Classificação de Tweets sobre Desastres: Comparação de Abordagens em PLN

Marco Antônio Santolin (198769)
Igor Zanette (198862)

15 de setembro de 2025

1 Introdução

O desafio escolhido para este trabalho foi a competição *Natural Language Processing with Disaster Tweets*, disponível na plataforma Kaggle. Trata-se de uma competição do tipo *Getting Started*, voltada para iniciantes em Processamento de Linguagem Natural (PLN), cujo objetivo é desenvolver modelos de aprendizado de máquina capazes de identificar se uma determinada mensagem publicada no Twitter está relacionada a um desastre real ou não.

A motivação por trás desse problema é clara: em situações de emergência, as redes sociais, em especial o Twitter, tornaram-se canais fundamentais de comunicação em tempo real. Órgãos de defesa civil, organizações de ajuda humanitária e veículos de comunicação podem se beneficiar de sistemas automáticos que consigam filtrar rapidamente mensagens relevantes sobre desastres. No entanto, essa tarefa apresenta desafios, já que muitas vezes o mesmo vocabulário pode ser usado de forma figurativa ou metafórica, sem qualquer relação com eventos reais.

O conjunto de dados fornecido contém aproximadamente 10.000 tweets, classificados manualmente em duas categorias:

- **1** – Tweet relacionado a um desastre real;
- **0** – Tweet sem relação com desastre.

Além do texto das mensagens, o dataset inclui metadados como o identificador do tweet, palavra-chave associada (quando disponível) e localização informada. Esses atributos possibilitam a exploração de diferentes estratégias de pré-processamento e modelagem. A partir desse material, buscamos desenvolver e comparar diferentes abordagens, avaliando seu desempenho na tarefa de classificação.

1.1 Descrição do Dataset

O conjunto de dados disponibilizado pela competição contém aproximadamente 10.000 tweets rotulados manualmente. Cada instância representa uma mensagem publicada no Twitter, acompanhada de alguns metadados. A Tabela 1 mostra um recorte ilustrativo das primeiras linhas do dataset.

Tabela 1: Exemplo de instâncias do dataset de treino.

id	keyword	location	text	target
1	NaN	NaN	Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all	1
4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
5	NaN	NaN	All residents asked to 'shelter in place' are being notified by officers. No other evacuation or shelter in place orders are expected	1
6	NaN	NaN	13,000 people receive #wildfires evacuation orders in California	1
7	NaN	NaN	Just got sent this photo from Ruby #Alaska as smoke from #wildfires pours into a school	1

O dataset possui as seguintes colunas:

- **id**: Identificador único do tweet;
- **keyword**: Palavra-chave associada ao tweet (quando disponível);
- **location**: Localização informada pelo usuário (quando disponível);
- **text**: Texto original do tweet;
- **target**: Rótulo da instância, sendo 1 para tweets relacionados a desastres reais e 0 caso contrário;
- **text_clean**: Versão pré-processada do texto, com remoção de pontuações e normalização. (disponível no codigo)

1.2 Stack do Projeto

1.2.1 Versão do Python

Para o desenvolvimento do projeto foi utilizada a linguagem de programação **Python**, na versão **3.12.11**, devido à sua ampla adoção em projetos de Ciência de Dados e ao suporte consolidado para bibliotecas de *Machine Learning* e *NLP*.

1.2.2 Gerenciador de pacotes

O gerenciamento de dependências foi realizado com o **uv**, um gerenciador de pacotes moderno e de alto desempenho que substitui ferramentas tradicionais como `pip` e `venv`. O `uv` permite instalar pacotes de forma rápida, criar ambientes virtuais isolados e manter a reprodutibilidade do projeto, além de possuir integração simplificada com o formato `pyproject.toml`, utilizado para organizar as dependências. Mais detalhes sobre a configuração podem ser consultados no arquivo `README.md` do projeto.

1.2.3 Estrutura do projeto

A estrutura do repositório foi organizada da seguinte forma:

- **datasets/**: contém as bases de dados utilizadas no desafio;
- **resultados/**: armazena os arquivos de predição gerados, prontos para submissão no Kaggle;
- **src/**: implementa funções auxiliares em Python que são utilizadas no pré-processamento e no treinamento dos modelos;
- **tf_idf_logistic_regression.ipynb**: notebook responsável pela implementação da abordagem baseada em TF-IDF e Regressão Logística;
- **tf_idf_xgboost.ipynb**: notebook responsável pela implementação da abordagem baseada em TF-IDF e XGBoost;
- **distil_bert_model.ipynb**: notebook responsável pela implementação da abordagem baseada no modelo pré-treinado DistilBERT.

Essa organização permite que o projeto seja facilmente reprodutível, além de separar de forma clara os dados, códigos utilitários, experimentos e resultados obtidos.

2 Abordagens

2.1 Logistic Regression — TF-IDF

Para a primeira abordagem, optamos pela combinação da técnica **TF-IDF** (*Term Frequency–Inverse Document Frequency*) com o classificador **Regressão Logística**. Essa escolha foi motivada pelo fato de a regressão logística ser um modelo linear eficiente, interpretável e bastante utilizado em tarefas de classificação de texto.

O pipeline da abordagem foi implementado em Python, utilizando as seguintes bibliotecas principais:

Listing 1: Dependências utilizadas para a implementação

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold,
    cross_val_predict
from sklearn.metrics import f1_score, accuracy_score,
    recall_score, confusion_matrix

from src.utils import (
    import_datasets, limpar_textos,
    plotar_matriz_confusao, plot_roc, plot_precision_recall
)
```

O procedimento adotado consistiu nos seguintes passos:

1. Pré-processamento dos textos, incluindo normalização e limpeza;

2. Vetorização dos tweets com `TfidfVectorizer`, explorando unigramas e bigramas;

3. Treinamento do modelo de Regressão Logística com validação cruzada estratificada (`StratifiedKFold`);

4. Avaliação do desempenho com métricas de classificação e geração de gráficos comparativos.

2.1.1 Resultados

Os resultados médios obtidos pela validação cruzada indicam que a abordagem TF-IDF + Regressão Logística serve como um forte *baseline* para o desafio, alcançando valores competitivos para as principais métricas:

- **Acurácia**: 79,64%;
- **F1-score (macro)**: 79,04%;
- **Recall**: 78,83%.

Além disso, foram gerados gráficos de análise, como a matriz de confusão (Figura 1), a curva ROC (Figura 2) e a curva de Precisão-Recall (Figura 3), que permitem visualizar melhor o desempenho do classificador.

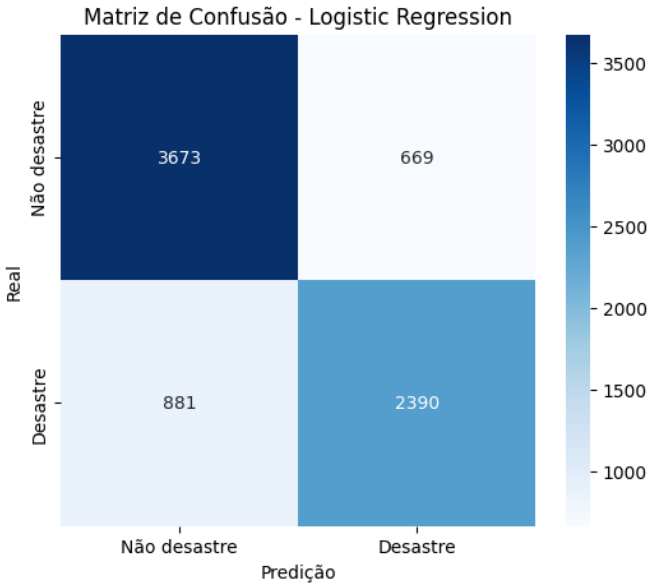

Figura 1: Matriz de confusão da abordagem TF-IDF + Regressão Logística.

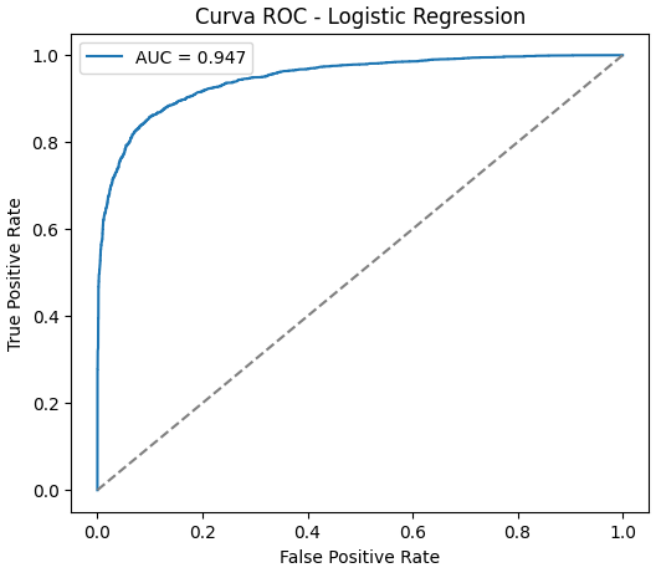

Figura 2: Curva ROC da abordagem TF-IDF + Regressão Logística.

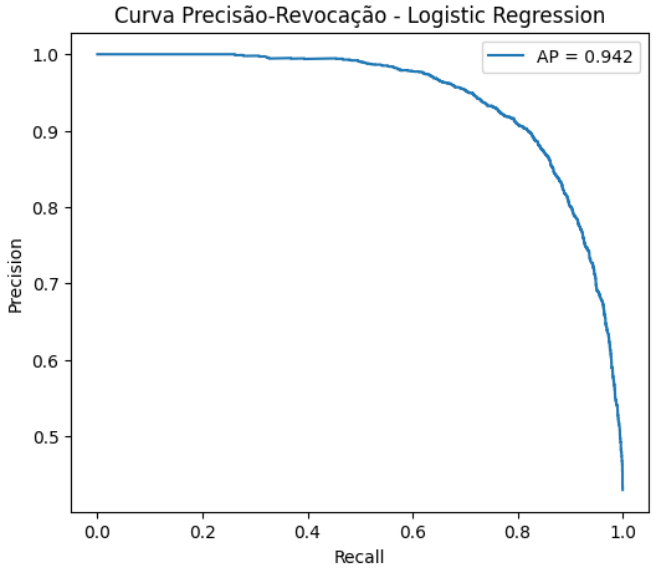

Figura 3: Curva Precisão-Recall da abordagem TF-IDF + Regressão Logística.

2.1.2 Conclusão dos resultados

A combinação de TF-IDF com Regressão Logística apresentou um desempenho bastante satisfatório no problema de classificação de tweets sobre desastres. O modelo alcançou **79,64% de acurácia**, **79,04% de F1-score (macro)** e **78,83% de recall**, mostrando-se consistente tanto na identificação de tweets relacionados a desastres quanto naqueles sem relação.

A matriz de confusão evidencia um bom equilíbrio entre verdadeiros positivos e verdadeiros negativos, embora ainda haja uma quantidade considerável de falsos negativos, ou seja, casos em que tweets de desastre não foram identificados corretamente.

A análise das curvas de avaliação reforça a qualidade do modelo: a curva ROC apresentou uma área sob a curva de **0,947**, enquanto a curva de Precisão-Recall obteve uma

área de **0,942**, indicando que o classificador consegue manter alta precisão mesmo em diferentes limiares de decisão.

De modo geral, esta abordagem se mostrou um *baseline* robusto para o desafio, servindo de referência para a comparação com métodos mais sofisticados, como o DistilBERT e o XGBoost.

2.1.3 Submissão Kaggle

A abordagem TF-IDF + Regressão Logística gerou uma submissão no Kaggle com desempenho consistente com os resultados da validação cruzada, alcançando **F1-score (macro) de 0,79068**.

Figura 4: Resultado da submissão no Kaggle para a abordagem TF-IDF + Regressão Logística.

2.2 DistilBERT Pré-Treinado

Nesta abordagem, utilizamos o modelo **DistilBERT**, uma versão reduzida e otimizada do BERT, pré-treinado na língua inglesa (`distil_bert_base_en_uncased`). O pré-processamento foi realizado por meio do `DistilBertPreprocessor`, que realiza tokenização, truncamento e padding das sequências, padronizando-as para comprimento máximo de 160 tokens.

O classificador foi implementado com a classe `DistilBertClassifier`, configurado para a tarefa de classificação binária. O trecho de código abaixo ilustra a inicialização do modelo e seu pré-processador:

Listing 2: Configuração do DistilBERT pré-treinado com KerasNLP

```
preset= "distil_bert_base_en_uncased"

preprocessor = keras_nlp.models.DistilBertPreprocessor.
   from_preset(
    preset,
    sequence_length=160,
    name="preprocessor_4_tweets"
)

classifier = keras_nlp.models.DistilBertClassifier.from_preset(
    preset,
    preprocessor = preprocessor,
    num_classes=2
)

classifier.summary()
```

2.2.1 Resultados

Os resultados obtidos no conjunto de validação indicaram um desempenho inferior ao baseline estabelecido pela Regressão Logística com TF-IDF. As métricas alcançadas foram:

- **Acurácia**: 57,38%;
- **F1-score (macro)**: 36,46%;
- **Recall (macro)**: 50,00%.

Embora o modelo tenha conseguido identificar corretamente alguns exemplos positivos, os resultados gerais demonstram dificuldade em capturar padrões discriminativos relevantes, refletindo em baixo valor de F1-score e acurácia apenas ligeiramente acima da aleatoriedade.

Para avaliar o desempenho do modelo DistilBERT, foram gerados gráficos que permitem uma análise visual das métricas de classificação.

As figuras a seguir apresentam: a **matriz de confusão** (Figura 5), a **curva ROC**(Figura 6) e a **curva Precisão-Recall**(Figura 7), que possibilitam compreender melhor o comportamento do classificador frente a diferentes limiares de decisão.

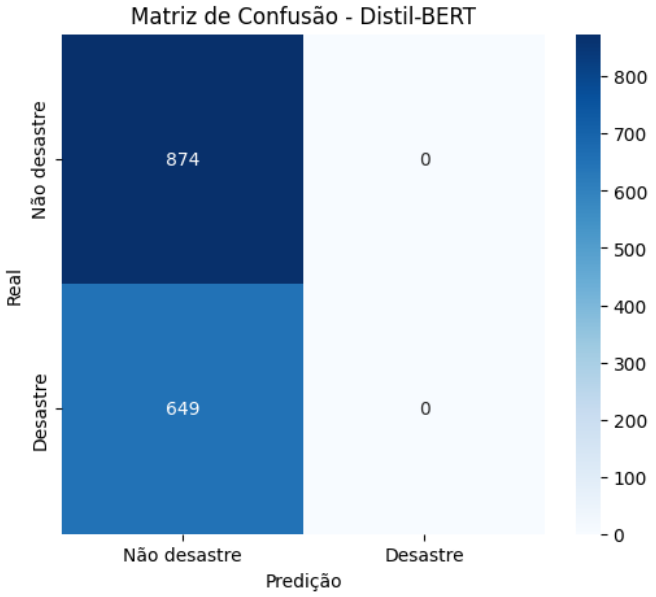

Figura 5: Matriz de confusão da abordagem DistilBERT.

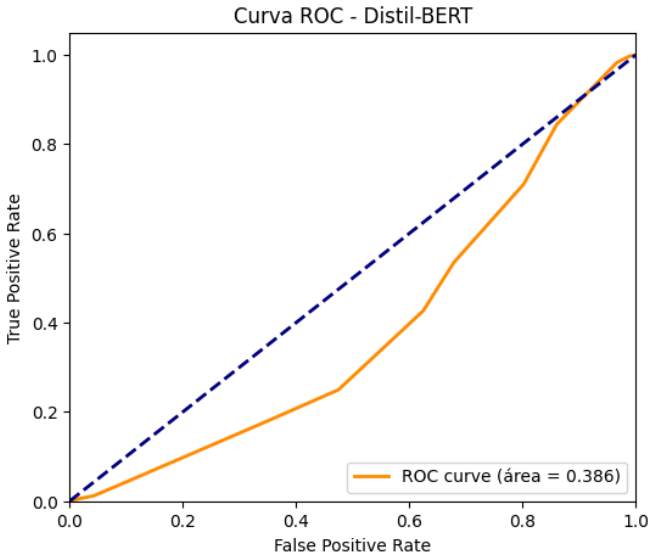

Figura 6: Curva ROC da abordagem DistilBERT.

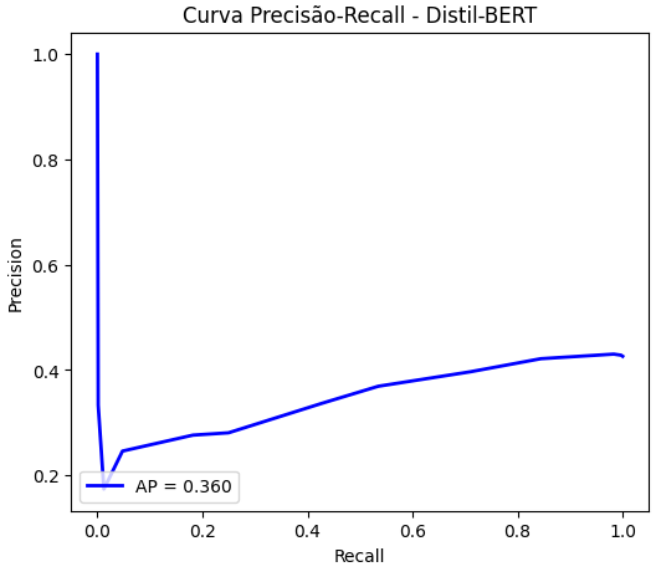

Figura 7: Curva Precisão-Recall da abordagem DistilBERT.

2.2.2 Conclusão dos resultados

Os experimentos com o DistilBERT evidenciam que, apesar do custo computacional elevado e da sofisticação da arquitetura baseada em transformadores, o modelo não superou o desempenho da abordagem mais simples com TF-IDF e Regressão Logística.

Isso pode ser explicado pela configuração restrita do treinamento (apenas duas épocas) e pela ausência de ajustes mais avançados, como *fine-tuning* com técnicas de regularização, uso de *learning rate schedules* ou aumento do número de épocas.

Dessa forma, este resultado destaca um ponto importante: modelos complexos como o DistilBERT não necessariamente garantem melhor desempenho quando não são adequadamente ajustados ou quando os recursos computacionais disponíveis limitam o processo

de otimização. Assim, no contexto deste trabalho, a abordagem linear permanece como uma solução mais eficaz e eficiente para o problema em questão.

2.2.3 Submissão Kaggle

A abordagem utilizando o DistilBERT resultou em uma submissão no Kaggle com desempenho inferior ao observado na regressão logística. O modelo alcançou um **F1-score (macro) de 0,57033**, valor consistente com os resultados obtidos na validação local, indicando que, apesar do potencial do modelo pré-treinado, sua configuração e tempo de treinamento impactaram negativamente no desempenho final.

Figura 8: Resultado da submissão no Kaggle para a abordagem DistilBERT.

2.3 XGBoost — TF-IDF

2.3.1 Resultados

Para esta abordagem, utilizou-se o *TfidfVectorizer* com até 50.000 características, considerando uni e bigramas, e removendo termos com frequência mínima de 3 ocorrências. Após a vetorização, os dados foram convertidos para o formato `DMatrix`, que é otimizado para o treinamento do XGBoost.

O modelo foi configurado com profundidade máxima de 6, taxa de aprendizado (*eta*) igual a 0.1, e técnicas de regularização como *subsample* e *colsample_bytree* para reduzir overfitting.

Listing 3: Hiperparâmetros utilizados no modelo XGBoost

```
# Hiperparametros do modelo
params = {
    "objective": "binary:logistic",
    "eval_metric": "logloss",
    "max_depth": 6,
    "eta": 0.1,
    "subsample": 0.8,
    "colsample_bytree": 0.8,
    "seed": 42,
}
```

Durante o treinamento, foi aplicado *early stopping* com paciência de 20 iterações, monitorando a métrica de validação (*logloss*).

Os resultados obtidos no conjunto de validação foram:

- F1 (macro): **0.7803**
- Acurácia: **0.7905**
- Recall (macro): **0.7756**

As figuras a seguir ilustram métricas gráficas que complementam a avaliação quantitativa do modelo XGBoost com TF-IDF. A matriz de confusão (Figura 9) revela um desempenho satisfatório do classificador, com 766 verdadeiros negativos e 438 verdadeiros positivos corretamente identificados. Observa-se, contudo, um número considerável de falsos negativos (216), indicando que o modelo apresenta certa dificuldade em identificar tweets relacionados a desastres, classificando-os erroneamente como não-desastre. Os falsos positivos (103) são em menor quantidade, sugerindo que o modelo é mais conservador ao classificar tweets como relacionados a desastres.

A curva ROC (Figura 10) demonstra um desempenho robusto, com AUC de 0.849, indicando boa capacidade discriminativa do modelo entre as duas classes. A curva apresenta formato típico de um classificador eficiente, distanciando-se significativamente da diagonal (classificador aleatório).

Por fim, a curva Precisão-Recall (Figura 11) confirma a qualidade do modelo com AP (Average Precision) de 0.846. A curva mostra que o modelo mantém precisão elevada mesmo com o aumento do recall, embora apresente declínio mais acentuado a partir de recall 0.6, refletindo o trade-off típico entre essas métricas e corroborando a observação sobre os falsos negativos identificados na matriz de confusão.

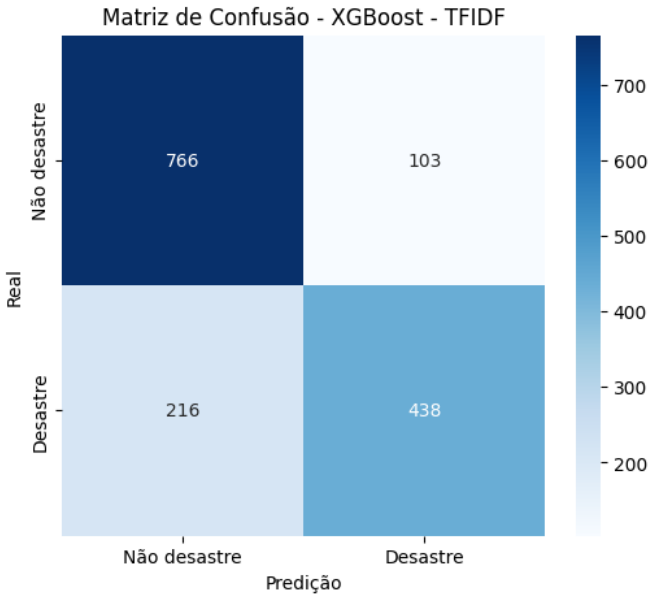

Figura 9: Matriz de confusão da abordagem TF-IDF XGBoost.

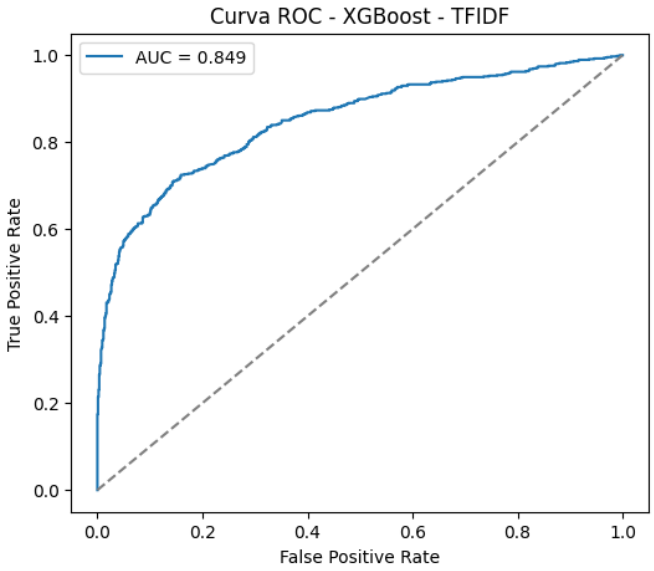

Figura 10: Curva ROC da abordagem TF-IDF XGBoost.

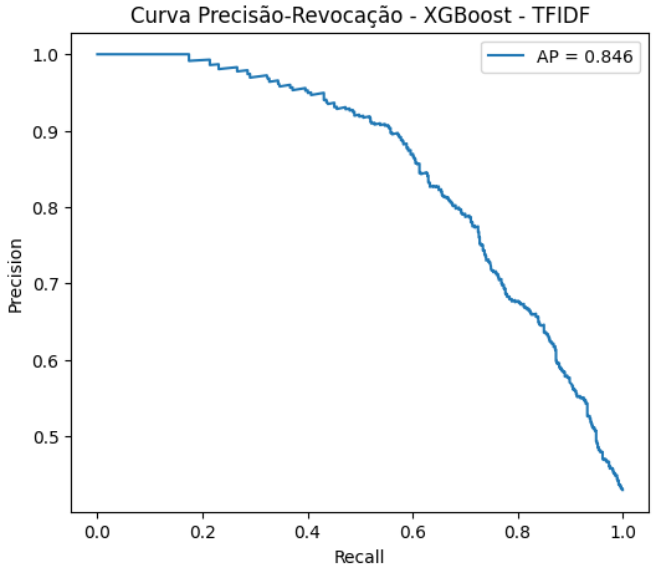

Figura 11: Curva Precisão-Recall da abordagem TF-IDF XGBoost.

2.3.2 Conclusão dos resultados

O modelo XGBoost obteve resultados sólidos e próximos aos alcançados pela regressão logística, mas com melhor equilíbrio entre precisão e recall. Isso evidencia que, embora mais complexo, o XGBoost conseguiu capturar relações não lineares nos dados de forma eficiente. Além disso, o uso de *early stopping* foi essencial para evitar overfitting e garantir boa generalização.

2.3.3 Submissão Kaggle

A submissão do modelo XGBoost com TF-IDF no Kaggle alcançou um F1-score de 0.77137, resultado que reflete adequadamente o desempenho observado durante a va-

lidação. Este resultado demonstra a capacidade de generalização do modelo para dados não vistos, mantendo consistência com as métricas obtidas no conjunto de validação. O F1-score de 0.771 posiciona a abordagem de forma competitiva na competição, evidenciando que a combinação de XGBoost com representação TF-IDF constitui uma estratégia eficaz para a classificação de tweets sobre desastres.

Figura 12: Resultado da submissão no Kaggle para a abordagem XGBoost + TF-IDF (F1-score = 0.77137).

O resultado obtido sugere que o modelo possui boa capacidade de discriminação entre tweets relacionados e não relacionados a desastres, com potencial para melhorias através de ajustes de hiperparâmetros, engenharia de features adicionais ou técnicas de ensemble.

3 Conclusão

Este trabalho apresentou uma análise comparativa de três abordagens distintas para a classificação automática de tweets relacionados a desastres: TF-IDF com Regressão Logística, DistilBERT pré-treinado e XGBoost com TF-IDF. Os experimentos realizados forneceram insights relevantes sobre a aplicabilidade e a eficácia de diferentes técnicas de Processamento de Linguagem Natural em cenários reais.

3.0.1 Síntese dos Resultados

A Regressão Logística com TF-IDF demonstrou ser a abordagem mais eficaz, alcançando um F1-score de 0.79068 no Kaggle e 79.64% de acurácia na validação local. Esse resultado evidencia que técnicas tradicionais de representação textual, quando bem aplicadas, podem superar métodos mais sofisticados em determinados contextos. A combinação de simplicidade, interpretabilidade e eficiência computacional consolidou esta abordagem como o melhor baseline do estudo.

O XGBoost com TF-IDF apresentou desempenho competitivo, com F1-score de 0.77137 no Kaggle, posicionando-se como a segunda melhor abordagem. Embora tenha ficado ligeiramente abaixo da regressão logística, o modelo demonstrou maior estabilidade e capacidade de capturar relações não lineares, sugerindo potencial de melhorias por meio de ajustes mais refinados de hiperparâmetros.

Por outro lado, o DistilBERT pré-treinado apresentou resultados inferiores, com F1-score de apenas 0.57033. Esse desempenho evidencia que modelos baseados em transformadores, apesar de sua sofisticação arquitetural, requerem configurações mais elaboradas, maior tempo de treinamento e possivelmente maior volume de dados para atingirem seu desempenho máximo.

3.0.2 Considerações Finais

Os resultados obtidos demonstram que maior complexidade algorítmica não se traduz, necessariamente, em melhor desempenho prático. A abordagem TF-IDF + Regressão

Logística, embora conceitualmente simples, mostrou-se a mais eficaz para a tarefa proposta, oferecendo o melhor equilíbrio entre performance, interpretabilidade e eficiência computacional.

Este estudo reforça a importância de estabelecer baselines sólidos antes de explorar arquiteturas mais complexas, além de evidenciar que o sucesso em tarefas de PLN depende tanto da qualidade do pré-processamento e da adequação da abordagem ao problema quanto da sofisticação do modelo escolhido.

Para aplicações práticas de classificação de tweets sobre desastres, recomenda-se adotar a abordagem TF-IDF + Regressão Logística como solução inicial, utilizando-a como baseline para futuras otimizações e comparações com métodos mais avançados.