UNIVERSIDADE FEDERAL DO AMAPÁ
CAMPUS MARCO ZERO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

UNIFAP

CLASSIFICADOR NAIVE BAYES MULTINOMIAL

MACAPÁ
2023

ALEX BARBOSA CORRÊA
JOSEPH SILVA DA CUNHA
KÁSSIO ANDREY GUIMARÃES TRINDADE

CLASSIFICADOR NAIVE BAYES MULTINOMIAL

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal do Amapá, em cumprimento às exigências legais como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.
Orientador: Prof. Dr. José Walter Cárdenas Sotil

MACAPÁ
2023

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Central/UNIFAP-Macapá-AP
Elaborado por Maria do Carmo Lima Marques – CRB-2 / 989

C823c Corrêa, Alex Barbosa.
Classificador Naive Bayes Multinomial. / Alex Barbosa Corrêa, Joseph Silva Da Cunha, Kássio Andrey Guimarães Trindade. Macapá: Unifap, 2023.
1 recurso eletrônico. 58 folhas.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Amapá, Bacharelado em Ciência da Computação, Macapá, 2022.
Orientador: Prof. Dr. José Walter Cárdenas Sotil.

Modo de acesso: World Wide Web.
Formato de arquivo: Portable Document Format (PDF).

1. Aprendizado de Máquina. 2. Naive Bayes. 3. Notícias Falsas. I. Da Cunha, Joseph Silva. II.Trindade, Kássio Andrey Guimarães. III. Sotil, José Walter Cárdenas, Orientador. IV. Universidade Federal do Amapá. V. Título.

CDD 23. ed. – 004

CORRÊA, Alex Barbosa; DA CUNHA, Joseph Silva; TRINDADE, Kássio Andrey Guimarães. **Classificador Naive Bayes Multinomial**. Orientador: Prof. Dr. José Walter Cárdenas Sotil. 58 f. Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Amapá, Macapá, 2023.

UNIVERSIDADE FEDERAL DO AMAPÁ
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

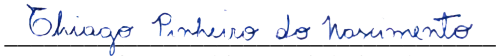
ATA DE DEFESA DE TCC

Realizou-se, no dia 02 de fevereiro de 2023, às 20h, via "Streamyard", a defesa do TCC intitulado: **"Classificador Naive Bayes Multinomial"**, dos discentes **Alex Barbosa Corrêa, Joseph Silva da Cunha e Kássio Andrey Guimarães Trindade**. A Banca Examinadora foi composta pelo Prof. Dr. José Walter Cárdenas Sotil, presidente da banca e orientadora; Prof. Me. Thiago Pinheiro do Nascimento e Prof. Me. Adeildo Telles da Silva, examinadores. Concluída a defesa, foram realizadas as arguições e comentários. Em seguida, procedeu-se o julgamento pelos membros da Banca Examinadora, tendo o trabalho sido APROVADO.

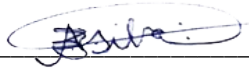
E, para constar, eu, Prof. Dr. José Walter Cárdenas Sotil, orientador e presidente da Banca Examinadora, lavrei a presente ata que, após lida e achada conforme, foi assinada por mim e demais membros da Banca Examinadora.

Macapá-Ap., 03 de fevereiro de 2023.

Prof. Dr. José Walter Cárdenas Sotil
Orientador



Prof. Me. Thiago Pinheiro do Nascimento Examinador (UNIFAP)



Prof. Me. Adeildo Telles da Silva
Examinador (UNIFAP)

AGRADECIMENTOS

Eu, Alex Barbosa Corrêa, agradeço à Risalva Barbosa Corrêa, minha mãe, e à Telma Medeiros Rabelo, minha segunda mãe, por tudo o que fizeram por mim e por serem essências em todas as minhas conquistas. Sou grato aos meus familiares e amigos por acreditarem e me apoiarem sempre. Gratidão, a todos os meus professores, em especial, ao José Walter Cárdenas Sotil que nos orientou brilhantemente e forneceu tudo o que foi necessário para a conclusão deste trabalho. A Bianca De Matos Ferreira, minha namorada, por todo o amor, apoio e companheirismo.

Eu, Joseph Silva da Cunha, gostaria de agradecer a minha família, principalmente meus pais, José Pereira da Cunha e Socorro das Graças Nogueira da Silva, que me deram todo o apoio necessário para os meus estudos. Agradeço ao meu namorado André Alonso Célis que sempre esteve comigo nas horas difíceis e me ajudou muito no desenvolvimento desse trabalho. Gostaria de agradecer imensamente ao meu orientador José Walter Cárdenas Sotil, pelo apoio, conhecimento, paciência e pela oportunidade de ter sido orientado por ele. Sou grato também aos meus amigos da faculdade que são alguns dos tesouros que esse curso me proporcionou.

Eu, Kassio Andrey Guimarães Trindade, gostaria de agradecer a toda equipe docente da universidade pelo ótimo trabalho realizado no decorrer do curso. Agradeço em especial ao nosso orientador José Walter Cárdenas Sotil por ter nos acompanhado até o fim e por toda a ajuda e conhecimento que recebemos. A todos aqueles que contribuíram, de alguma forma, para a realização e conclusão do trabalho de conclusão de curso.

RESUMO

Neste trabalho apresentamos a base teórica do classificador de aprendizado de máquina Naive Bayes Multinomial. Não foram usadas no desenvolvimento dos algoritmos do classificador Naive Bayes bibliotecas de aprendizado de máquina como o sklearn do Python, estes algoritmos foram implementados usando argumentos probabilísticos, a regra de Bayes e a hipótese de independência entre as variáveis. Isto permite implementar o classificador de Naive Bayes em qualquer linguagem de programação. Para testar o classificador usamos dados com notícias falsas e reais sobre a COVID-19 no Brasil, recopilados de postagens em redes sociais e os algoritmos foram implementados nas linguagens de programação Python e Javascript. Para testar a eficiência do classificador foram realizadas 100 simulações para detectar as notícias falsas, os resultados em ambas as linguagens de programação mostram uma acurácia em torno de 85%, precisão de 91%, sensibilidade de 80% e F1-score de 85%.

Palavras-chave: Aprendizado de Máquina; Naive Bayes; Notícias Falsas.

ABSTRACT

In this work, we present the theoretical basis of the Multinomial Naive Bayes machine learning classifier. Machine learning libraries such as Python's sklearn were not used in the development of the algorithms of the Naive Bayes classifier, these algorithms were implemented using probabilistic arguments, the Bayes rule, and the hypothesis of independence between the variables. This allows Naive Bayes classifier implementation in any programming language. To test the classifier, we used data with fake and real news about COVID-19 in Brazil, compiled from posts on social networks, and the algorithms were implemented in Python and JavaScript programming languages. To verify the efficiency of the classifier, 100 simulations were performed to detect fake news, the results in both programming languages shows an accuracy level around 85%, precision of 91%, sensitivity of 80%, and F1-score of 85%.

Keywords: Machine Learning; Naive Bayes; Fake News.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

O aprendizado de máquina é uma ramificação da Inteligência Artificial, na qual tem como intuito elaborar sistemas que sejam capazes tanto de aprender quanto de aperfeiçoar tarefas por meio da vivência de certas experiências (SOUZA, 2022). Desse modo, máquinas otimizam os processos sem a necessidade de intervenção humana agindo conforme a identificação de padrões. A aprendizagem de máquina pode ser utilizada em filtros de spam e respostas automáticas, recomendações de conteúdos e de produtos, detecção de fraudes e notícias falsas.

Com uma comunidade cada vez maior que usa métodos de aprendizado de máquina, são disponibilizadas por várias empresas de tecnologia bibliotecas prontas. Algumas das bibliotecas em Python são (GÉRON, 2022):

TensorFlow. Uma biblioteca de aprendizado de máquina criada pelo Google, que oferece suporte a redes neurais profundas e computação em larga escala;

Scikit-learn. Uma biblioteca simples e fácil de usar para aprendizado de máquina, que inclui uma ampla gama de algoritmos de classificação, regressão e agrupamento;

PyTorch. Uma biblioteca de aprendizado de máquina desenvolvida pelo Facebook, que é especialmente útil para tarefas relacionadas a visão computacional e processamento de linguagem natural;

Keras. Uma biblioteca de aprendizado de máquina de alto nível, que permite a criação de modelos de redes neurais complexos com poucas linhas de código;

XGBoost. Uma biblioteca de aprendizado de máquina de alta performance, especialmente projetada para acelerar o treinamento de modelos de árvore de decisão.

O JavaScript tem uma boa oferta de bibliotecas de aprendizado de máquina. Existem bibliotecas para vários cenários com diferentes pontos fortes (KOPRA, 2019):

Brain.js. É uma biblioteca de código aberto de redes neurais escritas em JavaScript. Pode ser usado com Node.js ou no navegador e se anuncia como simples e fácil de usar;

Synaptic. é outra biblioteca de rede neural escrita em JavaScript para node.js e navegador. Ele se diferencia por ser livre de arquitetura, portanto, há liberdade para os desenvolvedores escolherem o tipo de arquitetura de rede neural que desejam construir;

ConvNetJS. é uma biblioteca JavaScript para redes neurais. O que torna essa biblioteca única é que ela é usado inteiramente no navegador, ou seja, não há necessidade de instalar nada ou ter um computador potente para executar o treinamento do modelo.

No Java, algumas das bibliotecas de aprendizado de máquina mais populares incluem (BHATIA; KALUZA, 2018):

WEKA. Uma biblioteca de aprendizado de máquina de código aberto que fornece uma ampla gama de algoritmos de aprendizado de máquina, incluindo classificação, regressão, agrupamento e redução de dimensionalidade;

DeepLearning4j. Uma biblioteca de aprendizado de máquina de código aberto, especialmente projetada para deep learning em Java. É compatível com o TensorFlow e o Theano e permite a criação de modelos de rede neural profunda em larga escala;

MOA. Uma biblioteca de aprendizado de máquina de código aberto que fornece uma ampla gama de algoritmos de aprendizado de máquina on-line e off-line, incluindo classificação, regressão, agrupamento, detecção de anomalias e outros.

A linguagem de programação R é uma das mais populares para a análise de dados e aprendizado de máquina, e existem muitas bibliotecas prontas disponíveis para isso, entre elas (GARETH et al., 2013):

caret. Uma biblioteca para construir e avaliar modelos de aprendizado de máquina em R. É uma das bibliotecas mais amplas e completas disponíveis para aprendizado de máquina em R, cobrindo técnicas de classificação, regressão, agrupamento e muito mais;

randomForest. Uma biblioteca de aprendizado de máquina de floresta aleatória que fornece uma implementação eficiente do algoritmo random forest para classificação e regressão;

gbm. Uma biblioteca de gradient boosting que implementa o algoritmo gradient boosting para classificação e regressão;

xgboost. Uma biblioteca de gradient boosting que implementa o algoritmo XGBoost para classificação e regressão.

Para a maior parte dos usuários estas bibliotecas são caixas pretas, entretanto práticas pois com poucas linhas de código pode-se treinar o algoritmo. Desenvolver novos algoritmos requer o conhecimento teórico da técnica ou método proposto, em particular neste trabalho desenvolvemos o classificador Naives Bayes Multinomial sem depender de bibliotecas prontas.

Testamos o classificador Naive Bayes Multinomial na detecção de notícias falsas sobre o covid 19 no Brasil. As eleições americanas de 2016 e o referendo para a saída do Reino Unido da União Europeia (Brexit) demonstraram a existência de campanhas organizadas de desinformação na internet e suas consequências (POSETTI; MATTHEWS, 2018). No Brasil, a pandemia da COVID-19 mostrou que as mesmas técnicas de desinformação podem ser aplicadas independentemente de linguagem ou contexto (JÚNIOR et al., 2020).

A checagem de fatos realizada por veículos tradicionais de informação mostrou-se lenta e ineficiente demais para lidar com a velocidade e o volume de informações produzidas pelas plataformas de rede social (HASSAN et al., 2015). A detecção automática de notícias falsas por meio de aprendizado de máquina tem sido capazes de detectar, a partir das palavras e do contexto utilizado nas notícias, a veracidade do conteúdo (BRASIL et al., 2021).

1.1 OBJETIVOS

Este trabalho tem como objetivo principal apresentar os fundamentos teóricos e práticos do classificador de aprendizado de máquina Naive Bayes Multinomial de modo que o classificador possa ser implementado em qualquer linguagem de programação sem recorrer a bibliotecas de aprendizado de máquina como o sklearn do Python. O classificador é testado na detecção de notícias falsas sobre a COVID-19 no Brasil. Para alcançar esse objetivo, se faz necessário:

- Apresentar conceitos probabilísticos, regra de bayes e hipotese de independência;
- Desenvolver todas as etapas e algoritmos do classificador de aprendizado de máquina Naive Bayes multinomial;
- Implementar estes algoritmos do classificador Naive Bayes multinomial nas linguagens de programação Python e JavaScript;
- Elaborar uma base de dados com postagens sobre o Covid-19 no Brasil;
- Aplicar os dados nos algoritmos do classificador Nive Bayes multinomial, obtendo um detetor de noticias falsas nas linguagens Python e JavaScript;
- Validar o sistema por meio de testes e expor os resultados usando métricas de eficiência.

Estas são apenas algumas das muitas bibliotecas disponíveis em Python, JavaScript, Java e R para aprendizado de máquina. A escolha da biblioteca certa depende das necessidades específicas e do tipo de problema que deseja-se-resolver. Os algoritmos do classificador Naive Bayes Multinomial, desenvolvidos neste trabalho, podem ser implementados em qualquer linguagem de programação, sem necessidade de recorrer a bibliotecas prontas de aprendizado de máquina.

1.2 TRABALHOS RELACIONADOS

Existem vários trabalhos com foco no classificador Naive Bayes. Destaca-se o estudo de LEAL (2018) em que comparou o desempenho do classificador Naive Bayes com outros algoritmos como J48, JRIP e o IBK para a identificação de Fake News referente as eleições do Brasil em 2018. Para alcançar seu objetivo, fez um compilado de postagens da rede social Twitter e utilizou a linguagem de programação JavaScript em conjunto com a biblioteca WEKA para

testar os algoritmos. O resultado superou as expectativas e 3 dos 4 algoritmos testados, inclusive o algoritmo Naive Bayes, obtiveram acurácia acima de 90%.

Outro trabalho em destaque é a pesquisa de DIAS (2018), O autor comparou os algoritmos Naive Bayes e Rede de Crenças (K2), utilizando a ferramenta WEKA, na predição de período chuvoso em Blumenau. Como resultado, a predição realizada pelo algoritmo Naive Bayes obteve 85,36% de acuraria. Outros trabalhos no Brasil com o algoritmo Naive Bayes incluem: Classificação de documentos em domínios específicos; Análise de sentimentos; Previsão de risco de crédito; Detecção de spam em e-mails utilizando o algoritmo Naive Bayes.

Algo em comum nesses trabalhos é a utilização de bibliotecas prontas para implementar o classificador. Diferentemente dos trabalhos citados, esta pesquisa não utiliza bibliotecas prontas como WEKA e Scikit-Learn para a construção do classificador. Entende-se que ao desenvolver totalmente o classificador, desde a base de dados até o código, seguindo as instruções do algoritmo, o trabalho se torna mais didático, unindo a prática com a teoria, assim podendo ter uma explicação mais precisa de cada passo.

1.3 ORGANIZAÇÃO DO TRABALHO

A fim de organizar o trabalho de forma a facilitar o entendimento do tema abordado, o conteúdo presente foi dividido nos seguintes tópicos:

- Capítulo 2: Conceitos Probabilísticos. Apresenta os conceitos fundamentais para o entendimento deste trabalho. São abordadas definições e propriedades de conceitos probabilísticos necessários para o entendimento da Regra de Bayes. Por fim, os conceitos apresentados são aplicados em um exemplo de detecção de Spam.
- Capítulo 3: Classificador Naive Bayes. É apresentado conceitos de aprendizado de máquina e explicado o funcionamento do algoritmo Naive Bayes com exemplo de código implementado na linguagem de programação Python.
- Capítulo 4: Classificador Naive Bayes na Detecção de Fake News. É demonstrado o desenvolvimento de uma aplicação utilizando a linguagem de programação Python para testar o classificador Naive Bayes na detecção de Fake News relacionadas à pandemia do Covid-19. A implementação é baseada nos conceitos explicados nos capítulo 2 e 3 e não utiliza bibliotecas prontas.
- Capítulo 5: Implementação Alternativa em JavaScript. É apresentado uma implementação alternativa da aplicação utilizando a linguagem de programação JavaScript.
- Capítulo 6: São apresentadas as considerações finais do trabalho.

2 CONCEITOS PROBABILISTICOS

O conceito de **probabilidade** é fundamental na área de aprendizado de máquina porquê a mesma lida com incerteza (HÜLLERMEIER; WAEGEMAN, 2021). São necessárias métricas definidas com base nas probabilidades do problema escolhido, tanto em modelos de classificação como em modelos de regressão para alcançar bons resultados. Em geral, para criar modelos de aprendizado de máquina é necessária uma ampla variedade de técnicas estatísticas (SUGIYAMA, 2015). O classificador Naive Bayes tem como base a Regra de Bayes e uma hipótese de independência condicional. Apesar da hipótese não ser realista o classificador apresenta resultados satisfatórios em prática (GRANIK; MESYURA, 2017).

Definimos o conceito de probabilidade no caso finito e a definição axiomática que permite estender o conceito para os casos discreto e contínuo. Apresentamos as propriedades de probabilidade, probabilidade condicional, eventos independentes, probabilidade de Bayes, Regra de Bayes e finalizamos com um exemplo de detecção de spam (HASLWANTER, 2016).

2.1 PROBABILIDADE

Em aprendizado de máquina trabalhamos com experimentos aleatórios, que diferem de experimentos determinísticos. Se repetimos nas mesmas condições um experimento determinístico o resultado será sempre o mesmo, por exemplo ao deixar cair varias vezes uma pedra de uma altura de 2 metros o tempo da queda é o mesmo para cada um dos lançamentos.

No experimento aleatório, mesmo que se conheça todos os resultados possíveis, não há a certeza de qual será o resultado ao repetir o experimento. O lançamento de uma moeda é um experimento aleatório, pois conhecemos que o resultado deve ser cara ou coroa, mas não temos como prever qual será o resultado do próximo lançamento (FONSECA; MARTINS, 1996).

Um **espaço amostral** Ω é o conjunto formado por todos os resultados possíveis de um experimento aleatório. Por exemplo, no lançamento de uma moeda, o espaço amostral é $\Omega = \{\text{cara, coroa}\}$. Qualquer subconjunto Ω do espaço amostral é denominado um **evento**, no exemplo do lançamento de uma moeda temos os eventos $E_1 = \emptyset$, $E_2 = \{\text{cara}\}$, $E_3 = \{\text{coroa}\}$ e $E_4 = \Omega$. Assim, sempre o conjunto nulo e o espaço amostral são eventos de Ω. No caso do espaço amostral ser finito de n elementos, tem-se 2^n eventos.

É mais simples trabalhar cálculo de probabilidades se os elementos do espaço amostral são números, em casos diferentes, se introduz o conceito de variável aleatória. Uma **variável aleatória** é uma função $X : \Omega \to R$ que leva cada evento do espaço amostral a um número real. Por exemplo, no caso do lançamento de uma moeda, definimos $X(\text{cara}) = 0$ e $X(\text{coroa}) = 1$.

Um espaço amostral Ω finito com n elementos é dito igualmente provável se cada

elemento tem a mesma chance $\frac{1}{n}$ de ser escolhido. Por exemplo, no lançamento de uma moeda honesta, a chance do resultado ser cara é $\frac{1}{2}$ e a chance de ser coroa é também $\frac{1}{2}$. Definimos agora a probabilidade para eventos igualmente prováveis.

Definição 2.1 (Caso finito equiprovável)**.** *Se Ω é um espaço amostral finito de resultados igualmente prováveis e E é um evento, então a probabilidade de E é:*

$$p(E) = \frac{n(E)}{n(\Omega)} \tag{2.1}$$

onde,

- $n(E)$*: número de elementos do evento E, e*
- $n(\Omega)$*: número de elementos do espaço amostral Ω.*

Usualmente, esta probabilidade na equação (2.1) é interpretada como:

$$p(E) = \frac{\text{número de resultados favoráveis}}{\text{número de resultados possíveis}}.$$

Exemplo 2.1. *Qual é a probabilidade, quando dois dados honestos são jogados, da soma dos números ser igual a 7?*

Solução Se os dados são honestos todos os eventos individuais tem a mesma probabilidade de serem escolhidos.

Os casos favoráveis onde a soma dos dados somam 7 são: $E = \{(1,6), (2,5), (3,4), (4,3), (5,2), (6,1)\}$.

Logo, o número de casos favoráveis são: $n(E) = 6$.

O número de casos totais são: $n(\Omega) = 6 \times 6 = 36$.

A probabilidade calculada pela Equação (2.1) é: $p(E) = \frac{n(E)}{n(\Omega)} = \frac{6}{36} = \frac{1}{6}$. ■

Se verifica que se Ω é um espaço amostral finito igualmente provável, então $p(\Omega) = 1$. Com efeito:

$$p(\Omega) = \frac{\text{número de resultados favoráveis}}{\text{número de resultados possíveis}} = \frac{n(\Omega)}{n(\Omega)} = 1.$$

E^c é o evento complementar do evento E se $E \cup E^c = \Omega$. A probabilidade complementar é calculada como:

Propriedade 2.1 (Probabilidade complementar)**.** *Seja E um evento do espaço amostral Ω. A probabilidade do evento complementar E^c, é:*

$$P(E^c) = 1 - P(E). \tag{2.2}$$

Demonstração. Como $E \cup E^c = \Omega$ e $E \cap E^c = \emptyset$, então

$$n(E^c) = n(\Omega) - n(E)$$

$$P(E^c) = \frac{n(E^c)}{n(\Omega)} = \frac{n(\Omega) - n(E)}{n(\Omega)} = 1 - \frac{n(E)}{n(\Omega)} = 1 - P(E). \blacksquare$$

Exemplo 2.2. *Uma sequência de 10 bits é construída aleatoriamente. Qual é a probabilidade de que pelo menos um desses bits ser 0?*

Solução.

Seja E: a sequência têm pelo menos um bit 0.

E^c: a sequência não tem nenhum bit 0 = a sequência têm todos os bits 1s.

Temos um único caso favorável 1111111111, logo $n(E^c) = 1$.

Cada posição tem 2 possibilidades: 0 ou 1. Logo, $n(\Omega) = 2^{10}$.

Da Equação (2.2), temos $P(E) = 1 - P(E^c) = 1 - \frac{n(E^c)}{n(\Omega)} = 1 - \frac{1}{2^{10}} = 1 - \frac{1}{1024} = \frac{1023}{1024}$. $\blacksquare$

A probabilidade da união de eventos é calculada usando o principio da contagem (ROSEN, 2019):

$$n(E_1 \cup E_2) = n(E_1) + n(E_2) - n(E_1 \cap |E_2).$$

Propriedade 2.2 (Probabilidade da união)**.** *Considere E_1 e E_2 eventos do espaço amostral Ω. Então,*

$$p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2) \tag{2.3}$$

Demostração

$$P(E_1 \cup E_2) = \frac{n(E_1 \cup E_2)}{n(\Omega)}$$

$$= \frac{n(E_1) + n(E_2) - n(E_1 \cap E_2)}{n(\Omega)}$$

$$= \frac{n(E_1)}{n(\Omega)} + \frac{n(E_2)}{n(\Omega)} - \frac{n(E_1 \cap E_2)}{n(\Omega)}$$

$$= p(E_1) + p(E_2) - p(E_1 \cap E_2). \blacksquare$$

Exemplo 2.3. *Qual é a probabilidade de um número inteiro positivo selecionado aleatoriamente a partir do conjunto dos números inteiros positivos não maior a 100 ser divisível por 2 ou 5?*

Solução

Se E_1 são os números divisíveis por 2, então $E_1 = \{2\cdot 1,\ 2\cdot 2, \cdots, 2\cdot 50\}$ e portanto $n(E_1) = 50$.

Se E_2 são os números divisíveis por 5, então $E_2 = \{5\cdot 1,\ 5\cdot 2, \cdots, 5\cdot 20\}$ e portanto $n(E_2) = 20$.

$E_1 \cap E_2$ são os números divisíveis por 10, então $E_1 \cap E_2 = \{10\cdot 1,\ 10\cdot 2, \cdots, 10\cdot 10\}$, portanto $n(E_1 \cap E_2) = 10$.

Como $n(\Omega) = 100$, resulta da Equação (2.3),

$$P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2) = \frac{50}{100} + \frac{20}{100} - \frac{10}{100} = 0,6. \ \blacksquare$$

A Equação (2.1) calcula a probabilidade para espaços amostrais finitos equiprováveis, para o caso geral a probabilidade é definida de maneira axiomática (FONSECA; MARTINS, 1996).

Definição 2.2 (Probabilidade Axiomática)**.** *Seja Ω um espaço amostral. A probabilidade p é uma função $p : \Omega \to R$ que verifica os três axiomas:*

1. $p(E) \geq 0$*, para qualquer evento* E.
2. $p(\Omega) = 1$.
3. $p(E_1 \cup E_2) = p(E_1) + p(E_2), \quad$ *se* $E_1 \cap E_2 = \emptyset$.

Esta definição generaliza o conceito de probabilidade para conjuntos finitos igualmente prováveis. A definição é válida se o espaço amostral é discreto (finito ou infinito enumerável) ou contínuo (intervalo ou coleção de intervalos de números reais).

Propriedade 2.3. *Se E e F são eventos, se verificam as seguintes propriedades:*

1. $p(E^c) = 1 - p(E)$.
2. $p(\emptyset) = 0$.
3. $p(E \cup F) = p(E) + p(F) - p(E \cap F)$.
4. *Se* E_1, E_2, $\cdots$ *for uma sequência de eventos disjuntos em um espaço amostral S, então*

$$p\left(\bigcup_i E_i\right) = \sum_i p(E_i)$$

Exemplo 2.4. *Quais as probabilidades devemos determinar para os resultados cara (C) ou coroa (K) quando uma moeda é viciada para que as caras saiam duas vezes mais que as coroas?*

Solução. Seja $\Omega = \{c,k\}$ o espaço amostral. Como a moeda é viciada, e as caras devem sair duas vezes mais que as coroas, temos

$$p(\{C\}) = 2p(\{K\})$$

logo,

$$\begin{aligned} 1 = p(\Omega) &= p(\{C\}) + p(\{K\}) \\ &= 2p(\{K\}) + p(\{K\}) \\ &= 3p(\{K\}). \end{aligned}$$

portanto, $p(\{K\}) = \dfrac{1}{3}$ e $p(\{C\}) = \dfrac{2}{3}$. ■

2.2 PROBABILIDADE CONDICIONAL

Um conceito importante no classificador Naive Bayes é a probabilidade de um evento E dado que já se conhece que ocorreu, o evento F. Em decorrência disso, se restringe o espaço amostral de Ω para F.

Definição 2.3 (Probabilidade Condicional)**.** *Sejam E e F como eventos de um espaço amostral Ω. com $P(F) > 0$. A Probabilidade Condicional de E dado F, denotada por $p(E|F)$, é definida como*

$$p(E|F) = \frac{p(E \cap F)}{p(F)} \tag{2.4}$$

Se o espaço amostral Ω é finito com elementos igualmente prováveis, temos

$$p(E|F) = \frac{\dfrac{n(E \cap F)}{n(\Omega)}}{\dfrac{n(F)}{n(\Omega)}} = \frac{n(E \cap F)}{n(F)}. \tag{2.5}$$

Exemplo 2.5. *Qual é a probabilidade que uma cadeia de bits de extensão quatro, tenha pelo menos dois zeros consecutivos, dado que o primeiro bit é zero? (assumimos que os bits 0 e 1 são igualmente prováveis).*

Solução

- Ω: cadeia de 4 bits = ????
- $n(\Omega) = 2 \cdot 2 \cdot 2 \cdot 2 = 16$

- F: cadeias com primeiro bit igual a zero = 0???
- $n(F) = n(0???) = 1 \cdot 2 \cdot 2 \cdot 2 = 8$
- E: cadeia de bits que tem pelo menos dois zeros consecutivos
- $E \cap F$: cadeia com o primeiro bit igual a zero e que tem pelo menos dois zeros consecutivos
- $n(E \cap F) = n(00??) + n(0100) = 1 \cdot 1 \cdot 2 \cdot 2 + 1 \cdot 1 \cdot 1 \cdot 1 = 4 + 1 = 5$

$$p(E|F) = \frac{n(E \cap F)}{n(F)} = \frac{5}{8}. \blacksquare$$

O cálculo da probabilidade condicional na Equação (2.4), permite calcular a probabilidade da interseção de dois eventos em função das probabilidades condicionais.

Propriedade 2.4. *Se E e F são dois eventos de um espaço amostral Ω, então*

$$p(E \cap F) = p(E|F) \cdot p(F), \quad e \tag{2.6}$$

$$p(E \cap F) = p(F|E) \cdot p(E). \tag{2.7}$$

2.3 EVENTOS INDEPENDENTES

No caso da ocorrência de um evento não afetar a probabilidade de outro evento, define-se o conceito de eventos independentes.

Definição 2.4 (Eventos Independentes)**.** *Dois eventos são independentes se conhecida a probabilidade de um evento, este não fornece informação sobre a probabilidade do outro evento. Isto é, os eventos E e F são independentes se*

$$p(E|F) = p(E). \tag{2.8}$$

Se dois eventos não são independentes, dizemos que eles são eventos dependentes

Exemplo 2.6. *No lançamento de duas moedas:*

- *na primeira moeda: p(c)=p(k)=1/2*
- *na segunda moeda: p(c)=p(k)=1/2*

logo, as probabilidades para a segunda moeda não são alteradas se conhecemos as probabilidades da primeira moeda. Portanto, o lançamento das moedas são eventos independentes.

Propriedade 2.5. *Se E e F são dois eventos independentes, então:*

i) $p(E|F) = p(E)$, *e*

ii) $p(E \cap F) = p(E) \cdot p(F)$.

Exemplo 2.7. *Qual é a probabilidade de que no lançamento de duas moedas o resultado seja obter duas caras?*

Solução

- Espaço amostral: S=(c,c), (c,k), (k,c), (k,k).
- $p(\text{E= 1º lançamento cara}) = p((c,c),\ (c,k)) = \dfrac{2}{4}$.
- $p(\text{F =2º lançamento cara}) = p((c,c),\ (k,c)) = \dfrac{2}{4}$.
- Probabilidade de obter duas caras: $p(E \cap F = (c,c)) = \dfrac{1}{4}$■.

Nota: como $p(E \cap F) = p(E) \cdot p(F)$, os eventos E e F são independentes.

Exemplo 2.8. *Em uma pesquisa realizada pelo IBGE em 2020, a proporção de pessoas obesas com 20 anos ou mais foi de 26,8% no Brasil. Assumindo que a proporção se mantenha constante, qual é a probabilidade que duas pessoas com 20 ou mais anos, escolhidas aleatoriamente no Brasil, sejam ambas obesas?*

Solução

- Se uma pessoa é obesa, existe a possibilidade que outra pessoa que more na mesma casa, com a mesma rotina e hábitos alimentares também seja obesa.
- Mas como as pessoas são escolhidas aleatoriamente, a probabilidade de escolher duas pessoas da mesma casa é bastante pequena. Logo, podemos considerar que a escolha das duas pessoas são eventos independentes.

$$p(\textit{ambas obesas}) = p(1^a \textit{obesa}) \times p(2^a \textit{obesa}) = 0,268 \times 0,268 = 0,072 = 7,2\%. \blacksquare$$

O conceito de independência para mais de dois eventos se generaliza pela seguinte propriedade (ROSEN, 2019):

Propriedade 2.6. *Se os eventos $E_1,\ E_2,\ \cdots, E_n$ são n eventos independentes, então*

$$p(E_1 \cap E_2 \cap \cdots \cap E_n) = p(E_1) \cdot p(E_2) \cdot \cdots \cdot p(E_n). \tag{2.9}$$

Exemplo 2.9. *Um número binário é formado por n dígitos. Se a probabilidade que apareça um dígito incorreto é p e sabendo que os erros em dígitos diferentes são independentes um do outro, qual é a probabilidade de obter um número incorreto?*

Solução

- p: probabilidade de um dígito ser incorreto.
- $q = 1 - p$: probabilidade de um dígito ser correto.
- E: "obter um número incorreto".
- E^c: obter um número correto", isto só acontece se todos os n dígitos são corretos.
- Da Equação (2.9): $p(E^c) = q \cdot q \cdot \cdots \cdot q = q^n = (1-p)^n$.
- $p(E) = 1 - p(E^c) = 1 - (1-p)^n$. ■

2.4 PROBABILIDADE TOTAL

Definimos a partição de um espaço amostral e o cálculo da probabilidade de um evento com base numa partição do espaço amostral.

Definição 2.5 (Partição)**.** *Consideremos os eventos E_i, $i = 1, \cdots, n$, uma partição do espaço amostral Ω, isto é, uma coleção de eventos que satisfazem as propriedades:*

1. $E_i \cap E_j = \emptyset, \quad se \quad i \neq j$
2. $E_1 \cup E_2 \cup \cdots \cup E_n = \Omega$

Definição 2.6. *Se $\bigcup_{i=1}^{n} E_i$ é uma partição do espaço amostral Ω e A um evento, a Probabilidade Total do evento A é a soma das probabilidades de A interceptados com os elementos da partição (ver Figura 2.1), isto é:*

$$P(A) = \sum_{i=1}^{n} P(A \cap E_i). \tag{2.10}$$

ou, em termos da probabilidade condicional,

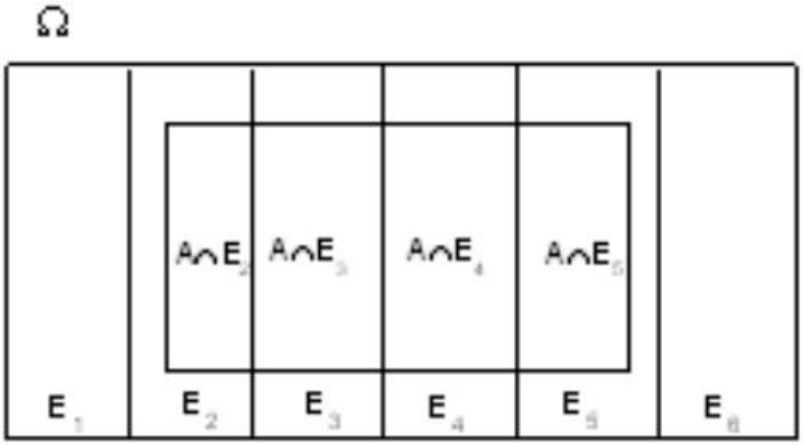
$$P(A) = \sum_{i=1}^{n} P(A|E_i) \cdot P(E_i). \tag{2.11}$$

Exemplo 2.10. *Admita a seguinte configuração:*

	urna1	urna2	urna3
amarela	3	4	2
branca	1	3	3
vermelha	5	2	3

Escolheu-se uma urna ao acaso e dela extraiu-se uma bola ao acaso. Admitindo-se que as urnas tem a mesma probabilidade de serem escolhidas, qual é a probabilidade

Figura 2.1 – Probabilidade Total



Fonte: Elaborado pelos Autores.

a) *da bola ser amarela?*

b) *da bola ser branca?*

c) *da bola ser vermelha?*

Solução

a) da bola ser amarela?

A bola amarela pode ser proveniente da urna 1, da urna 2 ou urna 3 as quais formam uma partição do espaço amostral. Logo,

$$p(A) = p(A \cap u_1) + p(A \cap u_2) + p(A \cap u_3)$$
$$p(A) = p(A|u_1) \cdot p(u_1) + p(A|u_2) \cdot p(u_2) + p(A|u_3) \cdot p(u_3).$$

da tabela temos que:

- u_1: urna 1, u_2: urna 2, u_3: urna 3
- - A: bola amarela, B: bola branca, V: bola vermelha
- $\Omega = \{u_1, u_2, u_3\}$
- $p(u_1) = p(u_2) = p(u_3) = \frac{1}{3}$
- $p(A|u_1) = \frac{3}{9}, \quad p(A|u_2) = \frac{4}{9}, \quad p(A|u_3) = \frac{2}{8}$
- $p(B|u_1) = \frac{1}{9}, \quad p(B|u_2) = \frac{3}{9}, \quad p(B|u_3) = \frac{3}{8}$
- $p(V|u_1) = \frac{5}{9}, \quad p(V|u_2) = \frac{2}{9}, \quad p(V|u_3) = \frac{3}{8}$

logo,

$$p(A) = p(A \cap u_1) + p(A \cap u_2) + p(A \cap u_3)$$

$$p(A) = p(A|u_1) \cdot p(u_1) + p(A|u_2) \cdot p(u_2) + p(A|u_3) \cdot p(u_3)$$

$$p(A) = \frac{3}{9} \cdot \frac{1}{3} + \frac{4}{9} \cdot \frac{1}{3} + \frac{2}{8} \cdot \frac{1}{3} = \frac{37}{108}$$

logo.

$$p(A) = p(A \cap u_1) + p(A \cap u_2) + p(A \cap u_3)$$

$$p(A) = p(A|u_1) \cdot p(u_1) + p(A|u_2) \cdot p(u_2) + p(A|u_3) \cdot p(u_3)$$

$$p(A) = \frac{3}{9} \cdot \frac{1}{3} + \frac{4}{9} \cdot \frac{1}{3} + \frac{2}{8} \cdot \frac{1}{3} = \frac{37}{108}.$$

b) da bola ser branca?

$$p(B) = p(B \cap u_1) + p(B \cap u_2) + p(B \cap u_3)$$

$$p(B) = p(B|u_1) \cdot p(u_1) + p(B|u_2) \cdot p(u_2) + p(B|u_3) \cdot p(u_3)$$

$$p(B) = \frac{1}{9} \cdot \frac{1}{3} + \frac{3}{9} \cdot \frac{1}{3} + \frac{3}{8} \cdot \frac{1}{3} = \frac{59}{216}.$$

c) da bola ser vermelha?

$$p(V) = p(V \cap u_1) + p(V \cap u_2) + p(V \cap u_3)$$

$$p(V) = p(V|u_1) \cdot p(u_1) + p(V|u_2) \cdot p(u_2) + p(V|u_3) \cdot p(u_3)$$

$$p(V) = \frac{5}{9} \cdot \frac{1}{3} + \frac{2}{9} \cdot \frac{1}{3} + \frac{3}{8} \cdot \frac{1}{3} = \frac{83}{216}.$$

Observação

- Conhecemos dos dados a probabilidade condicional: $p(A|u_1) = 3/9$: a probabilidade da bola ser amarela dado que a urna escolhida é a urna 1.
- Como calcular: $p(u_1|A)$: a probabilidade da bola ser da urna 1 dado que ela é amarela?
- A resposta é dada pela Regra de Bayes.

2.5 REGRA DE BAYES

Consideremos a partição $\bigcup_{i=1}^{n} E_i$ do espaço amostral Ω, e A um outro evento. Da Lei Multiplicativa temos,

$$\begin{aligned} P(A \cap E_i) &= P(A|E_i) \cdot P(E_i) \\ P(E_i \cap A) &= P(E_i|A) \cdot P(A) \end{aligned}$$

igualando os lados direitos temos,

$$P(A|E_i) \cdot P(E_i) = P(E_i|A) \cdot P(A)$$

obtendo a Regra de Bayes:

$$p(E_i|A) = \frac{p(A|E_i) \cdot p(E_i)}{p(A)} \tag{2.12}$$

da lei da probabilidade total, a Regra de Bayes pode-se escrever como:

$$p(E_i|A) = \frac{P(A|E_i) \cdot P(E_i)}{\sum_{j=1}^{n} [P(A|E_j) \cdot P(E_j)]} \tag{2.13}$$

Exemplo 2.11. *Quando um teste para esteroides é aplicado em jogadores de futebol, 98% dos jogadores que tomam esteroides têm teste positivo e 12% dos jogadores que não tomam esteroides têm teste positivo. Suponha que 5% dos jogadores tomem esteroides. Qual é a probabilidade de um jogador que tenha teste positivo realmente tomar esteroides?*

Solução

- E: jogador toma esteroide
- E^c: jogador não toma esteroide
- $+$: teste positivo
- -$p(+|E) = 0,98$
- $p(+|E^c) = 0,12$
- $p(E) = 0,05$
- $p(E^c) = 0,95$

$$p(E|+) = \frac{p(+|E) \cdot p(E)}{p(+)} = \frac{p(+|E) \cdot p(E)}{p(+|E) \cdot p(E) + p(+|E^c) \cdot p(E^c)}$$

$$p(E|+) = \frac{0,98 \cdot 0,05}{0,98 \cdot 0,05 + 0,12 \cdot 0,95} \approx 0,30.$$

Exemplo 2.12. *Suponha que a palavra Rolex aparece em 250 das 2000 mensagens que são spam e em 5 das 1000 mensagens que não são spam.*
Faça a estimativa da probabilidade de uma mensagem recebida que contém a palavra Rolex ser spam, assumindo que é igualmente provável uma mensagem recebida ser ou não spam.
Se nosso percentual mínimo para rejeitar a mensagem como spam é de 0,9, você rejeitará essa mensagem?

Solução

- S: a mensagem é spam
- NS: a a mensagem não é spam
- $\Omega = \{S,\ NS\}$
- $p(S) = p(NS) = 1/2$
- $p(rolex|S) = \dfrac{250}{2000} = \dfrac{1}{8}$
- $p(rolex|NS) = \dfrac{5}{1000} = \dfrac{1}{200}$

$$p(S|rolex) = \frac{p(rolex|S)\cdot p(S)}{p(rolex)}$$

$$= \frac{p(rolex|S)\cdot p(S)}{p(rolex|S)\cdot p(S) + p(rolex|NS)\cdot p(NS)}$$

$$= \frac{\frac{1}{8}\cdot\frac{1}{2}}{\frac{1}{8}\cdot\frac{1}{2} + \frac{1}{200}\cdot\frac{1}{2}} = \frac{25}{26} \approx 0,9615$$

Como $p(S|rolex) = 0,9615 > 0,9$, rejeitamos esta mensagem como spam.

2.6 DETECÇÃO DE SPAM

Todos os conceitos de probabilidade definidos neste capítulo são aplicados na detecção de spam, principalmente a Regra de Bayes e da independência condicional. O seguinte exemplo de detecção de spam tem os fundamentos de um algoritmo de classificação de aprendizado de máquina.

Recebemos mensagens e desejamos saber o que pode ser classificado como spam ou mensagem normal. Considere 4 mensagens spam e 8 mensagens normais. Nas mensagens, aparecem diversas palavras com a seguinte frequência:

Para decidir se a mensagem "Prezado amigo"é normal ou spam adotamos os seguinte passos:

	prezado	amigo	almoço	dinheiro
spam	2	1	0	4
normal	8	5	3	1

- Observa-se na tabela um caso de probabilidade nula: Na linha spam a palavra *almoço*, não aparece. Como a probabilidade é nula, se a regra de Bayes for aplicada sem nenhum tipo de mecanismo corretivo, mensagens incluindo a palavra almoço nunca serão classificadas como spam.

  Existem várias técnicas para corrir este problema, uma delas é a suavização por adição, também conhecida como Suavização de Laplace (YUAN; CONG; THALMANN, 2012). O método consiste em somar uma constante ao número de ocorrências em todos os elementos da tabela. O valor padrão escolhido na maioria dos casos é 1.

	prezado	amigo	almoço	dinheiro
spam	3	2	1	5
normal	9	6	4	2

- Sejam
  - $p_1 = p(\text{spam} \mid [\text{prezado e amigo}])$, e
  - $p_2 = p(\text{normal} \mid [\text{prezado e amigo}])$

  Decisão:
  - se $p_1 > p_2$ a mensagem é classificada como spam
  - se $p_1 < p_2$ a mensagem é classificada como normal
- Usando Bayes:

$$p_1 = p(\text{spam} \mid [\text{prezado e amigo}]) = \frac{p([\text{prezado e amigo}] \cap \text{spam})}{p([\text{prezado e amigo}])}$$

$$p_2 = p(\text{normal} \mid [\text{prezado e amigo}]) = \frac{p([\text{prezado e amigo}] \cap \text{normal})}{p([\text{prezado e amigo}])}$$

- como p_1 e p_2 tem o mesmo denominador, para efeito de comparação consideremos:

$$p_1 = p([\text{prezado e amigo}] \cap \text{spam}) = p([\text{prezado e amigo}] \mid \text{spam}) \cdot p(\text{spam}), \text{ e}$$

$$p_2 = p([\text{prezado e amigo}] \cap \text{normal}) = p([\text{prezado e amigo}] \mid \text{normal}) \cdot p(\text{normal})$$

- Supondo que as palavras 'prezado' e 'amigo' sejam independentes (uma suposição bastante restritiva e que não necessariamente é verdadeira), temos que:

$$p_1 = p([\text{prezado}] \mid \text{spam}) \cdot p([\text{amigo}] \mid \text{spam}) \cdot p(\text{spam}), \text{ e}$$

$$p_2 = p([\text{prezado}] \mid \text{normal}) \cdot p([\text{amigo}] \mid \text{normal}) \cdot p(\text{normal})$$

- cálculo das probabilidades com os dados do problema e da tabela:

$$\begin{aligned} p(\text{normal}) &= 8/12 \\ p(\text{spam}) &= 4/12 \\ p([\text{prezado}] \mid \text{spam}) &= 3/11 \\ p([\text{amigo}] \mid \text{spam}) &= 2/11 \\ p([\text{prezado}] \mid \text{normal}) &= 9/21 \\ p([\text{amigo}] \mid \text{normal}) &= 6/21 \end{aligned}$$

- logo,

$$\begin{aligned} p_1 &= p([\text{prezado}] \mid \text{spam}) \cdot p([\text{amigo}] \mid \text{spam}) \cdot p(\text{spam}) \\ &= \frac{3}{11} \cdot \frac{2}{11} \cdot \frac{4}{12} = 0,01653 \end{aligned}$$

$$\begin{aligned} p_2 &= p([\text{prezado}] \mid \text{normal}) \cdot p([\text{amigo}] \mid \text{normal}) \cdot p(\text{normal}) \\ &= \frac{9}{21} \cdot \frac{6}{21} \cdot \frac{8}{12} = 0,08163 \end{aligned}$$

como $p_1 < p_2$, a mensagem é classificada como normal.

Até então foram apresentados os fundamentos básicos do funcionamento do Algoritmo Naive Bayes. O processo de implementação deste algoritmo em software, em larga escala pode ser descrita pelos seguintes passos:

1. Tem-se um banco de dados com mensagens de dois tipos: spam e normal;
2. Divide-se o banco de dados em 2 grupos, um de treino e outro de teste;
3. Se calculam todas as probabilidades no grupo de treino;
4. Com estas probabilidades se classificam as mensagens do grupo de teste;
5. Se calculam as métricas avaliadoras do processo;
6. Qualquer nova mensagem (não incluída no banco de dados) pode ser classificada como spam ou normal com a probabilidade de acerto do algoritmo.

3 CLASSIFICADOR NAIVE BAYES

O aprendizado de máquina é um subcampo da inteligência artificial que envolve o uso de algoritmos para reconhecer padrões em dados e fazer predições ou decisões com base nesse padrões aprendidos. Envolve o treinamento de um modelo a partir de um conjunto de dados, o que permite que o modelo faça predições ou tome ações com base em dados não vistos antes (DEISENROTH; FAISAL; ONG, 2020).

Os tipos de aprendizado de máquina, incluem aprendizado supervisionado, não supervisionado, semi-supervisionado, por reforço e aprendizado por transferência. O tipo específico de aprendizado de máquina usado depende da natureza do problema e dos objetivos do modelo. Segundo HARRINGTON (2012):

No **aprendizado supervisionado**, o algoritmo é treinado a partir de um conjunto de dados rotulado, onde a saída correta é fornecida para cada exemplo no conjunto de treinamento. O objetivo do algoritmo é fazer previsões sobre exemplos novos, não vistos, que são extraídos da mesma distribuição do conjunto de treinamento. Exemplos de aprendizado supervisionado incluem regressão, árvores de decisão e máquinas de vetores de suporte.

No **aprendizado não supervisionado**, o algoritmo não é fornecido com exemplos de treinamento rotulados. Em vez disso, deve descobrir a estrutura subjacente dos dados por meio de técnicas como agrupamento ou redução de dimensionalidade. Exemplos de aprendizado não supervisionado incluem agrupamento k-médias e análise de componentes principais.

No **aprendizado semi-supervisionado**, o algoritmo é fornecido com um pequeno número de exemplos rotulados e um grande número de exemplos não rotulados. O objetivo é aprender com os exemplos rotulados e não rotulados para fazer previsões mais precisas. O aprendizado semi-supervisionado é frequentemente usado quando é caro ou demorado rotular um grande conjunto de dados.

No **aprendizado por reforço**, um agente aprende interagindo com seu ambiente e recebendo recompensas ou punições por determinadas ações. O objetivo é que o agente aprenda uma política que maximize a recompensa cumulativa ao longo do tempo. O aprendizado por reforço é frequentemente usado em robótica e sistemas de controle.

No **aprendizagem por transferência**, um modelo treinado em uma tarefa é ajustado ou adaptado para executar uma tarefa relacionada. Isso pode ser útil quando há escassez de dados rotulados para uma tarefa específica ou quando a tarefa de destino é semelhante à tarefa de origem. O aprendizado por transferência tem sido bem-sucedido em uma variedade de aplicações, incluindo processamento de linguagem natural e visão computacional.

O aprendizado de máquina possui diversase aplicações, incluindo reconhecimento de

imagem e fala, processamento de linguagem natural, detecção de fraude e descoberta de medicamentos. Tem potencial para automatizar e melhorar muitas tarefas que atualmente são realizadas por humanos, e é uma área ativa de pesquisa em ciência da computação e análise de dados (JOEL, 2016).

3.1 PASSOS PARA DESENVOLVER UM ALGORITMO SUPERVISIONADO DE APRENDIZADO DE MÁQUINA

No desenvolvimento de um algoritmo de aprendizado de máquina supervisionado segue-se os passos (JOEL, 2016):

Coleta de dados. Para alimentar a entrada do algoritmo são necessários dados de qualidade. Estes dados podem ser públicos, jornais, redes sociais ou gerados por um aplicativo ou instrumentos de medição como a velocidade do vento por exemplo;

Formatação dos dados. Depois de obter os dados, é necessário garantir que eles estejam em um formato utilizável. Implementações diferentes podem utilizar formatos diferentes e os dados devem ser processados para se adequar ao formato exigido;

Análise dos dados. Verificar que os dados obtidos satisfazem as condições do problema e que possam ser lidos corretamente. Evitar entradas vazias e identificar entradas que diferem muito do restante do conjunto (outliers). Para descrever o dataset podem ser utilizadas tabelas, gráficos e dados estatísticos;

Preparação dos dados. Nesta etapa os dados analisados são divididos em dois conjuntos. O primeiro conjunto, geralmente com mais de 70% dos dados, é usado para a etapa de treino e o restante para a etapa de teste;

Etapa de treino. Nesta etapa ocorre o aprendizado de máquina. O algoritmo é alimentado com os dados reservados, que incluem os rótulos com a saída correta esperada e extrai padrões e probabilidades dos dados. Estas informações são armazenadas e serão utilizados na etapa de teste do aprendizado supervisionado;

Etapa de teste. Utilizando o restante dos dados reservados e sem o rótulo da saída esperada, o algoritmo agora deve com base na informação obtida na fase de treino determinar a classificação correta de dados que não viu antes. Nessa etapa também são coletadas as métricas para avaliar o desempenho do algoritmo na tarefa desempenhada;

Aplicação. Depois do treino e verificação do desempenho do algoritmo o mesmo pode passar a receber informações externas que não fazem parte do dataset original e fazer previsões da mesma maneira que fez na etapa de teste.

3.2 ALGORITMOS DE CLASSIFICAÇÃO

Um algoritmo de classificação é um algoritmo de aprendizado de máquina usado para prever a classe ou categoria de um determinado ponto de dados. É um tipo de aprendizado supervisionado, o que significa que o algoritmo é treinado em um conjunto de dados rotulado, onde a classe ou categoria correta para entrada já é conhecida.

Existem muitos algoritmos de classificação diferentes, incluindo Regressão Logística, Naive Bayes, Máquinas de Vetores de Suporte (SVMs), Árvores de Decisão e Florestas Aleatórias. Esses algoritmos funcionam aprendendo com os dados de treinamento e identificando padrões e relacionamentos que podem ser usados para prever a classe de dados novos e não vistos. Segundo DEISENROTH; FAISAL; ONG (2020):

A **Regressão Logística** é um classificador linear usado para tarefas de classificação binária. Ele usa uma função logística para mapear os dados de entrada para uma probabilidade entre 0 e 1 e, em seguida, faz uma previsão com base em se a probabilidade está acima ou abaixo de um determinado limite.

A **Árvore de Decisão** é um classificador baseado em árvore que usa uma série de decisões baseadas nas características dos dados de entrada para prever a classe da entrada.

A **Máquina de Vetores de Suporte** (SVMs) é um classificador linear que usa um limite de decisão para separar as diferentes classes nos dados de entrada.

O **Naive Bayes** é um classificador probabilístico que faz previsões com base na probabilidade de cada classe, dados os dados de entrada.

A **Floresta Aleatória** é um classificador de conjunto que consiste em uma coleção de árvores de decisão treinadas em diferentes subconjuntos dos dados de entrada. A previsão final é feita agregando as previsões das árvores individuais.

Para treinar um algoritmo de classificação, é necessário ter um conjunto de dados rotulado com exemplos das diferentes classes que deseja prever. Pode-se então dividir o conjunto de dados em um conjunto de treinamento e um conjunto de teste. O desempenho do algoritmo é normalmente avaliado usando métricas como exatidão, precisão e sensibilidade após a fase de teste (COELHO; RICHERT, 2015).

O desempenho do algoritmo também pode ser visualizado usando uma matriz de confusão, que mostra o número de previsões verdadeiro positivo, verdadeiro negativo, falso positivo e falso negativo feitas pelo classificador. A matriz de confusão é uma representação conveniente do desempenho do algoritmo e a facilita o cálculo de outras métricas como acurácia e sensibilidade.

Algoritmos de classificação são amplamente usados em uma variedade de aplicações, incluindo filtragem de spam, análise de sentimento e diagnóstico médico. Eles são uma ferramenta importante no campo do aprendizado de máquina e podem ser usados para resolver diversos problemas.

Por exemplo, um algoritmo de classificação pode ser usado para prever se um e-mail é spam ou não, com base nas palavras e frases do e-mail. Outro exemplo, prever as espécies de um determinado tipo de planta com base em suas características, como o formato de suas folhas e a cor de suas flores.

3.3 ALGORITMO NAIVE BAYES

Naive Bayes é um algoritmo de aprendizado de máquina baseado no princípio da probabilidade bayesiana. É um classificador probabilístico que usa o teorema de Bayes para prever a probabilidade de um determinado ponto de dados pertencer a uma determinada classe. O algoritmo assume que todos os recursos no conjunto de dados são independentes uns dos outros, o que é conhecido como suposição "ingênua". Essa suposição simplifica o cálculo da probabilidade de um ponto de dados pertencer a uma classe, mas nem sempre pode ser verdadeira em conjuntos de dados do mundo real.

Apesar dessa suposição, os classificadores Naive Bayes ainda podem ter um bom desempenho em muitas aplicações práticas, principalmente nos casos em que o número de recursos é grande em comparação com o número de exemplos de treinamento. Eles também são relativamente simples de implementar e podem ser bem dimensionados para grandes conjuntos de dados.

No contexto da classificação de texto, um documento é representado como um vetor binário, onde cada elemento do vetor corresponde a uma palavra ou termo específico do vocabulário. Se uma determinada palavra estiver presente no documento, o elemento correspondente do vetor é definido como 1; se a palavra não estiver presente, o elemento é definido como 0.

Existem várias variações do algoritmo Naive Bayes, incluindo Naive Bayes Gaussiano, Naive Bayes Multinomial e Naive Bayes Bernoulli. Essas variações diferem na maneira como estimam a probabilidade de um ponto de dados pertencer a uma determinada classe com base nos recursos (DANGETI, 2017).

O algoritmo **Naive Bayes de Bernoulli** usa essa representação binária dos documentos para fazer previsões sobre a classe de um determinado documento. Ele faz isso calculando a probabilidade de cada classe, dado o vetor de recursos do documento e, em seguida, selecionando a classe com a probabilidade mais alta.

O algoritmo **Naive Bayes Multinomial** é usado quando os recursos são dados de contagem (por exemplo, contagens de palavras em um documento). Ele assume que a probabilidade de cada característica é uma distribuição multinomial e usa essa suposição para fazer previsões sobre a classe de uma determinada amostra. O algoritmo Naive Bayes de Bernoulli é um caso particular do Naive Bayes Nultinomial, o qual por ser mais geral foi escolhido no desenvolvimento deste trabalho.

O algoritmo **Naive Bayes Gaussiano** é usado quando os recursos são variáveis contínuas

normalmente distribuídas. Ele assume que a distribuição de cada característica é gaussiana (curva de sino) e usa essa suposição para fazer previsões sobre a classe de uma determinada amostra.

O algoritmo Naive Bayes é frequentemente usado para classificação de texto, filtragem de spam e diagnóstico médico. Também é uma boa escolha para tarefas de classificação quando a quantidade de dados de treinamento é pequena ou quando o número de recursos é grande. A seguir apresentamos o desenvolvimento do cálculo das probabilidades no algoritmo Naive Bayes.

3.3.1 Cálculo das probabilidades com vetores de palavras

Desejamos calcular a probabilidade $p(c_i|w)$ de um vetor de palavras w ser da i-ésima classe c_i. Para o calculo desta probabilidade aplicamos a regra de Bayes (2.12):

$$p(c_i|w) = \frac{p(w|c_i) \cdot p(c_i)}{p(w)}, \tag{3.1}$$

onde,

- w é um vetor de palavras
- c_i é a classe i-ésima
- $p(w|c_i)$ é a probabilidade do vetor w dado que está na classe c_i

Por exemplo, se c_1 é a classe de noticias fake, c_2 é a classe de notícias reais sobre um determinado assunto e w é um vetor de palavras que identifica uma noticia, então $p(w|c_1)$ é a probabilidade de w dado que a noticia é fake e $p(w|c_2)$ é a probabilidade de w dado que a noticia é real. Enquanto, dada uma noticia representada pelo vetor de palavras w, desejamos classificar ela como fake ou real, isto é desejamos calcular $p(c_1|w)$ e $p(c_2|w)$ usando a regra de Bayes. A tomada de decisão é realizada comparando as probabilidades:

- se $p(c_1|w) > p(c_2|w)$, a probabilidade de w ser fake é maior, portanto a noticia é classificada como fake, e
- se $p(c_2|w) > p(c_1|w)$, a probabilidade de w ser real é maior, portanto a noticia é classificada como real.

Na etapa de treino há um conjunto de noticias e outro conjunto identificadas como reais. Pode-se calcular $p(c_i)$ somando quantas vezes vemos a classe i (noticias fakes ou reais) e depois dividindo pelo número total de noticias já identificadas a priori como fake ou real.

Dada uma noticia, constrói-se um vetor $w = [w_1, w_2, \cdots, w_N]$, onde cada componente w_k é uma palavra pré-processada representativa do tema em estudo. O cálculo da probabilidade condicional $p(w|c_i)$ é dada por:

$$p(w|c_i) = p(w_1, w_2, \cdots, w_N|c_i)$$

No caso particular de $w = [w_1, w_2]$, da equação da probabilidade condicional (2.4):

$$p(w_1, w_2|c_i) = \frac{p(w_1, w_2, c_i)}{p(c_i)}$$
$$p(w_1, w_2|c_i) = \frac{p(w_1|w_2, c_i)p(w_2, c_i)}{p(c_i)}$$
$$p(w_1, w_2|c_i) = \frac{p(w_1|w_2, c_i)p(w_2|c_i)\cancel{p(c_i)}}{\cancel{p(c_i)}}$$

logo,

$$p(w_1, w_2|c_i) = p(w_1|w_2, c_i)p(w_2|c_i) \tag{3.2}$$

Na equação (3.2) o termo $p(w_2|c_1)$ são calculados dos dados de treinamento, enquanto o termo $p(w_1|w_2, c_1)$ é difícil de ser calculada. Introduzimos o conceito de **independência condicional** definido em DEISENROTH; FAISAL; ONG (2020).

Definição 3.1 (Independência Condicional)**.** *Dois variáveis aleatórias X e Y são condicionalmente independentes dado Z se, e somente se,*

$$p(x, y|z) = p(x|z)\ p(y|z), \qquad \forall\, z \in Z. \tag{3.3}$$

Da definição (3.1), se w_1 e w_2 são independentes, então:

$$p(w_1, w_2|c_i) = p(w_1|c_i)\ p(w_2|c_i). \tag{3.4}$$

Comparando as equações (3.2) e (3.4) temos que se w_1 e w_2 são independentes, então:

$$p(w_1|w_2, c_1) = p(w_1|c_1). \tag{3.5}$$

A equação (3.5) é uma definição alternativa de independência condicional, e mostra que o fato de ocorrer w_2 não modifica a probabilidade de ocorrer w_1.

Pode-se generalizar a definição de probabilidade condicional para mais de dois eventos:

Definição 3.2 (Probabilidade Condicional Generalizada)**.** *As variáveis aleatórias $X_1, X_2, \cdots, X_N$ são condicionalmente independentes dado Z se, e somente se,*

$$p(x_1, x_2, \cdots, x_n|z) = p(x_1|z)\ p(x_2|z) \cdots p(x_N, z), \qquad \forall\, z \in Z. \tag{3.6}$$

3.3.2 Hipótese do Classificador Naive Bayes

A hipótese ingênua ou hipótese de Naive Bayes assume que as palavras $w_1, w_2, \cdots, w_n$ do vetor w são independentes dadas as classes c_i (DANGETI, 2017), isto é:

$$p(w_1, w_2, \cdots, w_N|c_i) = p(w_1|c_i)\ p(w_2|c_i) \cdots p(w_N|c_i). \tag{3.7}$$

A hipótese de independência (3.7) é chamada de naive ou ingênua pelo fato das suposições não serem realistas. Esta-se assumindo que cada palavra é tão provável individualmente como em conjunto com outras palavras: $p(w_1|w_2,\cdots w_n,c_i) = p(w_1|c_i)$. A outra suposição é que as classes são igualmente prováveis, o qual nem sempre é verdade. Mesmo com essas falhas nas suposições, o classificador Naive Bayes funciona bem na prática (HARRINGTON, 2012).

Substituindo a equação (3.7) em (2.12) a probabilidade $p(c_i|w_1,w_2,\cdots,w_N)$ é calculada por:

$$p(c_i|w_1,w_2,\cdots,w_N) = \frac{p(w_1|c_i)p(w_2|c_i)\cdots p(w_N|c_i)p(c_i)}{p(w_1,w_2,\cdots,w_N)}. \tag{3.8}$$

Na equação (3.8) as probabilidade $p(c_i|w)$ tem todas o mesmo denominador $p(w)$ independente da classe c_i, pelo qual para efeitos de comparação prescindir do denominador não muda o resultado. Simplificamos assim (3.8) para a equação:

$$p(c_i|w_1,w_2,\cdots,w_N) = p(w_1|c_i)p(w_2|c_i)\cdots p(w_N|c_i)p(c_i). \tag{3.9}$$

Para evitar underflow, situação onde os números ultrapassam os limites de precisão da estrutura de dados do computador (devido ao produto de probabilidades pequenas) (3.9), aplicamos a função logaritmo natural:

$$ln(p(c_i|w_1,w_2,\cdots,w_N)) = ln(p(c_i)) + ln\left(\sum_{k=1}^{N} p(w_k|c_i)\right). \tag{3.10}$$

3.4 CÓDIGO PYTHON PARA O CÁLCULO DAS PROBABILIDADES

O cálculo das probabilidades para a etapa de treino é descrito no Código 3.1 com a função **trainNB** (HARRINGTON, 2012). Descrevemos os detalhes do Código 3.1:

- A entrada **trainMatrix**, na linha 1, é um array $k_1 \times k_2$, onde k_1 é o número de documentos do conjunto de treino e k_2 é o comprimento da lista de todos os itens representativo do conjunto de dados. Por exemplo, se queremos detectar se um e-mail é ou não spam, k_1 é o número de e-mails do conjunto de treino e k_2 são todas as palavras presentes nos e-mails logo de ser aplicado o pre-processamento. Cada documento tem por tanto dimensão k_2 e é um vetor binário, onde cada componente recebe o valor 1 se este item está presente no documento e 0 em caso contrário.

- A entrada **trainCategory** na linha 1, são as classes ao qual pertencem os dados. O código é desenhado para a classificação binária, mais pode ser facilmente modificado para mais de duas classes. Consideremos as classes $c_0 = 0$ e $c_1 = 1$.

- **numTrainDocs**, na linha 2, é a dimensão k_1 do array **trainMatrix**.

- **numWords**, na linha 3, é a dimensão k_2 do array **trainMatrix**.
- **pAb2** é a probabilidade $p(c_1)$ da classe c_1 e $(1 - pAb2)$ é a probabilidade $p(c_0)$ da classe c_0).
- A filtragem para evitar probabilidades nulas, adicionando uma unidade na contagem dos casos favoráveis nas probabilidades $p(w|c_i)$ é efetivada nas linhas 5 a 8. **p0Num** e **p1Num** são vetores de 1's de dimensão k_2, enquanto **p0Denom** e **p1Denom** é a contagem total das probabilidades condicionais que iniciam com 2 por ter duas classes.
- Nas linhas 9 a 15 se calculam os casos favoráveis e totais das probabilidades $p(w|c_0)$ e $p(w|c_1)$ para cada um dos k_2 itens w do array **trainMatrix**.
- Na linha 16, **p1Vect** é um vetor de dimensão k_2 contendo o logaritmo das probabilidades condicionais $p(w|c_1)$.
- Na linha 17, **p0Vect** é um vetor de dimensão k_2 contendo o logaritmo das probabilidades condicionais $p(w|c_0)$.
- Na linha 18, a função retorna o array das probabilidades condicionais $p(w|c_0)$, $p(w|c_1)$ e a probabilidade condicional $p(c_1)$.

Código 3.1 – Probabilidades do treino

```
1  def trainNB(trainMatrix,trainCategory):
2      numTrainDocs = len(trainMatrix)
3      numWords = len(trainMatrix[0])
4      pAb2 = sum(trainCategory)/float(numTrainDocs)
5      p0Num = np.ones(numWords)
6      p1Num = np.ones(numWords)
7      p0Denom = 2.0
8      p1Denom = 2.0
9      for i in range(numTrainDocs):
10         if trainCategory[i] == 1:
11             p1Num += trainMatrix[i]
12             p1Denom += sum(trainMatrix[i])
13         else:
14             p0Num += trainMatrix[i]
15             p0Denom += sum(trainMatrix[i])
16     p1Vect = np.log(p1Num/p1Denom)
17     p0Vect = np.log(p0Num/p0Denom)
18     return p0Vect,p1Vect,pAb2
19
```

Estas probabilidades, obtidas com a função trainNB na etapa de treino, são usadas na etapa de teste para obter as métricas que avaliam o desempenho do classificador.

4 CLASSIFICADOR NAIVE BAYES NA DETECÇÃO DE FAKE NEWS

Em dezembro de 2019, na China, foram identificados os primeiros casos do Covid-19, doença causada pelo vírus SARS-CoV-2. A doença espalhou-se rapidamente para outros países, atingindo milhões de pessoas e levando a Organização Mundial da Saúde (OMS) a decretar a situação mundial como pandêmica no dia 11 de março de 2020 (GALHARDI et al., 2020).

No Brasil, o primeiro caso da doença foi reportado em 25 de fevereiro de 2020, sendo também o primeiro caso da América do Sul (BURKI, 2020). Em março, foi implementado o distanciamento social para frear a transmissão do vírus, junto com algumas medidas como a obrigatoriedade do uso de máscaras em lugares públicos e higienização das mãos com álcool em gel. Porém, mesmo com essas medidas, o número de casos aumentava exponencialmente.

Há vários fatores que contribuíram para esse aumento e um deles pode ser atribuído à propagação de notícias falsas relacionadas à Covid-19, principalmente, por meio das redes sociais. Esse movimento atingiu o mundo inteiro, a ponto de a Organização Mundial da Saúde (OMS) denominá-lo como “infodemia” (GALHARDI et al., 2020).

A desinformação é muito prejudicial para a área da saúde. Em 2008, por exemplo, ocorreu no Brasil um surto do vírus da febre amarela e o Ministério da Saúde tinha a meta de vacinar 80% da população, porém, chegou apenas em 55% da população. Segundo a Organização Mundial da Saúde, as notícias falsas que questionavam a eficácia e a segurança das vacinas podem ter sido umas das causas que influenciaram para que a meta não fosse atingida (GALHARDI et al., 2020).

Com base no problema que as fakes news são para a população, será testado o algoritmo Naive Bayes (desenvolvido no Capítulo 3) na detecção de notícias falsas sobre a pandemia do Covid-19 ocorrido no Brasil. Para isso, foram coletadas publicações da rede social Twitter no período entre os meses janeiro e julho de 2021. Essas publicações podem ser classificadas de duas formas: Notícias verdadeiras sobre assuntos relacionados ao Covid-19 retiradas de perfis confiáveis (CNN, Ministério da Saúde, Coronavírus Brasil, prefeituras, pessoas verificadas) que reproduzem notícias com conteúdos verdadeiros ou notícias falsas retiradas de postagens com intuito de desinforma a população. Para validar se as notícias são falsas, foi utilizado sites como “G1 Fato ou Fake”, “E-Farsas” e “Agência Lupa” que analisam essas notícias e com base em evidências definem se é verdadeira ou falsa.

4.1 BANCO DE DADOS

O banco de dados **BD_NB** é um arquivo de texto de extensão csv (Comma-Separated Values) que separa os dados com vírgulas. O arquivo consiste das seguintes informações:

1. **Id**: Índice que numera a posição da linha ou notícia.
2. **Título**: O título da notícia.
3. **Corpo**: O corpo da notícia.
4. **Classificação**: A classificação, na qual cada notícia é considerada verdadeira (TRUE) ou falsa (fake), e
5. **Link**: O link da notícia.

Do banco de dados **BD_NB** é feita somente a leitura dos campos ID, Corpo e Classificação, os outros campos ficam como registros dos titulares e das fontes das noticias que foram usadas no desenvolvimento desta aplicação. Um dataframe **df** com os três campos é gerada no Python com o comando:

```
df = pd.read_csv('BD_NB.csv', sep = '\t', usecols = ['id', 'corpo', '
    classificacao'])
```

Na Tabela 4.1 visualizamos o cabeçalho dos dados com df.head().

Tabela 4.1 – Cabeçalho do dataframe df

	id	corpo	classificacao
0	1	O que é imunidade de grupo? Por que se fala ta...	TRUE
1	2	Não podemos esquecer que o próprio STF proibiu...	fake
2	3	Por falta de oxigênio sete pessoas da mesma fa...	TRUE
3	4	A Vachina pode ter 50% de eficácia, mas a verb...	fake
4	5	Vacina Pfizer tem entre 19% e 29% de eficácia ...	fake

Fonte: Elaborado pelos Autores.

A função len(df) indica que o dataframe **df** tem 450 linhas e o código:

```
df['classificacao'].value_counts()
```

classifica o dataframe com 242 notícias falsas (fake) e 208 notícias verdadeiras (TRUE).

Para preparar a divisão dos dados em um conjunto de treino e outro de testes, o dataframe é misturado aleatoriamente com o comando df = df.sample(frac=1).reset_index(drop=True). Na Tabela 4.2 se mostra os dados embaralhados ao aplicar a função df.head().

Tabela 4.2 – Cabeçalho do dataframe df embaralhado

	id	corpo	classificacao
0	339	ButanVac: Instituto Butantan inicia testes clí...	TRUE
1	397	Infeção por coronavírus causa alterações grave...	fake
2	258	negacionismo do Doria durou quanto tempo? Há q...	fake
3	342	Pfizer e autoridades de saúde dos EUA discutir...	TRUE
4	226	Informamos que amanhã, quarta-feira (17), cheg...	TRUE

Fonte: Elaborado pelos Autores.

4.2 PRÉ-PROCESSAMENTO

As notícias tem que ser pré-processadas para transformar cada noticia em um array de palavras. Primeiro pode-se limpar eliminando caracteres como virgulas, traços, símbolos especiais, a exemplo de @, <, >=. As palavras que são derivadas de uma raiz comum podem ser substituídas pela raiz, evitando assim fazer contagem de palavras similares como sendo diferentes. Pode-se também eliminar palavras que tem somente uma ou duas letras, como 'a', 'se', 'de' por exemplo, pois elas não são decisivas para definir se uma noticia é real ou fake. Esse processo é conhecido como tokenização (tokenization) (WEBSTER; KIT, 1992).

Fazemos uso de algumas ferramentas do pacote NLTK (Natural Linguagem Toolkit), o qual é um conjunto de bibliotecas e programas para processamento simbólico e estatístico de linguagem natural para português escrito na linguagem Python. A biblioteca **stopwords** do NLTK, é um conjunto de palavras que podem ser eliminadas por serem palavras usadas frequentemente que não influenciam no significado do texto, armazenando o conteúdo na variável **stop_words**. Para melhorar o desempenho, foi criado uma lista na variável **new_stopwords** com novas palavras para complementar a biblioteca.

A função preprocessamento remove as palavras em stop_words, as palavras com um ou 2 caracteres e a biblioteca **gensim** particiona cada noticia em um array com as palavras resultantes da limpeza realizada (ver Código 4.1.

Código 4.1 – Pré processamento

```
nltk.download("stopwords")
from nltk.corpus import stopwords
stop_words = stopwords.words('portuguese')
new_stopwords = ["ficar","letra","passada","maior","havia","alguns","
    abrir","contudo","oitenta","deixar","propria","novos","poderia","
    sexta","domingo","segunda","terca","quarta","quinta","sabado","feira"
    ,"vidas","partes","parte","longa","varios","anos","todo","otimo","
    apoio","fase","pros","ultimo","forma","lado","tempo","encurtar","
    nenhum","todos","sempre","final","comeco","dois","tres","quatro","
    cinco","seis","sete","oito","nove","dez","onze","categoria","link","
    mail","variant","publicidade","aalx","quase","completa","existem","
    meio","ultima","medica","deste","grande","voltar","ontem","horas","
```

```
    algum","geral","https","outras","cbsn","krewe","igual","proxima","
    cerca","olho","nivel","primeira","ainda","mulher","agora","brasil","
    amapa","doze","quantidade","marco","fevereiro","junho","julho","
    janeiro","abril","maio","agosto","setembro","outubro","dezembro","
    novembro","maximo","todas","homem","leia","menos","mais","primeiro","
    ultimo","tipo","numeros","numero", "mudou", "motivo"]
stop_words.extend(new_stopwords)
def preprocessamento(text):
    result = []
    for token in gensim.utils.simple_preprocess(text):
        if token not in gensim.parsing.preprocessing.STOPWORDS and len(
    token) > 3 and token not in stop_words:
            result.append(token)

    return result
```

Aplicamos a função preprocessamento a **df['corpo']** e a salvamos num array **corpo**:

```
corpo = df['corpo'].apply(preprocessamento)
```

por exemplo, **corpo[0]** tem como pré-processamento o array:
['levantamento', 'indica', 'aumento', 'expressivo', 'novos', 'casos', 'covid', 'após', 'réveillon', 'aglomerações', 'festas', 'algumas', 'praias', 'procuradas', 'país', 'apuração', 'agência', 'pública'].

O campo **df['classificacao']** é armazenada no array **classes**, assim nossa base de dados inicial após o pre-processamento ficou reduzida aos arrays **corpo** e **classes**. Transformamos os valores **'TRUE'** e **'false'** para **1** e **0** respectivamente:

```
classes = df['classificacao']
for j in range(len(classes)):
    if classes[j] == 'TRUE':
        classes[j] = 1
    else:
        classes[j] = 0
```

Consideremos 70% dos dados para treinamento e 30% para teste, para isto definimos **n = len(corpo)** o número total de noticias, **n_train = 0.7*n** o número de noticias para treino e **n_test = n - n_train** o número de noticias para teste. Os resultados mostram que $n = 450$, $n\_train = 315$ e $n\_test = 135$.

As 315 primeiras noticias definem o conjunto de treino **X_train** e **y_train**, enquanto as ultimas 135 noticias o conjunto de teste **X_test** e **y_test**:

```
n = len(corpo)
n_train = int(0.7*n)
n_test = n - n_train
X_train = corpo[0:n_train]
```

```
y_train = classes[0:n_train]
X_test = corpo[n_train:n]
y_test = classes[n_train:n]
y_test
```

4.3 ETAPA DE TREINAMENTO

As funções desenvolvidas no capítulo 3 para o classificador Naive Bayes serão aplicadas na etapa de treinamento com os arrays **X_train** e **y_test**. Cada componente de **X_train** é um array formado por palavras obtidas no pre-processamento. A união de todas estas palavras formam um array **ListaVocab** gerada com a função **criaListaVocab** (ver Código 4.2).

Código 4.2 – Cria lista de palavras

```
def criaListaVocab(dados):
  vocab = set([])
  for documento in dados:
    vocab = vocab | set(documento)
  return list(vocab)

ListaVocab = criaListaVocab(X_train)
```

Transformamos uma nova noticia em um vetor binário comparando-o com o vocabulário usando a função **WordsVec** no Código 4.3.

Código 4.3 – Transformação de uma noticia em vetor binário

```
def WordsVec(ListaVocab, inputSet):
  Vec = [0]*len(ListaVocab)
  for word in inputSet:
    if word in ListaVocab:
      Vec[ListaVocab.index(word)] = 1
  return Vec
```

Aplicando a função **len(ListaVocab)** se observa que o array é formado por 2495 palavras. A função **print(ListaVocab)** imprime o array, mostramos o inicio e o final deste array:

['golpe', 'sabiam', 'tosse', 'mostra', 'dinamarca',$\cdots$, 'preferem', 'méxico', 'filho']

Cada componente de **X_train[k]**, que é formada pelas palavras da késima noticia, é transformada a um vetor binário de igual dimensão ao array **ListaVocab** usando a função **WordsVec**. Este vetor binário terá o valor **1** para as palavras de **X_train[k]** e **0** em caso contrário, os resultados são armazenados no vetor **trainMat**. Calculam-se as probabilidades associadas ao classificador Naive Bayes usando função **trainNB** do Código 3.1, usando como entradas os arrays **trainMat** e **y_train** (ver o Código 4.4).

Código 4.4 – Performance do Classificador Naive Bayes

```
trainMat=[]
for postinDoc in X_train:
    trainMat.append(WordsVec(ListaVocab, postinDoc))
p0V,p1V,pAb=trainNB(trainMat,y_train)
```

Pode-se visualizar estas probabilidades com a função **print**:

pAb = 0.45396825396825397
p0V = array([-6.1723, -7.2710, -7.2710, ···, -7.2710, -7.2710, -7.2710])
p1V = array([-7.7557, -7.7557, -7.0626, ···, -7.7557, -7.7557, -7.7557])

Os valores das probabilidades das componentes do array **trainMat** apresentam valores negativos devido a que foram calculadas usando a função logaritmo natural. Os resultados para efeitos de comparação se mantém devido a que a função logaritmo é crescente. Por exemplo **p0V[0] = -6.1723** e **p1V[0] = -7.7557**, como **p0V[0] < p1V[0]** a palavra **trainMat[0]** tem maior probabilidade de pertencer a uma noticia fake.

4.4 ETAPA DE TESTE

Nesta etapa são comparadas as decisões tomadas pelo classificador Naive Bayes das noticias ser ou não fake do conjunto **X_test** com os valores conhecidos de **y_test** e vários indicadores são calculados para analisar a performance do classificador Naive Bayes na detecção de noticias fake sobre o covid-19. Para tomar a decisão se uma noticia é ou não **fake** usamos a função **NB** no Código 4.5.

Código 4.5 – Decisão no Classificador Naive Bayes

```
def NB(corpo, classes, testEntry):
    myVocabList = criaListaVocab(corpo)
    trainMat = []
    for postinDoc in corpo:
        trainMat.append(WordsVec(myVocabList, postinDoc))
    p0V, p1V, pAb = trainNB(np.array(trainMat), np.array(classes))
    thisDoc = np.array(WordsVec(myVocabList, testEntry))
    valor = classificaNB(thisDoc, p0V, p1V, pAb)
    return valor
```

O Código 4.6 define a função **acertos** que calcula a performance do conjunto teste:

Código 4.6 – Performance do Classificador Naive Bayes

```
def acertos(X_train,y_train,X_test,y_test,n,n_train):
    count00 = 0
    count01 = 0
    count10 = 0
    count11 = 0
```

```
    for k in range(len(X_test)):
        decisao = NB(X_train, y_train, X_test[k+n_train])
        real = y_test[k+n_train]
        if decisao == 0 and real == 0:
            count00 = count00 + 1
        elif decisao == 0 and real == 1:
            count01 = count01 + 1
        elif decisao == 1 and real == 0:
            count10 = count10 + 1
        else:
            count11 = count11 + 1

    return count00, count01, count10, count11
```

onde,

- **count00**: é um contador que se incrementa quando a decisão é **fake** sendo que a noticia é **fake**;
- **count01**: é um contador que se incrementa quando a decisão é **fake** sendo que a noticia é **verdadeira**;
- **count10**: é um contador que se incrementa quando a decisão é **verdadeira** sendo que a noticia é **fake**;
- **count11**: é um contador que se incrementa quando a decisão é **verdadeira** sendo que a noticia é **verdadeira**.

Para cada componente de **X_test** se aplica a função NB(X_train, y_train, X_test[k+n_train]) para obter uma decisão se a noticia é fake ou verdadeira. Cada decisão é comparada com o valor real **y_test[k+n_train]** sendo atualizada cada um dos 4 contadores. A chamada da função **acertos** é: VP, FP, FN, VN = acertos(X_train,y_train,X_test,y_test,n,n_train) , onde

- **VP**: Verdadeiro-Positivo, associado ao contador **count00** são as noticias cuja decisão são fake quando realmente elas são fake;
- **FP**: Falso-Positivo, associado ao contador **count01** são as noticias cuja decisão são fake quando realmente elas são verdadeiras;
- **FN**: Falso-Negativo, associado ao contador **count10** são as noticias cuja decisão são verdadeirasq uando elas são realmente fake;
- **VN**: Verdadeiro-Negativo, associado ao contador **count11** são as noticias que são verdadeiras, quando elas são realmente verdadeira.

Os resultados obtidos são: **VP = 67**, **FP = 8**, **FN = 5** e **VN = 55**.

4.4.1 Matriz de Confusão

A **matriz de confusão** é uma métrica que exibe visualmente o número de previsões verdadeiro positivo, verdadeiro negativo, falso positivo e falso negativo feitas pelo modelo. Normalmente usada em combinação com outras métricas, como precisão e sensibilidade, para avaliar o desempenho de um modelo de classificação. Essas métricas podem ajudar a identificar áreas específicas onde o modelo está cometendo erros e podem ser usadas para melhorar o modelo.

No Python geramos a matriz confusao = np.array([[VP, FN],[FP, VN]]), a qual tem valores:

$$\begin{bmatrix} VP & PN \\ FP & VN \end{bmatrix} = \begin{bmatrix} 67 & 8 \\ 5 & 55 \end{bmatrix}$$

Transformamos o array **confusao** num dataframe para mudar o nome dos dados, **0** muda para **Fake** e **1** muda para **Real**. Os resultados do dataframe **confusao_df** são mostrados na Tabela 4.4.

Tabela 4.3 – Dataframe da matriz confusão

	Fake	Real
Fake	67	8
Real	5	55

Fonte:Elaborado pelos Autores.

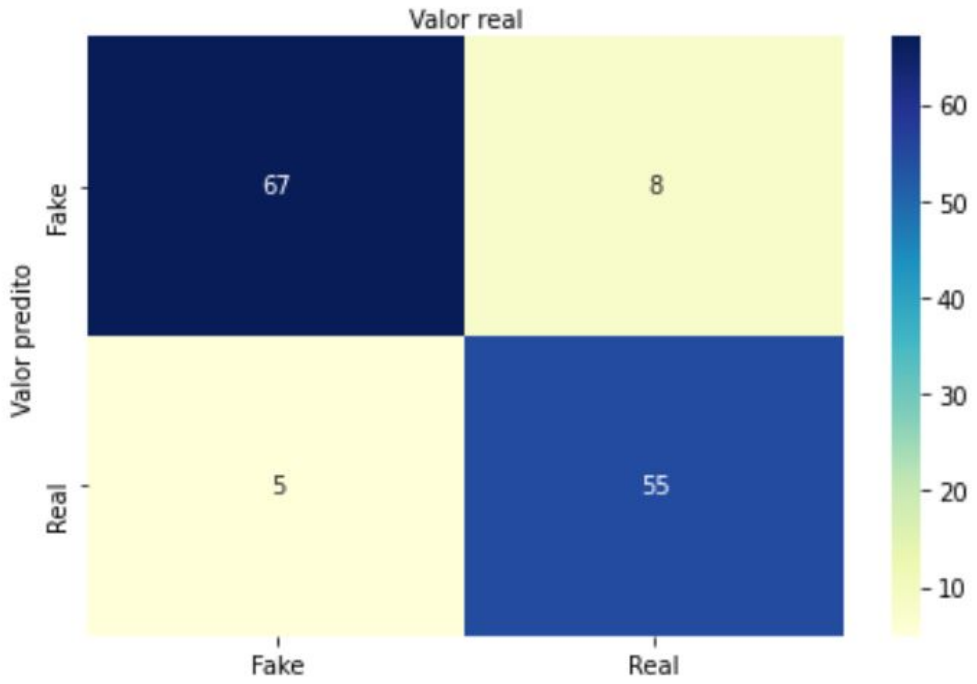
Geramos um gráfico da matriz confusão usando a função **heatmap** do pacote **seaborn** do Python, com o Código 4.7. A Figura 4.1 mostra os resultados da matriz confusão.

Código 4.7 – Performance do Classificador Naive Bayes

```
VP, FP, FN, VN = acertos(X_train,y_train,X_test,y_test,n,n_train)
confusao = np.array([[VP, FN],[FP, VN]])
classe =['Fake', 'Real']
confusao_df = pd.DataFrame(confusao, columns=classe, index=classe)
fig, ax = plt.subplots()
sns.heatmap(pd.DataFrame(confusao_df),annot=True,cmap="YlGnBu",fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Matriz de Confusao', y=1.1)
plt.ylabel('Valor predito')
plt.xlabel('Valor real')
plt.show()
```

A matriz de confusão possibilita visualizar com mais clareza os resultados do algoritmo de aprendizado de máquina. As cores mais escuras representam os acertos e as cores claras os erros de predição. Assim, seguem-se as seguintes observações:

Figura 4.1 – Matriz confusão



Fonte: Elaborado pelos Autores.

- São 67+8+5+55 = 135 noticias que foram usada como teste;
- a primeira coluna indica que foram testadas 67+5 = 72 noticias fake;
- a segunda coluna indica que foram testadas 8+55 = 63 noticias reais;
- a diagonal principal indica que houve 67+55 = 122 acertos;
- a diagonal secundária indica que houve 8+5 = 13 erros na predição;
- foram preditas 67+8 = 75 noticias fake, das quais:
  - 67 acertos, já que as noticias eram realmente fake;
  - 8 erros, já que as noticias eram na verdade reais;
- foram preditas 5+55 = 60 noticias reais, das quais:
  - 8 erros, já que as noticias eram na realidade fake;
  - 58 acertos, já que as noticias sim eram reais.

4.4.2 Métricas de Avaliação de Classificadores

As métricas de avaliação são formas de medir o desempenho de um modelo de aprendizado de máquina. Algumas métricas de avaliação comuns incluem acurácia, precisão, sensibilidade e pontuação F1-score. A escolha da métrica de avaliação geralmente depende da tarefa

específica para a qual o modelo está sendo usado, bem como das características dos dados. Por exemplo, a precisão é uma boa métrica a ser usada quando as classes nos dados estão bem balanceadas, mas pode ser enganosa se as classes estiverem desequilibradas. Nesse caso, a precisão e a sensibilidade podem ser métricas mais úteis.

Acurácia. É uma medida de desempenho de um modelo ou sistema. É tipicamente expressada em percentual e reflete o número de previsões ou classificações corretas do modelo comparadas ao número total de previsões ou classificações realizadas. Isto é, a **acurácia** é uma medida de quão precisamente um modelo o sistema pode prever ou classificar novos dados.

Alta exatidão significa que a medida o cálculo é muito próximo ao valor real, enquanto baixa acurácia significa que a medida está longe do valor verdadeiro. Por exemplo, se um modelo de detecção de noticias fake prevê acerta 190 de um total de 200 noticias, então a acurácia do modelo seria 190/200 = 95%. Ela é calculada pela fração:

$$\text{Acurácia } = \frac{VP+VN}{VP+VN+FP+FN} = \frac{67+55}{67+55+8+5} = 90,37\%.$$

O resultado mostra que a acurácia do classificador Naive Bayes na detecção de noticias fake do covid-19 foi de 90,37%. No Python, calculamos a acurácia com as linhas de código:

```
acuracia = (VP+VN)/(VP+VN+FP+FN)
print('acuracia = {}%'.format(acuracia*100))
```

Precisão Fake. Precisão é uma medida de quão preciso é um modelo ou sistema. É frequentemente usada no contexto de tarefas de classificação em aprendizado de máquina, onde mede a fração de exemplos positivos previstos corretamente de todos os exemplos que são classificados como positivos. Por exemplo, se um modelo de detecção de noticias fake prevê que existem 100 noticias fake e na verdade existem 110 noticias fake, a precisão do modelo seria 100/110 = 91%. Ela é calculada pela fração:

$$\text{Precisão Fake } = \frac{VP}{VP+FP} = \frac{67}{67+5} = 93,06\%.$$

O resultado mostra que a precisão fake do classificador Naive Bayes na detecção de noticias fake do covid-19 foi de 93.06%. No Python, calculamos a precisão com as linhas de código:

```
precisao_fake = VP/(VP+FP)
print('precisao fake = {}%'.format(precisao_fake*100))
```

Sensibilidade Fake É a capacidade de um modelo ou sistema de recuperar todas as informações relevantes de um determinado conjunto de dados. É frequentemente usado no contexto de aprendizado de máquina e processamento de linguagem natural, onde mede a fração de exemplos positivos previstos corretamente de todos os exemplos previstos como positivos. Por exemplo, se um modelo de detecção de noticias fake prevê que existem 100 noticias fake e na verdade

existem 85 noticias fake, a sensibilidade do modelo seria 85/100 = 85%. Ela é calculada pela fração:

$$\text{Sensibilidade Fake } = \frac{VP}{VP+FN} = \frac{67}{67+8} = 89,33\%.$$

O resultado mostra que a sensibilidade fake do classificador Naive Bayes na detecção de noticias fake do covid-19 foi de 89,33%. No Python, calculamos a sensibilidade com as linhas de código:

```
sensibilidade_fake = VP/(VP+FN)
print('sensibilidade fake = {}%'.format(sensibilidade_fake*100))
```

F1-Score Fake O F1-score, também conhecido como F-score ou F-medida, é uma métrica para avaliar a precisão de um modelo. É a média harmônica entre a precisão e a sensibilidade, sendo calculada tomando a média dos dois. O F1-score é comumente usado no campo do aprendizado de máquina, especialmente no contexto da classificação binária. O F1-score é um valor intermediário entre a precisão e a sensibilidade, sendo calculada pela fração:

$$\text{F1-Score Fake } = \frac{2}{\frac{1}{\text{precisão fake}} + \frac{1}{\text{sensibilidade fake}}} = \frac{2}{\frac{1}{0.9306} + \frac{1}{0,8933}} = 91,16\%.$$

O resultado mostra que o F1-score fake do classificador Naive Bayes na detecção de noticias fake do covid-19 foi de 91,16%. No Python, calculamos o F1-score com as linhas de código:

```
F1_score_fake = 2/(1/precisao_fake + 1/sensibilidade_fake)
print('F1-score fake = {}%'.format(F1_score_fake*100))
```

4.5 RESULTADOS

Para testar a eficácia do algoritmo desenvolvido, foram coletados os resultados de a acurácia, precisão fake, sensibilidade fake e F1-Score Fake de 100 iterações do aplicativo e ao final foi calculado a média desses resultados. A cada iteração o algoritmo mistura os dados do banco de dados aleatoriamente para que não haja vícios e resultados iguais.

Tabela 4.4 – Resultados do Teste

Acurácia	84,62222222%
Precisão Fake	90,79182804%
Sesibilidade Fake	79,62825579%
F1-Score Fake	84,76520242%

Fonte: Elaborado pelos Autores.

O resultado, levando em consideração a proporção de 70% dos dados para treinamento e 30% para teste, apresentou uma acurácia de 84,62%. As medidas precisão e F1-Score obtiveram bons resultados demonstrando que o classificador possui uma boa concordância entre a classe predita para cada objeto e a classe fake. A sensibilidade mesmo apresentando o menor resultado dos testes, ainda denotando uma boa habilidade na identificação de notícias fakes.

5 IMPLEMENTAÇÃO ALTERNATIVA EM JAVASCRIPT

Utilizando os fundamentos apresentados anteriormente nos capítulos 2 e 3 é possível implementar o classificador em qualquer linguagem de programação . No exemplo a seguir foi utilizada a linguagem de programação JavaScript (nodeJS).

5.1 BANCO DE DADOS

O mesmo banco de dados foi utilizado com pequenas alterações. Foram extraídos apenas os campos id, corpo e classificação. Na coluna classificação **TRUE** foi substituído por "1"e **fake** por "0".

5.2 PRÉ PROCESSAMENTO 4.2

Para o pré processamento de dados foram utilizadas as bibliotecas **Natural** (NATURALNODE, 2022) para a geração de tokens e **stopword** (FERGIEMCDOWALL, 2022) para a remoção de 4.2 stopwords.

Código 5.1 – Funções para o pré-processamento

```
const CustomWords = () => {
    const stopwords = readFileSync('stopwords.txt', 'utf8');
    const tokenizer = new natural.AggressiveTokenizerPt()
    return tokenizer.tokenize(stopwords);
}
const Tokenize = (news) => {
    const customStopWords = CustomWords();
    const tokenizer = new natural.AggressiveTokenizerPt()
    const output = tokenizer.tokenize(news)
    const custom = removeStopwords(output,[...porBr, ...customStopWords
    ]);
    return custom.map((word) => {
        return word.toLowerCase();
    });
}
```

O texto *Três pacientes foram identificados com a nova cepa do coronavírus* quando usado como argumento na função **Tokenize** retorna a seguinte estrutura:

```
[ 'pacientes', 'identificados', 'cepa', 'coronavirus' ]
```

A função **Tokenize** é aplicada em todas as entradas do banco **BD_NB** 4.1 e armazenada como um array de objetos com dois campos: corpo (o conteúdo da notícia) e tipo (verdadeiro ou falso) com o seguinte código:

Código 5.2 – Aplicação das funções no banco de dados

```
import {readFileSync} from "fs";
import Papa from 'papaparse';
import Tokenize from './Tokenize.js'
const CSVData = () => {
    const csv = readFileSync('bd.csv', 'utf8');
    const parsed = Papa.parse(csv);
    const output = parsed.data.map((data) => {
        //se possuir corpo e tipo
        if (data[2]) {
            return {
                'corpo': Tokenize(data[2]),
                'tipo': data[3],
            }
        } else {
            return {
                'corpo': '',
                'tipo': 3,
            }
        }
    });
    return output.filter(entry => entry.tipo != 3);
}
```

A estrutura dos objetos possui o seguinte formato:

```
  {
    corpo: [
    'pacientes', 'identificados', 'cepa', 'coronavirus'
    ],
    tipo: '1'
  },
```

5.3 ETAPA DE TREINAMENTO

O Dataset gerado a partir do banco é dividido em treino e teste utilizando uma proporção 70/30 com o seguinte código:

```
    const split = 0.70
    const seed = Math.random();
    const alphaPad = 1;
    const csv = CSVData();
    const [train, test] = trainTestSplit(csv, split, seed);
```

A função de treino é executada duas vezes para gerar um classificador para cada tipo de notícia, verdadeira e falsa.

Código 5.3 – Filtragem do dataset

```
    const verdadeiras = train.filter(entry => entry.tipo == 1);
    const falsas = train.filter(entry => entry.tipo == 0);
    const baseVerdadeira = verdadeiras.length / quantidade;
    const baseFalsa = falsas.length / quantidade;
    const trained = Train(verdadeiras, falsas);
```

Código 5.4 – Função Train

```
const Classifier = (dataSet) => {
    let vocabulario = [];
    const classifiedData = [];
    dataSet.map((data) => {
        data.corpo.map((palavra) => {
            const current = classifiedData.find(entry => entry.palavra
   === palavra);
            if (current) {
                current.contagem += 1;
            } else {
            vocabulario.push(palavra);
                classifiedData.push({
                    'palavra': palavra,
                    'contagem': 1,
                    'probabilidade': 0.0,
                });
            }
        });
    });
    classifiedData.map((data) => {
        data.probabilidade = data.contagem/ (classifiedData.length || 1)
   ;
    });
    return { 'data' : classifiedData, 'vocabulario' : vocabulario };
}
const Train = (dataTrue, dataFalse) => {
    const verdadeiras = Classifier(dataTrue);
    const falsas = Classifier(dataFalse);
    const vocab = _.union(verdadeiras.vocabulario, falsas.vocabulario);
    return ({
        'classV': verdadeiras.data,
        'classF': falsas.data,
        'vocab': vocab.length,
    });

}
```

Código 5.5 – Estrutura retornada pela função train

```
{
  vocab: 2625,
  classV: [
    { palavra: 'estudo', contagem: 5, probabilidade: 0.004 },
    { palavra: 'variante', contagem: 36, probabilidade: 0.0288 },
    { palavra: 'delta', contagem: 11, probabilidade: 0.0088 },
    { palavra: 'anticorpos', contagem: 3, probabilidade: 0.0024 },
    ...
  ],
  classF: [
    {palavra: 'inescrupulosos', contagem: 2, probabilidade: 0.00088},
    {palavra: 'virus',contagem: 37, probabilidade: 0.0164},
    {palavra: 'chines',contagem: 7, probabilidade: 0.00310},
    {palavra: 'pegar',contagem: 2, probabilidade: 0.000888},
    ...
  ],
}
```

Onde:

- vocab: Total de palavras encontradas no dataset
- classV e classF: Classificadores para notícias verdadeiras e falsas respectivamente. Cada item é um objeto com a palavra e a probabilidade da mesma aparecer em uma notícia de seu tipo

5.4 ETAPA DE TESTE

4.4 Antes de iniciar a etapa de teste os dados são organizados na seguinte estrutura:

Código 5.6 – Estrutura para realização dos testes

```
    const model = {
        'probV': baseVerdadeira,
        'probF': baseFalsa,
        'classV': trained.classV,
        'classF': trained.classF,
        'alpha': alphaPad / (trained.vocab),
        'conMatrix': {
            'VP': 0,
            'VN': 0,
            'FP': 0,
            'FN': 0,
        }
    }
```

Na fase de teste cada noticia passa pelos classificadores utilizando a função ClassifierTest.

Código 5.7 – Função ClassifierTest

```
const ClassifierTest = (classifier, prob, alpha, noticia) => {
    prob = Math.log(prob);
    let corpo = noticia.corpo;
    corpo.map((palavra) => {
        const current = classifier.find(entry => entry.palavra ===
   palavra);
        if (current) {
            prob += Math.log(current.probabilidade + alpha);
        } else {
            prob += Math.log(alpha);
        }
    });
    return prob;
}
```

Os dados são então comparados com a função Test, que também preenche a matriz de confusão:

Código 5.8 – Função Test

```
const Test = (model, test) => {
    test.map((noticia) => {
        let probV = ClassifierTest(model.classV, model.probV, model.
   alpha, noticia)
        let probF = ClassifierTest(model.classF, model.probF, model.
   alpha, noticia)
        let prediction = (probV > probF);
        if (prediction == 1) {
            if (noticia.tipo == 1) {
                model.conMatrix.VP += 1;
            } else {
                model.conMatrix.FP += 1;
            }
        } else {
            if (noticia.tipo == 0) {
                model.conMatrix.VN += 1;
            } else {
                model.conMatrix.FN += 1;
            }}});
    return model;
}
```

5.5 RESULTADOS

4.5 A aplicação retorna os resultados no seguinte formato

Código 5.9 – Resultados de um teste

```
{ VP: 36, VN: 60, FP: 4, FN: 10 }
Acuracia geral: 87.27%
Precisao fake: 85.71%
Sensibilidade: 93.75%
f1 score: 90.00%
```

Onde:

- VP - Verdadeiros Positivos
- VN - Verdadeiros negativos
- FP - Falsos positivos
- FN - Falsos negativos

Após executar 500 iterações da aplicação e calcular a média os seguintes resultados foram obtidos:

Código 5.10 – Código para iteração e cálculo da média

```
import NB from './NB.js'
const iterations = 500;
let acc, f1, sens, fPrec = 0.0;
let i, t, instance, total;
for (i = 0; i < iterations; i++) {
    instance = NB();
    t = instance.conMatrix;
    total = t.VP + t.VN + t.FP + t.FN;
    acc += (((t.VP + t.VN) / total) * 100);
    fPrec += ((t.VN / (t.VN + t.FN)) * 100);
    sens += ((t.VN / (t.VN + t.FP))*100);
    f1 += ((2*t.VP / (2*t.VP + t.FP + t.FP))* 100);
}
```

Código 5.11 – Saída do código. Resultado em 500 iterações

```
Acuracia em 500 iteracoes: 85.10%
Precisao fake em 500 iteracoes: 86.29%
Sensibilidade em 500 iteracoes: 86.68%
F1 Score em 500 iteracoes: 84.46%
```

Os resultados demonstram que a aplicação atinge altos níveis de precisão e um comportamento idêntico ao da aplicação desenvolvida em Python no capítulo 4 4.5.

6 CONSIDERAÇÕES FINAIS

O trabalho teve como objetivo apresentar a base teórica necessária e desenvolver e validar um classificador utilizando Naive Bayes Multinomial, o qual pode ser implementado em qualquer linguagem de programação. O classificador foi testado na detecção de notícias falsas sobre a COVID-19 no Brasil. Para o desenvolvimento desta pesquisa, as seguintes etapas metodológicas foram empregadas: levantamento bibliográfico, seleção da base de dados, desenvolvimento do classificador, etapa de treino e etapa de teste. Considerou ainda as seguintes métricas de avaliação: acurácia, precisão, sensibilidade e F1-Score. O levantamento bibliográfico, teve como base em um arcabouço teórico que envolve os conceitos probabilísticos, aprendizado de máquina, o algoritmo Naive Bayes.

Diante dos resultados das métricas em ambas linguagens de programação (Python e JavaScript), a aplicação do classificador de aprendizado de máquina Naive Bayes Multinomial, com métricas em torno de 90%, pode ser aplicado em qualquer linguagem de programação. Consideramos importante o poder ter a facilidade de gerar os próprios códigos e algoritmos, por permitir sair de ser um usuário de bibliotecas prontas, a poder desenvolver sua próprias bibliotecas. Com a possibilidade de propor e testar novas variantes do classificador.

Uma das limitações que pode ter impactado no desempenho do classificador Naive Bayes Multinomial na detecção de noticias falsas é no pré-processamento de texto na base de dados. A lista de stopwords obtida pela biblioteca NLTK (que foi a única biblioteca externa usada nesta aplicação) para o idioma português é inferior do que a de outros idiomas como o inglês, trazendo a necessidade de abrir uma pesquisa complementar para gerar uma lista de stopwords mais completa. Outro fator que podemos destacar é a qualidade e quantidade de dados na base que poderia ser maior nas etapas de treino e teste.

A continuação deste trabalho aponta a abrir a caixa preta de outras técnicas de aprendizado de máquina, de modo a ter uma maior autonomia e possibilidades de desenvolver novos métodos nesta área.

REFERÊNCIAS

BHATIA, A.; KALUZA, B. *Machine Learning in Java: Helpful techniques to design, build, and deploy powerful machine learning applications in Java*. [S.l.]: Packt Publishing Ltd, 2018.

BRASIL, L. C. et al. Comparação entre modelos com diferentes abordagens para classificação de fake news. Universidade Federal de Campina Grande, 2021.

BURKI, T. Covid-19 in latin america. *The lancet infectious diseases*, Elsevier, v. 20, n. 5, p. 547–548, 2020.

COELHO, L. P.; RICHERT, W. *Building machine learning systems with Python*. UK: Packt Publishing Ltd, 2015.

DANGETI, P. *Statistics for machine learning*. UK: Packt Publishing Ltd, 2017.

DEISENROTH, M. P.; FAISAL, A. A.; ONG, C. S. *Mathematics for machine learning*. Cambridge: Cambridge University Press, 2020.

DIAS, B. B. Os métodos bayesianos de aprendizado de máquina pelos algoritmos naïve bayes e redes de crença na predição de período chuvoso na cidade de blumenau. 2018.

FERGIEMCDOWALL. *stopword, a module for node and the browser that allows you to strip stopwords from an input text*. 2022. <https://web.archive.org/web/20221225184644/https://www.npmjs.com/package/stopword>. Acessado: 2022-12-25.

FONSECA, J.; MARTINS, G. *Curso de Estatística*. SP: Ed. Atlas, 1996.

GALHARDI, C. P. et al. Fato ou fake? uma análise da desinformação frente à pandemia da covid-19 no brasil. *Ciência & Saúde Coletiva*, SciELO Brasil, v. 25, p. 4201–4210, 2020.

GARETH, J. et al. *An introduction to statistical learning: with applications in R*. [S.l.]: Spinger, 2013.

GÉRON, A. *Maõs a Obra: Aprendizado de Máquina com Scikit-Learn, Keras, and TensorFlow*. RJ: "O'Reilly Media, Inc.", 2022.

GRANIK, M.; MESYURA, V. Fake news detection using naive bayes classifier. In: IEEE. *2017 IEEE first Ukraine conference on electrical and computer engineering (UKRCON)*. [S.l.], 2017. p. 900–903.

HARRINGTON, P. *Machine learning in action*. USA: Simon and Schuster, 2012.

HASLWANTER, T. An introduction to statistics with python. *With applications in the life sciences. Switzerland: Springer International Publishing*, Springer, 2016.

HASSAN, N. et al. The quest to automate fact-checking. In: *Proceedings of the 2015 computation+ journalism symposium*. [S.l.: s.n.], 2015.

HÜLLERMEIER, E.; WAEGEMAN, W. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, Springer, v. 110, n. 3, p. 457–506, 2021.

JOEL, G. *Data Science do Zero: Primeiros Princípios com Python*. RJ: O'Reilly Media, 2016.

JÚNIOR, J. H. de S. et al. Da desinformação ao caos: uma análise das fake news frente à pandemia do coronavírus (covid-19) no brasil. *Cadernos de Prospecção*, v. 13, n. 2 COVID-19, p. 331–331, 2020.

KOPRA, J. Machine learning with javascript. 2019.

LEAL, I. H. D. S. O uso de aprendizagem de máquina para identificação e classificação de fake news no twitter referentes a eleição presidencial de 2018. 2018.

NATURALNODE. *Natural, a general natural language facility for nodejs*. 2022. <https://web.archive.org/web/20221225183748/http://naturalnode.github.io/natural/>. Acessado: 2022-12-25.

POSETTI, J.; MATTHEWS, A. A short guide to the history of 'fake news' and disinformation. *International Center for Journalists*, v. 7, n. 2018, p. 2018–07, 2018.

ROSEN, K. H. *Discrete mathematics & applications*. NY: McGraw-Hill, 2019.

SOUZA, D. P. d. Aplicação de técnicas de aprendizado de máquina para classificar potenciais quebras na linha de produção de lentes oftalmológicas. 2022.

SUGIYAMA, M. *Introduction to statistical machine learning*. USA: Morgan Kaufmann, 2015.

WEBSTER, J. J.; KIT, C. Tokenization as the initial phase in nlp. In: *COLING 1992 volume 4: The 14th international conference on computational linguistics*. [S.l.: s.n.], 1992.

YUAN, Q.; CONG, G.; THALMANN, N. M. Enhancing naive bayes with various smoothing methods for short text classification. In: *Proceedings of the 21st international conference on world wide web*. [S.l.: s.n.], 2012. p. 645–646.