use o modelo com os melhores hiperparâmetros para fazer previsões no conjunto de testes
y_pred = grid_search.predict(X_test)
```

Construção do modelo CatBoost aplicado

- Resultados dos Algoritmos de Aprendizado Supervisionado:

Diante das etapas exemplificadas acima, dividimos nossos dados para realizar o treinamento e avaliação do nosso modelo, com base na estruturação realizada. Sendo dividimos em duas seções

- Dados de treino: separação com o intuito de fazer o modelo aprender as características e os padrões dos dados que permitem fazer previsões ou classificações.
- Dados de teste: separação usada como método de verificação do modelo, com base nas previsões ou classificações corretas e precisas.

Naive Bayes:

```
Accuracy: 0.5552361396303901
Classification Report:
              precision    recall  f1-score   support

          -1       0.39      0.80      0.53       513
           0       0.80      0.48      0.60      1073
           1       0.57      0.50      0.53       849

    accuracy                           0.56      2435
   macro avg       0.59      0.59      0.55      2435
weighted avg       0.63      0.56      0.56      2435
```

```
Acurácia de treinamento: 0.5484135948249307
Acurácia de teste: 0.5552361396303901
```

Os resultados conferidos pelo Naive Bayes foram satisfatórios mas não ideais. Com 54% de acurácia de treinamento e 55% de acurácia total

CatBoost:

```
Accuracy: 0.7215605749486653
Classification Report:
              precision    recall  f1-score   support

          -1       0.64      0.64      0.64       501
           0       0.76      0.75      0.76      1038
           1       0.73      0.73      0.73       896

    accuracy                           0.72      2435
   macro avg       0.71      0.71      0.71      2435
weighted avg       0.72      0.72      0.72      2435
```

```
Acurácia de treinamento: 0.9519457849881918
Acurácia de teste: 0.7232032854209446
```

```
[[317  77 119]
 [ 93 801 179]
 [ 82 124 643]]
```

Os resultados conferidos pelo CatBoost foram satisfatórios. Entretanto, apresenta um overfitting, já que existe 95% de acurácia de treinamento e 72% de acurácia total, tendo uma diferença grande entre as duas separações, portanto, sendo necessário entender o motivo. Também foi obtido resultados satisfatórios na matriz de confusão.

8. Bag Of Words

O modelo Bag-of-Words é uma abordagem comum no pré-processamento de texto usada para representar documentos de texto como vetores numéricos. É uma técnica simples e amplamente utilizada em tarefas de processamento de linguagem natural.

No contexto do Google Colab, o pré-processamento com o modelo Bag-of-Words envolve as seguintes etapas:

Tokenização: O texto é dividido em unidades menores chamadas "tokens". Geralmente, os tokens são palavras individuais, mas também podem ser caracteres, n-grams (sequências de n tokens consecutivos) ou outras unidades, dependendo do caso de uso.

Construção do vocabulário: O vocabulário é criado coletando todos os tokens únicos presentes nos documentos de texto. Cada token único é atribuído a um índice único no vocabulário.

Codificação dos documentos: Cada documento de texto é codificado como um vetor numérico de acordo com o vocabulário construído. O tamanho do vetor é igual ao tamanho do

vocabulário. Cada posição no vetor representa uma palavra do vocabulário, e o valor naquela posição indica a frequência ou outra medida de importância do termo no documento.

Matriz de documentos-termos: Todos os documentos são representados em uma matriz, em que cada linha corresponde a um documento e cada coluna corresponde a um termo do vocabulário. Os valores da matriz são geralmente contagens de frequência, mas também podem ser pesos TF-IDF (term frequency-inverse document frequency) ou outros esquemas de ponderação.

Essa representação baseada no modelo Bag-of-Words permite que os algoritmos de aprendizado de máquina trabalhem com dados de texto, que normalmente requerem entrada numérica. No Google Colab, você pode implementar essas etapas usando bibliotecas de processamento de texto, como NLTK (Natural Language Toolkit), e aplicá-las aos seus dados de texto para prepará-los para tarefas de classificação, agrupamento ou outras análises.

Funções utilizadas:

O código fornecido realiza a vetorização de texto usando o CountVectorizer da biblioteca scikit-learn. Vejamos o que cada linha faz:

- Importação das bibliotecas: -Essas linhas importam as classes CountVectorizer e TfidfVectorizer da biblioteca sklearn.feature_extraction.text, necessárias para realizar a vetorização de texto. from sklearn.feature_extraction.text import CountVectorizer from sklearn.feature_extraction.text import TfidfVectorizer
- Instanciação do vetorizador: -Aqui, um objeto CountVectorizer é criado e atribuído à variável vectorizer. O CountVectorizer é usado para converter o texto em uma matriz de contagens de palavras. vectorizer.fit(frases_pre)
- Ajuste do vetorizador aos dados de entrada: -Essa linha ajusta o vetorizador aos dados de entrada frases_pre. Ele analisa o texto fornecido, constrói o vocabulário e atribui um índice numérico único a cada palavra encontrada nas frases. vectorizer.fit(frases_pre)
- Exibição do vocabulário ordenado: -Essa linha imprime o vocabulário ordenado alfabeticamente. O vocabulário é um dicionário que mapeia as palavras encontradas nas frases para seus respectivos índices numéricos. print(sorted(vectorizer.vocabulary_))

- Transformação dos dados em uma representação vetorial:

Aqui, o método transform é chamado para converter as frases pré-processadas frases_pre em uma matriz vetorial. Cada linha da matriz representa uma frase, e cada coluna representa uma palavra do vocabulário. O valor em cada posição da matriz representa a contagem de ocorrências da palavra correspondente na frase. "vector = vectorizer.transform(frases_pre)"

- Sumarização dos resultados: Essas linhas exibem a forma (shape) da matriz resultante, que indica o número de frases e o tamanho do vocabulário. Em seguida, é impressa a representação em formato de array da matriz vetorizada, mostrando as contagens de palavras para cada frase. print(vector.shape) print(vector.toarray())

6.2. Comparação entre os modelos e escolha do modelo final

Os modelos foram submetidos aos seus respectivos processos de vetorização e, em seguida, aos mesmos algoritmos, sendo eles, regressão logística e naive bayes, a fim de garantir a avaliação conforme a mesma otimização. Além do mais, ressalta-se a utilização do recall como métrica central de avaliação, pois, para a problemática estabelecida, é importante identificar verdadeiros positivos.

Portanto, observa-se, respectivamente, as maiores precisões dos modelos: Doc2Vec, GloVe, Bert, TF-IDF e Fast-Text. Dessa forma, nota-se esse resultado como possibilidade de realizar melhores combinações no modelo Doc2Vec e os algoritmos associados, com o objetivo de resultados mais precisos.

Comparação entre modelos

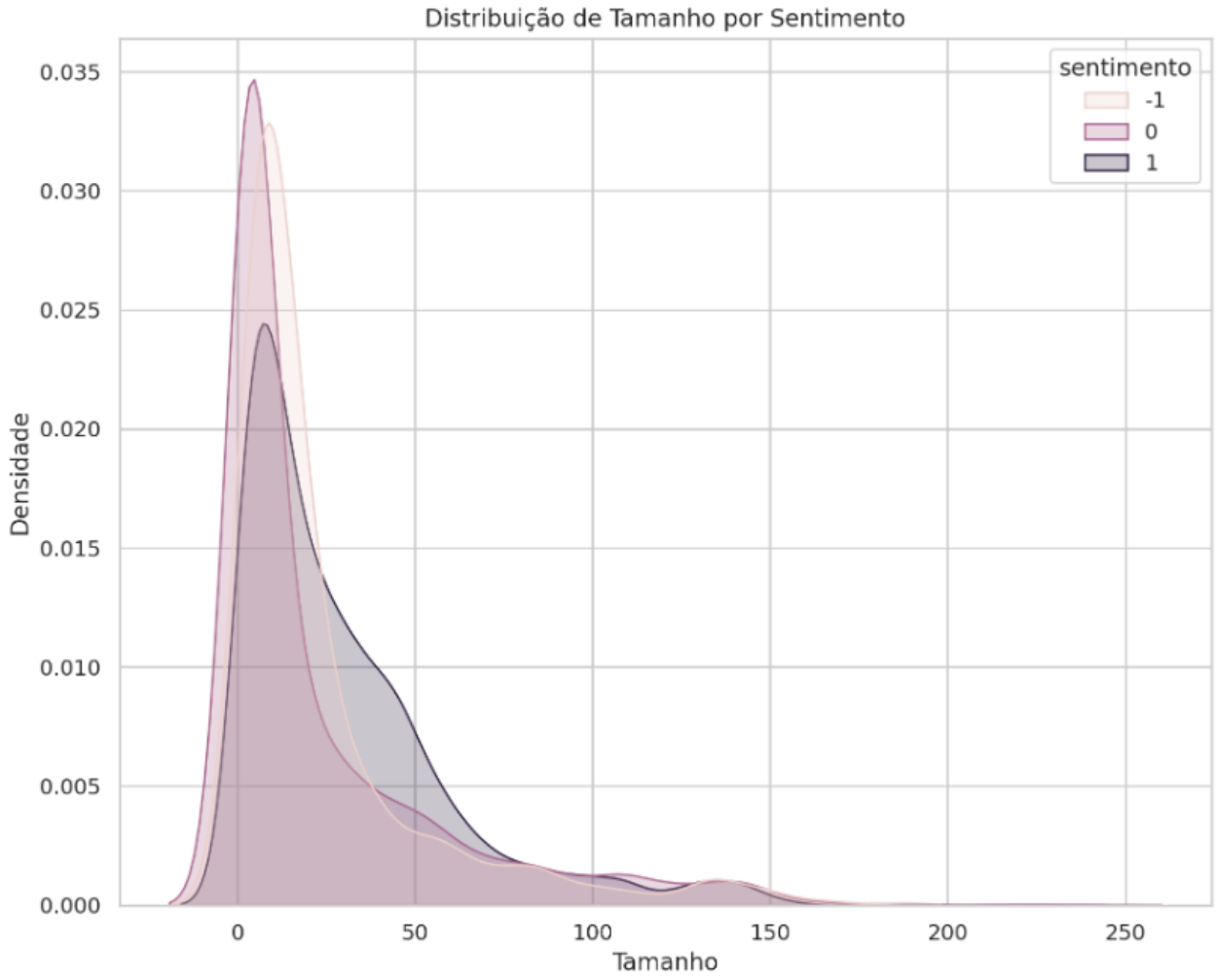
Modelos	Recall	Ordem de assertividade
GloVe	0.74	2º
Doc2Vec	0.77	1º
BERT	0.73	3º
TF-IDF	0.72	4º
FastText	0.68	5º
ELMo	-	-

Esclarece-se que os resultados do ELMo não são expostos na comparação entre os modelos, dado que, durante a exploração do mesmo, foi possível perceber a complexidade no qual está equiparado, além da utilização de redes neurais profundas, diferente dos demais modelos, que utilizam algoritmos.

6.3. Adicionando Features Novas

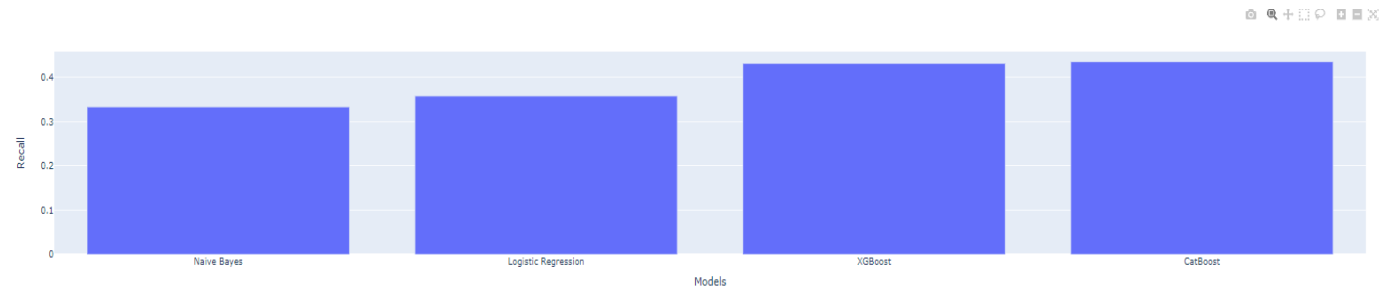
Foram criadas novas features baseadas no tamanho das frases, em particular o número de tokens por frase, para comparar com modelos que incluam ou não essas features. Quatro algoritmos foram utilizados para esta comparação: regressão logística, cat-boost, naive-bayes e xg-boost. A vetorização escolhida para esta comparação foi o TF-IDF devido sua facilidade de aplicação e a combinação com os algoritmos escolhidos. Todos os modelos que utilizaram as novas features tiveram resultados inferiores em comparação aos modelos que não as utilizaram, apresentando uma média de recall de 38.7. No entanto, a inclusão das novas features permitiu a utilização do gráfico KDE (Estimativa de Gráfico Kernel) que demonstra a probabilidade de um sentimento, dado o número de tokens de uma frase. Onde “-1” representa

sentimentos negativos, 0 representa sentimentos neutros e 1 representa sentimentos positivos.



Cálculo KDE para a Estimativa: KDE(x) = (1 / (n * h)) * Σ K((x - xi) / h) KDE(x) é o valor estimado da densidade de probabilidade para um determinado ponto x. n é o número total de pontos de dados no conjunto de dados. h é o parâmetro de suavização ou largura de banda (bandwidth). Σ representa a soma ao longo de todos os pontos de dados. K é a função kernel, que é uma função simétrica em torno de zero que define a forma da contribuição de cada ponto de dados para a estimativa de densidade.

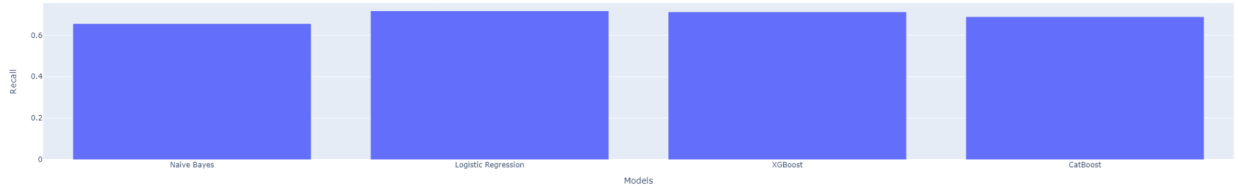
Comparação entre os modelos com a feature aplicada:



Comparação entre os modelos com a feature aplicada:

```
Train acc: 0.5170820799312419
Test acc: 0.4991408934707904
```

Modelos sem a adição da feature aplicada:



Maior valor (xg-boost):

```
Train acc: 0.8650623119896863
Test acc: 0.7332474226804123
```

6.4. Prototipação

Coloque aqui a lista dor frames relacionadas ao módulo.

Informe apenas o Número dos frames (não duplique a prototipação).

6.5. Diagramas

Casos de Uso, Sequência, Processos ou de implantação.

Seção opcional.

6.6. Avaliação

Apresente evidências de como os critérios de qualidade foram alcançados.

Aqui deve-se evidenciar os testes de aceitação das user stories, previamente escritos.

Teste

6.6.1 Avaliação e métricas relevantes para o modelo

Para avaliar o desempenho do nosso modelo, precisamos definir algumas métricas que nos permitam quantificar a sua capacidade de classificar corretamente os textos em categorias como positivo, negativo ou neutro. Essas métricas devem levar em conta não apenas a taxa de acerto do modelo, mas também cada tipo de erro e o balanceamento das classes nos dados.

Nesta seção, vamos apresentar as principais métricas que usamos para avaliar o nosso modelo, explicar como elas são calculadas e interpretadas, e mostrar os resultados obtidos com o nosso conjunto de teste. As métricas que utilizamos são:

- Acurácia: foi uma das métricas que mais olhamos na definição e relevância dos dados que foram passados. A acurácia, em termos mais objetivos, se refere a “taxa de acerto” do modelo. Ela é calculada dividindo o número de previsões corretas pelo número total de previsões. É importante ressaltar que seu balanceamento foi feito de acordo com a base de dados que nos foi deferida.
- Precisão: pode ser definido como a proporção de previsões positivas que são realmente positivas. Ela mede a confiabilidade do modelo em prever a classe positiva.
- ***Revocação**: é a proporção de positivos reais que são corretamente previstos como positivos. Ela mede a sensibilidade do modelo em capturar a classe positiva, mas pode ignorar a quantidade de falsos positivos.

- F1-score: é a média harmônica entre precisão e revocação.

Ela mede o equilíbrio entre essas duas métricas, dando mais peso aos valores baixos. Ela é útil para comparar modelos que têm trade-offs entre precisão e revocação.

**métrica de maior ênfase para o modelo*

Para termos referência, recomendamos a análise dos resultados obtidos com o modelo usando estas métricas como base.

Lembrando que diante da definição e alinhamento com o professor, pudemos definir:

- Verdadeiro Positivo: referem-se aos comentários negativos que são classificados como negativos;
- Falso Positivo: referem-se aos comentários positivos que são classificados como negativos;
- Falso Negativo: referem-se aos comentários negativos que são classificados como positivos;
- Verdadeiro Negativo: referem-se aos comentários positivos que são classificados como positivos.

É importante ressaltar que a partir de análises feitas, foi possível identificar as proporções de "falso negativo" (Recall) como as mais importantes, diante das predições que o modelo deve fazer, e com base na estratégia de negócio do parceiro.

7. Conclusões e Recomendações

Escreva, de forma resumida, sobre os principais resultados do seu projeto e faça recomendações formais ao seu parceiro de negócios em relação ao uso desse modelo. Você

pode aproveitar este espaço para comentar sobre possíveis materiais extras, como um manual de usuário mais detalhado na seção “Anexos”.

8. Referências

Nesta seção você deve incluir as principais referências de seu projeto, para que seu parceiro possa consultar caso ele se interessar em aprofundar.

Utilize a norma ABNT NBR 6023 para regras específicas de referências. Um exemplo de referência de livro:

SOBRENOME, Nome. **Título do livro**: subtítulo do livro. Edição. Cidade de publicação: Nome da editora, Ano de publicação.

Neuralmind. Bert-base-portuguese-cased. Disponível em: https://huggingface.co/neuralmind/bert-base-portuguese-cased. Acesso em: 13 jun. 2023.

Anexos

Utilize esta seção para anexar materiais como manuais de usuário, documentos complementares que ficaram grandes e não couberam no corpo do texto etc.

Sugestão:

Documentos que são alterados por cada sprint, como a Matriz de Riscos, devem ser movidas para a seção de anexo.

No corpo do documento deve permanecer o documento atual.

Separar os documentos por sprints.

Sugestão de divisão da seção Anexo:

ANEXO I – Sprint 1

Mova para essa seção os documentos produzidos na Sprint 1 que sofreram alterações na Sprint 2.

ANEXO II – Sprint 2

Mova para essa seção os documentos produzidos na Sprint 2 que sofreram alterações na Sprint 3.

ANEXO III – Sprint 3

Mova para essa seção os documentos produzidos na Sprint 3 que sofreram alterações na Sprint 4.

ANEXO IV – Sprint 4

Mova para essa seção os documentos produzidos na Sprint 4 que sofreram alterações na Sprint 5.