ESTUDO COMPARATIVO DOS ALGORITMOS DE APRENDIZADO DE MÁQUINA NAIVE BAYES E MÁQUINAS DE VETORES DE SUPORTE APLICADOS A ANÁLISE DE SENTIMENTO – IMPLEMENTAÇÃO COM QUANTEDA-R

Carmem Cynthia de Oliveira Buzzo[1], Júlio César Aparecido de Melo[2], Sandra Cristina Costa Prado[3]

[1] Discente do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
carmem.buzzo@fatec.sp.gov.br
[2] Discente do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
julio.melo@fatec.sp.gov.br
[3] Docente do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
sandra.costa@fatec.gov.br

RESUMO
O objetivo deste trabalho é comparar a performance preditiva entre os algoritmos naive Bayes e Máquina de Vetores de Suporte na tarefa de classificar a opinião contida em um texto sobre um determinado assunto. Na implementação deste trabalho, foi usada a linguagem de programação R e a biblioteca quanteda com a IDE RStudio. Para realizar o treinamento destes algoritmos de Machine Learning foram usados textos supervisionados com comentários de usuários sobre a compra de um produto. Por meio de uma análise da matriz de confusão foi possível concluir sob várias métricas que o algoritmo naive Bayes apresenta um poder preditivo maior além de ser mais rápido na fase de treinamento.

Palavras-chave: Análise de Sentimento 1; Aprendizado de Máquina 2; Quanteda 3. (tamanho 10)

1 INTRODUÇÃO

Processamento Natural de Linguagem (NLP) é um ramo da inteligência artificial com características fortemente interdisciplinar (MCROY, 2021). Ela envolve áreas do conhecimento tais como linguística, ciência da computação e estatística. Ela tem o objetivo de promover a interação das máquinas com a linguagem humana, programando computadores para processarem e analisarem grandes quantidades de dados textuais em linguagem natural. Entre as possibilidades de uso do NLP está interpretar o conteúdo de documentos bem como resumi-los e traduzi-los.

No presente projeto, foram usados algoritmos de Aprendizado de Máquina para classificar o conteúdo opinativo de dados textuais sobre determinado tema ou assunto. Este tipo de tarefa é chamado de Análise de Sentimento, ou Mineração de Opinão. A tarefa de extrair a opinião de um texto através de algum algoritmo de Aprendizado de Máquina pode apresentar muitas aplicações, como o monitoramento de midias sociais, gerenciamento de suporte a clientes e análise de feedback de clientes.

Em geral, o texto bruto a ser analisado deve passar por um pré-processamento antes de ser usado como dado de entrada de algum algoritmo de Machine Learning. Uma das formas mais populares de se preparar os textos para a Análise de Sentimento é a vetorização do vocabulário de um corpus textual através da técnica Bag-Of-Words.

Em geral, a Análise de Sentimento é um problema de classificação binário, em que o algoritmo de Machine Learning pode retornar como saída duas classes (positivo ou negativo) ao receber como entrada um texto já preparado. Entretanto classes adicionais podem existir para que haja um detalhamento maior sobre a opinião contida no texto. Desta forma, as classes podem ser graduadas em uma escala de intensidade que variam do muito positivo ao muito negativo ou, de forma equivalente, em intervalos numéricos representando um grau de intensidade, como é o caso dos dados utilizados neste TCC.

2 REFERENCIAL TEÓRICO

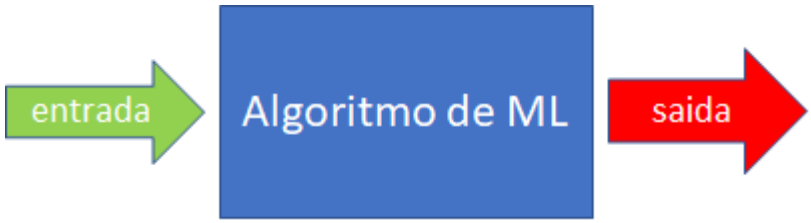
O uso de um algoritmo de Machine Learning neste problema de NLP requer que exista uma preparação dos dados textuais que servem como dados de entrada para o algoritmo. Neste caso, a manipulação dos dados textuais e, consequentemente, sua preparação ocupa papel central neste projeto. Para tanto, foi usada uma ferramenta muito versátil da linguagem R, a biblioteca Quanteda (BENOIT, 2020), que possibilita a implementação deste projeto de forma prática e rápida.

A biblioteca Quanteda, da linguagem de programação R, é própria para análise quantitativa de dados textuais e pode ser usada de forma livre com a licença GPL-3. Ela apresenta um grande número de funções bem acessíveis e intuitivas para a

realização de uma variedade de tarefas de NLP, desde a preparação dos dados textuais, análise de similaridade e distância, bem como vários algoritmos supervisionados e não-supervisionados em computação esparsa. Suas funções são, na maioria das vezes, escritas em C++ cujo código é amplamente paralelizado. Por estes motivos, o Quanteda apresenta velocidade e eficiência no processamento de grandes quantidades de dados textuais. O Quanteda foi projetado principalmente para permitir aos usuários um método rápido e conveniente de transformar um corpus de textos em uma matriz Bag-Of-Words (saco de palavras). O pacote torna mais fácil redefinir documentos da referida matriz, por exemplo, dividindo-os em frases ou parágrafos, ou por tags, bem como agrupá-los em documentos maiores por variáveis de documento, ou subdividi-los com base em condições lógicas ou combinações de variáveis de documento. O pacote também implementa funções comuns de seleção de features de NLP, como remoção de palavras irrelevantes, seleção de palavras encontradas em dicionários, tratamento de palavras como equivalentes com base em um "dicionário de sinônimos" definido pelo usuário e recursos de corte e ponderação com base na frequência das features.

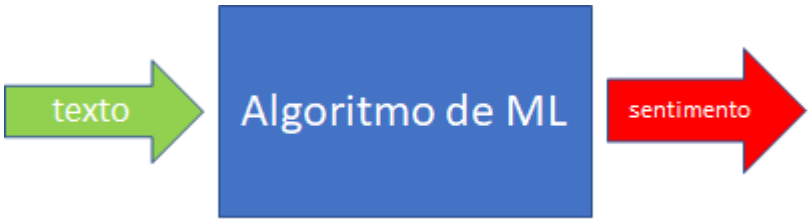
2.1 Funcionamento de um Algoritmo de Machine Learning

A utilização de um algoritmo de *Machine Learning* pode ser entendida da mesma maneira que a utilização de uma função. Ou seja, para usar o algoritmo deve-se ter incialmente uma entrada que é recebida e processada pelo algoritmo e que ao fim retornará uma saída, como mostra a figura abaixo:



No caso da aplicação de um algoritmo de Machine Learning ao problema de Análise de Sentimento tem-se como dado de entrada o texto a ser analisado, que deve ser preparado apropriadamente para ser recebido pelo algoritmo. A saída retornada

pelo algoritmo de ML será o sentimento, ou seja, a classificação do conteúdo opinativo do texto de entrada que pode assumir dois valores positivo ou negativo, conforme a figura abaixo ilustra:



Apesar do algoritmo de ML apresentar a estrutura de uma função, o algoritmo deve ser treinado antes de ser usado afim de que realize a tarefa de classificação do sentimento do texto de forma adequada. Para tanto, é necessário mostrar ao algoritmo vários exemplos com pares de entrada e saída. Ou seja, deve-se ter vários textos com os respectivos sentimentos para que o algoritmo possa aprender com eles a classificar outros que ele nunca viu. Estes dados são chamados de dados supervisionados e eles fazem o papel de professor para o algoritmo de ML.

2.2 Preparação dos Dados de Entrada

Neste projeto foi considerado um conjunto de textos com os respectivos sentimentos previamente classificados. Em uma situação real os textos a serem analisados, devem ser extraídos de algum banco de dados ou mesmo da internet usando técnicas de *web scraping*. No presente artigo, foi usado um corpo de textos brutos juntamente com informações relevantes sobre eles, que é chamado de Corpus. No pacote Quanteda (WATANABE, 2021) existem funções que constroem esta estrutura de dados, o Corpus. Nesta estrutura, o texto bruto é considerado uma grande string, ou seja, o texto bruto é simplesmente uma grande junção de caracteres.

Na sequência deve-se fazer a distinção de todas as unidades que compõe esta grande string (o texto bruto) contida no Corpus. As unidades que compõem um texto bruto, são as palavras, as pontuações, os números, emojis, etc. Estas unidades são chamadas de tokens. Portanto, esta fase é chamada de tokenização do texto bruto. A biblioteca Quanteda oferece várias funções para implementar a tokenização de um Corpus.

A próxima etapa é remover os tokens que contribuem pouco na determinação do sentimento do texto a ser analisado. Em geral, os tokens que apresentam pouco significado são as pontuações, artigos, preposições, pronomes, entre outros. Os tokens com pouco significado para a análise do texto são chamados de *stopwords*. Após a remoção das stopwords, a informação do texto original conterá apenas palavras que realmente contribuem para análise do texto. Estas palavras são chamadas de vocabulário de treinamento.

Na última etapa, é criada uma tabela cujas colunas são compostas pelas palavras do vocabulário de treinamento e as linhas correspondem à informação de cada texto a ser analisado. As entradas da tabela são os números de vezes que cada palavra aparece no texto. Esta tabela é chamada de matriz Bag-of-Words e é facilmente implementada por meio de funções que existem na biblioteca Quanteda.

Com a matriz Bag-of-Words construída a partir dos dados de treinamento, encerra-se a fase de preparação dos dados textuais. Na sequência é necessário escolher os algoritmos a serem usados e iniciar a fase de treinamento para cada um deles.

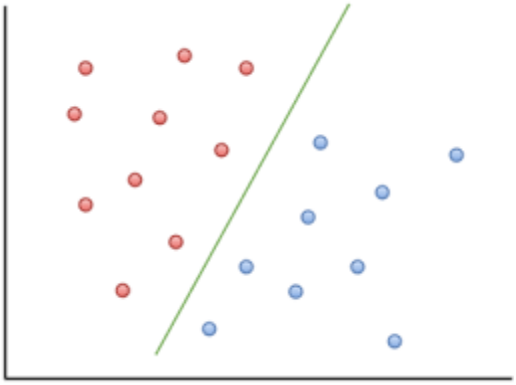
2.3 Naive Bayes

O algoritmo Naive Bayes (JURAFSKY, 2018)faz uso do cálculo de probabilidades condicionais avaliando a frequência das palavras que aparecem no vocabulário de treinamento. As probabilidades das palavras são calculadas impondo a condição de estarem em uma das classes de sentimento. Por exemplo, calcula-se a probabilidade da palavra “gostei”, dado como certo que ela apareça em um texto positivo. A fase de treinamento deste algoritmo compreende o cálculo de todas as probabilidades condicionais associadas às palavras do vocabulário de treinamento. Após a fase de treinamento usa-se uma aproximação do teorema de Bayes para inferir a probabilidade do texto nunca visto pertencer a classe das opiniões negativas e positivas. O texto analisado será classificado como positivo se a probabilidade do texto pertencer à classe positiva for maior que a probabilidade do mesmo texto pertencer à classe negativa. Caso contrario classifica-se o texto como negativo.

A linguagem R (R, 2021) tem este algoritmo implementado dentro da biblioteca *quanteda.textmodels*, já adaptada para a entrada da matriz bag-of-words.

2.4 Máquinas de Vetores de Suporte

O algoritmo Máquina de Vetores de Suporte (SVM) calcula uma superfície de separação entre os textos positivos e negativos (JAMES, 2021). Esta superfície se encontra dentro do espaço das features, onde cada texto é um ponto neste espaço. Conforme a figura abaixo ilustra, a divisão do espaço entre dados de uma classe (bolinhas vermelhas) e outra classe (bolinhas azuis) é feita por uma linha, sendo esta linha o que o algoritmo SVM calcula.



A superfície criada por este algoritmo pode ser linear (que corresponde a um plano no espaço das features) ou ainda pode ser não-linear (que corresponde a uma superfície com curvatura no espaço das features). No caso deste projeto, optou-se por usar SVM linear o qual está implementado no pacote quanteda.textmodels pela função textmodel_svmlin().

3 METODOLOGIA

A pesquisa e parte prática deste trabalho se deu utilizando a IDE RStudio que permitiu escrever os códigos em linguagem R para as três fases do projeto: a fase de preparação do dados, a fase de treinamento e teste dos algoritmos e ao fim deste projeto, a análise da performance dos algoritmos.

A codificação deste projeto não é extensa, visto que foram usado várias funcionalidades das bibliotecas específicas para cada fase do projeto o que facilita a codificação.

Inicia-se o código deste projeto de Análise de Sentimento carregando todos os pacotes necessários, conforme é mostrado no trecho de código abaixo:

```
1  library(quanteda)
2  library(quanteda.textmodels)
3  library(caret)
```

As bibliotecas utilizadas neste projeto foram quanteda (que permitiu a preparação dos dados), a biblioteca quanteda.textmodels (a qual contém os algoritmos de ML usados neste projeto) e a biblioteca caret (com funcionalidade para análise da performance dos algoritmos).

Na sequência inicia-se a fase de preparação dos dados textuais. Antes, são carregados os dados textuais brutos amazona_baby.csv que contém milhares de comentários de usuários sobre a compra de um produto (fraldas de bebê), sendo que os textos vêm acompanhados da classificação do sentimento, ou seja, se aquele texto opina sobre o produto de forma favorável (sentimento positivo) ou contrária (sentimento negativo). O trecho de código abaixo mostra a conversão do arquivo .csv para a estrutura de dados própria do R chamada de data frame.

```
7  data_raw <- read.csv("amazon_baby.csv")
8
```

Na sequência, é feita a separação dos dados para o treinamento dos algoritmos, deixando 30% de dados restantes para realização do teste da aprendizagem, onde é avaliada a performance do algoritmo. O trecho de código abaixo, mostra as linhas de código associado ao procedimento explicado acima:

```
10 ## 70% dos dados para treinamento
11 smp_size <- floor(0.7 * nrow(data_raw))
12
13 ## definindo a semente do sorteio, para fins de reprodutibilidade
14 set.seed(123)
15
16 ## sorteando as linhas do data frame reservadas para treinamento
17 train_ind <- sample(seq_len(nrow(data_raw)), size = smp_size)
18
19 # separando os dados de treinamento e teste
20 df_train <- data_raw[train_ind, ]
21 df_test <- data_raw[-train_ind, ]
```

Com os dados separados utilizamos o *workflow* proposto pelo pacote quanteda para obter a matriz bag-of-words. Deve-se realizar uma sequência de transformações sobre os dados de treinamento e teste. A primeira etapa consiste em criar o Corpus

que é uma estrutura de dados própria da biblioteca quanteda em que o texto é considerado uma única string acompanhada da correspondente classificação do sentimento. Na sequência, o ocorre a tokenização do texto, ou seja, passa-se a identificar todos os elementos que compõe o texto tais como palavras, pontuações, endereços, números etc. Nesta etapa, são removidos as pontuações, endereços de internet, números e deixamos apenas as palavras. Na última etapa, é criada a matriz bag-of-words com a função dfm() da biblioteca quanteda. Nesta etapa, foram removidas as palavras com pouco significado, as chamadas *stopwords*. Além disto, as palavras que apresentam mesmo radical foram contabilizadas como mesma feature.

Todas as etapas de preparação de dados explicadas acima estão no trecho de código abaixo:

```
#criando o corpus de treinamento e teste
corpus_train <- corpus(df_train, text_field = "texto")
corpus_test <- corpus(df_test, text_field = "texto")

#criando os tokens de cada um dos corpus
toks_train <- tokens(corpus_train, remove_punct = TRUE, remove_symbols = TRUE,
                     remove_numbers = TRUE, remove_url = TRUE)

toks_test <- tokens(corpus_test, remove_punct = TRUE, remove_symbols = TRUE,
                     remove_numbers = TRUE, remove_url = TRUE)
```

A próxima fase do projeto é a fase de treinamento dos algoritmos. Nesta fase, são usados os dados textuais de treinamento, preparados como uma matriz bag-of-words. Foram usados dois algoritmos, o Naive Bayes e a Máquina de Vetores de Suporte, implementados pelas funções textmodel_nb() e textmodel_svmlin() da biblioteca quanteda.textmodels. O trecho de código abaixo mostra a implementação desta fase:

```
#a aprendizagem do algoritmo Naive Bayes
tmod_NB <-textmodel_nb(bow_train,bow_train$sent)

#a aprendizagem do algoritmo Support Vector Machine
#bow_train_trimmed <- dfm_trim(bow_train, min_docfreq
tmod_SVM <- textmodel_svmlin(bow_train,bow_train$sent)
```

Na próxima seção será mostrada a análise da performance dos algoritmos bem como o código que implementa esta análise.

4 RESULTADOS E DISCUSSÃO

Após o treinamento do algoritmo inicia-se a fase de teste do algoritmo, onde é possível realizar as predições de sentimentos em textos nunca vistos pelo algoritmo. Nesta fase, foram usados os dados de teste que foram reservados exclusivamente para este fim. Antes de testar a predição dos algoritmos treinados, deve-se considerar apenas as features (palavras) contidas nos dados de treinamento. Ou seja, é apenas o vocabulário de treinamento que deve ser levado em conta ao entrar com os textos de teste no algoritmo.

O procedimento explicado acima é mostrado abaixo pelo trecho de código:

```
53 #preparando os dados de teste
54 #considera-se apenas as features que aparecem nos dados de treinamento
55 bow_teste_matched <- dfm_match(bow_teste, features =  featnames(bow_train))
```

Para avaliar os acertos e erros do algoritmo foi calculada a Matriz de Confusão. As entradas desta matriz apresentam o número de acertos e erros que o algoritmo performou. No caso deste projeto, o algoritmo pode acertar de duas formas distintas, como também, errar de duas maneiras. Cada uma destas formas, de erro e acerto, é contabilizada e colocada como entrada na Matriz de Confusão. As entradas da diagonal contabilizam os números de acertos e as entradas fora da diagonal contabilizam os números de erros. O trecho de código abaixo mostra a implementação das matrizes de confusão para cada um dos algoritmos:

```
63 #o sentimento verdaeiro dos dados de teste
64 actual_sent <- bow_teste_matched$sent
65
66 #calculando erros e acertos com a Matriz de Confusão
67 tab_sent_NB <- table(predicted_NB_sent,actual_sent)
68 tab_sent_SVM <- table(predicted_SVM_sent,actual_sent)
69
```

Após montar as Matrizes de Confusão para cada um dos algoritmos, foi usada a função confusionMatrix() do pacote caret que possibilita a análise da performance de cada um dos dois algoritmos usado neste projeto. Além de apresentar várias métricas de performance preditiva, esta função calcula o intervalo de confiança da acurácia como outras métricas estatísticas que avaliam a confiabilidade dos resultados obtidos.

O trecho de código abaixo, mostra esta etapa final do projeto:

```
70  # analise da matriz de confusão
71  confusionMatrix(tab_sent_NB, mode = "everything", positive = '1')
72  confusionMatrix(tab_sent_SVM, mode = "everything", positive = "1")
73
```

As linhas 71 e 72 apresentam as seguintes saídas com as análises de performance de cada algoritmo.

```
                 actual_sent
predicted_NB_sent     0     1
                0  1339   882
                1   998 13196

               Accuracy : 0.8855
                 95% CI : (0.8805, 0.8903)
    No Information Rate : 0.8576
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5211

 Mcnemar's Test P-Value : 0.007995

            Sensitivity : 0.9373
            Specificity : 0.5730
         Pos Pred Value : 0.9297
         Neg Pred Value : 0.6029
              Precision : 0.9297
                 Recall : 0.9373
                     F1 : 0.9335
             Prevalence : 0.8576
         Detection Rate : 0.8039
   Detection Prevalence : 0.8647
      Balanced Accuracy : 0.7552
```

```
                 actual_sent
predicted_SVM_sent    0    1
                 0  866 5109
                 1 1471 8969

               Accuracy : 0.5991
                 95% CI : (0.5916, 0.6067)
    No Information Rate : 0.8576
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0046

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6371
            Specificity : 0.3706
         Pos Pred Value : 0.8591
         Neg Pred Value : 0.1449
              Precision : 0.8591
                 Recall : 0.6371
                     F1 : 0.7316
             Prevalence : 0.8576
         Detection Rate : 0.5464
   Detection Prevalence : 0.6360
      Balanced Accuracy : 0.5038
```

5 CONSIDERAÇÕES FINAIS

As saídas mostradas na figura acima mostram que o poder preditivo do algoritmo Naive Bayes (NB) se mostrou maior quando comparado com o algoritmo Máquina de Vetores de Suporte (SVM). A acurácia (porcentagem de acerto) do primeiro foi de 88,55%, enquanto que o segundo foi de apenas 59,91%. A especificidade e sensitividade que mostram a quantidade relativa de falsos positivos e falsos negativos, mostram que o NB performou melhor que o SVM.

Como perspectiva futura, acredita-se ser interessante realizar análise de sentimento com dados textuais em português. Para tanto, é necessário obter dados por meio de técnicas de web scraping e rotulá-los manualmente. Além disto, há outros algoritmos que não foram possíveis de implementá-los e pode-se futuramente avaliá-los juntamente com os apresentados neste projeto.

Os algoritmos apresentam hiper parâmetros que não foram experimentados com novos valores. Sabe-se que o ajuste dos hiper parâmetros pode levar a uma melhor performance dos algoritmos de ML. Como perspectiva futura, deve-se levar em conta estes ajustes bem como recorrer a técnicas que permitam encontrar o melhor ajuste.

REFERÊNCIAS

MCROY, S. **Principles of Natural Language Processing**., Susan McRoy, 2021.

MITCHELL, T. **Machine Learning**. New York. McGraw Hill, 1997.

BENOIT, K. et al. quanteda: An R package for the quantitative analysis of textual data. ***Journal of Open Source Software*** 3(30), 774 2018.

JURAFSKY, D.; JAMES H.M. ***Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition***. Capítulo 4, 2018.

R. https://www.r-project.org. Acesso em 20 de novembro de 2021.

WATANABE, K.; STEFAN, M. *quanteda tutorials*. https://tutorials.quanteda.io. Acesso em 20 de novembro de 2021.

JAMES, G. et al. **An Introduction to Statistical Learning**. 2. ed. Spring, 2021.