UNIVERSIDADE FEDERAL DE UBERLÂNDIA

João Jacó Santos Abreu

Identificação de *fake-news* utilizando Processamento de Linguagens Naturais e Aprendizado de Máquina

Uberlândia, Brasil

2024

João Jacó Santos Abreu

Identificação de *fake-news* utilizando Processamento de Linguagens Naturais e Aprendizado de Máquina

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Uberlândia, Brasil, 09 de Maio de 2024:

Profa. Maria Adriana Vidigal de Lima
Orientadora

Prof. Bruno Augusto Nassif Travençolo

Prof. Luiz Gustavo Almeida Martins

Uberlândia, Brasil
2024

Resumo

Com o crescimento exponencial da quantidade de informações disponíveis na internet, surgiu a necessidade de avaliar a veracidade das mesmas. É uma necessidade crucial e estudos tem sido feitos para tratar desse problema, como é o objetivo deste trabalho. Propõe-se então um estudo sobre a aplicação de técnicas de Processamento de Linguagem Natural (PLN) e Aprendizado de Maquina (AM) para uma classificação automática de notícias. O estudo irá abranger desde a coleta e pré-processamento dos dados até a implementação e avaliação de modelos de AM para classificação.

Palavras-chave: *fake news*, Classificação de textos, Processamento de Linguagem Natural, Aprendizado de Máquina.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PLN	Processamento de Linguagem Natural (*Natural Language Processing*)
AM	Aprendizado de Máquina (*Machine Learning*)
AP	Aprendizado Profundo (*Deep Learning*
PMV	Produto Mínimo Viável (*Minimum Viable Product*)
MVS	Máquina de Vetores de Suporte (*Support Vector Machine*)
BoW	Bolsa de Palavras *Bag of Words*
VP	Verdadeiro positivo (*True positive*)
FP	Falso positivo (*False positive*)
FN	Falso negativo (*False negative*)
VN	Verdadeiro negativo (*True negative*)
BERT	*Bidirectional Encoder Representations for Transformers*
TF-IDF	*Term Frequency–Inverse Document Frequency*

Sumário

1 Introdução

O mundo contemporâneo está imerso em um mar de informações, caracterizando-se pela Era da Informação (JAMIL, 2007), na qual notícias e conteúdos são gerados em um ritmo sem precedentes. Uma das características dessa era é a modificação no mercado, que rapidamente deixa de privilegiar a indústria tradicional e passa a valorizar a informação e a tecnologia em maior escala. Com o advento da internet e das redes sociais, a quantidade de dados disponíveis cresceu exponencialmente, aumentando a desconfiança sobre a confiabilidade desses dados e tornando-se desafiador para os usuários acessarem conteúdos realmente relevantes. Nesse contexto, a classificação automática de notícias torna-se uma necessidade para otimizar o processo de busca e acesso a informações.

A classificação de notícias, ou categorização de textos, é o processo de atribuir automaticamente uma categoria ou rótulo a um determinado artigo ou texto com base no seu conteúdo. Essa tarefa é fundamental para diversas aplicações, como sistemas de recomendação de conteúdo, filtragem de spam e muitas outras. No entanto, a classificação de notícias apresenta desafios únicos devido à sua natureza dinâmica e diversidade de tópicos e estilos de escrita.

Para lidar com essa complexidade, a combinação de técnicas de Processamento de Linguagem Natural (PLN) e Aprendizado de Máquina (AM) tem se mostrado uma abordagem eficaz. A PLN é uma área da inteligência artificial que se concentra na interação entre computadores e linguagem humana natural (GONZALEZ, 2003). Engloba uma variedade de técnicas, incluindo análise sintática, semântica e pragmática, que permitem extrair informações significativas de textos não estruturados. Por outro lado, o AM fornece os métodos e algoritmos necessários para treinar modelos capazes de aprender com os dados e fazer previsões ou tomar decisões automaticamente.

Ao combinar PLN e AM, torna-se possível desenvolver sistemas automatizados capazes de analisar grandes volumes de textos de notícias e atribuir categorias relevantes a eles de forma precisa e eficiente. Isso não apenas facilita a organização e o acesso a informações, mas também abre portas para uma série de aplicações práticas, como personalização de conteúdo, monitoramento de mídia e detecção de *fake news*, que é onde este trabalho irá se concentrar.

1.1 Motivação

A explosão da era digital trouxe consigo uma quantidade massiva de informações disponíveis online, abrangendo uma ampla variedade de tópicos e fontes. Nesse cenário, encontrar

e acessar conteúdos relevantes tornou-se uma tarefa cada vez mais desafiadora. A crescente necessidade de otimizar esse processo de busca e acesso a informações pertinentes motiva o desenvolvimento de sistemas inteligentes capazes de classificar automaticamente notícias.

Além disso, a disseminação de desinformação e *fake news* na internet destaca a importância crítica de ferramentas capazes de filtrar e identificar conteúdos confiáveis. A classificação automatizada de notícias não apenas melhora a eficiência na organização e acesso a informações, mas também contribui para mitigar os efeitos prejudiciais da desinformação na sociedade.

Expandindo um pouco mais neste tópico, a seguir são apresentados alguns exemplos de efeitos prejudiciais oriundos da desinformação na sociedade, ou das famosas *fake news* (TEIXEIRA AMANDA DUARTE MARCOS,):

1. **Desestabilização da democracia**: A disseminação de informações falsas pode minar a confiança nas instituições democráticas, diminuindo a credibilidade do processo eleitoral e levando a decisões políticas baseadas em informações distorcidas ou falsas.

2. **Polarização e divisão social**: *fake news* muitas vezes são projetadas para explorar divisões sociais existentes e criar conflitos entre diferentes grupos da sociedade. Isso pode levar a uma polarização ainda maior, dificultando o diálogo e a cooperação entre diferentes partes da população.

3. **Impactos na saúde pública**: Notícias falsas relacionadas à saúde, como teorias da conspiração sobre vacinas ou remédios milagrosos, podem ter consequências graves para a saúde pública. Elas podem desencorajar as pessoas a buscar tratamento médico adequado, disseminar informações perigosas sobre doenças e contribuir para surtos de doenças evitáveis.

4. **Prejuízos econômicos**: A propagação de *fake news* também pode ter impactos econômicos significativos, especialmente quando se trata de notícias falsas sobre empresas ou mercados financeiros. Investidores podem tomar decisões com base em informações errôneas, levando a flutuações no mercado e prejuízos financeiros.

5. **Dano à reputação e segurança**: Indivíduos e organizações podem sofrer danos irreparáveis à sua reputação devido à disseminação de informações falsas sobre os mesmos. Isso pode afetar a segurança pessoal, a integridade profissional e a confiança do público nas instituições.

6. **Erosão da confiança na mídia**: A proliferação de *fake news* pode levar à desconfiança generalizada em relação aos meios de comunicação tradicionais, tornando-se

mais difícil para o público discernir entre fontes confiáveis e não confiáveis de informação.

Em resumo, a desinformação representa uma ameaça séria e multifacetada para a sociedade, minando os fundamentos da democracia, prejudicando a saúde pública, causando danos econômicos e sociais e minando a confiança nas instituições. A luta contra a desinformação requer esforços coordenados e abrangentes, incluindo educação pública, regulamentação de plataformas online e desenvolvimento de ferramentas tecnológicas para detectar e combater *fake news*.

1.2 Objetivo

O objetivo do trabalho de conclusão de curso é aplicar e analisar a viabilidade de classificação de notícias em duas classes - verdadeiras e falsas - com o intuito de detectar possíveis *fake news*. A análise será conduzida utilizando três métodos de entrada do Aprendizado de Máquina: Naive Bayes, Máquina de Vetores de Suporte e Regressão Logística.

1.3 Organização da monografia

O capítulo 2 apresenta toda a fundamentação teórica utilizada e conveniente a leitura deste trabalho.

O capítulo 3 apresenta três trabalhos correlatos e um breve detalhamento de seus escopos, objetivos e conclusões.

O capítulo 4 traz o detalhamento dos experimentos e resultados obtidos via código Python.

O capítulo 5 elucida uma conclusão e dispõe de análises sobre os resultados obtidos no capítulo 4, além da proposição de trabalhos futuros.

2 Fundamentação teórica

A integração de assistentes virtuais como Amazon Alexa, Google Assistente ou a Siri da Apple em nossas vidas exemplifica a interação crescente entre humanos e máquinas por meio da linguagem natural, uma habilidade inata que é utilizada desde tempos primordiais. No entanto, é importante notar que os computadores operam em uma lógica binária, lidando apenas com dados representados por 0s e 1s. Portanto, surge a questão: como as máquinas podem compreender a complexidade da linguagem humana?

A resposta a essa pergunta reside no campo do Processamento de Linguagem Natural (PLN), uma disciplina da Ciência da Computação dedicada ao desenvolvimento de métodos para analisar, modelar e compreender a linguagem humana (VAJJALA et al., 2020). Essencialmente, toda aplicação inteligente que envolve interação por meio da linguagem natural depende, em alguma medida, do processamento de linguagem natural (SANTOS et al., 2022). Existem diversas alternativas para solucionar esse problema e serão exploradas algumas delas neste projeto.

2.1 Casos de uso comuns

Para se ter uma maior noção do quão profunda é a integração do PLN na vida das pessoas serão exemplificadas inúmeras aplicações de PLN em cenários do mundo real a seguir.

- Classificação de spams, caixas de prioridades e *auto-complete* em plataformas de e-mail como Gmail, Outlook, etc.
- Assistentes de voz como Siri, Alexa, Google Assistente e Microsoft Cortana usam técnicas de PLN para interagir com o usuário, entender seus comandos e responder de acordo.
- Sistemas de busca modernos como Google e Bing, que são os pilares do acesso a informação da internet atual, usam intensivamente PLN para inúmeras sub tarefas como por exemplo: compreensão de consultas, respostas a perguntas, recuperação de informações e classificação e agrupamento dos resultados, dentre muitas outras.
- Serviços de tradução automática, como o Google Tradutor, o Bing Microsoft Tradutor e o Amazon Tradutor, estão sendo cada vez mais utilizados no mundo atual para realizar uma ampla gama de tarefas.

A PLN situa-se numa área do campo da Inteligência Artificial que se dedica a compreender, analisar e, eventualmente, gerar linguagem humana de modo a possibilitar inte-

Figura 1 – Tarefas e aplicações do PLN. Adaptado de (VAJJALA et al., 2020)

rações semelhantes às realizadas entre indivíduos e computadores (JURAFSKY; MARTIN, 2008). Nos últimos anos, essa disciplina vem ganhando considerável popularidade, em grande parte devido à fácil acessibilidade à informação proporcionada pela Internet, tornando-se assim não apenas atrativa, mas também essencial (JACKSON; MOULINIER, 2007).

2.2 Tarefas principais de PLN

Em diversos projetos de Processamento de Linguagem Natural, é comum encontrar um conjunto de tarefas fundamentais que surgem repetidamente. Dada a sua importância central, essas tarefas foram minuciosamente estudadas (VAJJALA et al., 2020). Dominá-las de maneira adequada é fundamental para capacitar o desenvolvimento de uma ampla gama de aplicações de PLN em diferentes campos. A seguir, tais tarefas são apresentadas sucintamente:

- **Modelagem da linguagem**: Esta é a função de prever qual será a palavra seguinte em uma frase com base no contexto das palavras anteriores. O objetivo é compreender a probabilidade de uma sequência de palavras ocorrer em um determinado idioma. A modelagem de linguagem é valiosa para o desenvolvimento de soluções em uma ampla gama de problemas, como reconhecimento de voz, reconhecimento óptico de caracteres, reconhecimento de escrita manual, tradução automática e correções de ortografia.

- **Classificação de textos**: Trata-se da atividade de classificar o texto em diferentes conjuntos de categorias pré-definidas com base em seu conteúdo. A classificação de texto é amplamente reconhecida como uma das tarefas mais comuns e essenciais em PLN, sendo empregada em uma diversidade de ferramentas, que vão desde a identificação de spam em e-mails até a análise de sentimentos.

- **Extração de informações**: Conforme o próprio nome indica, esta é a atividade de recuperar informações pertinentes a partir de textos, como eventos de calendário de e-mails ou nomes de indivíduos mencionados em postagens em redes sociais.

- **Recuperação de informações**: Esta tarefa consiste em localizar documentos relevantes para uma consulta feita pelo usuário em uma extensa coleção. Exemplos conhecidos de aplicação da recuperação de informações incluem serviços como a Pesquisa do Google.

- **Agentes de conversação**: Esta é a tarefa de construir sistemas de diálogo que possam conversar em linguagens humanas. Alexa, Siri, etc., são algumas aplicações comuns desta tarefa.

- **Resumo de texto**: Esta tarefa tem como objetivo criar resumos curtos de documentos mais extensos, mantendo o conteúdo principal e preservando o significado geral do texto.

- **Responder a perguntas**: Esta é a atividade de desenvolver um sistema capaz de responder automaticamente a perguntas formuladas em linguagem natural.

- **Tradução automática**: Esta é a tarefa de converter um trecho de texto de um idioma para outro. Ferramentas como o Google Tradutor são aplicações comuns desta tarefa.

- **Modelagem de tópicos**: Esta tarefa consiste em revelar a estrutura temática de uma extensa coleção de documentos. A modelagem de tópicos é uma ferramenta comum de mineração de texto e é empregada em diversos domínios, que vão desde a literatura até a bioinformática.

A Figura 2 ilustra uma sequência de tarefas dispostas em ordem de dificuldade, começando das mais simples até aquelas mais complexas:

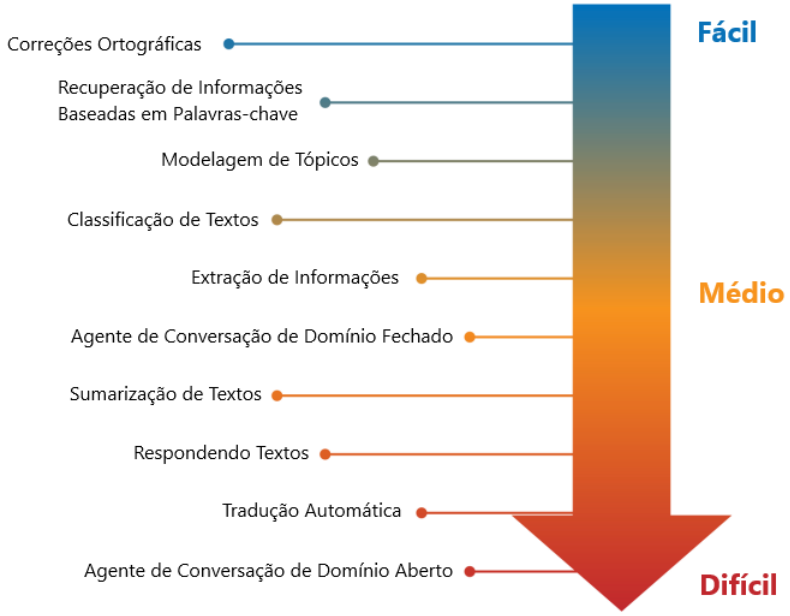

Figura 2 – Tarefas ordenadas de acordo com a sua respectiva dificuldade. Adaptado de (VAJJALA et al., 2020)

2.3 Aprendizado de máquina

Em termos gerais, inteligência artificial é um ramo da ciência da computação que tem como objetivo construir sistemas capazes de realizar tarefas que exigem inteligência humana. Isso às vezes também é chamado de "inteligência de máquina". As bases da IA foram estabelecidas na década de 1950 em um workshop organizado no Dartmouth College (VAJJALA et al., 2020). A IA inicialmente foi amplamente construída com base em sistemas lógicos, heurísticas e baseados em regras. A aprendizagem de máquina (AM) é um ramo da IA que lida com o desenvolvimento de algoritmos que podem aprender a realizar tarefas automaticamente com base em um grande número de exemplos, sem a necessidade de regras elaboradas manualmente.

Os algoritmos de aprendizado de máquina podem ser classificados em uma taxonomia com base nos resultados desejados do algoritmo (AYODELE, 2010). Dentre os tipos comuns de algoritmos estão:

- Aprendizado supervisionado: o algoritmo gera uma função que mapeia entradas para saídas desejadas, como o problema de classificação, em que busca-se aprender (ou aproximar) o comportamento de uma função que mapeia um vetor para uma das várias classes, com base em exemplos de entrada e saída.
- Aprendizado não supervisionado: este modelo lida com um conjunto de entradas sem exemplos rotulados.

- Aprendizado semi-supervisionado: combina exemplos rotulados e não rotulados para gerar uma função ou classificador apropriado.
- Aprendizado por reforço: o algoritmo aprende uma política para agir com base em observações do mundo. Cada ação afeta o ambiente, que fornece *feedback* para orientar o aprendizado.

O *Deep Learning* se enquadra principalmente na categoria de Aprendizado supervisionado, embora também possa ser aplicado em outras categorias, como Aprendizado não supervisionado e Aprendizado por reforço. O DL refere-se ao ramo do Aprendizado de Máquina que se baseia em redes neurais. AM, DL e PLN são todos subcampos dentro da IA.

A compreensão da interconexão entre AM, IA, *Deep Learning* e Processamento Natural de Linguagem revela a importância fundamental dos dados de entrada. Especificamente na análise textual, a qualidade desses dados exerce um impacto significativo no desempenho dos modelos empregados. A limpeza de texto emerge, portanto, como uma etapa de fundamental importância nesse contexto. A remoção de ruídos, a padronização da formatação e a extração de informações relevantes por meio deste processo promovem uma maior eficácia dos modelos de IA na interpretação e análise de texto.

2.4 Pré-Processamento

A etapa principal no contexto deste projeto de limpeza de dados é o pré-processamento, que envolve a remoção de informações que possivelmente não adicionam nada de útil ao classificador. Isso ocorre porque todo programa de PLN tipicamente opera em um nível de sentenças, esperando uma mínima separação das palavras.

A extração e limpeza dos textos se refere ao processo de extrair texto bruto dos dados de entrada, removendo todas as outras informações não textuais, como marcações, metadados, etc., e convertendo o texto para o formato de codificação necessário. Tipicamente, isso depende do formato dos dados disponíveis na organização (por exemplo, dados estáticos de PDF, HTML ou texto, algum tipo de fluxo contínuo de dados, etc.).

Será realizada a limpeza de todas as notícias avaliadas da base. Para isso, serão empregados métodos como:

- **Tokenização**: Consiste na transformação das palavras de cada texto em tokens, que são unidades individuais, como palavras ou pontuações.
- **Conversão para minúsculas**: Envolve a conversão de todas as letras de todos os tokens para minúsculas, a fim de garantir consistência e evitar a distinção entre palavras com letras maiúsculas e minúsculas.

- **Remoção de pontuação e stopwords**: Envolve a eliminação de pontuações, como vírgulas e pontos finais, bem como stopwords, que são palavras comuns que geralmente não contribuem para o significado de um texto, como “é”, “de”, “a”, entre outras.

- **Lematização**: Consiste em reduzir as palavras aos seus lemas, que são suas formas básicas de dicionário. Por exemplo, “correndo”, “corre” e “correu” seriam reduzidos ao mesmo lema “correr”.

2.5 Representação de Textos

A representação dos tokens criados anteriormente é uma etapa importante para qualquer problema de aprendizado de máquina. Não importa o quão bom seja o algoritmo de classificação utilizado, se a entrada fornecida for inadequada, os resultados obtidos serão ruins. Na ciência da computação, isso é frequentemente referido como “lixo entra, lixo sai” (VAJJALA et al., 2020).

A transformação de um texto em forma numérica, para que possa alimentar algoritmos de PLN e AM, é fundamental para o processamento eficiente da linguagem natural. Na terminologia de PLN, essa conversão de texto bruto para uma forma numérica é denominada representação de texto.

Na figura a seguir observa-se como ficaria esse fluxo:

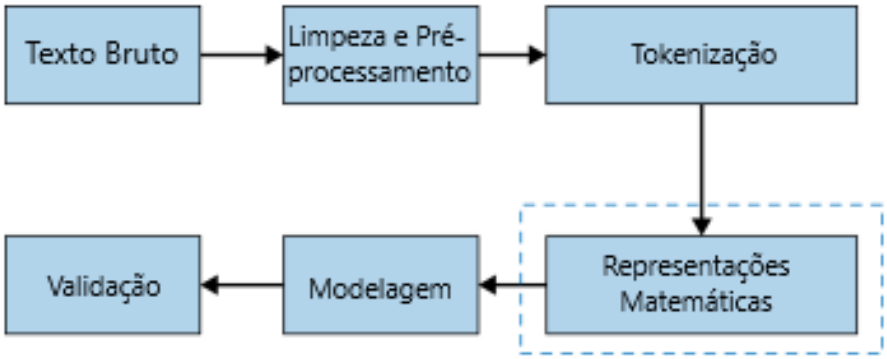

Figura 3 – Fluxo da PLN. Adaptado de (VAJJALA et al., 2020)

2.5.1 *Bag of Words*

A abordagem *Bag of Words* (BoW) é uma técnica clássica de representação de texto que tem sido amplamente utilizada em PLN, especialmente em problemas de classificação de texto. A ideia-chave por trás é a seguinte: representar o texto como uma coleção de palavras, ignorando a ordem e o contexto. A intuição básica é que ela assume que o

texto pertencente a uma classe específica no conjunto de dados é caracterizado por um conjunto único de palavras. Se duas porções de texto têm palavras quase idênticas, então elas pertencem ao mesmo saco (classe). Assim, ao analisar as palavras presentes em um pedaço de texto, pode-se identificar a classe (saco) a que pertence.

A BoW mapeia palavras para IDs inteiros únicos entre 1 e |V|. Cada texto é então convertido em um vetor de dimensões |V|, onde no componente i-ésimo desse vetor, $i = W_{id}$ é simplesmente o número de vezes em que a palavra W ocorre no texto.

Em um exemplo composto por quatro textos:

ID	**Texto**
T1	Dog bites man.
T2	Man bites dog.
T3	Dog eats meat.
T4	Man eats food.

Tabela 1 – Exemplos de textos (T1, T2, T3, T4). Fonte: (VAJJALA et al., 2020)

Dessa forma, ao aplicar a lógica da BoW, coloca-se um ID em cada palavra ocorrente em cada texto da tabela 1. Assim, se dog = 1, bites = 2, man = 3, meat = 4, food = 5, eats = 6, T1 resultará em [1 1 1 0 0 0]. Isso porque as primeiras três palavras aparecem exatamente apenas uma vez em T1 e as últimas três não aparecem. Já T4 seria [0 0 1 0 1 1].

Vantagens da BoW:

- Relativamente simples de entender e fácil de implementar.
- Com essa representação, textos que possuem as mesmas palavras terão suas representações vetoriais mais próximas umas das outras em comparação com textos que possuem palavras completamente diferentes. Utilizando-se da distância euclidiana, tem-se que a distância entre T1 e T2 é 0, em comparação com a distância entre T1 e T4, que é 2. Assim, o espaço vetorial resultante do esquema BoW captura a similaridade semântica dos textos. Portanto, se dois textos possuem vocabulários semelhantes, eles estarão mais próximos um do outro no espaço vetorial, e vice-versa.

Desvantagens da BoW:

- O tamanho do vetor aumenta com o tamanho do vocabulário. Assim, a dispersão continua sendo um problema. Uma maneira de controlá-la é limitar o vocabulário para um número n das palavras mais frequentes.
- Não captura a similaridade entre palavras diferentes que significam a mesma coisa. Supõe-se três documentos:“Eu corro”, “Eu corri” e “Eu comi”. Os vetores BoW de

todos os três documentos estarão igualmente distantes, ainda mais considerando a lematização do pré-processamento.

- Como o nome indica, é um "saco" de palavras, isto é, a informação da ordem das palavras é perdida nesta representação. Tanto T1 quanto T2 terão a mesma representação neste esquema.

2.5.2 Tarefa de classificação

E-mails são verificados diariamente por muitas pessoas, muitas vezes várias vezes ao dia. Uma funcionalidade útil encontrada em muitos provedores de serviços de e-mail é a capacidade de separar automaticamente e-mails de spam dos e-mails regulares. Isso exemplifica o uso de uma tarefa comum de PLN conhecida como classificação de texto, que consiste em atribuir uma ou mais categorias a um determinado trecho de texto a partir de um conjunto maior de categorias possíveis.

No exemplo de identificação de spam de e-mail, existem duas categorias - spam e não spam - e cada e-mail recebido é atribuído a uma dessas categorias. Esta tarefa de categorizar textos com base em algumas propriedades tem uma ampla variedade de aplicações em diversos domínios, como mídias sociais, comércio eletrônico, saúde, direito e marketing, citando-se apenas alguns. Embora o propósito e a aplicação da classificação de texto possam variar de um domínio para outro, o problema abstrato subjacente permanece o mesmo. Essa invariabilidade do problema central e suas aplicações em uma infinidade de domínios tornam a classificação de texto amplamente utilizada na indústria e pesquisada na academia. Neste projeto, serão utilizados métodos conhecidos de classificação de textos para lidar com a identificação de notícias falsas.

Em aprendizado de máquina, a classificação é o problema de categorizar uma instância de dados em uma ou mais classes conhecidas. Os dados podem ser originalmente de diferentes formatos, como texto, fala, imagem ou numérico. A classificação de texto é uma instância especial do problema de classificação, onde o ponto de dados de entrada são textos e o objetivo é categorizar cada texto em um ou mais grupos (chamados de classe) de um conjunto de grupos predefinidos. O texto pode ter comprimento arbitrário: um caractere, uma palavra, uma frase, um parágrafo ou um documento completo. No cenário deste projeto, deseja-se classificar todas as notícias de um dado repositório em duas categorias: verdadeira ou falsa. O desafio da classificação de texto é "aprender" essa categorização a partir de uma coleção de exemplos para cada uma dessas categorias e prever as categorias para novas notícias. Fora desse contexto, essa categorização nem sempre precisa resultar em uma ou duas categorias; pode haver qualquer número de categorias disponíveis.

A abordagem de classificação supervisionada, incluindo a classificação de texto, pode ser distinguida em três tipos com base no número de categorias envolvidas: classificação

binária, multiclasse ou multirrótulo (CERRI; FERREIRA et al., 2019). Se o número de classes for dois, é chamada de classificação binária. Se o número de classes for maior que dois, é chamada de classificação multiclasse. Assim, classificar um e-mail como spam ou não spam é um exemplo de configuração de classificação binária, assim como classificar uma notícia em verdadeira ou falsa.

Classificar o sentimento de uma análise de um filme por exemplo como negativo, neutro ou positivo é um exemplo de classificação multiclasse. Tanto em configurações binárias quanto multiclasse, cada documento pertence exatamente a uma classe de C, onde C é o conjunto de todas as classes possíveis.

Na classificação multirrótulo, um documento pode ter um ou mais rótulos/classes associados a ele. Por exemplo, um artigo de notícias sobre um jogo de futebol pode pertencer a mais de uma categoria, como "esportes" e "futebol", simultaneamente, enquanto outro artigo de notícias sobre eleições nos EUA pode ter os rótulos "política", "EUA" e "eleições". Assim, cada documento tem rótulos que são um subconjunto de C. Cada artigo pode estar em nenhuma classe, exatamente em uma classe, em várias classes ou em todas as classes. Às vezes, o número de rótulos no conjunto C pode ser muito grande (conhecido como "classificação extrema"). Neste projeto, o foco será apenas na classificação binária.

A seguir serão expostos alguns classificadores comuns e de entrada. Não está no escopo deste projeto se aprofundar em uma explicação detalhada do passo a passo dos classificadores.

2.5.2.1 Classificador Naive Bayes

O Naive Bayes é um classificador probabilístico que utiliza o teorema de Bayes para classificar textos com base nas evidências observadas nos dados de treinamento. Ele estima a probabilidade condicional de cada característica de um texto dado para cada classe com base na ocorrência dessa característica naquela classe e multiplica as probabilidades de todas as características de um texto dado para calcular a probabilidade final de classificação para cada classe. Por fim, ele escolhe a classe com a maior probabilidade. Sendo assim, Naive Bayes é um classificador generativo.

2.5.2.2 Máquina de Vetores de Suporte

Descreve-se a regressão logística como um classificador discriminativo que aprende os pesos para cada característica individual e prevê uma distribuição de probabilidade sobre as classes. O classificador MVS, inventado no início dos anos 1960, é um classificador discriminativo assim como a regressão logística. No entanto, ao contrário da regressão logística, ele busca encontrar um hiperplano ótimo em um espaço de dimensão superior, que pode separar as classes nos dados por uma margem máxima possível. Além disso, as MVSs são capazes de aprender até mesmo separações não lineares entre classes, ao

contrário da regressão logística. No entanto, elas também podem levar mais tempo para treinar.

2.5.2.3 Regressão Logística

Em contraste ao classificador Naive Bayes, há um classificador que visa aprender a distribuição de probabilidade sobre todas as classes, chamado de discriminativo. A regressão logística é um exemplo de classificador discriminativo e é comumente usado na classificação de texto como uma linha de base em pesquisas e como um PMV em cenários da indústria do mundo real.

Diferentemente do Naive Bayes, que estima probabilidades com base na ocorrência de características em classes, a regressão logística "aprende" os pesos para características individuais com base em quão importantes são para tomar uma decisão de classificação. O objetivo da regressão logística é aprender um separador linear entre classes nos dados de treinamento com o objetivo de maximizar a probabilidade dos dados. Esse "aprendizado" dos pesos das características e da distribuição de probabilidade sobre todas as classes é feito por meio de uma função chamada função "logística", e (daí o nome) regressão logística.

2.5.3 Treinamento e teste

Na fase de treinamento e teste de um modelo de classificação de texto, é preciso dividir o conjunto de dados em duas partes distintas: conjunto de treinamento e conjunto de teste. Essa divisão permite avaliar o desempenho dos modelos em dados não vistos, ou seja, em dados que não foram usados durante o treinamento.

- Conjunto de Treinamento: Esta parte do conjunto de dados é utilizada para treinar o modelo. Durante o treinamento, o modelo é exposto a exemplos rotulados, ou seja, exemplos de texto juntamente com suas respectivas classes ou rótulos. O modelo ajusta seus parâmetros com base nessas informações para aprender a relação entre os recursos, no caso palavras e as classes.

- Conjunto de Teste: Após o treinamento, o modelo é avaliado no conjunto de teste. Este conjunto contém exemplos de texto que o modelo não viu durante o treinamento. O objetivo é verificar se o modelo fez uma boa generalização para dados não vistos e é capaz de fazer previsões precisas em situações do mundo real.

É importante garantir que a divisão entre os conjuntos de treinamento e teste seja feita de forma aleatória para evitar viés nos conjuntos. A proporção entre os dois conjuntos pode variar dependendo do tamanho do conjunto de dados e da complexidade do problema, mas

uma prática comum é utilizar cerca de 70-80% dos dados para treinamento e o restante para teste.

2.5.4 Medidas de desempenho

Nesta seção, serão analisadas algumas métricas de avaliação, que são comumente usadas para medir sistemas de PLN. Para a maioria das métricas dessa categoria, assume-se um conjunto de teste onde tem-se a verdade absoluta ou rótulo (anotações humanas, respostas corretas). Os rótulos no caso deste projeto são binários (0 ou 1 para falso ou verdadeiro respectivamente). A saída do modelo de PLN em um ponto de dados é comparada com o rótulo correspondente para esse ponto de dados, e as métricas são calculadas com base na correspondência (ou falta de correspondência) entre a saída e o rótulo. Para a maioria das tarefas de PLN, a comparação pode ser automatizada, portanto, a avaliação pode ser automatizada.

2.5.4.1 Matriz de confusão

A matriz de confusão é uma tabela que permite visualizar o desempenho de um modelo de classificação em um conjunto de dados de teste. Ela mostra a contagem de verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN) e falsos negativos (FN) para cada classe do problema de classificação. Essa matriz é útil para entender como o modelo está classificando corretamente e incorretamente as instâncias de cada classe.

Na matriz de confusão, as linhas representam as classes reais ou verdadeiras, enquanto as colunas representam as classes previstas pelo modelo. Cada célula da matriz contém o número de instâncias que foram classificadas de acordo com a interseção das classes reais e previstas. Uma representação comum da matriz de confusão é a seguinte:

	Classe prevista	
Classe real	Positiva	Negativa
Positiva	VP	FP
Negativa	FN	VN

Tabela 2 – Matriz de Confusão

2.5.4.2 Acurácia, precisão, revocação e F1

A matriz de confusão é uma ferramenta essencial para avaliar o desempenho de modelos de classificação. Ela apresenta a contagem de verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN) e falsos negativos (FN). Com base nesses valores, pode-se calcular várias medidas de desempenho importantes:

- **Acurácia**: é a proporção de previsões corretas feitas pelo modelo em relação ao total de previsões. Ela é calculada como:

$$accuracy = \frac{VP + VN}{VP + FP + VN + FN}$$

A acurácia é útil para avaliar o desempenho geral do modelo, especialmente quando as classes são balanceadas.

- **Precisão**: mede a proporção de exemplos positivos previstos corretamente em relação a todos os exemplos previstos como: positivos. Ela é calculada como

$$precision = \frac{VP}{VP + FP}$$

A precisão é importante em casos onde os falsos positivos são custosos, como em diagnósticos médicos.

- **Revocação**: também conhecida como sensibilidade, mede a proporção de exemplos positivos que foram corretamente identificados pelo modelo em relação a todos os exemplos positivos reais. Ela é calculada como:

$$recall = \frac{VP}{VP + FN}$$

A revocação é crucial em cenários onde é importante recuperar todos os exemplos positivos, como em sistemas de busca.

- **F1-score**: é uma métrica que combina precisão e revocação em um único valor, capturando o equilíbrio entre ambas. É calculado como a média harmônica de precisão e revocação, ou seja:

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

O F1-score é útil quando se quer uma única métrica que leve em consideração tanto a completude quanto a exatidão das previsões.

3 Trabalhos Relacionados

As notícias falsas têm se tornado uma preocupação crescente nos últimos anos, pois podem ter sérias consequências para indivíduos, organizações e sociedades. Várias técnicas de aprendizado de máquina foram propostas para detectar e classificar notícias falsas, com o objetivo de melhorar a precisão e eficiência do processo de classificação. Nesta seção de trabalhos relacionados, serão analisados três trabalhos recentes sobre classificação de notícias falsas usando aprendizado de máquina, publicados entre 2019 e 2023.

3.1 *Fake news* Classification Using Machine Learning

O trabalho de Mohammed et al. (2023) apresenta uma proposta baseada em aprendizado de máquina para a classificação de *fake news*, utilizando uma combinação de técnicas de processamento de linguagem natural (PLN) e algoritmos de classificação.

Inicialmente, os artigos de notícias brutos são pré-processados e características relevantes são extraídas usando técnicas de PLN, como tokenização, stemming e marcação de partes do discurso.

Em seguida, diversos modelos de classificação são treinados e avaliados, incluindo regressão logística, árvore de decisão e máquina de vetores de suporte, utilizando várias métricas de desempenho, como precisão, revocação e pontuação F1.

Os resultados experimentais demonstraram que a melhor abordagem foi a MVS, que alcançou uma acurácia de 92% no conjunto de dados de teste na classificação de artigos de *fake news* de fontes Indianas. Os autores destacam que a abordagem pode ser aplicada em diversas áreas, como monitoramento de redes sociais, filtragem de notícias e moderação de conteúdo, além de contribuir para o combate à disseminação de *fake news*.

3.2 Comparação de métodos de aprendizado de máquina para classificação automática de notícias em português.

A dissertação de Saavedra (2023) propõe a comparação de métodos classificadores de textos aplicados à notícias em português. Inicialmente foi desenvolvido um coletor personalizado que recuperou mais de 18 mil notícias atualizadas de portais brasileiros, as quais foram rotuladas manualmente em 19 classes temáticas distintas.

Foram exploradas diversas técnicas de pré-processamento e representação textual, como Word2Vec, Doc2Vec, *Bag of Words* com TF-IDF e fine-tunning de um modelo Bidirecti-

onal Encoder Representations for Transformers (BERT) pré-treinado. Para treinamento e avaliação foram empregados algorítimos como Regressão Logística e Support Vector Machine.

O modelo BERT obteve o melhor desempenho, com 93% de acurácia e 98% de F1. No entanto, combinações como *Bag of Words* + MVS e Doc2Vec + regressão logística emergiram como métricas satisfatórias do ponto de vista de custo-benefício computacional.

3.3 Classificação de notícias digitais utilizando Processamento de Linguagem Natural

No contexto do trabalho de Lima (2023), também foi utilizado o repositório Corpus Fake.Br (SANTOS; MONTEIRO; PARDO, 2018) e o objetivo foi criar um modelo computacional para classificar notícias digitais em português brasileiro utilizando técnicas de pré-processamento similares às vistas no capítulo 2 deste projeto, porém com outra técnica de representação textual: Word Embeddings.

Para melhor validação dos conjuntos de testes e treinamento, foi utilizado uma metodologia chamada Validação Cruzada K-fold, onde ocorre a separação dos dados em vários subconjuntos (daí a letra K, onde K é o número de subconjuntos), então várias iterações são feitas, onde em cada uma delas um subconjunto é determinado como o conjunto de testes e os demais de treinamento, até que todas as combinações tenham sido feitas. Isso permite verificar se o modelo está sofrendo algum tipo de influência em relação aos dados de treinamento. O método para classificação utilizado foi a Árvore de decisão da função *entropy* da biblioteca scikit-learn do Python.

Os testes foram feitos em um escopo de 500 notícias escolhidas aleatoriamente do corpus Fake.Br, divididos meio a meio entre notícias falsas e verdadeiras. A média da acurácia sobre o conjunto de treinamento foi de 98.75%, enquanto que a média de acurácia para o conjunto de testes foi de 55.59%.

4 Classificação de informações falsas

4.1 O corpus Fake.br

Nesta seção, será discutido o corpus "Fake.Br", composto por notícias em português (brasileiro), tanto verdadeiras quanto falsas. Conforme indicado pelos autores, não há outro corpus semelhante disponível para esta língua (SANTOS; MONTEIRO; PARDO, 2018). O repositório do corpus "Fake.Br' está disponível em <https://github.com/roneysco/Fake.br-Corpus>.

Os autores destacam que a coleta de textos para este corpus foi uma tarefa desafiadora. Demandou vários meses para encontrar e verificar manualmente as notícias falsas disponíveis na web e, em seguida, procurar semi-automaticamente por notícias verdadeiras correspondentes para cada uma das falsas. A etapa manual foi essencial para examinar detalhadamente as notícias falsas, garantindo assim a qualidade e a confiabilidade do corpus.

No total, foram coletadas 7.200 notícias, sendo exatamente 3.600 verdadeiras e 3.600 falsas. Todas estão em formato de texto simples, cada uma em um arquivo diferente. Mantiveram a homogeneidade de tamanho o máximo possível, mas algumas notícias verdadeiras são mais longas do que as falsas. Por esse motivo, também foi fornecido uma versão do corpus com normalização de tamanho, na qual, para cada par de notícias verdadeiras e falsas, o texto mais longo é truncado (em número de palavras) para o tamanho do texto alinhado mais curto. Foi estabelecido um intervalo de tempo de 2 anos para as notícias, de janeiro de 2016 a janeiro de 2018, mas houve casos de notícias falsas neste período que se referiam a notícias verdadeiras de um tempo anterior a isso. Porém isso não foi considerado um problema.

As notícias falsas foram analisadas e coletadas manualmente de 4 sites: Diário do Brasil, A Folha do Brasil, The Journal Brasil e Top Five TV. Por fim, foram filtradas as notícias que apresentavam meias verdades, mantendo apenas aquelas que eram totalmente falsas.

Já as notícias verdadeiras foram coletadas de maneira semiautomática. Foi usado um rastreador para coletar notícias das principais agências de notícias do Brasil, nomeadamente, G1, Folha de São Paulo e Estadão. O rastreador procurou nas páginas da web correspondentes dessas agências por palavras-chave das notícias falsas. Cerca de 40.000 notícias verdadeiras foram coletadas desta forma. Para cada notícia falsa, foi aplicado uma medida de similaridade lexical, escolhendo as mais semelhantes às notícias falsas, e por fim uma última verificação manual foi feita para garantir que as notícias falsas e verdadeiras estivessem relacionadas ao assunto. É interessante observar que houve casos

em que as notícias verdadeiras negavam explicitamente as falsas correspondentes, mas outras eram meramente sobre o mesmo tópico.

4.2 Metodologia de classificação

Como mencionado na seção anterior, será utilizado neste projeto o repositório "Fake.Br-Corpus" (SANTOS; MONTEIRO; PARDO, 2018) de 7200 notícias, 3600 sabidamente falsas e 3600 sabidamente verdadeiras para treinar métodos de classificação com teor binário, isto é, recebendo um conjunto de notícias como entrada e serão atribuído as mesmas quais são verdadeiras e quais são falsas. Para criação do código, será utilizado Python na ferramenta Google Colab, com acesso à Gemini API, onde é possível variar a entrada entre textos, códigos, imagens e áudios (Google Gemini, 2024).

A Figura 4 mostra o código, em linguagem Python, elaborado para a clonagem do repositório "Fake.Br" a partir do GitHub para a utilização dos seus dados.

```
!pip install nltk
!pip install git-python  # Instalar a biblioteca git

import os
import nltk
import git

repo_dir = "/content/repo"

# Verificar se o diretório do repositório já existe
if not os.path.exists(repo_dir):
  # Clonar o repositório do GitHub
  git.Repo.clone_from("https://github.com/roneysco/Fake.br-Corpus.git", "/content/repo")
```

Figura 4 – Clonando do Github

Para todo o processo de classificação das notícias, serão empregadas bibliotecas tradicionais da linguagem Python. Estas ferramentas, conhecidas por sua eficiência e robustez, facilitarão a implementação dos métodos de classificação, bem como a execução da limpeza e organização dos dados. O uso dessas bibliotecas não apenas simplifica o desenvolvimento, mas também garante uma base sólida e confiável para a análise e interpretação dos resultados obtidos. São elas:

- **os**: manipulação de diretórios da máquina virtual
- **nltk**: funções de pré-processamento dos textos
- **sklearn**: classificadores e outras funções relacionadas

- **matplotlib/seaborn**: plotar a matriz de confusão
- **random**: randomizar a ordem dos dados

A Figura 5 mostra como foi feita a organização dos diretórios de notícias falsas e verdadeiras em pastas diferentes sabendo-se assim qual classe está sendo lida.

```
import os
import nltk
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import matplotlib.pyplot as plt
import seaborn as sns
import random

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# Diretórios dos textos verdadeiros e falsos
diretorio_verdadeiras = "/content/repo/full_texts/true"
diretorio_falsas = "/content/repo/full_texts/fake"
```

Figura 5 – Bibliotecas utilizadas e organização de diretórios

O presente trabalho deve empregar uma abordagem que envolve a aplicação de métodos de classificação tradicionais, incluindo Naive Bayes, Máquina de Vetores de Suporte e Regressão Logística. No entanto, antes da implementação desses métodos, uma etapa fundamental de limpeza e organização dos dados será conduzida. Esta fase inicial é fundamental para garantir a qualidade e a confiabilidade dos resultados obtidos posteriormente, permitindo uma análise mais precisa e significativa dos dados.

4.2.1 Pré-processamento

O pré-processamento dos dados é uma etapa importante e segue um roteiro específico para garantir a qualidade e a eficácia das análises subsequentes. Este processo inclui diversas etapas, tais como a remoção de dados duplicados, a correção de erros, a padronização de formatos, a tokenização de texto e a remoção de *stopwords*. Além disso, técnicas de normalização, como a lematização e a *stemming*, são aplicadas para reduzir as variações linguísticas. A etapa final envolve a vetorização dos textos, transformando-os em representações numéricas adequadas para a entrada nos algoritmos de classificação.

Esse roteiro, cuidadosamente elaborado, assegura que os dados estejam prontos para serem processados pelos métodos de classificação selecionados, permitindo uma análise

precisa e significativa das notícias. Neste contexto, serão utilizadas as seguintes etapas para pré-processamento dos textos:

- Tokenização
- Conversão para minúsculas
- Remoção de *stopwords* e pontuações
- Lematização

A Figura 6 ilustra como as etapas de pré-processamento foram implementadas no ambiente da linguagem Python.

```python
# Função para pré-processamento dos textos
def pre_processamento(texto):
    # Tokenização
    tokens = word_tokenize(texto)

    # Conversão para minúsculas
    tokens = [token.lower() for token in tokens]

    # Remoção de pontuação e stopwords
    stop_words = set(stopwords.words('portuguese'))
    tokens = [token for token in tokens if token.isalnum() and token not in stop_words]

    # Lematização
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(token) for token in tokens]

    return ' '.join(tokens)
```

Figura 6 – Pré-processamento dos textos

4.2.2 Construção da *Bag of Words* (BoW)

Seguindo o referencial teórico do capítulo 2, será abordada agora a construção da *Bag of Words*. Para isso utiliza-se a função de pré-processamento da seção anterior, aplicando-a para cada notícia do repositório. Assim que aplicado, as notícias são armazenadas em um vetor cru de notícias, juntamente com os seus devidos rótulos, 1 para verdadeiras e 0 para falsas. Em seguida aplica-se uma randomização na ordem do vetor de notícias para que aconteça um embaralhamento.

A Figura 7 mostra a função `construir_bow` com o rotulamento, randomização e a separação do vetor em 2 listas, uma contendo apenas as notícias, e outra contendo apenas os rótulos, porém respeitando a ordem em que foram distribuídos no vetor. Utiliza-se então um objeto `CountVectorizer` juntamente com o seu método `fit_transform` para criar a representação numérica descrita na subseção 2.5.1. Por fim, retorna-se a *Bag of Words* criada.

```python
# Função para construção da BoW
def construir_bow(diretorio_verdadeiras, diretorio_falsas):
    noticias = []
    # Processar notícias verdadeiras
    for arquivo in os.listdir(diretorio_verdadeiras):
        with open(os.path.join(diretorio_verdadeiras, arquivo), 'r', encoding='utf-8') as f:
            texto = f.read()
            texto_processado = pre_processamento(texto)
            noticias.append((texto_processado, 1))

    # Processar notícias falsas
    for arquivo in os.listdir(diretorio_falsas):
        with open(os.path.join(diretorio_falsas, arquivo), 'r', encoding='utf-8') as f:
            texto = f.read()
            texto_processado = pre_processamento(texto)
            noticias.append((texto_processado, 0))

    # Randomizar a ordem das notícias mantendo a correspondência dos rótulos
    random.shuffle(noticias)

    # Obter listas separadas de notícias e rótulos
    lista_noticias = [noticia[0] for noticia in noticias]
    lista_rotulos = [noticia[1] for noticia in noticias]

    # Construir a BoW
    vectorizer = CountVectorizer()
    matriz_bow = vectorizer.fit_transform(lista_noticias)

    return matriz_bow, lista_rotulos

# Construir a BoW de todas as notícias
bow_completa, rotulos_completos = construir_bow(diretorio_verdadeiras, diretorio_falsas)
```

Figura 7 – Construção da BoW

4.2.3 Divisão de dados em Treinamento e Teste

Como visto na seção 2.5.3, na fase de treinamento e teste de um modelo de classificação de texto, é preciso dividir o conjunto de dados em duas partes distintas: conjunto de treinamento e conjunto de teste. Essa divisão permite avaliar o desempenho do modelo em dados não vistos, ou seja, em dados que não foram usados durante o treinamento. Para este fim, foi aplicada uma média de 80% para dados de treinamento e 20% para teste, conforme código contido na Figura 8.

O parâmetro `random_state` é um parâmetro usado para controlar a aleatoriedade. Quando se define `random_state` para um valor específico, garante-se que o resultado da função seja reproduzível, ou seja, executar a função várias vezes com o mesmo valor de `random_state` produzirá sempre o mesmo resultado.

```
# Dividir a BOW combinada e os rótulos em conjuntos de treinamento e teste
bow_treino, bow_teste, rotulos_treino, rotulos_teste = train_test_split(
    bow_completa, rotulos_completos, test_size=0.2, random_state=52)
```

Figura 8 – Divisão de dados

4.3 Análise e comparação dos resultados obtidos

Para comparação e análise dos resultados, foram realizadas três baterias de testes, cada uma contendo três execuções, variando apenas a ordem em que as notícias foram treinadas e interpretadas pelos modelos.

4.3.1 Máquina de Vetores de Suporte

4.3.1.1 Execução 1

- **Matriz de confusão**:

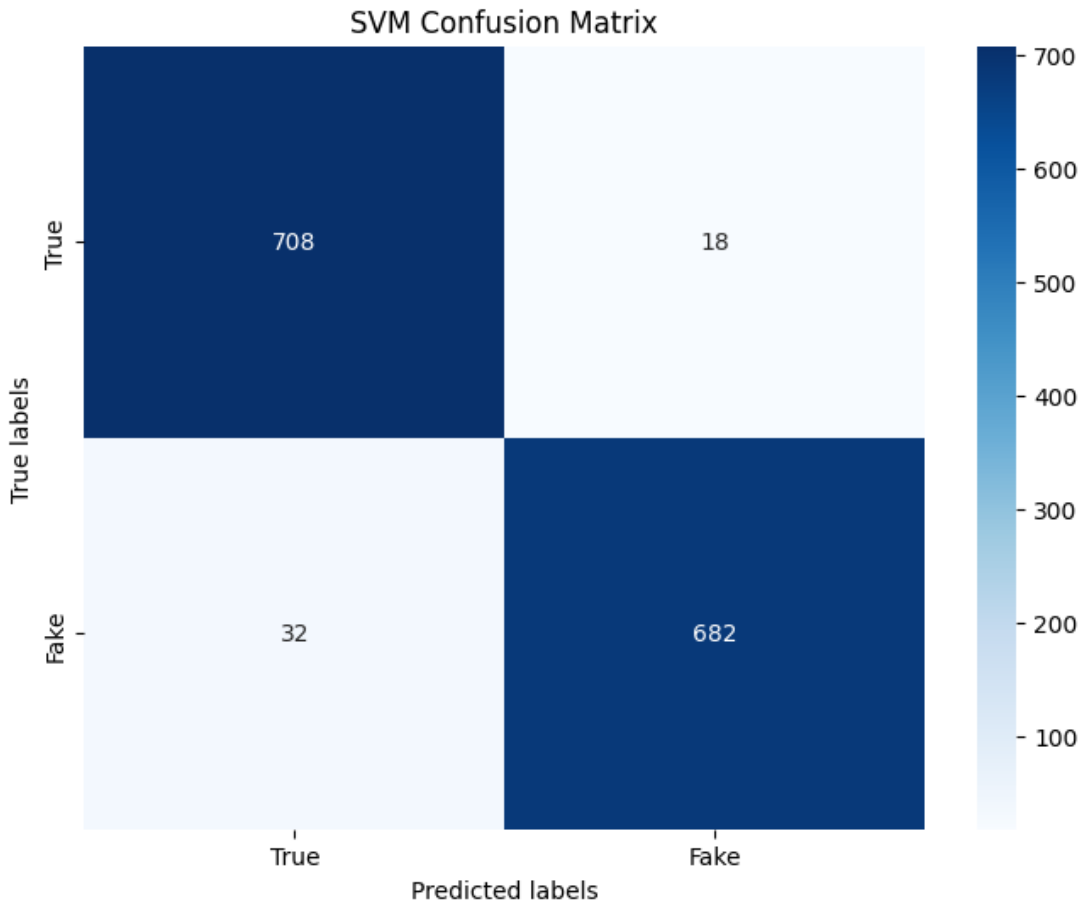

Figura 9 – Matriz de confusão MVS: 1

- **Tempo de execução**: 29s
- **Acurácia**: 96.53%
- **Precisão** : 95.68%
- **Revocação** : 97.52%
- **F1**: 96.59%

4.3.1.2 Execução 2

- **Matriz de confusão**:

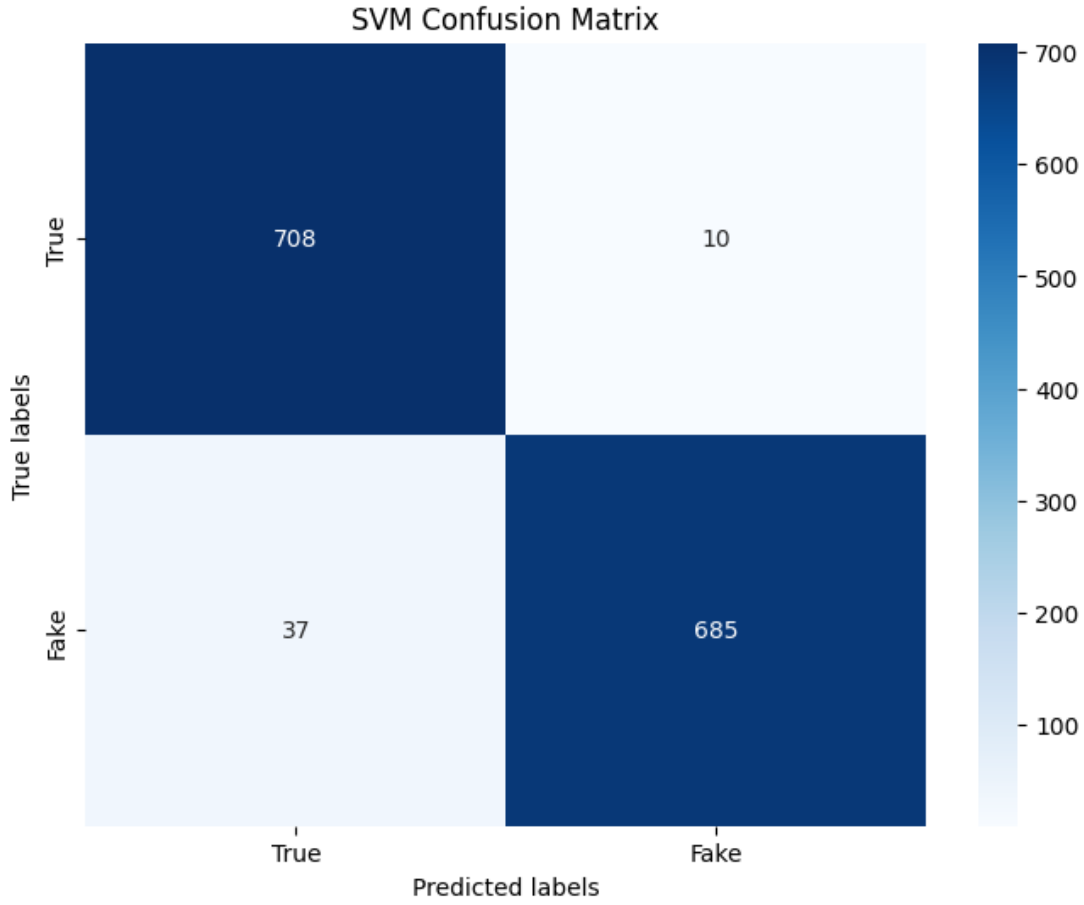

Figura 10 – Matriz de confusão MVS: 2

- **Tempo de execução**: 26s
- **Acurácia**: 96.74%
- **Precisão** : 95.03%
- **Revocação** : 98.61%
- **F1**: 96.79%

4.3.1.3 Execução 3

- **Matriz de confusão**:

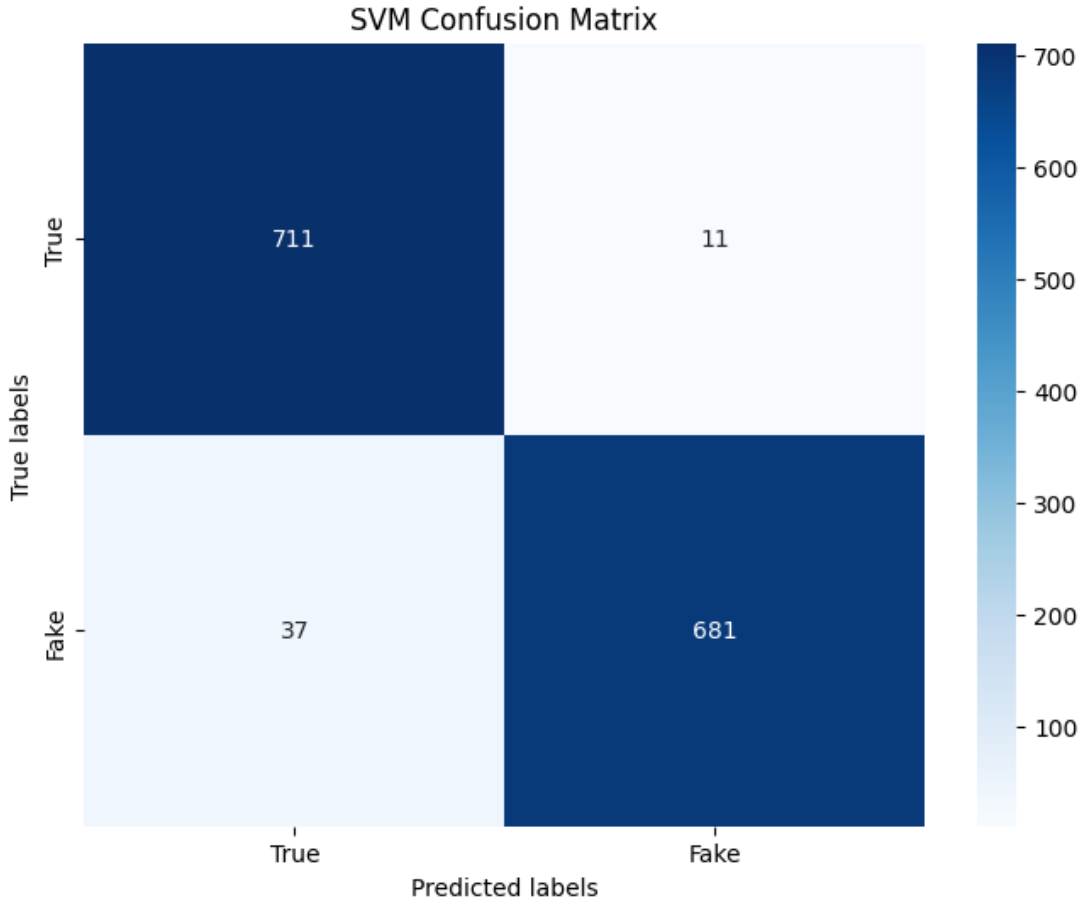

Figura 11 – Matriz de confusão MVS: 3

- **Tempo de execução**: 23s
- **Acurácia**: 96.67%
- **Precisão** : 95.05%
- **Revocação** : 98.48%
- **F1**: 96.73%

4.3.1.4 Média Final MVS

Em média, o classificador MVS proporcionou:

- 26s de tempo de execução
- 96.65% de acurácia
- 95.25% de precisão
- 98.20% de revocação
- 96.70% de F1

4.3.2 Naive Bayes

4.3.2.1 Execução 1

- **Matriz de confusão**:

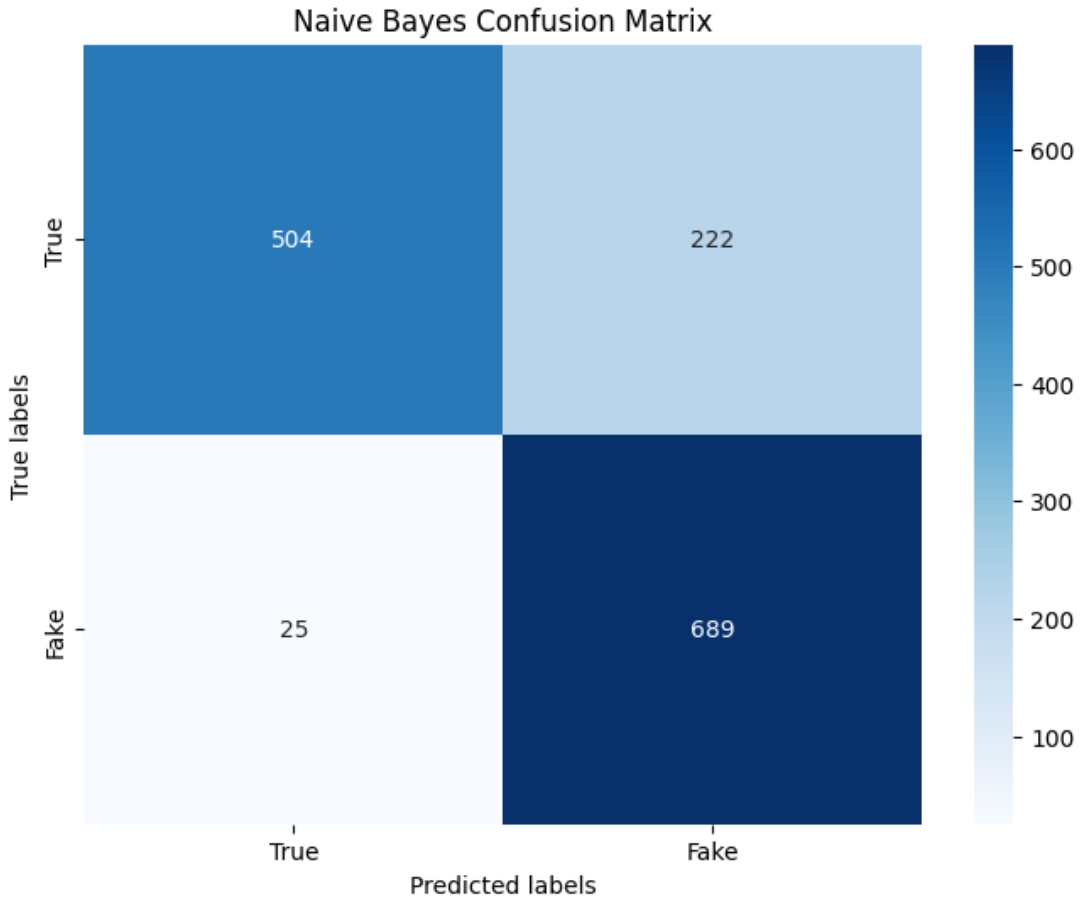

Figura 12 – Matriz de confusão Naive Bayes: 1

- **Tempo de execução**: 1s

- **Acurácia**: 82.85%
- **Precisão** : 95.27%
- **Revocação** : 69.42%
- **F1**: 80.32%

4.3.2.2 Execução 2

- **Matriz de confusão**:

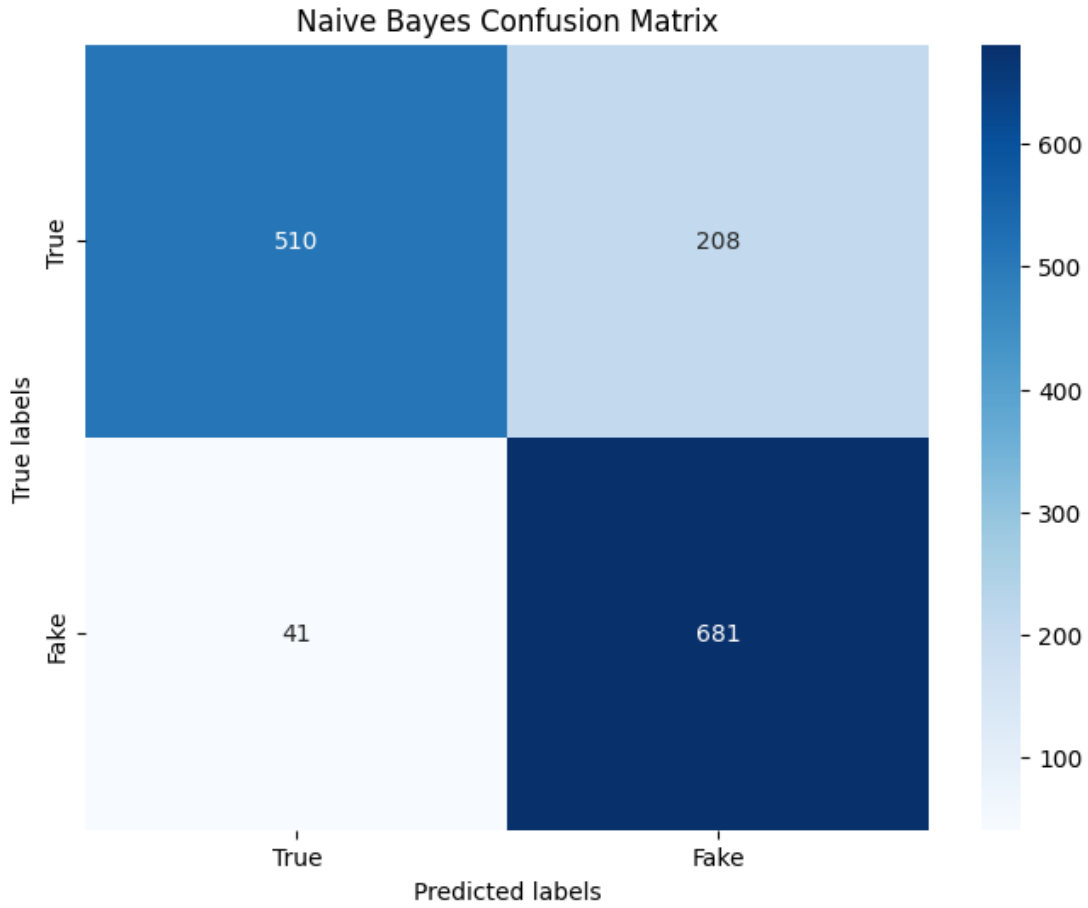

Figura 13 – Matriz de confusão Naive Bayes: 2

- **Tempo de execução**: 1s
- **Acurácia**: 82.71%
- **Precisão** : 92.56%
- **Revocação** : 71.03%
- **F1**: 80.38%

4.3.2.3 Execução 3

- **Matriz de confusão**:

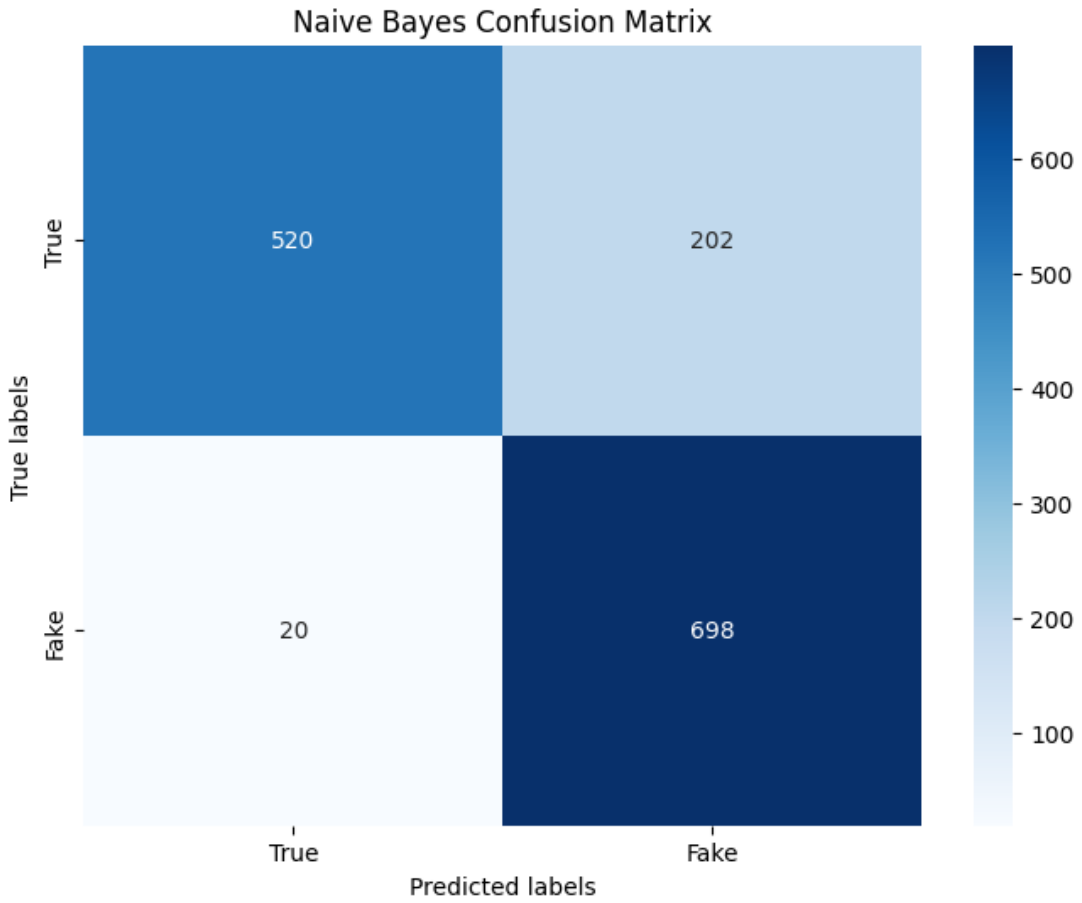

Figura 14 – Matriz de confusão Naive Bayes: 3

- **Tempo de execução**: 1s
- **Acurácia**: 84.58%
- **Precisão** : 96.30%
- **Revocação** : 72.02%
- **F1**: 82.41%

4.3.2.4 Média Final Naive Bayes

Em média, o classificador Naive Bayes proporcionou:

- 1s de tempo de execução
- 83.38% de acurácia

- 94.04% de precisão
- 70.49% de revocação
- 81.04% de F1

4.3.3 Regressão Logística

4.3.3.1 Execução 1

- **Matriz de confusão**:

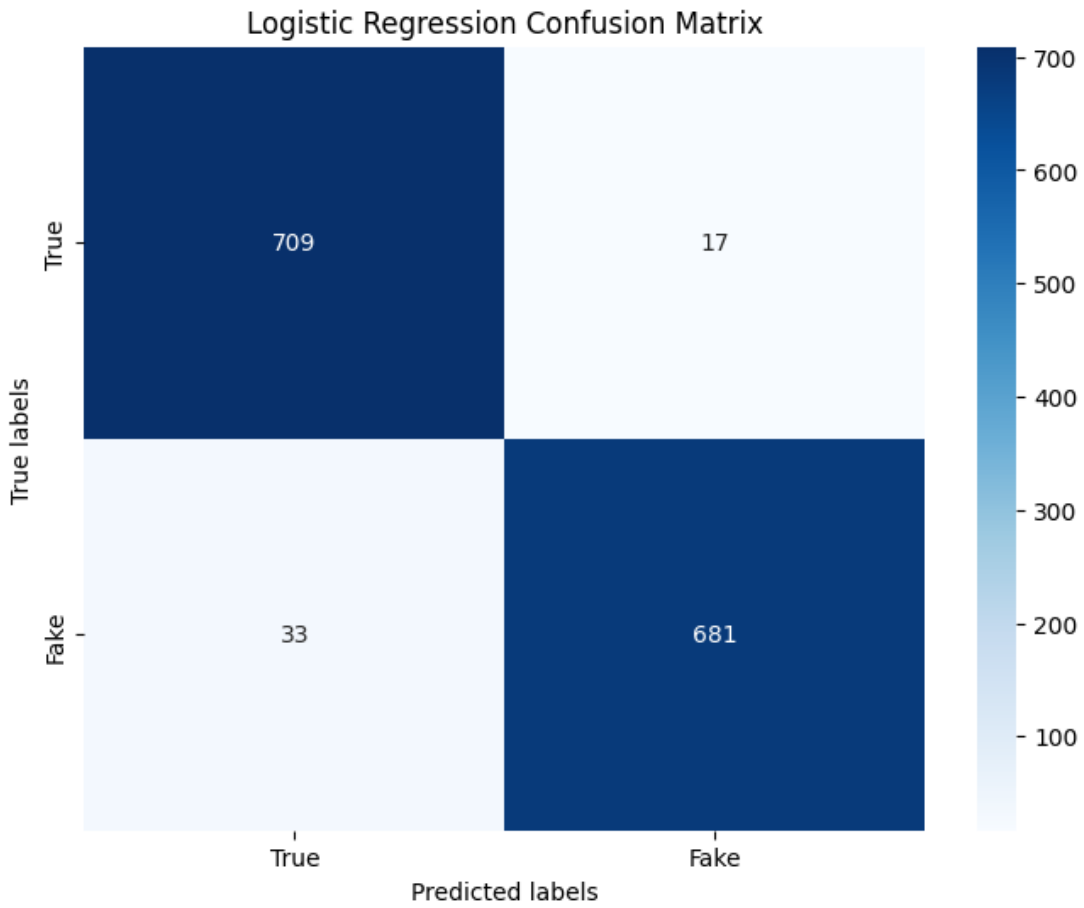

Figura 15 – Matriz de confusão Regressão Logística: 1

- **Tempo de execução**: 7s
- **Acurácia**: 96.53%
- **Precisão** : 95.55%
- **Revocação** : 97.66%
- **F1**: 96.59%

4.3.3.2 Execução 2

- **Matriz de confusão**:

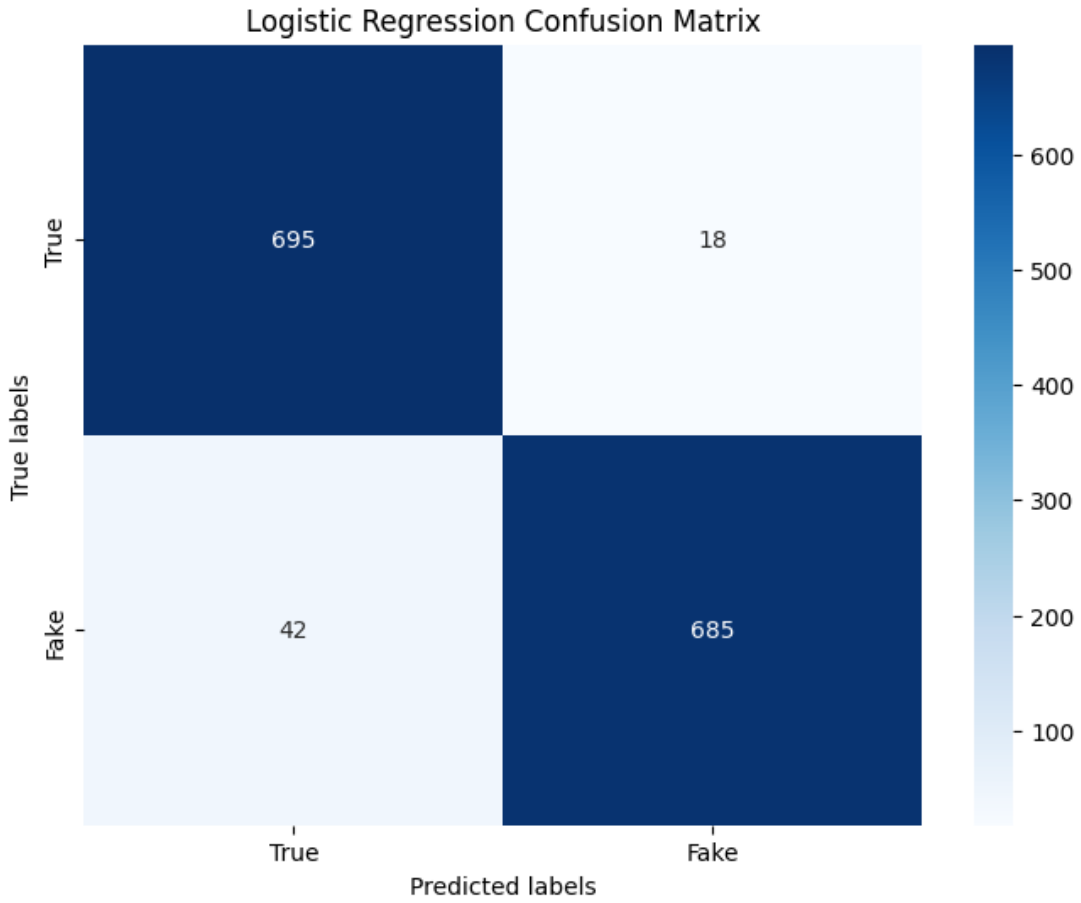

Figura 16 – Matriz de confusão Regressão Logística: 2

- **Tempo de execução**: 8s
- **Acurácia**: 95.83%
- **Precisão** : 94.30%
- **Revocação** : 97.48%
- **F1**: 95.86%

4.3.3.3 Execução 3

- **Matriz de confusão**:
- **Tempo de execução**: 6s
- **Acurácia**: 96.81%

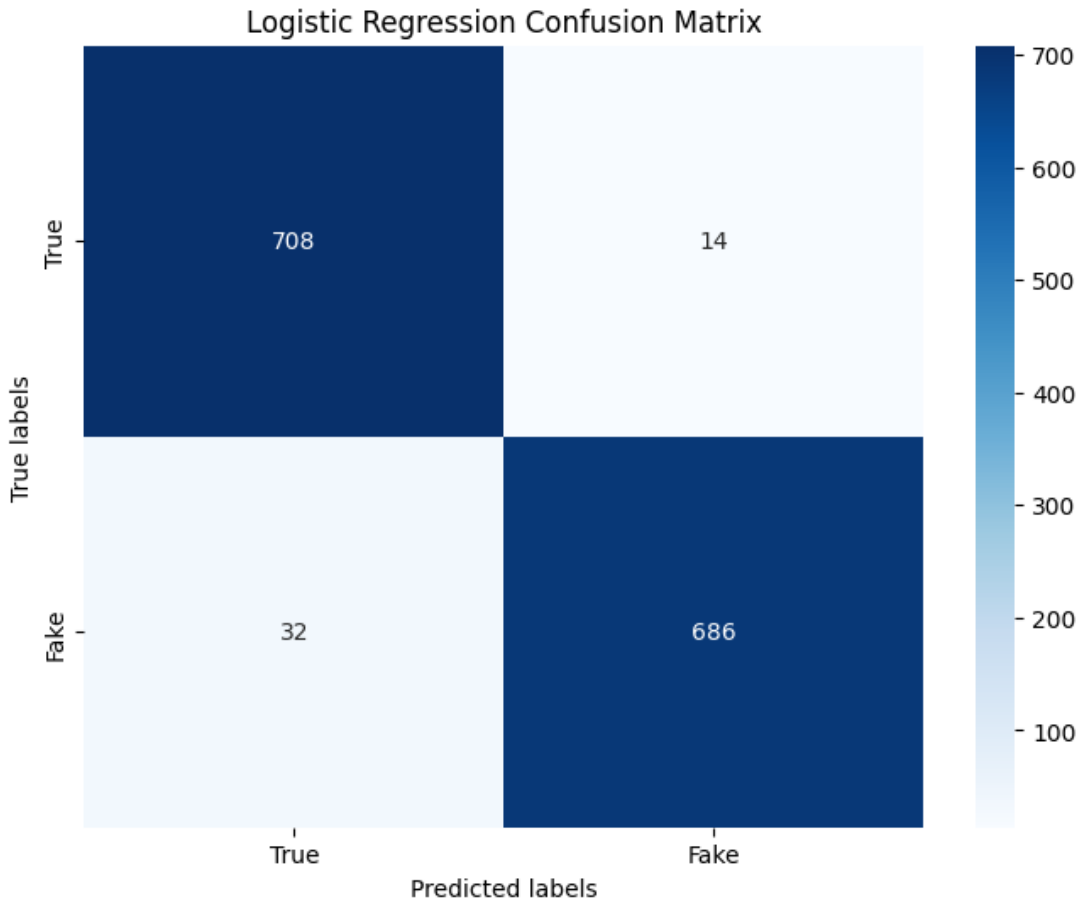

Figura 17 – Matriz de confusão Regressão Logística: 3

- **Precisão** : 95.68%
- **Revocação** : 98.06%
- **F1**: 96.85%

4.3.3.4 Média Final Regressão Logística

Em média, o classificador Regressão Logística proporcionou:

- 7s de tempo de execução
- 96.39% de acurácia
- 95.51% de precisão
- 97.07% de revocação
- 96.10% de F1

5 Conclusão

Com base nos resultados obtidos, conclui-se que:

- O classificador MVS apresentou um desempenho geral bastante consistente, com alta acurácia (96.65%), precisão (95.25%), revocação (98.20%) e F1-score (96.70%). No entanto, o tempo de execução foi relativamente maior em comparação com os outros modelos, com uma média de 26 segundos, se provando custoso computacionalmente.
- O classificador Naive Bayes demonstrou uma acurácia menor em comparação com o MVS, atingindo uma média de 83.38%. Apesar disso, apresentou um tempo de execução muito baixo, apenas 1 segundo. A precisão e o F1-score também foram razoavelmente altos, mas a revocação foi menor, sugerindo que o modelo pode ter dificuldades em identificar corretamente alguns casos positivos.
- A Regressão Logística obteve resultados semelhantes ao MVS em termos de acurácia (96.39%), precisão (95.51%) e F1-score (96.10%). O tempo de execução foi intermediário, com uma média de 7 segundos. A revocação foi ligeiramente menor em comparação com o MVS, indicando uma possível tendência de subestimar os casos positivos. Porém em uma situação de mundo real provavelmente seria o método preferível dentre os três citados devido ao seu custo-benefício computacional.

Entretanto é preciso ressaltar alguns pontos:

- A aleatoriedade dos experimentos foi decidida exclusivamente pela ordem de leitura das notícias.
- Todo e qualquer ajuste na etapa de pré-processamento tem influência direta no resultado das classificações, logo é válido explorar diferentes estratégias nessa parte.

5.1 Trabalhos futuros

Como futuros trabalhos, pretende-se explorar as seguintes abordagens:

1. **Conjunto de testes balanceado**: Em vez de usar um conjunto de testes desbalanceado, considerar a criação de um conjunto de testes com 50% de notícias verdadeiras e 50% de notícias falsas. Isso pode ajudar a reduzir possíveis vieses dos modelos, em um cenário onde aleatoriamente foram escolhidas mais notícias verdadeiras do que falsas por exemplo.

2. **Variação nos Parâmetros de Aleatoriedade**: Investigar diferentes fontes de aleatoriedade, como o parâmetro *random_state* e os grupos de notícias selecionados em métodos de validação cruzada, como o K-fold. Ajustar esses parâmetros pode fornecer insights sobre a estabilidade e a generalização dos modelos em diferentes cenários.

3. **Exploração de Outros Métodos de Representação de Textos**: Além da abordagem de *Bag of Words* (BoW), considerar outros métodos de representação de textos, como:

    - **TF-IDF (Term Frequency-Inverse Document Frequency)**: Que leva em consideração não apenas a frequência das palavras em um documento, mas também sua importância em relação ao corpus inteiro.
    - **Word Embeddings (Word2Vec, Doc2Vec), etc.**: Técnicas que mapeiam palavras em vetores de alta dimensionalidade, capturando relações semânticas e contextuais entre as palavras.
    - **Modelos de Linguagem Pré-Treinados (BERT, GPT)**: Utilizar modelos de linguagem pré-treinados e com grande confiabilidade para extrair representações ricas e contextualizadas das palavras.

Todas as abordagens adicionais citadas podem trazer melhores insights, melhorando ou trazendo os modelos mais pertos da realidade, fornecendo resultados mais condizentes com o mundo real.

Referências

AYODELE, T. O. Types of machine learning algorithms. In: ZHANG, Y. (Ed.). *New Advances in Machine Learning*. Rijeka: IntechOpen, 2010. cap. 3. Disponível em: <https://doi.org/10.5772/9385>. Citado na página 14.

CERRI, R.; FERREIRA, A. C. P. de L. et al. Aprendizado de máquina: breve introdução e aplicações. *Cadernos de Ciência & Tecnologia*, v. 34, n. 3, p. 297–313, 2019. Citado na página 19.

GONZALEZ, V. L. S. d. L. M. *Recuperação de Informação e Processamento da Linguagem Natural*. 2003. Disponível em: <https://d1wqtxts1xzle7.cloudfront.net/43022678/minicurso-jaia2003-libre.pdf?1456349219=&response-content-disposition=inline%3B+filename%3DRecuperacao_de_Informacao_e_Processament.pdf&Expires=1717085943&Signature=PN1kHa2Wdc06QjEAw55~cDQnXu0CpTIkGQTCqYijTwMl3ayeNwk8l~GNniNN7xAj6SqV2uofLjK_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>. Citado na página 8.

Google Gemini. *Google Gemini Cookbook*. 2024. Disponível em: <https://github.com/google-gemini/cookbook>. Citado na página 26.

JACKSON, P.; MOULINIER, I. *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization*. John Benjamins Pub., 2007. (Natural language processing). ISBN 9789027249920. Disponível em: <https://doi.org/10.1075/nlp.5>. Citado na página 12.

JAMIL, J. T. d. R. N. G. L. *A era da informação: considerações sobre o desenvolvimento dastecnologias da Informação*. 2007. Disponível em: <https://periodicos.ufmg.br/index.php/pci/article/view/23309/18844>. Citado na página 8.

JURAFSKY, D.; MARTIN, J. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2. ed. [S.l.]: Prentice Hall, 2008. Citado na página 12.

LIMA, G. d. S. *Classificação de notícias digitais utilizando processamento de linguagem natural*. 29 f. Monografia (Trabalho de Conclusão de Curso) — Universidade Federal de Uberlândia, Monte Carmelo, 2023. Citado na página 24.

MOHAMMED, M. A. et al. Fake news classification using machine learning. *International Journal of Engineering Research & Technology (IJERT)*, Volume 11, Issue 03, 2023. Citado na página 23.

SAAVEDRA, J. D. O. Comparação de métodos de aprendizado de máquina para classificação automática de notícias em português. *Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação*, 2023. Citado na página 23.

SANTOS, F. A. et al. Processamento de linguagem natural em textos de mídias sociais: Fundamentos, ferramentas e aplicações. *Sociedade Brasileira de Computação*, 2022. Citado na página 11.

SANTOS, R. L. S.; MONTEIRO, R. A.; PARDO, T. A. S. *OpenCor2018: The Fake.Br corpus - a corpus of fake news for Brazilian Portuguese*. 2018. Disponível em: <https://sites.icmc.usp.br/taspardo/OpenCor2018-SantosEtAl.pdf>. Citado 3 vezes nas páginas 24, 25 e 26.

TEIXEIRA AMANDA DUARTE MARCOS, M. L. H. G. M. H. L. T. B. C. V. M. *AS FAKE NEWS E SUAS CONSEQUÊNCIAS NOCIVAS À SOCIEDADE*. Disponível em: <https://d1wqtxts1xzle7.cloudfront.net/97153149/407925573_As_Fake_News_e_Suas_Consequencias_Nocivas_a_Sociedade-libre.pdf?1673462754=&response-content-disposition=inline%3B+filename%3DAs_Fake_News_e_Suas_Consequencias_Nociva.pdf&Expires=1717086370&Signature=HjlLosbUP-ksk5yzU8FkjFL6-M-rQ3KTwltZLWY~FndsyA6aeGPTDaOOx8jBdJQlCLcXbAHcG2BM_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>. Citado na página 9.

VAJJALA, S. et al. *Practical natural language processing: a comprehensive guide to building real-world NLP systems*. Sebastopol, California, USA: O'Reilly Media, 2020. Citado 7 vezes nas páginas 3, 4, 11, 12, 14, 16 e 17.