AVALIAÇÃO E OTIMIZAÇÃO – RELATÓRIO 12/11/2025

ALLAN MESSIAS PINTO
(FATEC BAIXADA SANTISTA – RUBENS LARA)
allan.messias01@fatec.sp.gov.br

CARLOS EDUARDO FRANÇA AMADOR
(FATEC BAIXADA SANTISTA – RUBENS LARA)
carlos.amador@fatec.sp.gov.br

JEFFERSON LEONARDO DOS SANTOS SEPULVEDA
(FATEC BAIXADA SANTISTA – RUBENS LARA)
jefferson.sepulveda@fatec.sp.gov.br

ROBERT RICHARD DAS NEVES CORREIA DOS SANTOS
(FATEC BAIXADA SANTISTA – RUBENS LARA)
robert.santos01@fatec.sp.gov.br

VICTOR BARBOSA GONÇALVES
(FATEC BAIXADA SANTISTA – RUBENS LARA)
victor.goncalves4@fatec.sp.gov.br

VICTOR ROMA VIANNA FERREIRA
(FATEC BAIXADA SANTISTA – RUBENS LARA)
victor.ferreira38@fatec.sp.gov.br

PREPARAÇÃO, ROTULAGEM E TREINAMENTO DOS MODELOS

1. Preparação dos Dados

O código desenvolvido em Python, por meio do Google Colab, foi utilizado para ler os arquivos processados anteriormente, extraindo a coluna “texto_original” e aplicando o tokenizador nltk.sent_tokenize para dividir as críticas em sentenças individuais. Após a etapa de limpeza textual, organizamos as sentenças em um novo DataFrame, que foi salvo no arquivo sentencas_para_rotular.csv, destinado ao uso na etapa seguinte, o Assistente de Rotulagem.

Essa fase foi essencial para a preparação dos dados, pois nela foi construído o Ground Truth, que serviu como base de referência para o treinamento dos modelos supervisionados.

2. Assistente de Rotulagem

Nessa etapa, foi desenvolvido um assistente interativo de rotulagem, responsável por realizar o upload do arquivo sentencas_para_rotular.csv e exibir cada sentença individualmente, permitindo que atribuíssemos uma polaridade (positiva, negativa ou neutra) por meio de códigos numéricos.

Ao final do processo, todas as classificações foram reunidas em uma nova coluna chamada “polaridade”, e o resultado foi salvo no arquivo sentencas_rotuladas.csv, que posteriormente serviu de base para o treinamento dos modelos de aprendizado de máquina. Além disso, realizamos uma análise da distribuição das classes, verificando a proporção de sentenças positivas, negativas e neutras obtidas após o processo de rotulagem.

3. Vetorização das Sentenças

Após a etapa de rotulagem, realizamos o processo de vetorização dos textos. As features (X) foram obtidas por meio da conversão das sentenças em uma matriz numérica de pesos, utilizando o método TF-IDF (TfidfVectorizer), que também removeu as stopwords e considerou bigramas (como “não gostei”), possibilitando uma representação mais precisa do contexto semântico das frases.

Já os rótulos (y) foram convertidos de texto (“Positivo”, “Negativo”, “Neutro”) para valores numéricos com o uso do LabelEncoder, permitindo que os algoritmos de classificação interpretassem corretamente cada classe durante o treinamento.

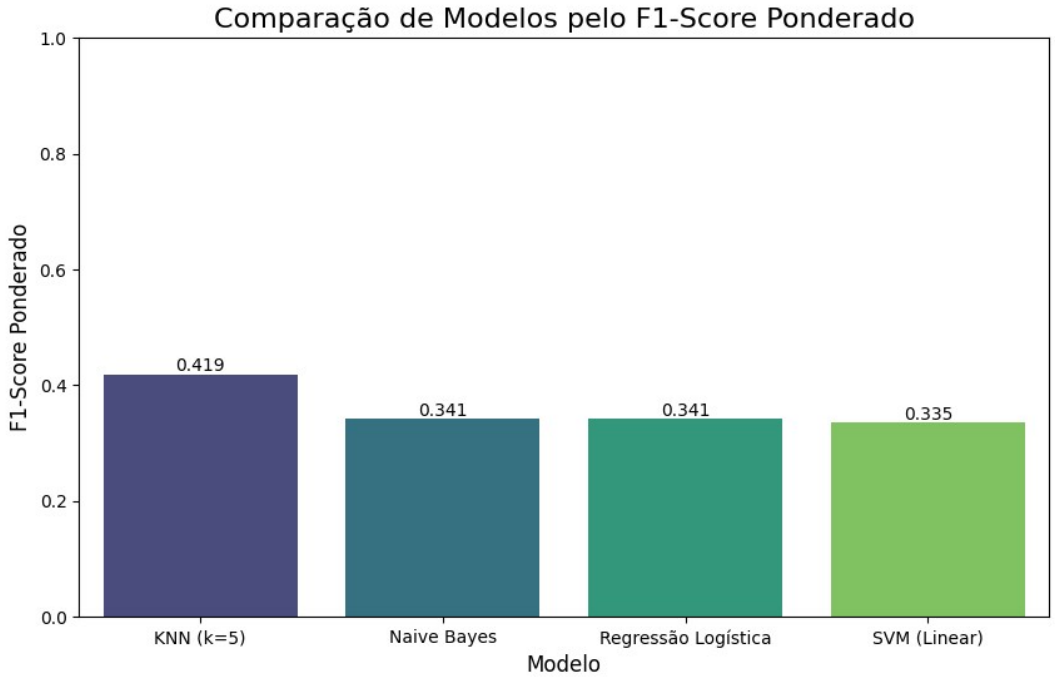
4. Treinamento e Avaliação dos Modelos (Baseline)

Nessa etapa, foram executados e comparados quatro modelos de classificação: Naive Bayes, SVM (Linear), KNN (k=5) e Regressão Logística. Para garantir uma avaliação mais robusta e confiável, especialmente diante do possível desbalanceamento do conjunto de dados, optamos pela utilização da Validação Cruzada Estratificada (StratifiedKFold) em vez de uma simples divisão entre treino e teste. Em seguida, foram calculadas as médias das principais métricas de desempenho (F1-Score, Precisão e Recall) para cada modelo. Por fim, geramos uma tabela comparativa com os resultados obtidos, classificando os modelos do melhor ao pior desempenho, a fim de identificar o modelo vencedor que seria utilizado na etapa de otimização.

```
--- Tabela Comparativa de Performance (Métricas Ponderadas) ---
| Modelo              |   Acurácia |   Precisão |   Recall |   F1-Score |
|:--------------------|-----------:|-----------:|---------:|-----------:|
| KNN (k=5)           |      0.453 |      0.420 |    0.453 |      0.419 |
| Naive Bayes         |      0.507 |      0.258 |    0.507 |      0.341 |
| Regressão Logística |      0.507 |      0.258 |    0.507 |      0.341 |
| SVM (Linear)        |      0.493 |      0.254 |    0.493 |      0.335 |
```

5. Visualização dos Resultados

A etapa de visualização faz uso da biblioteca Seaborn para plotar um gráfico de barras que apresenta o F1-Score ponderado de cada modelo. As barras são ordenadas do melhor para o pior desempenho, e cada uma possui anotações com os valores exatos das métricas, permitindo uma interpretação visual clara sobre qual modelo obteve a melhor performance de baseline.



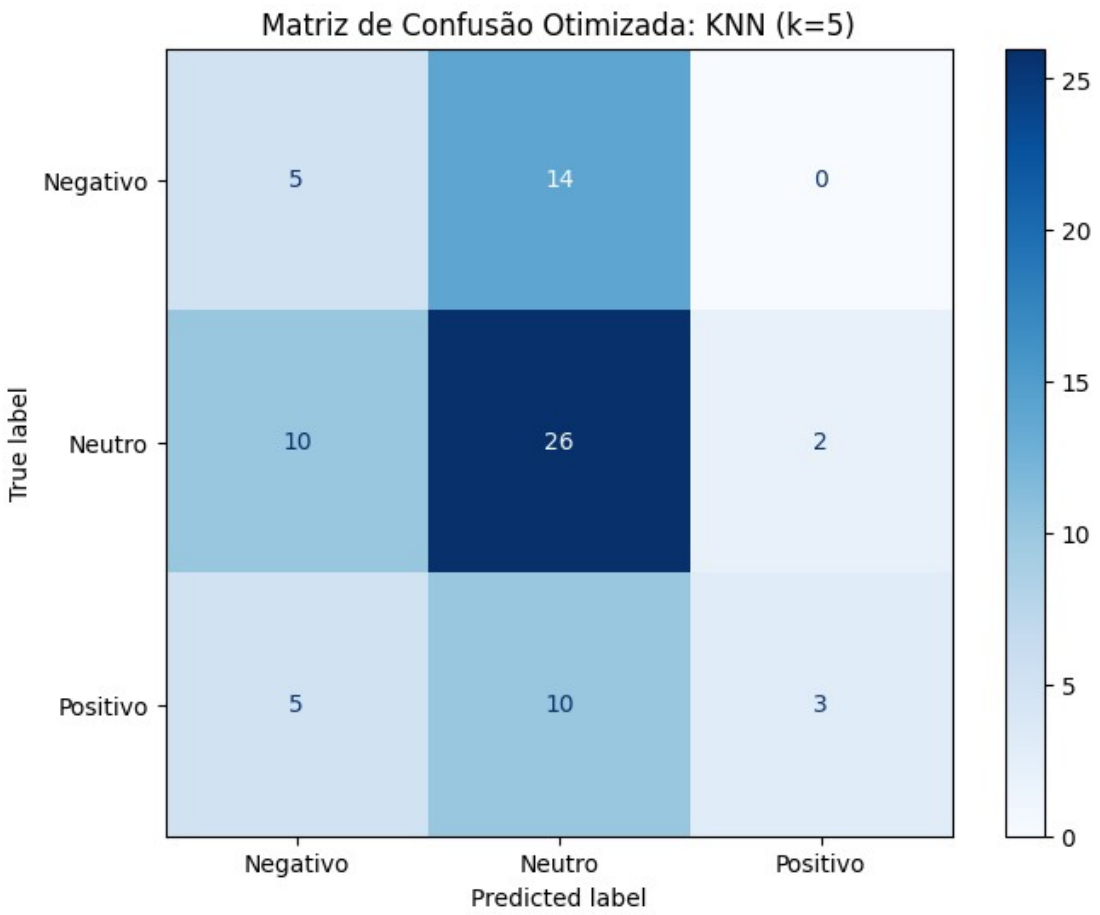
6. Pipeline Final de Machine Learning

Todo o fluxo das etapas anteriores foi consolidado em um pipeline completo de Machine Learning, estruturado para integrar todas as fases do processo de forma automatizada. Inicialmente, realizamos o carregamento e a vetorização dos dados, por meio do upload do arquivo sentencas_rotuladas.csv, convertendo as sentenças em vetores TF-IDF (X) e os rótulos em valores numéricos (y).

Em seguida, efetuamos a comparação dos modelos de baseline, aplicando a validação cruzada aos quatro classificadores e gerando uma tabela comparativa para identificar o modelo com melhor desempenho em F1-Score. Na sequência, realizamos a otimização com o método GridSearchCV, testando diferentes combinações de hiperparâmetros para encontrar a configuração mais eficiente.

Assim, geramos o relatório final de desempenho, apresentando o classification_report, com as métricas de Precisão e Recall para cada classe, e a Matriz de Confusão, que consolidou a avaliação definitiva do modelo otimizado.

```
--- Tabela Comparativa (Base) ---
|    | Modelo              |   F1-Score (Base) |
|---:|:--------------------|------------------:|
|  2 | KNN (k=5)           |             0.419 |
|  0 | Naive Bayes         |             0.341 |
|  3 | Regressão Logística |             0.341 |
|  1 | SVM (Linear)        |             0.335 |

--- Modelo Vencedor (Maior F1-Score): KNN (k=5) ---

Iniciando otimização (GridSearchCV) para o KNN (k=5)...
--- Otimização Concluída ---
Melhores parâmetros encontrados: {'n_neighbors': 5}
Melhor F1-Score (Ponderado) após otimização: 0.4190
Score original (base): 0.4190
Melhoria: 0.0000

--- Relatório de Classificação Detalhado (Modelo Otimizado) ---
              precision    recall  f1-score   support

    Negativo       0.25      0.26      0.26        19
      Neutro       0.52      0.68      0.59        38
    Positivo       0.60      0.17      0.26        18

    accuracy                           0.45        75
   macro avg       0.46      0.37      0.37        75
weighted avg       0.47      0.45      0.43        75
```



7. Conclusão

A matriz de confusão resultante mostra que o KNN (k=5), modelo com melhor desempenho na etapa de comparação dos baselines, obteve bom desempenho na classe “Neutro”, mas apresentou certa confusão entre as classes “Negativo” e “Neutro”, além de dificuldade em identificar corretamente as sentenças positivas. Esses resultados indicam uma tendência do KNN a concentrar previsões na classe neutra, possivelmente influenciada pelo desbalanceamento do conjunto de dados.

8. Disponibilização do Código-Fonte

Por fim, disponibilizamos todo o código desenvolvido no GitHub, a fim de garantir transparência, reprodutibilidade e fácil acesso ao material utilizado durante o projeto.

Link:
https://github.com/robertrichard86/Avalia-o-Otimza-o-PI4/blob/main/Avalia%C3%A7%C3%A3o_e_Otimiza%C3%A7%C3%A3o_PI.ipynb