UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

Luiz Hemrique Cruz dos Santos

RA: 191251003

Criação de Modelo Preditivo utilizando Python e Análise da Solução

Graduação em Ciência da Computação

FCT - UNESP

Presidente Prudente

2024

Introdução

No cenário atual, a disseminação de informações ocorre de maneira rápida e em larga escala, especialmente por meio da internet e redes sociais. No entanto, essa facilidade de distribuição trouxe também um desafio significativo: a propagação de notícias falsas, ou “fake news”. As fake news representam um problema sério, pois podem influenciar a opinião pública, causar pânico, manipular comportamentos e até afetar processos democráticos, como eleições. Portanto, a capacidade de identificar e distinguir notícias reais das falsas tornou-se uma necessidade extremamente importante para garantir a integridade da informação.

Contextualização do Problema

A propagação de fake news é um problema complexo. As fake news são frequentemente projetadas para parecer autênticas, usando fontes e formatos que imitam notícias reais. Elas podem ser motivadas por razões políticas, financeiras ou simplesmente por malícia. Identificar essas notícias exige mais do que uma simples análise superficial; é necessário empregar técnicas avançadas de processamento de linguagem natural (NLP) e aprendizado de máquina para detectar padrões que diferenciam notícias reais das falsas.

Neste projeto, analisamos um grande volume de notícias. A base de dados utilizada é composta por milhares de notícias, cada uma já marcada com uma etiqueta indicando se é real ou falsa. Esta classificação previa permitirá a construção e treinamento de modelos preditivos capazes de realizar essa distinção de maneira eficiente e precisa.

Base de Dados Utilizada

A base de dados utilizada neste projeto contém milhares de notícias coletadas de diversas fontes, cada uma rotulada como "real" ou "fake". Esta base de dados oferece um rico conjunto de textos que variam em estilo, conteúdo e origem, proporcionando um desafio interessante para o desenvolvimento de modelos de classificação.

As fontes variam, incluindo jornais online e redes sociais, proporcionando uma ampla gama de estilos de escrita e níveis de credibilidade.

Metodologia

Para abordar o problema de detecção de fake news, seguiremos um pipeline de análise de texto composto pelas seguintes etapas:

Pré-processamento de Dados

Nesta etapa, foi realizada a limpeza dos textos para remover ruídos, como pontuações desnecessárias, números e caracteres especiais. Além disso, foi feita a conversão de todos os textos para letras minúsculas, a fim de facilitar a visualização.

Extração de Características

Outra etapa no pré-processamento de dados foi a transformação dos textos em representações numéricas utilizando o método TF-IDF (Term Frequency — Inverse Document Frequency). Essa medida quantifica a importância ou relevância de representações de strings em um documento entre uma coleção de outros documentos.

Visto que a base de dados utilizada é totalmente formada por texto, foi necessário aplicar esta técnica, transformando as informações presentes em valores numéricos, pois ao vetorizar estes dados, é possível realizar várias tarefas, como identificação de documentos relevantes, classificações, agrupamentos, etc.

A TF (Term Frequency) refere-se à quantas vezes um termo aparece no documento. Já a IDF, ou seja, a frequência inversa dos documentos, refere-se à relevância e importância de um termo. Isso significa que o peso das palavras-chave com alta frequência é reduzido, enquanto que o peso dos termos que aparecem com menos regularidade é aumentado.

Além disso, foi realizada a técnica de clusterização utilizando o método Mini Batch K-Means, uma variação do método K-Means, que é principalmente destinada para o tratamento de conjuntos de dados muito grandes (caso da base de dados que foi utilizada neste processo). Este método seleciona pequenos conjuntos aleatórios dentro do conjunto principal, permitindo que o algoritmo convirja mais rapidamente, e também que use menos memória. Para facilitar a visualização, foi utilizada a técnica de redução de dimensionalidade PCA (Principal Component Analysis). O PCA reduziu as características para 2 componentes principais, facilitando a visualização dos clusters.

```python
# Inicializando o TfidfVectorizer com 1000 features
# pois o conjunto todo é muito grande e resulta em sobrecarga de memória
vectorizer = TfidfVectorizer(max_features=1000)

# Processamento em partes menores (Chunking) para evitar sobrecarregar a memória.
chunk_size = 1000
chunks = [df_clean['Text'][i:i + chunk_size]
          for i in range(0, df_clean.shape[0], chunk_size)]

# Cada parte é transformada usando o TfidfVectorizer.
tfidf_chunks = [vectorizer.fit_transform(chunk) for chunk in chunks]
x = vstack(tfidf_chunks)

# Utilizando o MiniBatchKMeans, que é uma variação de K-means
# projetada para trabalhar com grandes conjuntos de dados.
# Ele processa os dados em mini-batches, reduzindo o uso de memória.
num_clusters = 10
kmeans = MiniBatchKMeans(n_clusters= num_clusters,
                         random_state=0, batch_size=chunk_size).fit(x)

# # Adicionando os rótulos dos clusters ao DataFrame original
df_clean['Cluster'] = kmeans.labels_
```

Figura 1: código em python referente à clusterização do conjunto de dados

```python
# Visualizando os clusters com PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(x.toarray())

plt.scatter(principal_components[:, 0], principal_components[:, 1],
            c=kmeans.labels_, cmap='rainbow')
plt.title('Clusters de Notícias')
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
# plt.savefig('../images/clusterizacao_num10.png', dpi=300)
plt.show()
```

Figura 2: código em python referente redução de dimensionalidade utilizando o PCA

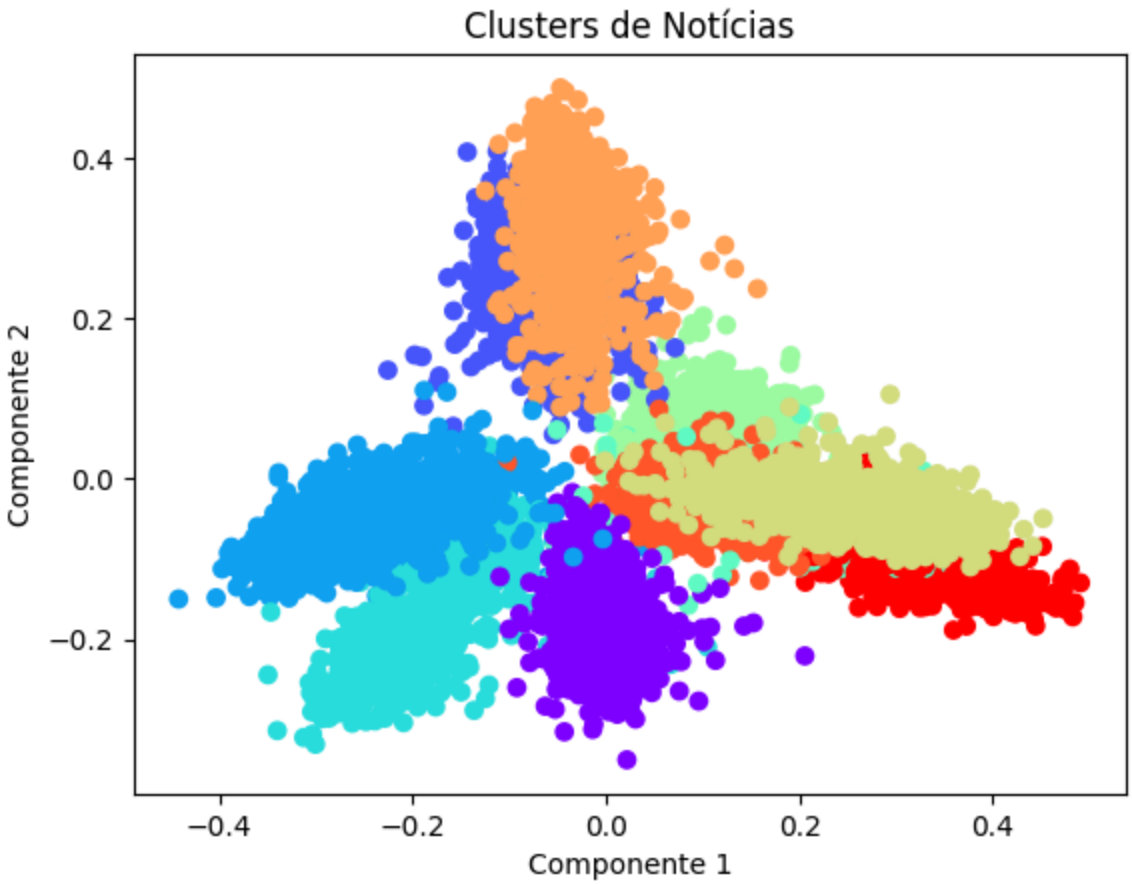


Figura 3: Clusterização da base de dados utilizando PCA para facilitar a visualização

Modelagem

Um passo extremamente necessário para aplicar qualquer modelo preditivo é a separação dos dados em um conjunto de treinamento e outro conjunto de testes. Isto é feito com a função 'train_test_split' da biblioteca Scikit-learn. Esta divisão é importante pois, ao aplicar um modelo preditivo num conjunto de dados, estaremos avaliando a capacidade de generalização do modelo para dados não vistos durante o treinamento. Isso quer dizer que se tentarmos avaliar o modelo apenas com os dados de treinamento, podemos obter uma estimativa não realista do desempenho. Além disso, o modelo pode se ajustar demais aos dados de treino, o que é chamado de overfitting.

```python
# Separar as features e labels
X = df_clean['Text'].copy()
y = df_clean['label'].copy()

# Dividir os dados em conjuntos de treinamento e teste
x_treino, x_teste, y_treino, y_teste = train_test_split(X, y, test_size=0.2, random_state=42)

# Converter o texto para vetores TF-IDF
vectorizer = TfidfVectorizer()
x_treino_tfidf = vectorizer.fit_transform(x_treino)
x_teste_tfidf = vectorizer.transform(x_teste)

# Verificar os rótulos presentes em y_teste
rotulos_unicos = y_teste.unique()
print(f"Rótulos únicos em y_teste: {rotulos_unicos}")
```

Figura 4: código em python referente à
divisão do conjunto de dados em treinamento e teste

Modelos aplicados

Os modelos aplicados no conjunto de dados foram o Decision Tree, o Naive Bayes, o LDA, o KNN e a Regressão Logística.

Decision Tree (Árvore de Decisão)

Uma árvore de decisão é um modelo de aprendizado supervisionado. O objetivo deste método é dividir os dados em subconjuntos menores com base em valores de atributo, criando uma estrutura de árvore. Nesta árvore, a raiz representa todo o conjunto de dados, os nós internos representam um atributo em um ponto de decisão, os ramos representam a decisão tomada após verificar um atributo e as folhas representam o resultado final.

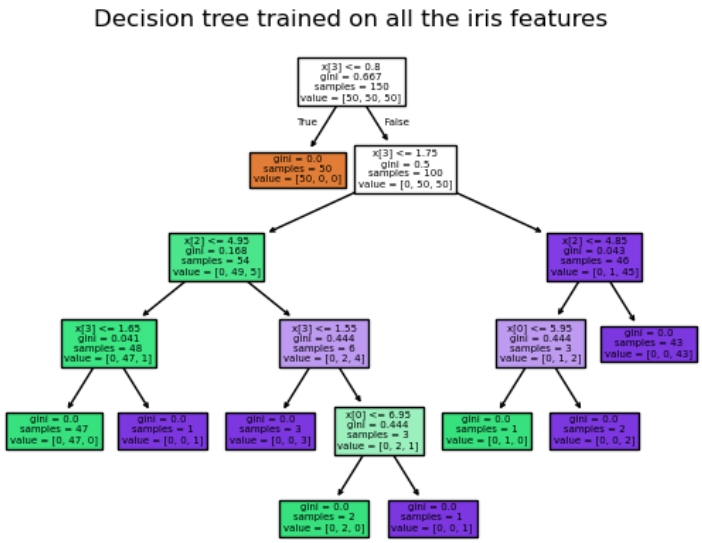


Figura 5: exemplo de uma árvore de decisão

Regressão Logística

A Regressão Logística é um modelo estatístico para a previsão de probabilidade de uma variável binária com base em uma ou mais variáveis independentes.

Naive Bayes

O algoritmo Naive Bayes é um algoritmo classificador probabilístico baseado no Teorema de Bayes, que assume que as features são independentes entre si. Ele funciona muito bem com grandes volumes de dados e é extremamente eficaz em problemas de classificação de texto, como por exemplo na detecção de spam.

Linear Discriminant Analysis (LDA)

O LDA é um método usado para encontrar uma combinação linear de características que melhor separa duas ou mais classes.

K-Nearest Neighbors (KNN)

O KNN é um algoritmo que classifica uma nova amostra com base na maioria dos votos de seus *k* vizinhos mais próximos. No nosso caso, o número k utilizado foi de 5 vizinhos.

Métricas de Avaliação

Para avaliar o desempenho dos modelos de classificação foram utilizadas as métricas de acurácia, precisão, recall e F1 score.

A Acurácia é a proporção de previsões corretas (tanto verdadeiras quanto falsas) em relação ao total de previsões feitas. Uma acurácia alta indica que o modelo é bom em prever corretamente.

A Precisão é a proporção de verdadeiros positivos em relação ao total de previsões positivas. Uma alta precisão significa que quando o modelo prevê uma notícia como falsa, é mais provável que ela realmente seja falsa.

O Recall (Revogação ou Sensibilidade) é a proporção de verdadeiros positivos em relação ao total de casos positivos reais. Um valor de recall alto indica que o modelo é melhor em identificar notícias falsas reais.

O F1 Score é a média harmônica entre precisão e recall, proporcionando um equilíbrio entre ambas. Um F1 score alto indica que o modelo tem um bom equilíbrio entre precisão e recall.

Resultados

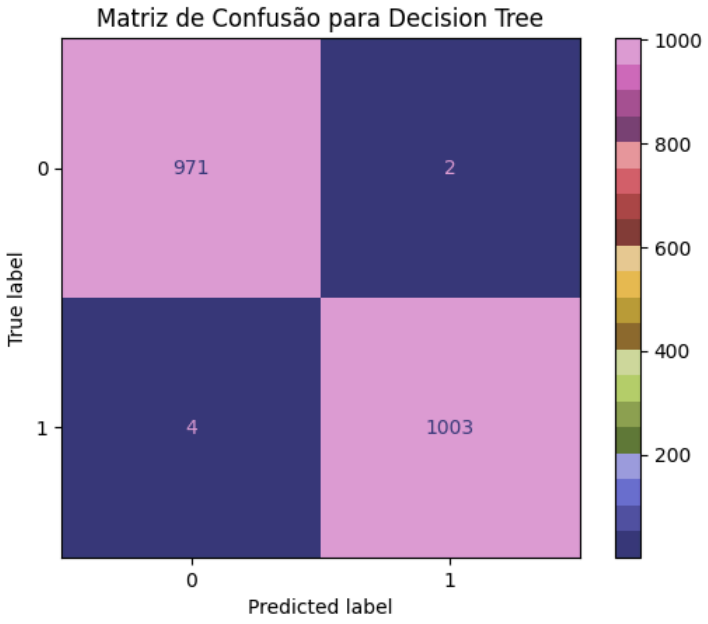
Após a aplicação dos modelos no conjunto de dados, os resultados são apresentados a seguir. A avaliação de desempenho foi realizada utilizando as métricas acurácia, precisão, recall e F1 Score. Além disso, as matrizes de confusão fornecem uma visão detalhada da performance de cada modelo.

Árvore de decisão (Decision Tree)

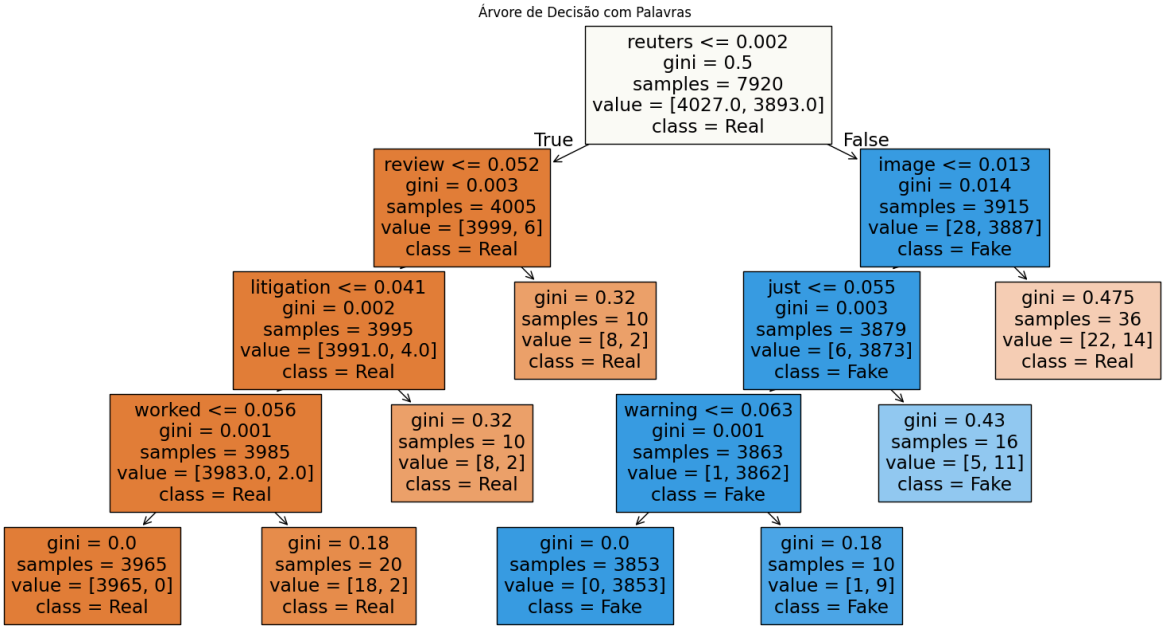
Desempenho

- Acurácia :0.996969696969697
- Precisão: 0.9980099502487563
- Recall: 0.9960278053624627
- F1 Score: 0.9970178926441352

Matriz de Confusão



Visualização da Árvore de Decisão



Análise

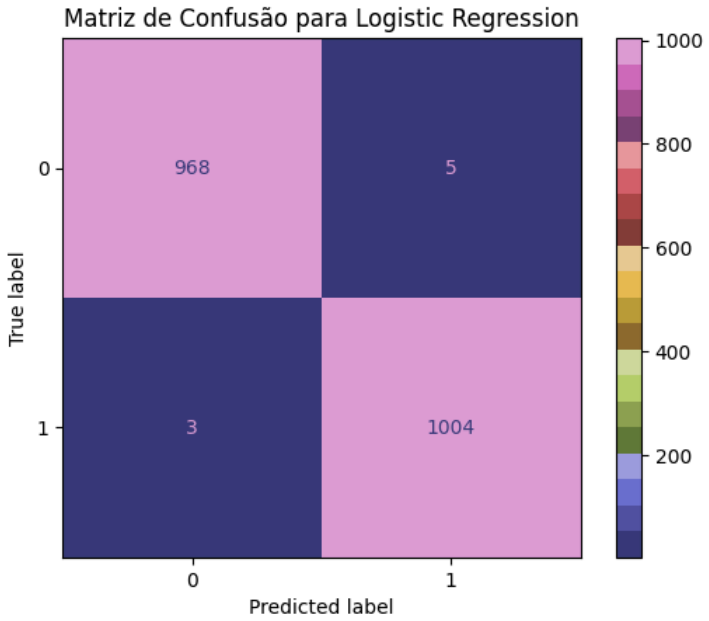
O modelo de Árvore de Decisão apresentou um desempenho excelente, com uma acurácia de 99.60%. A alta precisão e recall indicam que o modelo é muito eficaz em distinguir entre notícias reais e falsas. A matriz de confusão confirma a capacidade do modelo de realizar previsões precisas.

Regressão Logística

Desempenho

- Acurácia: 0.9959595959595959
- Precisão: 0.9950445986124876
- Recall: 0.997020854021847
- F1 Score: 0.996031746031746

Matriz de Confusão



Análise

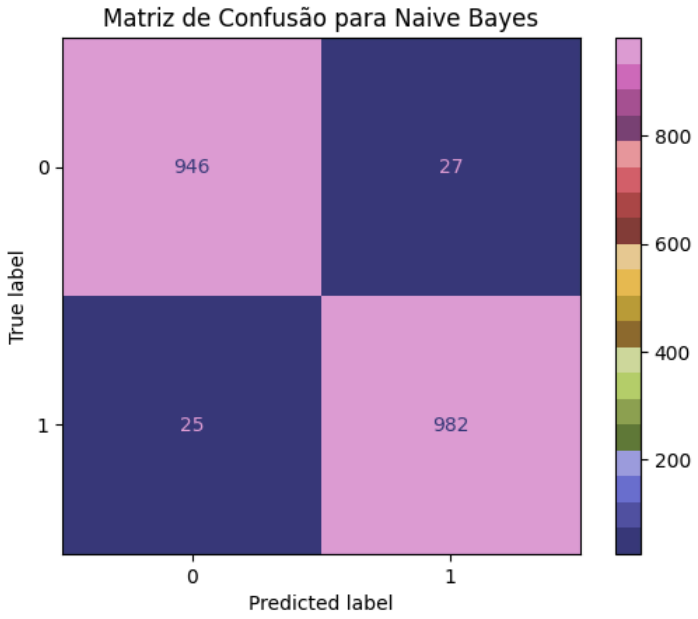
O modelo de Regressão Logística também apresentou um desempenho excelente, com acurácia, precisão, recall e F1-score todas muito próximas de 100%. A matriz de confusão mostra um balanço equilibrado entre os falsos positivos e falsos negativos, destacando a eficácia do modelo.

Naive Bayes

Desempenho

- Acurácia: 0.9737373737373738
- Precisão: 0.9732408325074331
- Recall: 0.9751737835153923
- F1 Score: 0.9742063492063492

Matriz de Confusão



Análise

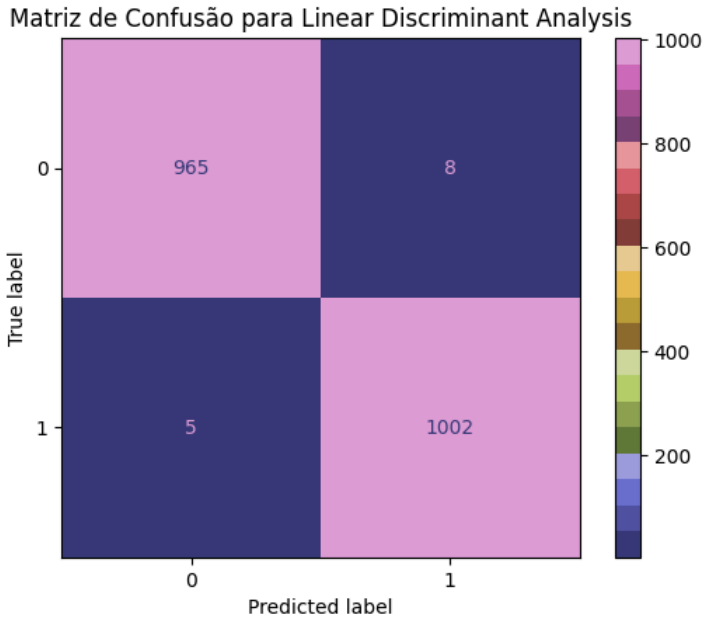
O modelo Naive Bayes teve um desempenho bom, com uma acurácia de 97.37%. Apesar de ser um modelo mais simples, ele ainda é bastante eficaz para a classificação de notícias. A matriz de confusão revela que o modelo tem um leve desequilíbrio, mas mantém um bom equilíbrio geral entre precisão e recall.

Linear Discriminant Analysis (LDA)

Desempenho

- Acurácia: 0.9934343434343434
- Precisão: 0.9920792079207921
- Recall: 0.9950347567030785
- F1 Score: 0.993554784333168

Matriz de Confusão



Análise

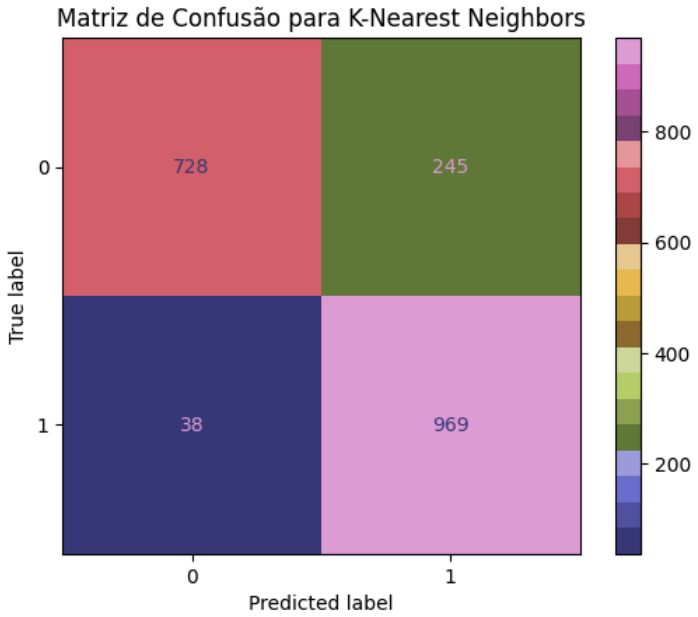
O modelo LDA apresentou uma acurácia de 99.44%. As métricas de precisão, recall e F1-score indicam um desempenho robusto, com a matriz de confusão confirmando a eficácia do modelo na classificação correta das notícias.

K-Nearest Neighbors (KNN)

Desempenho

- Acurácia: 0.8570707070707071
- Precisão: 0.7981878088962109
- Recall: 0.9622641509433962
- F1 Score: 0.8725799189554255

Matriz de Confusão



Análise

O modelo KNN teve o desempenho mais baixo entre os modelos avaliados, com uma acurácia de 85.71%. A precisão é significativamente menor que o recall, indicando uma tendência do modelo a classificar muitas notícias como verdadeiras. A matriz de confusão reflete este desequilíbrio.

Comparação Geral

Modelo	Acurácia	Precisão	Recall	F1 Score
Árvore de Decisão	0.996969696969697	0.9980099502487563	0.9960278053624627	0.9970178926441352
Regressão Logística	0.9959595959595959	0.9950445986124876	0.997020854021847	0.996031746031746
Naive Bayes	0.9737373737373738	0.9732408325074331	0.9751737835153923	0.9742063492063492
LDA	0.9934343434343434	0.9920792079207921	0.9950347567030785	0.993554784333168
KNN	0.8570707070707071	0.7981878088962109	0.9622641509433962	0.8725799189554255

Discussão

Entre os pontos fortes da solução proposta temos a diversidade de modelos. A aplicação de diferentes modelos preditivos (Árvore de Decisão, Regressão Logística, Naive Bayes, LDA e KNN) permite a exploração de diferentes abordagens e características dos dados, a comparação de diferentes técnicas e a identificação de qual modelo se ajusta melhor ao conjunto de dados específico.
Outro ponto forte é a validação cruzada, que proporciona uma estimativa mais robusta da performance do modelo ao lidar com dados não vistos. Além disso, a validação cruzada também ajuda a evitar problemas de underfitting e overfitting.
A avaliação dos modelos utilizando múltiplas métricas (acurácia, precisão, recall e f1 score) fornece uma visão completa do desempenho dos modelos. Já a matriz de confusão ajuda a visualizar detalhadamente os erros de classificação para cada um dos modelos preditivos.
O uso da técnica TF-IDF pode ajudar a capturar a relevância das palavras nos documentos melhorando a qualidade das representações textuais usadas pelos modelos, porém pode não capturar bem o contexto ou significado semântico completo das palavras.
Uma possível melhoria seria a aplicação de outros métodos, como embeddings de palavras (Word2Vec) ou transformadores (BERT), oferecendo representações textuais mais ricas.
Além disso, a adição de features adicionais ao conjunto de dados (como fonte, data de publicação, etc) pode ajudar os modelos a capturar mais informações e nuances dos dados.

Viabilidade para Aplicação Prática

Os modelos desenvolvidos possuem um potencial interessante para aplicação prática na classificação de notícias reais e falsas. No entanto, a implementação bem-sucedida depende de várias considerações como qualidade dos dados, capacidade computacional, entre outros.