Equipe Sentimetry

Projeto 1 - SentimentAPI

Data Science:

Ana Paula Silva

Anne Pimentel

Eduardo Marchi

Rodrigo Pereira

Back End:

Andreia Semedo

Carlos Eduardo Souza Araujo

Patricia Starck Bernadi

Matheus Espírito Santo

Pedro Wandrey

Roberto Filho

20 de Janeiro de 2026

Hackathon ONE

RESUMO

O presente projeto tem como objetivo o desenvolvimento de uma solução automatizada para classificação de sentimentos em textos, aplicada ao contexto de atendimento ao cliente, marketing e operações. Empresas que coletam grandes volumes de avaliações, comentários em redes sociais e pesquisas de satisfação enfrentam dificuldades para analisar manualmente essas informações e extrair *insights* relevantes em tempo hábil. Nesse cenário, técnicas de Ciência de Dados e Processamento de Linguagem Natural (PLN) tornam-se fundamentais para apoiar a tomada de decisão.

A proposta consiste na criação de uma API simples, capaz de receber textos escritos por clientes e aplicar um modelo de classificação de sentimentos, retornando o resultado no formato JSON. A classificação será realizada de forma binária (Positivo ou Negativo), permitindo sua integração com diferentes aplicações e sistemas corporativos. O modelo foi treinado a partir de dados rotulados extraídos do *dataset* B2W Reviews*,* contendo avaliações reais de e-commerce, passando por etapas de limpeza, pré-processamento, vetorização com TF-IDF e treinamento do modelo.

A solução atende à necessidade de identificar rapidamente reclamações, priorizar atendimentos críticos e acompanhar a satisfação dos clientes ao longo do tempo. Além disso, a análise de sentimentos apresenta valor estratégico para o monitoramento de campanhas de marketing e avaliação da imagem da marca. Mesmo utilizando um modelo inicial, o projeto demonstra viabilidade prática e potencial de aplicação, especialmente para pequenas e médias empresas que buscam automatizar a análise de *feedbacks* sem a necessidade de grandes equipes especializadas.

Este projeto foi desenvolvido no contexto do Hackathon ONE, promovido pela Oracle para alunos do programa educacional, com o objetivo de aplicar conceitos de Data Science e Back End em um problema real de negócio.

Palavras-chave: Análise de dados; Machine Learning; Processamento de Texto; Classificação de Sentimentos; PNL.

1. Introdução

Com o crescimento das plataformas digitais, empresas passaram a receber um volume cada vez maior de comentários, avaliações e opiniões de seus clientes por meio de redes sociais, aplicativos e pesquisas de satisfação. Embora essas informações sejam valiosas para a melhoria de produtos e serviços, a análise manual desse conteúdo torna-se inviável à medida que o volume de dados aumenta.

Nesse contexto, a análise de sentimentos surge como uma ferramenta relevante para identificar automaticamente o *feedback* dos clientes, permitindo classificar textos como positivos ou negativos.

O presente trabalho tem como objetivo desenvolver uma solução baseada em técnicas de Análise de Dados e Classificação de Sentimentos através do Processamento de Linguagem Natural (PLN) capaz de classificar automaticamente o sentimento de comentários textuais de clientes em Positivo ou Negativo, fornecendo os resultados por meio de uma API simples, acessível a outras aplicações.

A solução proposta busca atender empresas que desejam monitorar a satisfação de seus clientes, priorizar atendimentos críticos e acompanhar a imagem da marca ao longo do tempo. Essa abordagem auxilia áreas como atendimento ao cliente, marketing e operações na tomada de decisões mais rápidas e assertivas.

2. Fundamentação Teórica

2.1 Processamento de Linguagem Natural (PLN)

O PLN é uma subárea da Ciência da Computação e da Inteligência Artificial que tem como objetivo permitir que computadores entendam, interpretem e se comuniquem com a linguagem humana (IBM, s.d.). Segundo Jurafsky e Martin (2023), o PLN busca modelar a linguagem natural para que sistemas computacionais possam executar tarefas como classificação de textos, tradução automática, reconhecimento de fala e análise de sentimentos.

A linguagem humana apresenta características complexas, como ambiguidade, polissemia, variações culturais e contextuais, o que torna o seu processamento computacional um desafio significativo. Dessa forma, técnicas de PLN combinam conceitos de linguística, estatística e aprendizado de máquina para transformar textos não estruturados em representações que possam ser analisadas por algoritmos.

Segundo o IBM, a aplicação do PLN também desempenha um papel cada vez maior em soluções empresariais que ajudam a otimizar e automatizar operações de negócios, aumentar a produtividade dos funcionários e simplificar processos empresariais.

Os principais benefícios do PLN são:

- Automação de tarefas repetitivas

- Análise e insights de dados aprimorados

- Busca aprimorada

- Geração de conteúdo

Entre as principais etapas do PLN, destacam-se:

- Tokenização: divisão do texto em unidades menores, como palavras ou sentenças;

- Normalização: padronização do texto, incluindo conversão para letras minúsculas e remoção de caracteres especiais;

- Remoção de stopwords: eliminação de palavras com pouco valor semântico (como preposições e artigos);

- Vetorização: transformação do texto em representações numéricas, como Bag of Words, TF-IDF ou embeddings.

Essas etapas são fundamentais para preparar os dados textuais para tarefas de classificação e predição, como a análise de sentimentos.

2.2 Análise de Sentimentos

A Análise de Sentimentos é uma aplicação do PLN juntamente com o aprendizado de máquina (ML) que visa identificar e classificar opiniões, treinando um software de computador para analisar e interpretar textos de forma semelhante aos humanos (IBM, s.d.). De acordo com Liu (2012), a análise de sentimentos busca compreender o ponto de vista do autor em relação a um determinado tema, produto ou serviço.

No contexto empresarial, a análise de sentimentos é amplamente utilizada para interpretar avaliações de clientes, comentários em redes sociais, respostas a pesquisas de satisfação e interações com serviços de atendimento ao consumidor. A automatização desse processo permite que organizações analisem grandes volumes de dados textuais de forma rápida e escalável.

Os métodos de análise de sentimentos podem ser classificados em três abordagens principais:

- Baseada em léxico: utiliza dicionários de palavras previamente rotuladas com polaridade positiva ou negativa;

- Baseada em aprendizado de máquina: utiliza algoritmos supervisionados treinados com dados rotulados;

- Baseada em aprendizado profundo: emprega redes neurais e modelos avançados, como embeddings e transformers.

Neste projeto, a análise de sentimentos é aplicada a comentários de clientes, com foco na classificação automática do sentimento expresso no texto de forma binária, como Positivo ou Negativo, permitindo identificar padrões de satisfação e insatisfação de forma eficiente.

2.3 TF-IDF (Term Frequency-Inverse Document Frequency)

O TF-IDF é uma das técnicas de vetorização de texto mais utilizadas em Processamento de Linguagem Natural. Sua principal função é transformar textos não estruturados (como comentários, avaliações, etc.) em representações numéricas, que possam ser utilizadas por algoritmos de aprendizado de máquina.

O TF-IDF combina duas medidas:

- Term Frequency (TF) — Frequência do Termo:
Representa a frequência com que um termo aparece em um documento em particular. A ideia é que termos que aparecem mais vezes em um documento podem ser mais representativos do conteúdo desse documento do que termos raros.

- Inverse Document Frequency (IDF) — Frequência Inversa do Documento:
Reflete o quão comum ou raro um termo é em todo o conjunto de documentos (corpus). Termos muito comuns (como “e”, “de”, “o”) aparecem em muitos documentos e por isso recebem menor peso; termos raros recebem maior peso, pois têm maior poder discriminatório.

2.3.1 Formulação Matemática

A fórmula clássica do TF-IDF para um termo t no documento d dentro do corpus D é:

$$TF - IDF(t, d, D) = TF(t, d) \times log(\ |D| \ / \ (1 + df(t))\ )$$

TF(t, d) representa a frequência do termo t no documento d, definida por:

TF(t, d) = f(t, d)

sendo f(t, d) o número de vezes que o termo t aparece no documento d.

A Frequência Inversa do Documento (IDF) é calculada como:

IDF(t, D) = log (|D| / (1 + df(t)))

em que:

|D| é o número total de documentos no corpus;

df(t) é o número de documentos que contêm o termo t;

o valor 1 é adicionado ao denominador para evitar divisão por zero.

2.3.2 Interpretação

O TF-IDF pondera variáveis de duas maneiras:

Pré-ponderação local (TF) — captura a importância do termo no contexto do documento.

Pós-ponderação global (IDF) — reduz a influência de termos muito frequentes em todo o corpus.

Assim, um termo que aparece muitas vezes em um documento, mas raramente em outros, receberá um valor TF-IDF elevado, indicando que ele é importante para caracterizar aquele documento de forma discriminativa.

2.3.3 Importância no Projeto

Em projetos de classificação de sentimentos, como o presente, o TF-IDF é crucial porque:

Converte texto em vetor numérico: Algoritmos de classificação não conseguem trabalhar diretamente com texto; eles necessitam de valores numéricos.

Reduz dimensionalidade relativa: Ao ponderar termos pelo seu valor discriminatório, o TF-IDF ajuda a destacar palavras úteis para a tarefa de classificação de sentimentos.

É eficiente e interpretável: Comparado a representações mais complexas (como embeddings de redes neurais), TF-IDF é simples de interpretar e eficiente em tarefas clássicas de classificação de texto.

2.4 Regressão Logística

A regressão logística é uma técnica de análise de dados que usa matemática para encontrar as relações entre dois fatores de dados. Em seguida, essa relação é usada para prever o valor de um desses fatores com base no outro. A previsão geralmente tem um número finito de resultados, como sim ou não (AWS, s.d.).

Na regressão logística, o modelo prevê a probabilidade de ocorrência de um resultado específico. Por exemplo, no nosso projeto, podemos prever a probabilidade de que um texto ou comentário seja positivo ou não. A saída do modelo é um valor entre 0 e 1. Com base em um limite (frequentemente 0,5), classificamos o resultado como "Positivo" ou "Negativo". Em vez de traçar uma linha reta através dos dados como faríamos na regressão linear, a regressão logística ajusta uma curva em forma de S para mapear valores de entrada para uma probabilidade.

A probabilidade estimada indica o grau de confiança do modelo de que uma determinada instância pertence à classe positiva.

2.4.2 Formulação Matemática

A Regressão Logística modela a probabilidade condicional de uma instância pertencer à classe positiva por meio da função sigmoide, aplicada a uma combinação linear das variáveis explicativas (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Matematicamente, a regressão logística pode ser expressa como:

Combinação linear:

$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$

Função sigmoide:

$P(y = 1 \mid x) = 1 / (1 + e^{(-z)})$

Ou

$P(y = 1 \mid x) = 1 / (1 + e^{-(\beta 0 + \beta 1 x 1 + \beta 2 x 2 + \ldots + \beta n x n)})$

Onde:

β_0 representa o termo de intercepto;

$\beta_1, \beta_2, \ldots, \beta_n$ são os coeficientes do modelo;

$x_1, x_2, \ldots, x_n$ são as variáveis independentes;

$P(y = 1 \mid x)$ é a probabilidade da instância pertencer à classe positiva.

A decisão final de classificação é feita a partir de um limiar, geralmente 0,5:

Probabilidade ≥ 0,5 → classe positiva

Probabilidade < 0,5 → classe negativa

Uma das principais vantagens da regressão logística é a facilidade de interpretação dos seus coeficientes, onde cada um indica o impacto da variável correspondente sobre a probabilidade da classe positiva:

- Coeficientes positivos aumentam a probabilidade da classe positiva.

- Coeficientes negativos reduzem essa probabilidade.

O valor absoluto do coeficiente indica a força da influência da variável.

Essa característica é especialmente relevante em projetos que exigem transparência e explicabilidade, como análises de opinião e tomada de decisão empresarial.

2.4.3 Aplicação em Classificação de Textos

Em tarefas de classificação de sentimentos, textos são convertidos em vetores numéricos por meio de técnicas como TF-IDF, resultando em matrizes de alta dimensionalidade e grande esparsidade. Estudos comparativos indicam que a regressão logística apresenta desempenho competitivo, frequentemente superior a modelos generativos, em tarefas de classificação textual (NG; JORDAN, 2002).

A regressão logística se adapta bem a esse cenário porque:

- Lida eficientemente com dados esparsos;

- Possui bom desempenho computacional mesmo com grande número de atributos;

- Apresenta resultados competitivos em tarefas de classificação textual;

- Permite análise dos termos mais relevantes para cada classe.

2.4.4 Relevância no Projeto

No contexto deste projeto, a regressão logística foi escolhida como modelo de classificação por combinar simplicidade, eficiência e fácil interpretação, sendo adequada para um cenário de análise de sentimentos aplicada a texto, como comentário de clientes. O modelo possibilita identificar automaticamente se uma mensagem expressa um sentimento positivo ou negativo, fornecendo uma base confiável para a automação do atendimento ao cliente e a análise da satisfação dos usuários.

Além disso, sua integração com representações vetoriais baseadas em TF-IDF permite a construção de uma solução robusta, escalável e facilmente interpretável, alinhada às necessidades práticas de aplicações corporativas.

3. Metodologia

Este projeto caracteriza-se como uma pesquisa aplicada com abordagem em *Data Science* e Aprendizado de Máquina supervisionado. O objetivo é desenvolver uma solução prática para classificação automática de sentimentos

em textos de clientes, visando apoiar processos de atendimento, marketing e tomada de decisão empresarial.

Ferramentas e tecnologias utilizadas ao longo do projeto:

Linguagem Python

Bibliotecas: Pandas, NumPy, Matplotlib, Scikit-learn, Sklearn, re, nltk

Ambiente: Google Colab / Jupyter Notebook

3.1 Carregamento dos Dados

O conjunto de dados utilizado é composto por avaliações e comentários de clientes extraídos de um *dataset* da B2W Reviews. Os dados incluem, principalmente, o texto do comentário e a nota atribuída pelo cliente, variando em uma escala numérica. O arquivo é importado no formato .csv com as dimensões de (132373, 14).

3.2 Pré-processamento dos Dados

Esta etapa teve como objetivo preparar os dados textuais para a modelagem. O pré-processamento de dados foi aplicado da seguinte forma:

Definição das colunas a serem trabalhadas no projeto:

foram selecionadas do *DataFrame* original.apenas as colunas "review_text" e "overall_rating";

essas colunas foram renomeadas para “texto” e “nota”, respectivamente, para facilitar o manuseio e a clareza.

Remoção de valores ausentes (*NaN*) em qualquer uma das colunas usando *df.dropna(inplace=True)*.

3.3 *Criação do Rótulo de Sentimento*

Criação da variável categórica sentimento, definida da seguinte forma:

- Notas maiores ou iguais a 4: "Positivo".

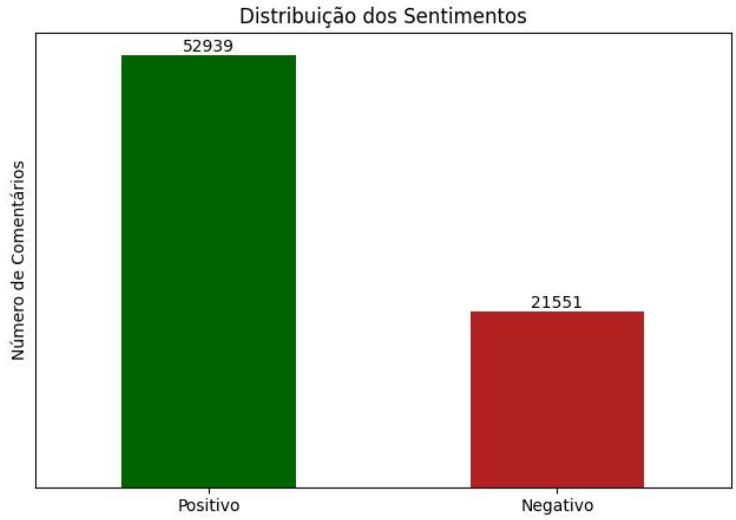
- Notas menores ou iguais a 2: "Negativo".

- Remoção de registros com nota neutra (nota = 3), visando reduzir ambiguidade na classificação de sentimentos.

3.4 Análise Exploratória dos Dados

Como pode-se observar no gráfico abaixo, após a rotulação de sentimentos “Positivo” e “Negativo” é possível verificar um desbalanceamento entre as classes no *dataset,* porém ainda aceitável para a aplicação da Regressão Logística.

Como sugestão, o *dataset* poderia ser balanceado no futuro (SMOTE, class_weight).



3.5 Pré-processamento de Texto

Esta etapa tem como objetivo limpar e normalizar o texto das avaliações para remover ruídos e preparar os dados textuais para a vetorização, otimizando o desempenho do modelo de *Machine Learning*.

Para isso, foi definida a função *limpar_texto* que aplica uma série de transformações como descrito a seguir:

Converte todo o texto para letras minúsculas *(texto.lower())*.

Remove valores monetários *(e.g., R$ 1994.20)* usando uma expressão regular *(re.sub(r'r\$\s*\d+[\.,]?\d*', '', texto))*.

Remove acentos e caracteres especiais, normalizando o texto *(remover_pontuacao).*

Remove caracteres que não são letras ou espaços *(re.sub(r'[^a-z\s]', '', texto)).*

Remove palavras sem valores semânticos (*stopwords*) da língua portuguesa *(stopwords_pt)*, que são carregadas da biblioteca *nltk* .

Uma nova coluna, “texto_processado”, foi criada no *DataFrame* df contendo os textos limpos e normalizados.

3.6 Vetorização com TF-IDF

Para a conversão dos textos em formato numérico, foi utilizada a técnica TF-IDF (*Term Frequency-Inverse Document Frequency*). Essa abordagem permite representar documentos textuais como vetores, atribuindo maior peso a termos relevantes e menos frequentes no corpus.

A escolha do TF-IDF justifica-se por sua eficiência em lidar com dados textuais de alta dimensionalidade e esparsidade, além de apresentar bom desempenho em tarefas clássicas de classificação de textos, como análise de sentimentos.

Foram considerados unigramas e bigramas *(ngram_range=(1,2))*, visando capturar não apenas palavras individuais, mas também expressões relevantes para análise de sentimento (ex.: “nao gostei”, “achei bom”).

O vocabulário foi limitado às 5.000 features mais relevantes, e termos muito raros foram descartados *(min_df=2)* para reduzir ruído e dimensionalidade.

3.7 Divisão em Treino e Teste

A divisão em treino e teste é uma prática padrão em Machine Learning para evitar o overfitting, onde o modelo memoriza os dados de treino e não generaliza bem para novos dados.

A reprodutibilidade é crucial em projetos de Data Science para que outros possam verificar e comparar os resultados do modelo

Para o treinamento do modelo foi feita a divisão do *dataset* em treino (80%) e teste (20%)) usando a função *train_test_split* do *scikit-learn* com uma semente fixa *(random_state=42)* garantindo a reprodutibilidade.

As variáveis criadas X_train e y_train contêm os dados para treinar o modelo, enquanto X_test e y_test são usados para avaliar o modelo após o treinamento.

Código utilizado:

3.8 Treinamento do Modelo

O modelo de classificação adotado foi a Regressão Logística, um algoritmo amplamente utilizado em problemas de classificação binária. A regressão logística modela a probabilidade de uma instância pertencer a uma determinada

classe por meio da função sigmoide, sendo adequada para dados vetorizados por TF-IDF.

A escolha desse algoritmo baseia-se em sua simplicidade, eficiência computacional e interpretabilidade, além de apresentar desempenho competitivo em tarefas de classificação textual com grandes volumes de atributos.

3.9 Métricas de Desempenho

Foram utilizadas métricas de desempenho adequadas para classificação, como acurácia e análise da capacidade de generalização do modelo em dados não vistos. Essas métricas permitiram verificar a efetividade do modelo na tarefa de classificação de sentimentos e sua viabilidade para uso prático.

A performance do modelo é mensurada através da Acurácia (proporção global de acertos), Precisão (fidelidade das predições positivas), *Recall* (capacidade de identificar instâncias positivas reais) e o F1-score (média harmônica entre Precisão e R*ecall*, ideal para conjuntos de dados desbalanceados). Os resultados do teste podem sem visualizados abaixo:

Acurácia do modelo: 0.9456

Relatório de Classificação:

	precision	recall	f1-score	support
Negativo	0.91	0.90	0.90	4244
Positivo	0.96	0.96	0.96	10654
accuracy			0.95	14898
macro avg	0.93	0.93	0.93	14898
weighted avg	0.95	0.95	0.95	14898

3.9 Resultados e Discussão

A acurácia do modelo foi calculada, resultando em aproximadamente 0.9484 (94.84%).

Um relatório de classificação (classification_report) foi gerado, fornecendo métricas detalhadas para cada classe “Positivo” e “Negativo”:

Precisão (*Precision*): A proporção de identificações positivas corretas entre todas as predições positivas. Para 'Negativo', foi de 0.92, e para 'Positivo', 0.96.

Recall (Sensibilidade): A proporção de identificações positivas corretas entre todas aspositivas reais. Para 'Negativo', foi de 0.91, e para 'Positivo', 0.97.

F1-score: A média harmônica da precisão e do *recall*, oferecendo um equilíbrio entre as duas métricas. Para 'Negativo', foi de 0.91, e para 'Positivo', 0.96.

Support: O número de ocorrências reais de cada classe no conjunto de teste.

As métricas indicam um bom desempenho geral do modelo, com alta acurácia e F1-scores equilibrados para ambas as classes, mesmo com o desbalanceamento observado.

Os resultados obtidos demonstram que o modelo de Regressão Logística, aliado à vetorização TF-IDF, apresentou desempenho altamente satisfatório na tarefa de classificação de sentimentos. A acurácia de aproximadamente 94,84% indica elevada capacidade de generalização do modelo sobre dados não vistos.

A análise das métricas de precisão, *recall* e *F1-score* revelou que o modelo foi eficaz tanto na identificação de comentários positivos quanto negativos, com desempenho ligeiramente superior para a classe “Positivo”. Ainda assim, os resultados obtidos para a classe “Negativo” também se mostraram consistentes, aspecto fundamental em aplicações voltadas ao atendimento ao cliente, onde a correta identificação de reclamações é essencial.

Dessa forma, o modelo desenvolvido se mostra adequado para aplicações práticas de análise de sentimentos, oferecendo uma solução eficiente, interpretável e computacionalmente viável para a automatização da análise de feedbacks textuais.

3.10 Serialização e Desserialização do Modelo

Esta etapa tem o objetivo de manter o modelo treinado e o vetorizador TF-IDF em arquivos para que possam ser reutilizados posteriormente sem a necessidade de re-treinamento ou re-ajuste, e carregá-los para uso em novas previsões.

A serialização é fundamental em projetos de Machine Learning para implantar modelos em ambientes de produção, permitindo que a aplicação consuma o modelo sem a necessidade de reexecutar todo o pipeline de treinamento. A desserialização garante que o modelo e o vetorizador carregados sejam

idênticos aos que foram treinados e salvos, assegurando consistência nas previsões.

A serialização e desserialização foram realizadas através das seguintes etapas:

O modelo de Regressão Logística treinado foi serializado (salvo) no arquivo *modelo_sentimento.joblib.*

O *TfidfVectorizer* ajustado foi serializado (salvo) no arquivo *vetorizador_tfidf.joblib.*

Ambos os objetos foram posteriormente desserializados (carregados) a partir de seus respectivos arquivos, demonstrando a capacidade de reutilizá-los.

3.11 Predição para um Novo Texto

Esta etapa valida o pipeline completo do modelo, desde o pré-processamento até a predição, utilizando os objetos persistidos. A formatação JSON da saída é crucial para a integração com APIs, pois é um formato padrão e facilmente consumível por outros sistemas.

O objetivo é demonstrar a funcionalidade do modelo treinado e serializado na prática, realizando uma predição de sentimento para um texto de exemplo, como pode ser visualizado nas etapas a seguir:

Um novo texto de exemplo foi definido: "Este produto é excelente, adorei! Recomendo a todos que procuram qualidade."

Este texto foi submetido ao mesmo processo de limpeza *(limpar_texto)* utilizado durante o treinamento para garantir consistência.

O texto limpo foi então transformado em um vetor numérico *(vetor_texto)* usando o *TfidfVectorizer* desserializado *(vetorizador_carregado)*.

O modelo de Regressão Logística desserializado *(modelo_carregado)* utilizou este vetor para prever o sentimento *(predicao)* e as probabilidades associadas a cada classe *(probabilidades).*

A saída foi formatada como um objeto JSON, incluindo o sentimento predito e as probabilidades formatadas para 'Positivo' e 'Negativo'. Para o texto de exemplo, a predição foi 'Positivo' com uma probabilidade de 1.0000 para a classe positiva e 0.0000 para a classe negativa.

3.12 Integração com API (Demonstração com Flask)

O objetivo é demonstrar como o modelo de sentimento pode ser empacotado em um microserviço *RESTful* usando Flask, permitindo que outras aplicações (como um *backend Spring Boot*) consumam suas predições.

A criação de uma *API REST* é o método padrão para disponibilizar modelos de Machine Learning para consumo por outras aplicações e sistemas, desacoplando o modelo do *backend* principal e permitindo escalabilidade.

Etapas realizadas:

Microserviço Flask: Um aplicativo Flask foi configurado para atuar como um servidor web simples.

Carregamento de Recursos: O modelo de Regressão Logística (modelo_sentimento.joblib) e o TfidfVectorizer (vetorizador_tfidf.joblib) foram carregados na inicialização do aplicativo Flask, garantindo que estejam prontos para uso em tempo real.

Pré-processamento: A função limpar_texto (a mesma usada no treinamento) foi incluída para garantir que qualquer texto de entrada para a API seja processado de forma consistente antes da predição.

*Endpoint /sentimen*t : Um *endpoint HTTP POST* foi criado em /*sentiment* para receber requisições com o texto a ser analisado.

> Ele valida se a requisição é *JSON* e se o campo *text* está presente e não vazio. O texto de entrada é limpo e vetorizado.
>
> O modelo faz a predição do sentimento e calcula as probabilidades.
>
> A resposta é formatada em JSON, contendo a previsao (Positivo/Negativo) e a probabilidade associada à classe predita.

Testes Internos Simulados: Dentro do ambiente *Colab*, foi demonstrada a funcionalidade da *API* através de simulações de requisições *POST* para a função *predict_sentiment* diretamente, testando textos positivos, negativos e casos de erro (texto vazio ou ausência do campo *text*).

4. Resumo final do Projeto

Construção Completa do Modelo: O projeto demonstrou a construção de um modelo de Análise de Sentimento desde o carregamento de dados até a integração com uma API REST.

Pipeline de Processamento: Incluiu carregamento e pré-processamento de dados (seleção, limpeza), criação de rótulos de sentimento binários, análise

exploratória (EDA), pré-processamento textual (normalização, remoção de stopwords), e vetorização com TF-IDF.

Treinamento e Avaliação do Modelo: Um modelo de Regressão Logística foi treinado e avaliado, alcançando uma acurácia de aproximadamente 94.84% no conjunto de teste, com métricas detalhadas (precisão, recall, F1-score) para as classes 'Positivo' e 'Negativo' indicando bom desempenho.

Serialização para Produção: O modelo treinado e o vetorizador TF-IDF foram serializados (joblib) para permitir sua persistência e fácil recarregamento em ambientes de inferência.

Demonstração de Predição: A capacidade do modelo de prever o sentimento de novos textos foi demonstrada, com a saída formatada em JSON, essencial para o consumo via API.

Prototipagem de API com Flask: Um microserviço Flask foi desenvolvido para exemplificar a interface REST, carregando o modelo serializado, pré-processando o texto de entrada e retornando predições em JSON. Testes internos simularam requisições para validar a funcionalidade do endpoint.

Referências

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. Introduction to Information Retrieval. Cambridge: Cambridge University Press, 2008.

JURAFSKY, D.; MARTIN, J. H. Speech and Language Processing. 3. ed. s.l.: Pearson, 2021.

SEBASTIANI, F. Machine Learning in Automated Text Categorization. ACM Computing Surveys, v. 34, n. 1, p. 1–47, 2002.

NG, A. Y.; JORDAN, M. I. On Discriminative vs. Generative Classifiers. Advances in Neural Information Processing Systems, 2002.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. The Elements of Statistical Learning. New York: Springer, 2009.

IBM, “O que é regressão logística?”<https://www.ibm.com/br-pt/think/topics/logistic-regression>

IBM, “O que é um saco de palavras?”<https://www.ibm.com/br-pt/think/topics/bag-of-words>

IBM, “O que é o NLP (processamento de linguagem natural)?”<https://www.ibm.com/br-pt/think/topics/natural-language-processing>