Análise Textual de Resumo de Notícias
Relatório Final
Introdução ao Aprendizado de Máquina - 2024/1

Caio Moraes Santos Moreira[1], **Daniel Arruda Ponte**[1], **Manoel Marcelo da Silva**[1]

[1] Instituto de Computação – Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil.

{caiomsm,danielap,manoelms}@ic.ufrj.br

1. Introdução

A classificação de textos é uma tarefa fundamental no campo do processamento de linguagem natural, com aplicações que variam desde a filtragem de e-mails até a análise de sentimentos em redes sociais. Este trabalho visa desenvolver modelos de aprendizado de máquina capazes de classificar resumos de notícias de acordo com o assunto abordado, por meio da vetorização de textos que consiste principalmente na contabilização de frequência das palavras contidas nos documentos, entre outras métricas. A partir disso, a motivação para este estudo envolve a exploração de como a utilização de diferentes algoritmos — Regressão Logística e Floresta Aleatória — pode impactar na acurácia da classificação das notícias (dados textuais), realizando comparações dos resultados obtidos pelos modelos aplicados.

2. Base de dados

A base de dados escolhida para esse trabalho contém cerca de 209.527 manchetes de notícias que datam de 2012 a 2022, sendo essas notícias advindas do HuffPost — agregador de blogues americano. Diante disso, esse é um dos maiores datasets de notícias que é utilizado como benchmark para uma variedade de problemas computacionais nessa área de linguística.[Misra and Grover 2021]

Infelizmente, o HuffPost interrompeu a manutenção desse arquivo de notícias em 2018, quando esse dataset foi gerado, de modo que não é possível replicar o processo de coleta desses dados nos dias de hoje. Devido a essas mudanças no website, há 200 mil manchetes de 2012 e maio de 2018 e 10 mil manchetes de maio de 2018 e 2022.[Misra 2022]

2.1. Conteúdo

Cada instância da base de dados consiste dos seguintes atributos:

- category: categoria em que o artigo foi publicado
- headline: o título da notícia
- authors: lista de autores que contribuíram para o artigo
- link: link para a notícia original
- short_description: resumo da notícia
- date: data de publicação do artigo

Para esse trabalho, serão utilizados desde o tratamento dos dados até o treinamento do modelo os atributos headline e short_description, pois entende-se que essas features tem maior potencial de identificação das notícias, que foram juntas como um único atributo; e o atributo category como variável target que identifica a classe da instância.

2.2. Classes

Há um total de 42 categorias de notícias nesse dataset. Para exemplificar, segue abaixo as 15 categorias que contém mais notícias associadas:

- POLITICS: 35602
- WELLNESS: 17945
- ENTERTAINMENT: 17362
- TRAVEL: 9900
- STYLE & BEAUTY: 9814
- PARENTING: 8791
- HEALTHY LIVING: 6694
- QUEER VOICES: 6347
- FOOD & DRINK: 6340
- BUSINESS: 5992
- COMEDY: 5400
- SPORTS: 5077
- BLACK VOICES: 4583
- HOME & LIVING: 4320
- PARENTS: 3955

3. Tratamento dos Dados

O tratamento inicial dos dados consistiu inicialmente na execução procedimentos genéricos feitos em qualquer dataset, como a remoção de instâncias duplicadas como um todo — 13 instâncias no total — e instâncias com atributos duplicados — 476 instâncias excluídas —, exclusivamente para os atributos headline e short_description, restando 209.038 instâncias após esse processo. Após isso, foram removidas amostras que tinham algum dos atributos-chave vazios, o que reduziu para 189.426 o número de instâncias — 2 headline + 19.610 short_description vazias excluídas. Por fim, foram selecionadas as notícias com as categorias mais frequentes — 84.220 instâncias no total:

- POLITICS: 32427
- WELLNESS: 17937
- ENTERTAINMENT: 14772
- STYLE & BEAUTY: 9667
- TRAVEL: 9417

Além disso, foi feito um tratamento um pouco mais específico desse tipo de problema, que consiste na remoção de textos que possuem um número de palavras discrepante da média. Na verdade, foram concatenados os atributos-chave e foi contabilizado o número de palavras dessa nova string que identifica a notícia. A partir disso, foram removidas as instâncias que possuem uma quantidade de palavras inferior a $1.5 \times$ IQR (remoção de outliers), restando 84.219 instâncias.

4. Pré-Processamento

No área de processamento de textos, a representação dos dados tem um grande impacto no desempenho do modelo de aprendizado. Dessa forma, o dado textual foi preparado por meio dos seguintes procedimentos de acordo com a ordem assinalada abaixo:

1. Tokenização
2. Conversão para apenas letras minúsculas
3. Remoção de Non-alpha text
4. Remoção de Stop-Words
5. Stemming

4.0.1. Tokenização

O processo de tokenização utilizado consiste na segmentação de um documento por meio da quebra do texto em fragmentos, nesse caso da segmentação em palavras que são divididas por caracteres de espaço. A partir disso, os textos conseguem ser transformados em pedaços de informação que podem ser considerados como elementos discretos, o que facilita a representação e análise vetorial desses documentos. No final dessa etapa, foram encontrados 2.996.906 tokens no total e 83.152 tokens distintos.

4.0.2. Remoção de Non-alpha text e Stop-Words

Após a tokenização, foram removidos caracteres não alfabéticos — números e sinais de pontuação —, reduzindo o número de tokens total para 2.522.077 e 54.877 tokens distintos. Em seguida, foram excluídas as Stop-Words que são palavras consideradas sem importância como artigos, pronomes e verbos de ligação, o que resultou em 1.479.689 tokens totais e 54.727 tokens distintos depois da remoção.

4.0.3. Stemming

Como última etapa do pré-processamento, foi utilizado o Stemming que se trata de um processo heurístico de remoção do final da palavra (afixos de derivação), ou seja, trechos responsáveis pela derivação de uma palavra primitiva em uma outra, como going em go e cars em car, por exemplo. Nesse trabalho, optou-se por esse método de normalização devido a sua velocidade e simplicidade. Após a aplicação do Stemming, restaram 1.460.138 tokens totais e 36.403 tokens distintos para serem utilizados na próxima etapa de processamento.

4.0.4. Vetorização

A vetorização de um documento textual geralmente consiste na criação de um vetor que mapeia a ocorrência de cada token no texto, alguns métodos computam valores binários de presença ou ausência do token (One-hot) e outros capturam a quantidade de vezes que o token aparece no texto (Bag of Words). Nesse trabalho, optou-se por uma variação do Bag of Words, pois foi considerado que a quantidade de ocorrência dos tokens nos documentos é importante para o problema.

Essa variação do Bag of Words chamada *TF-IDF* tem como base uma variável de frequência tf (Term Frequency), para cada termo t em um documento d, o que resulta no valor numérico $tf_{t,d}$; e também utiliza outro valor numérico chamado Inverse Document

Frequency idf_t que é uma função do número de documentos que contém o termo t — df_t — e do número total de documentos utilizados N, de modo que $idf_t = \log \frac{N}{df_t}$.

A combinação desses dois valores $tf_{t,d} - idf_t$, retorna uma grandeza que não apenas armazena a quantidade de vezes que um termo ocorreu em um documento, mas também faz um balanceamento com a frequência de aparição em outros documentos, de forma que seu valor numérico é:

- **Maior** quando termo t ocorre muitas vezes em um número pequeno de documentos
- **Menor** quando termo t ocorre poucas vezes em um documento ou quando ocorre em muitos documentos

Dessa forma, foi utilizado o *TfidfVectorizer* da biblioteca *scikit-learn* com a seleção das 10.000 palavras com maiores valores de TF-IDF juntamente com o filtro de min_df = 0.001 para excluir palavras que não aparecem em pelo menos 0.1% dos documentos. Como resultado do tratamento e pré-processamento de dados, foi gerada uma matriz (84219, 2625) com vetores TF-IDF, correspondentes a 84219 documentos e 2625 palavras distintas para serem analisadas nos modelos.

5. Métodos Usados

Como o problema trabalhado nesse projeto trata-se de classificação de um resumo de uma notícia (mais a manchete) em uma categoria pré-definida, após o tratamento dos dados, cada instância se tornou um vetor em dimensão $\mathbb{R}_{2625}$, ou seja, 2625 atributos. Levando em consideração a característica dos dados do problema, foram escolhidos dois modelos: A Regressão Logística e a Floresta Aleatória.

Para a aplicação dos modelos, foi usado majoritariamente a biblioteca *scikit-learn* na linguagem Python, disponível no Google Colaboratory.

5.1. Regressão Logística

A Regressão Logística é um dos modelos mais simples de classificação, ela consiste na previsão de uma categoria de uma variável dependente categórica com mais de duas classes possíveis, baseando-se em uma ou mais variáveis independentes. Em termos de espaço vetorial, a regressão logística encontra um hiperplano que separa as classes, onde os vetores de entrada são projetados para determinar a classe de saída com base na função sigmoide.

Assim, por conta da característica da vetorização das palavras, é um candidato ideal para o modelo de nosso problema. Na biblioteca usada, existem diversos hiper-parâmetros a serem escolhidos para o funcionamento ideal, sendo os mais relevantes:

- 'C' - Valor de regularização - penalização, sendo usado para evitar overfit -
- 'Solver' - Algoritmo de otimização

Um dos objetivos é encontrar os hiper-parâmetros ideais para a nossa base de dados, como veremos a seguir, será feito usando a função Grid Search da própria biblioteca do *scikit-learn*.

5.1.1. Floresta Aleatória

Uma das vantagens da Floresta Aleatória, por ser um ensemble de Árvores de Decisão, é que é possível ter um entendimento melhor da escolha de decisão em nosso problema — maior interpretabilidade dos resultados. Por exemplo, seria possível analisar as palavras mais relevantes para a classificação. Além do mais, por ser um ensemble, ou seja, agrega diversos classificadores para gerar a predição, é bem mais confiável que uma árvore de decisão por ser menos suscetível a ruídos, pois diminui a variância inerente da Árvore de decisão.

Porém, por ser um modelo mais complexo, é necessário ter certos cuidados com overfit. Para resolver isso, a própria biblioteca fornece ferramentas como hiperparâmetros do modelo:

- 'n_estimators' - Quantidade de árvores de decisão
- 'min_samples_leaf': Número mínimo de instâncias para ser uma folha
- 'criterion': Critério de escolha de atributos, podendo ser Gini ou Entropia
- 'max_features': Critério para o número máximo de atributos

Da mesma maneira que o modelo anterior, será feita uma busca para identificar os melhores parâmetros que maximizam o F1 Score na fase de treino e de validação.

6. Projetos Relacionados

No kaggle existem diversos trabalhos relacionados que usam o mesmo dataset. Dentre eles, em que há maior notoriedade no site são:

- News_articles_classif (Wordembeddings&RNN) [Avikumart 2022]
- News Category Classifier (val_acc 0.65)[Heng 2018]

Ambos os trabalhos usam o tratamento com tokenização e de stopwords, porém nenhum deles limitou o número de instâncias e de classes como foi feito neste trabalho, o que aumenta bastante a complexidade e o tempo de treinamento.

Outra diferença principal é o modelo usado. No primeiro, usa-se Redes Neurais Recorrentes (RNN), que basicamente é um tipo especial de rede neural que tenta manter o contexto de uma sequencia de dados do que os dados isolados. Apesar se sua alta complexidade, conseguiu apenas 50% de acurácia - considerando todas as classes.

Já o segundo trabalho, usou tres modelos distintos: TextCNN, Bidrectional LSTM com convolução e LSTM com 'Attention' e por fim gerou um ensemble com esses três modelos. O resultado foi uma acurácia de 65%, considerando todas as classes.

7. Experimentos Realizados

Para a realização dos testes com os modelos escolhidos, foi feita uma separação dos dados em base de treino, validação e teste, com 80% dos dados reservados para treino e validação (com K-Fold estratificado igual com k = 5) e o restante para teste. O uso do K-Fold estratificado é para manter sempre a mesma proporção de classes entre o treinamento e o teste, já que a base de dados é desbalanceada.

Ambos os modelos foram utilizados na função Grid Search para encontrar os melhores hiperparâmetros que serão passados como argumentos para os modelos selecionados. Essa função (Grid Search) é uma ferramenta usada para automatizar o processo de

ajuste dos parâmetros de um algoritmo, realizando uma busca exaustiva sobre as opções de parâmetros passadas. Dessa forma, o Grid Search treina e avalia um modelo com base num conjunto de dados passados, que consiste em 80% dos dados totais; além disso, foi executado um K-Fold estratificado com k = 5 sobre esses 80% para realizar essa busca, de forma que cada rodada do K-Fold utiliza cerca de 64% dos dados originais para treinamento e 16% para teste.

Como resultado, o método executa uma avaliação do *score* médio de teste de cada combinação existente — média das 5 rodadas de teste do K Fold —, com o intuito de retornar os parâmetros que maximizam esses *score*. Após isso, os parâmetros retornados são passados para os modelos finais que executam seu treinamento com K-Fold estratificado (k = 5) nos dados originais, passando ao final para uma etapa de avaliação desses modelos.

No caso da Floresta Aleatória, caso seja feito o treinamento sem nenhum limite em relação à podas das árvores geradas, ocorre um overfit, com uma acurácia de treino igual a 100%, enquanto o teste fica na margem de 84%. Por esse exato motivo, foi realizado o Grid Search neste modelo para identificar a melhor poda possível para evitar essa situação, além de diminuir o tempo de treinamento.

8. Resultados

8.1. Regressão Logística

Após os testes, dentre os 24 candidatos - 120 fits, os melhores parâmetros encontrados foram:

- 'C': [0.5, 1, 1.5, 2, 2.5, 3] = '2'
- 'solver': ['newton-cg', 'sag', 'saga', 'lbfgs'] = 'saga'

Tais parâmetros forneceu uma média no Cross-validation estratificado com K-Fold(5) — com 80% dos dados para treino e validação — de 92% acurácia de teste e 88% na acurácia de treino. Além disso, foi obtido o F1 Score (Macro) de 0.87, um valor bastante aceitável, o que é uma ótima taxa de acerto.

Por fim, ao treinar o modelo com todos os 80% dos dados e testar com os 20% separados inicialmente, é obtido o seguinte resultado:

- Acurácia de treino e teste são idênticas ao feito no cross-validation, com 92% e 88%, respectivamente.
- Recall = 86%
- F1 Score (Macro) = 0.87

Ao analisar a matriz de confusão observamos uma ótima taxa de acerto e precisão mesmo em uma base de dados desbalanceada. Além disso, o comportamento exibido pelas curvas de aprendizado sugere que o aumento do tamanho do conjunto de treinamento está ajudando a melhorar a performance do modelo de regressão logística. A curva de treinamento está diminuindo, indicando uma redução do overfitting, enquanto a curva de teste segue aumentando, mostrando uma melhora na generalização. Esse resultado é esperado devido ao grau de flexibilidade conhecido da Regressão Logística, confirmando que se trata de um modelo com baixa variância que se beneficia do fornecimento de mais dados.

Em comparação com os trabalhos relacionados de outros autores, é visível que, ao reduzir o número de categorias, foi possível obter um modelo muito satisfatório apesar de ser um modelo mais simples. Portanto, a Regressão Logística é uma ótima opção para resolver esse problema, principalmente quando se trabalha em uma base de dados mais comportada, sem necessitar de modelos mais complexos como foi feito nos trabalhos dos autores citados acima.

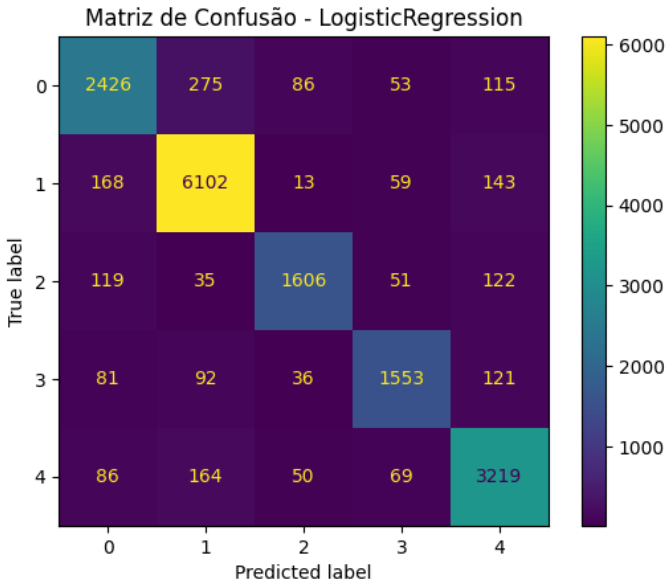

Figura 1. Matriz de Confusão da Regressão Logística

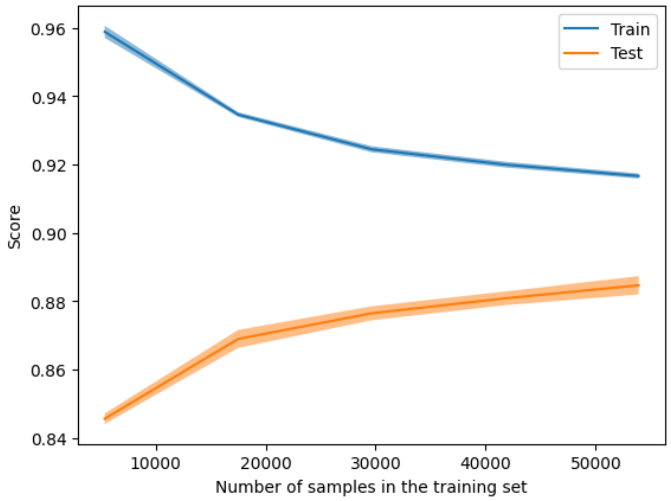

Figura 2. Curva de Aprendizado da Regressão Logística

8.2. Floresta Aleatória

Para a Floresta Aleatória, os parâmetros ideais encontrados, dentre os 36 candidatos - 180 fits, foram:

- 'n_estimators': [25, 50, 100] = 100 (Padrão - Pode ser diminuído para reduzir o tempo de treinamento)

- 'min_samples_leaf': [2, 5, 10] = 2
- 'criterion': ['gini', 'entropy'] = 'gini'
- 'max_features': ['sqrt', 'log2'] = 'log2'

Ao testar com o K-Fold estratificado (5), foi calculado uma média de acurácia de teste igual a 85% e de treino igual a 89%. Já o Score F1 (macro) foi igual a 0.82, o que demonstra que o modelo não se adaptou tão bem aos dados em relação à Regressão Logística.

Quando foi realizado o treinamento com todas as instâncias de treino, verificou-se uma acurácia similar, com uma acurácia de teste igual a 84% e 90% de treino. Além disso, também foi obtido o score de 0.83 no F1(macro), havendo uma pequena melhora na performance do modelo, mas ainda inferior ao modelo anterior.

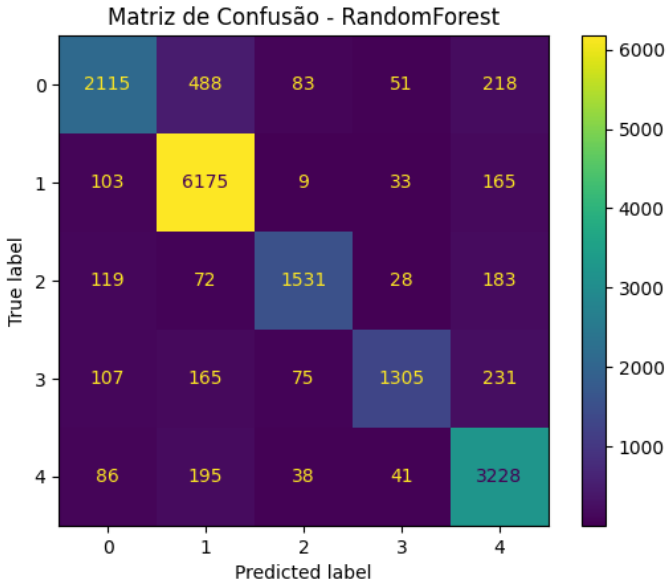

Figura 3. Matriz de Confusão da Floresta Aleatória

Observando a Matriz de confusão percebemos que há uma menor acurácia em relação à Regressão Logística, sendo visível na predição da classe 1 (POLITICS), em que a precisão é bem baixa - muito por conta da discrepância do número de instância desta classe em relação aos demais.

Ao analisar a curva de aprendizado na figura 4, percebemos que há um aumento gradativo na acurácia enquanto há mais árvores de decisão na floresta. Porém, há uma diferença significativa entre o treinamento e o teste, o que demonstra uma tendência ao overfitting do modelo, ainda mais pelo fato do score de treino aumentar juntamente com score de teste.

Como foi dito anteriormente, uma das vantagens desse modelo é seu alto grau de interpretabilidade, que possibilita a realização de análises de atributos (palavras) mais relevantes para a classificação. Na Figura 5, percebemos que as dez palavras mais importantes para determinar a classificação nas árvores geradas são, em ordem crescente: 'gop', 'democrat', 'style', 'republican', 'presid', 'fashion', 'travel', 'donald', 'photo', 'trump'. A partir desse resultado, torna-se visível a correlação existente com base de dados utilizada, já que a maior parte dos dados dessa época (2018) corresponde ao período em que

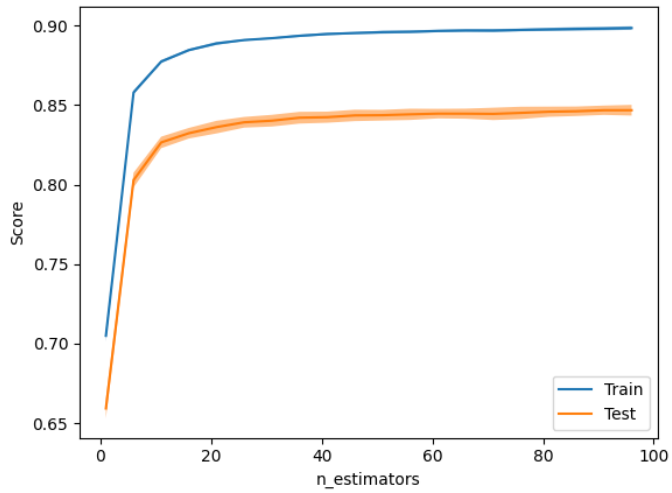

Figura 4. Curva de Aprendizado da Floresta Aleatória

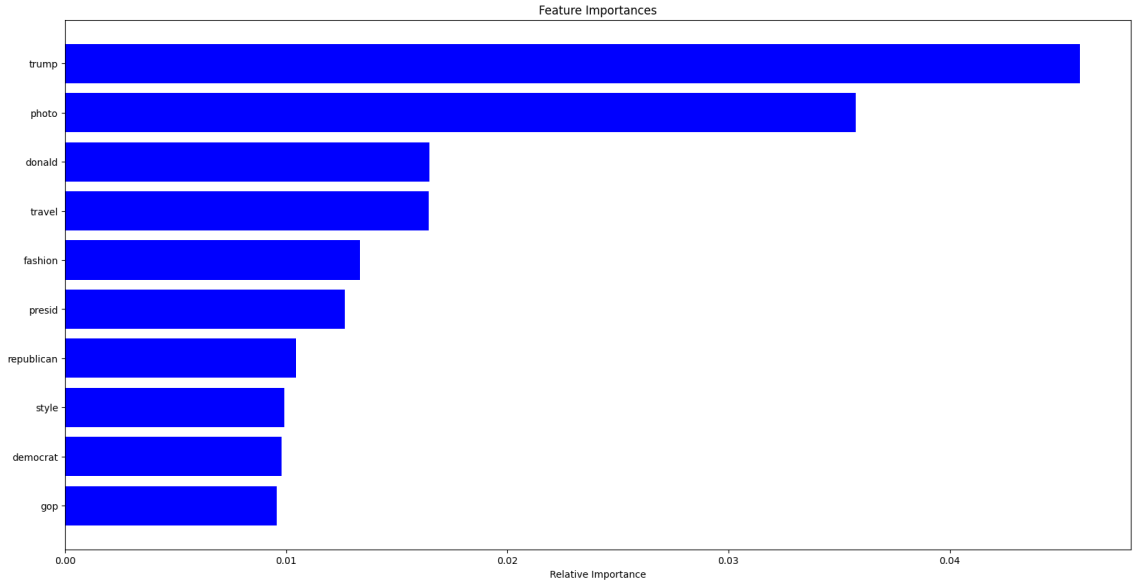

Figura 5. Palavras mais relevantes no texto

o presidente dos Estados Unidos era o republicano Donald Trump; além disso, há o fato de que as notícias do tema ”política”são os que possuem maior quantidade.

Por fim, apesar da Floresta Aleatória ser usualmente a melhor opção de modelo em razão de seu maior grau de flexibilidade, ele apresentou uma acurácia geral inferior ao modelo mais simples - Regressão Logística. Tal resultado pode ser explicado pela diferença pelo fenômeno do *Bias-Variance Tradeoff*.

Por outro lado, da mesma forma que o modelo anterior, ao comparar com os projetos de outros autores, é visível a diferença de performance quando usamos uma base mais com escopo reduzido de trabalho.

9. Conclusão

Conclui-se que o problema de classificação de textos é bastante desafiador, devido ao grande número de palavras e sua relação semântica e sintática com o texto. Nesse sentido, acreditamos que a etapa mais relevante de todo o projeto foi o tratamento desses dados não-estruturados, de forma que fosse viável a classificação dessas manchetes e resumos com um tempo e acurácia aceitáveis.

No presente estudo, os modelos de Regressão Logística e Floresta Aleatória demonstraram desempenho promissor na tarefa de classificação de resumos de notícias, principalmente quando a base de dados está simplificada e bem tratada. A Regressão Logística se destacou por sua simplicidade e acurácia, enquanto a Floresta Aleatória apresentou maior interpretabilidade, apesar de maior rigidez.

Referências

Avikumart (2022). [nlp]news_articles_classif (wordembeddings&rnn). Acessado em: 2024-06-13.

Heng (2018). News category classifier (val_acc 0.65). Acessado em: 2024-06-13.

Manning, C. D. (2008). *Introduction to information retrieval*. Syngress Publishing,. Capítulos 1, 2, 6 e 8.

Misra, R. (2022). News category dataset. *arXiv preprint arXiv:2209.11429*.

Misra, R. and Grover, J. (2021). *Sculpting Data for ML: The first act of Machine Learning*.