Explicabilidade na Detecção de Notícias Falsas

Isadora Alves de Salles[1]
Orientador: Fabrício Benevenuto[1]

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)

{isadorasalles, fabricio}@dcc.ufmg.br

Resumo. *A disseminação de desinformação se tornou um dos maiores problemas da sociedade atual, e tem impacto em diversas áreas cruciais da sociedade como política e saúde, como no caso da pandemia do COVID-19. Durante o último ano, foi compartilhado um tsunami de notícias falsas acerca da doença, sua cura, prevenção e causas. Em um momento como esse, é de suma importância que a sociedade esteja bem informada. Tendo isso como motivação principal, nesse trabalho decidimos explorar uma parte desse problema, criando um método de detecção de notícias falsas com o intuito de verificar o que distingue uma notícia verdadeira de uma falsa. Com as análises foi possível identificar que características sobre o conteúdo de uma notícia, localização da fonte e propagação em redes sociais foram importantes para diferenciar as classes. E foi encontrado maior potencial nas características que mapeiam estruturas sobre a fonte de notícias.*

1. Introdução

Redes sociais mudaram completamente a maneira como pessoas interagem com conteúdos online, em particular esses sistemas mudaram drasticamente a forma como notícias são produzidas, disseminadas e consumidas na nossa sociedade. A partir disso, muitas pessoas passaram a consumir notícias através de redes sociais ao invés de meios tradicionais. Possíveis razões para isso são: (1) é mais rápido consumir notícias em redes sociais; e (2) é mais fácil de compartilhar, comentar e discutir sobre. Porém, por ser mais barato prover conteúdos online e muito mais rápido de espalhá-los através de redes sociais, um grande volume de *fake news* (artigos com informações intencionalmente falsas) é produzido online para diferentes propósitos. Isso iniciou uma "guerra de informação" nos últimos anos, favorecendo campanhas de desinformação, reduzindo a credibilidade de veículos de informação confiáveis, e potencialmente afetando a opinião de novos leitores em assuntos críticos para a nossa sociedade [1]. A polarização política, o viés de confirmação e os próprios algoritmos usados pelas redes sociais implicaram no espalhamento de notícias falsas[1]. Que rapidamente evoluiu para um fenômeno mundial, envolvendo questões políticas, e mais atualmente, propagando desinformação sobre o COVID-19.

No início de 2020, enquanto a pandemia do COVID-19 se espalhava pelo mundo, ela foi acompanhada por um tsunami de notícias falsas, promovidas até por figuras públicas famosas, como celebridades e políticos. Em um momento tão delicado como esse, a disseminação de informações confiáveis é vital para a saúde pública e para a

[1] https://en.wikipedia.org/wiki/Fake_news

segurança da sociedade. Entretanto, notícias falsas sobre a doença, sua cura, prevenção e causas estão sendo espalhadas muito mais rápido do que os fatos, e atualmente, no Brasil temos mais de 580.000 mortes pela doença. Essa disseminação de desinformação é muito preocupante, pois afeta diretamente na maneira como as pessoas irão interpretar e responder às notícias verdadeiras, podendo colocar vidas em risco. Devido a isso, diversos estudos foram realizados com enfoque em detecção de *fake news* sobre COVID-19 [2, 3, 4].

A maioria dos esforços focados em detecção de notícias falsas reduzem esse problema a uma tarefa de classificação, onde aprendizado supervisionado é usado para separar histórias verdadeiras de falsas. Para isso, são extraídas *features* dos textos de notícias, como: a subjetividade, análise de sentimento das palavras usadas, contagem de *hashtags* usadas, entre outras, que serão utilizadas para realizar o treinamento do modelo de aprendizado de máquina. No entanto, além de conseguir construir um modelo com alta performance para detecção de *fake news*, sugere-se que as predições dos modelos sejam explicadas [5]. Nós acreditamos que explicar as decisões realizadas pelos modelos propostos é uma questão central para entender a estrutura por trás das notícias falsas, e conseguir separá-las de notícias verdadeiras.

Portanto, tendo como base o cenário atual da COVID-19 no Brasil como principal motivação para esse trabalho, o objetivo geral do projeto foi realizar uma análise de métodos de detecção de *fake news* quanto às *features* utilizadas, a fim de tentar explicar quais delas possuem mais relação com os dados, ou seja, quais as características que mais ajudam a separar notícias falsas de verdadeiras.

Para isso, foram exploradas as *features* já existentes na literatura para a tarefa de classificação de notícias falsas, e então foi seguido um processo de extração de *features* subdividido entre as categorias listadas pelos trabalhos relacionados. Essas *features* foram computadas sobre uma base de notícias e foram utilizadas como entrada para um modelo de classificação de notícias falsas. E para análise final sobre a explicação das decisões do modelo, foi utilizado o método SHAP (Shapley Additive exPlanations)[6], que tem como objetivo explicar as predições de uma instância X através da computação da contribuição de cada *feature* para a predição.

2. Trabalhos Relacionados

Existem várias pesquisas focadas em compreender melhor os impactos da disseminação de desinformação, e entender melhor esse fenômeno [7, 8, 9]. Em particular, no trabalho de Vosoughi *et al.* [9] eles mostraram que notícias falsas tendem a se espalhar mais rápido do que notícias verdadeiras. Uma outra vertente de estudos nessa área foca em propor alguma abordagem para detecção de notícias falsas [10, 11, 12, 13, 14, 15]. Para reduzir os efeitos da desinformação, pesquisadores desenvolveram diferentes modelos de predição para automaticamente tentar detectar informações falsas ou erradas. Detecção de notícias falsas ganhou bastante atenção ajudando verificadores de fatos a identificar histórias que valem a pena investigar e verificar [11, 16].

Em trabalhos relacionados a detecção de notícias falsas, modelos de aprendizado de máquina utilizando várias *features* como entrada são dominantes. Essas *features* costumam ser linguísticas (por exemplo, observando o estilo de escrita, subjetividade e autenticidade) e de contexto social (por exemplo, características e credibilidade dos usuários, conteúdo e variáveis extraídas de redes sociais, como número de compartilhamentos)

[10]. No trabalho de Pérez-Rosas *et al.* [17] foi conduzido um conjunto de experimentos de aprendizado para construir um detector de notícias falsas acurado usando *features* linguísticas. Similarmente, Volkova *et al.* [14] construíram modelos linguísticos para classificar notícias suspeitas e verdadeiras. No trabalho de Reis *et al.* [1], os autores propuseram o uso de *features* extraídas da localização do domínio, que foram criadas sob a hipótese de que algumas cidades se tornaram famosas por conta dos moradores que criavam e disseminavam notícias falsas[2].

Através de um estudo da literatura, foi possível subdividir as *features* que já foram exploradas em trabalhos passados nas seguintes três categorias:

1. Extraídas do conteúdo da notícia, por exemplo a partir do uso de técnicas de processamento de linguagem [14];
2. Extraídas do veículo (ou fonte) no qual a notícia foi publicada, como confiabilidade, credibilidade e localização [18];
3. Extraídas do ambiente, que normalmente envolve sinais de repercussão e espalhamento, obtidos por redes sociais [19].

Modelos como esses performam decisões que usualmente são difíceis de explicar. Porém, compreender porque um modelo fez uma decisão específica é essencial em qualquer cenário de detecção de notícias falsas, podendo ajudar as verificações de fatos, fornecendo entendimento sobre o que contribui para a decisão de classificar uma notícia como falsa. Entretanto, poucos estudos focam em melhorar a confiança na previsão do modelo incorporando uma explicação das *features* utilizadas [5]. Existem duas principais abordagens para prover explicações para modelos de aprendizado de máquina: *example-based* e *feature-based* [5]. Na abordagem baseada em *features*, que é o método que utilizamos neste trabalho, cada *feature* é caracterizada pelo valor de importância para uma predição particular [6].

No trabalho de Reis *et al.* [1], foi analisada uma quantidade grande e diversa de *features* utilizando SHAP para os dados da eleição de 2016 nos EUA. Eles descobriram que certos tipos de notícias falsas tendem a ser identificadas por modelos com uma combinação específica de *features*. Isso indica que técnicas que utilizam *ensemble* para combinar modelos de diferentes grupos são promissoras. Ayoub *et al.* [20] utilizaram SHAP para explicar as decisões de um modelo baseado em técnicas de processamento de linguagem natural no contexto de notícias falsas sobre o COVID-19.

Nosso trabalho se diferencia dos trabalhos relacionados na base de dados utilizada, que é composta apenas por notícias disseminadas no Brasil, principalmente escritas em português, e que estejam em um formato de texto, ou seja, que foram publicadas por portais de notícias, e não em redes sociais. Além disso, ao invés de utilizar apenas notícias verificadas como falsas, nós criamos uma abordagem, que será detalhada na próxima seção, para buscar por notícias de sites de baixa credibilidade. Também não encontramos nenhum trabalho similar acerca de explicabilidade utilizando SHAP considerando textos escritos em português. Por fim, algumas das *features* propostas neste trabalho não foram encontradas na literatura e possuem bastante potencial nessa tarefa.

[2] `https://www.bbc.com/news/magazine-38168281`

3. Metodologia

Para realizar análises de explicabilidade na detecção de notícias falsas são necessários três passos principais: (1) construção da base de dados; (2) extração de *features*; (3) construção do classificador de notícias. Nessa seção serão apresentados os desafios e a implementação desses passos em mais detalhes.

3.1. Base de dados

Neste trabalho nós focamos em construir uma base contendo notícias que foram disseminadas por veículos de comunicação online (como jornais e revistas). Dessa forma, as notícias falsas devem se parecer visualmente com notícias publicadas em veículos confiáveis. Ou seja, não tratamos de *fake news* espalhadas em redes sociais, por meio de *tweets* ou imagens no WhatsApp, por exemplo. Além disso, o trabalho tem como foco principal notícias disseminadas no Brasil.

Então, para estudar as diferenças entre notícias verdadeiras e falsas, a base de dados construída deveria conter uma quantidade representativa de notícias dessas duas classes. Entretanto, a obtenção de uma base de dados composta por notícias falsas é complexa. Para afirmar que as notícias coletadas são falsas, precisamos realizar um *match* entre a notícia e a verificação publicada por uma agência verificadora. Nossa abordagem inicial era essa, porém não conseguimos um número razoável de notícias, então partimos para uma abordagem de separação de sites de alta e de baixa credibilidade.

No nosso trabalho, consideramos que sites de baixa credibilidade são sites que já publicaram alguma notícia que foi verificada como falsa por organizações reconhecidas de checagem de fatos, como: Aos Fatos[3] e Agência Lupa[4]. Parte desses sites foram extraídos a partir de uma base de dados do IFCN (International Fact-Checking Network)[5], que contém notícias falsas sobre o COVID-19 e o link para a checagem referente a essa notícia. Sobre a base da IFCN selecionamos apenas os sites com notícias falsas disseminadas no Brasil, e para incrementar a base de dados foram aplicadas algumas técnicas de *matching* entre título de uma notícia falsa e títulos de reportagens postadas por agências de checagem. Assim, conseguimos 207 fontes consideradas de baixa credibilidade, e foi realizada uma coleta extensiva em busca de todas as notícias publicadas por esses sites. Sobre essa coleta, foram selecionadas as top-10 notícias mais populares de cada site de baixa credibilidade, com base da propagação das URLs dessas notícias numa base pré-computada de usuários do Twitter. Dessa forma, obtemos um conjunto de 2.045 notícias consideradas falsas.

Seguindo com essa abordagem, as notícias verdadeiras foram extraídas de sites considerados confiáveis, ou seja, sites de alta credibilidade, listados pela Associção Brasileira de Jornais (ANJ)[6]. A partir dos 97 associados a ANJ foi feita uma coleta similar à dos sites de baixa credibilidade, procurando por todas as notícias publicadas por esses sites. Porém, por conta de falta de recursos computacionais não foi possível *rankear* um top-N de notícias verdadeiras mais populares. Sendo assim, a escolha das notícias

[3] https://www.aosfatos.org/
[4] https://piaui.folha.uol.com.br/lupa/
[5] https://www.poynter.org/coronavirusfactsalliance/
[6] https://www.anj.org.br/associados/

populares foi feita de forma aleatória entre todas as coletadas, a fim de reunir um conjunto de 10.000 notícias consideradas verdadeiras. Note que, o desbalanceamento entre a quantidade de notícias verdadeiras e falsas é intencional, pois acreditamos que sites de alta credibilidade publicam um volume maior de notícias, bem como notícias de sites confiáveis são muito mais compartilhadas.

3.2. Extração de *Features*

A ideia dessa etapa é extrair algumas características (*features*), acerca das notícias, que possam evidenciar as diferenças entre notícias verdadeiras e falsas. E posteriormente, essas características serão usadas como variáveis de entrada para um modelo de classificação supervisionada. Nessa etapa, foi necessária uma revisão extensa da literatura, para saber o que já foi implementado e quais os espaços em aberto. Como dito na seção de Trabalhos Relacionados, existem três categorias de *features* comumente utilizadas em tarefas de classificação de notícias, são elas:

- Conteúdo da notícia;
- Fonte;
- Ambiente.

Para esse trabalho foram implementadas *features* de todos esses três grupos. Por se tratar de um processo bastante extenso, que levou a computação de 131 *features*, essa etapa foi realizada em conjunto com outros alunos, integrantes do laboratório que eu faço parte, e em colaboração com o projeto do Ministério Público. Portanto, as *features* dos grupos de conteúdo e de fonte, não foram desenvolvidas por mim. Sendo assim irei dar uma ideia geral sobre a implementação delas, mas não irei entrar em detalhes. Focarei mais na explicação da implementação das *features* de ambiente, implementadas por mim.

3.2.1. *Features* de Conteúdo

As *features* de conteúdo recebem um texto como entrada e como saída exibem algumas características acerca da estrutura do texto, levando em conta aspectos de sintaxe, lexicais, semânticos e etc. No nosso trabalho, *features* desse grupo foram implementadas sobre os títulos das notícias. A extração dos títulos de cada uma das URLs de notícias contidas na nossa base de dados ajudou a filtrar os dados, por exemplo, em casos de a URL da base não existir mais ou não se tratar de uma notícia. Dessa forma, após a extração dos títulos a base de dados final contém: 1951 notícias falsas e 9511 notícias verdadeiras. E essa base filtrada foi utilizada para a extração das demais *features*.

Features de conteúdo podem ser subdivididas nos seguintes subgrupos:

- **Sintaxe:** *features* em nível de sentença, são medidas de qualidade do texto, como índice de legibilidade.
- **Lexicais:** *features* em nível de caracteres e palavras, como o número de palavras eínúmero de *hashtags*.
- **Gramaticais:** como quantidade de pronomes e verbos.
- **Semântica:** como indicadores de toxicidade de um texto, que mensura o quão rude, desrespeitoso ou irracional um texto é; nível de ameaça e insulto. Para essas *features* foi utilizada a Perspective API[7].

[7]https://www.perspectiveapi.com/

- **Subjetividade:** como análises de sentimentos.
- **Psicolinguísticas:** que foram inspiradas no *Linguistic Inquiry and Word Count* (LIWC) que é um *software* de análises de texto que calcula o grau de uso para diferentes categorias de palavras através de um grande conjunto de textos.

Foram computadas algumas *features* para cada um desses subgrupos, gerando no total 103 *features* de conteúdo. Na Figura 1 temos um exemplo dessas *features*. Futuramente, pretendemos computar essas *features* também sobre o corpo da notícia, porém devido ao custo computacional da execução isso não foi possível nesse semestre.

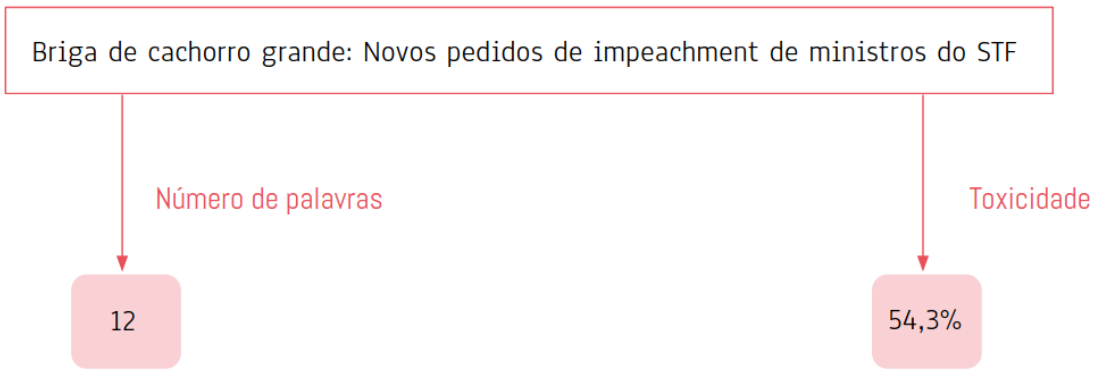


Figura 1. Exemplo *features* de conteúdo.

3.2.2. *Features* de Fonte

As *features* de fonte recebem como entrada a URL de uma notícia e como saída exibem características acerca da fonte dessa URL, ou seja, do veículo no qual a notícia foi publicada. As *features* computadas para esse grupo podem ser subdivididas em três categorias:

- **Credibilidade:** indicadores de credibilidade do domínio da URL, como o *ranking* do Alexa[8].
- **Localização:** indicadores de localização do subdomínio, como latitude e longitude.
- **Rede:** como o número de *hops* até resolver um domínio e o número de registros CAA ou TXT associados com o DNS.

É importante mencionar a diferença entre domínio e subdomínio de uma URL. Domínio é o nome de identificação de um site na internet, por exemplo, **globo.com**. Subdomínio é um endereço que faz parte do domínio, ou seja, é uma ramificação do domínio, por exemplo **g1.globo.com**.

Para ficar um pouco mais claro como as *features* desse grupo são computadas, no diagrama da Figura 2 temos um exemplo de algumas *features* de localização para uma notícia publicada no g1.globo.com.

[8] `https://www.alexa.com/`

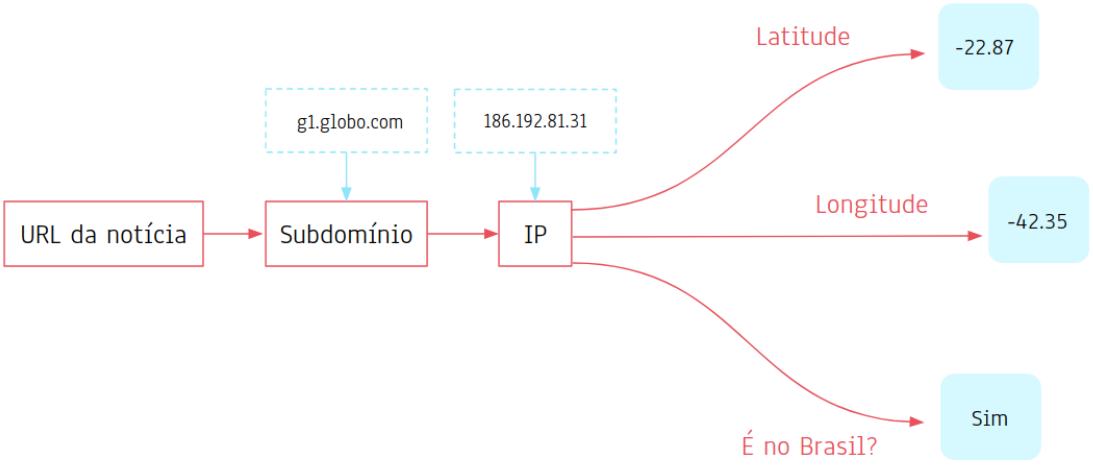


Figura 2. Exemplo *features* de fonte.

Foram computadas algumas *features* para cada um desses subgrupos, gerando no total 9 *features* de fonte. Note que as *features* de fonte sempre serão iguais para todas as URLs de notícias que possuem o mesmo subdomínio. A importância disso ficará clara na terceira etapa da nossa metodologia.

3.2.3. *Features* de Ambiente

As *features* de ambiente também recebem como entrada a URL de uma notícia e como saída exibem características acerca da propagação dessa URL em redes sociais. As *features* computadas para esse grupo foram subdivididas em duas categorias:

- **Propagação:** indicadores de propagação da notícia em diferentes ambientes, como números de curtidas ou *retweets* em um *tweet* que contém uma URL no corpo do texto.
- **Informações do usuário:** características dos usuários a nível individual e em grupo, em relação aos usuários que compartilharam as URLs, como número de seguidores e se a conta é verificada ou não.

Para a computar essas *features* usamos a API histórica do Twitter[9] para buscar por todos os *tweets*, em um dado período, que contém cada uma das URLs de notícias verdadeiras e falsas presentes na nossa base de dados. Para cada URL de notícia, essa busca retorna dois JSONs, o primeiro contendo as informações de todos os *tweets* com a URL, e o segundo contendo as informações dos usuários que postaram esses *tweets*. Em seguida, foi feito um processo de extração das informações desejadas nesses arquivos, como o número de *tweets* iniciais, soma de likes, soma de *retweets*, número médio de seguidores dos usuários que compartilharam cada notícia, e etc. Por fim, foram extraídos do segundo JSON os *usernames* dos usuários, e foi utilizada a API Botometer[10] para checar a atividade dessas contas no Twitter e obter a probabilidade das contas serem um

[9] https://developer.twitter.com/en/docs/twitter-api/tweets/search/introduction

[10] https://botometer.osome.iu.edu/

bot. No final desse processo, foram computadas 17 *features* de ambiente utilizando a API do Twitter como base. No esquema da Figura 3 podemos ver um exemplo dessas *features*.

Figura 3. Exemplo *features* de ambiente.

Entretanto, percebemos que o custo computacional desse processo seria inviável. Para cada URL, o processo demora no mínimo 5 minutos para executar. Como a base de dados contém as notícias falsas mais populares, a coleta demora ainda mais tempo para executar, pois quanto mais *tweets* contendo uma URL maior é o tempo gasto para coletá-los. Notícias verdadeiras também gastam muito mais tempo para serem encontradas, visto que comumente são mais compartilhadas. Com isso, algumas URLs gastaram mais de 3 horas para serem coletadas. Além disso, a API do Twitter está com alguma instabilidade, seja pelo número grande de dados consultados ou pelo extenso período de busca, a API estava retornando 0 *tweets* na maioria dos casos. Porém, ao testar novamente ela passava a encontrar *tweets* e em seguida voltada a retornar 0 para outras URLs. Devido a todos esses problemas, não conseguimos executar as *features* do Twitter na nossa base de dados, então elas não foram utilizadas na nossa análise final.

Para suprir esse grupo de *features* decidimos tentar coletar dados do Facebook. O CrowdTangle[11], que é a API mais conhecida do Facebook, possui uma limitação que permite coletar apenas dados de páginas que contenham pelo menos 50.000 de seguidores. Como páginas de notícias falsas normalmente são menos seguidas do que as de notícias verdadeiras, não foi possível utilizar essa API. Porém, recentemente o Facebook lançou uma nova API, que ainda está na fase beta, chamada Fort API[12], que permite a coleta de dados sobre as páginas sem essa limitação. O grupo de pesquisa que eu faço parte conseguiu acesso à essa API, e então decidimos usá-la para obter dados sobre o engajamento dessas URLs.

Como a Fort API ainda está na fase beta, não foi possível buscar pelo número de compartilhamentos das postagens contendo as URLs, visto que a API permite que a busca seja executada apenas sobre o texto das postagens e não sobre o campo "URL". Sendo assim, optamos por capturar dados das páginas dos veículos de notícias vinculados a cada URL da nossa base. Foi possível encontrar todas a página do Facebook referentes a cada site de notícia, e com isso conseguimos dados sobre o engajamento da página

[11] `https://www.crowdtangle.com/`
[12] `https://developers.facebook.com/docs/fort-pages-api/`

(número de curtidas) e se a página é verificada. Note que essas duas *features* de ambiente sempre serão iguais para todas as notícias que foram publicadas por um mesmo site.

3.3. Classificação

Nessa etapa, o objetivo foi criar um método de classificação supervisionada para detectar se uma dada notícia é falsa ou verdadeira, utilizando as *features* computadas como variáveis de entrada. A criação desse método depende de algumas decisões, como: tratamento de variáveis categóricas, divisão da base de dados em conjunto de treino e de teste, seleção de variáveis, seleção de modelos e escolha de hiperparâmetros.

O primeiro passo para tarefas que envolvam modelos de aprendizado de máquina é definir como transformar variáveis categóricas em numéricas. Existem duas opções para isso: *LabelEncoder* e *One-Hot-Encoder*. Com *LabelEncoder* é definida uma ordem para as possíveis categorias, e então são atribuídos números naturais, de 0 a N, para cada categoria. O problema dessa abordagem é que como temos diferentes números em uma mesma coluna, o modelo irá compreender o dado em algum tipo de ordem, $0 < 1 < 2$... Mas esse não é o caso, então é melhor utilizar o *One-Hot-Encoder*, que separa uma coluna que possui múltiplas categorias em múltiplas colunas binárias, com valor 1 na coluna que possui o valor da categoria. No caso de variáveis que possuem apenas dois estados possíveis (verdadeiro e falso), o *LabelEncoder* foi utilizado, pois a ideia é similar ao *One-Hot-Encoder*.

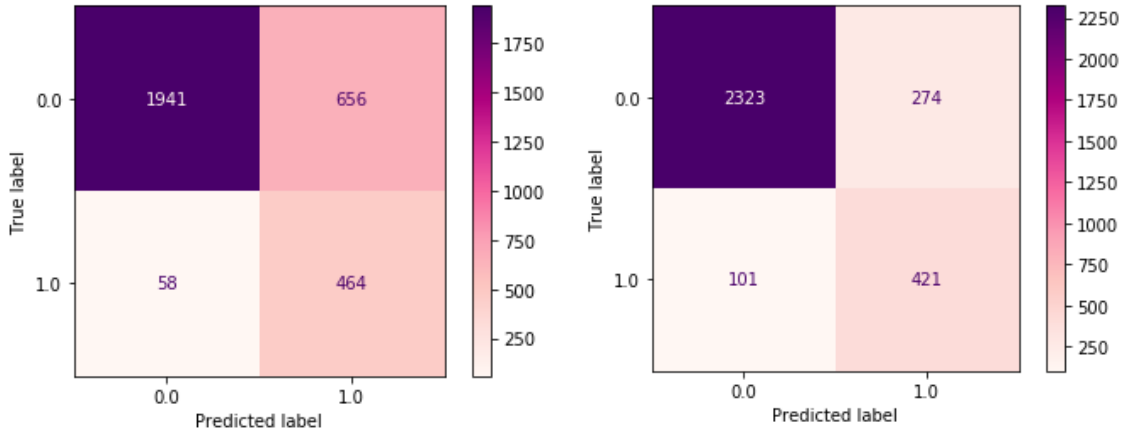
Depois de pré-processar os dados categóricos é necessário dividir a base. Como dito anteriormente, as *features* de fonte e as de ambiente são sempre iguais para as notícias publicadas por um mesmo site. Isso implica na divisão da base de treino e teste, pois se a base for separada sem levar esse fato em consideração, seria possível ter um caso de teste que já foi visto do treino. De forma que, não conseguiríamos mensurar o quão bom nosso modelo realmente é, visto que se ele apenas copiasse o resultado do treino ele iria acertar. Sendo assim, foi necessário particionar a base por site, de forma que os sites de notícias presentes no conjunto de teste nunca foram vistos durante o treinamento. Esse particionamento foi feito de maneira que o conjunto de treino possui cerca de 70% dos dados e o restante faz parte do conjunto de teste.

Para a classificação, foram usadas as 103 *features* de conteúdo, as 9 de fonte e as 2 de ambiente. Depois de realizar o *One-Hot-Encoder* nas colunas com múltiplas categorias, temos posse de 137 variáveis. Como não temos tantos dados, é necessário selecionar algumas dessas variáveis para que os modelos sejam capazes de generalizar, ou seja, não ocorra *overfitting*. Então, a seleção de variáveis para utilizar no treinamento do modelo constituiu-se dos seguintes dois passos:

1. **Mutual information:** a partir do conjunto de treino contendo as *features* computadas para cada URL descrito anteriormente, foi feito o cálculo de dependência mútua entre duas *features*. Isso será utilizado para selecionar o conjunto de *features* que terá melhor desempenho para a classificação.
2. **Abordagem gulosa:** as *features* foram ordenadas em ordem decrescente de *mutual information* e partir disso, foi seguida uma abordagem gulosa, que utiliza uma validação cruzada com cinco partições com o dado rotulado até então e um classificador *RandomForest*, para selecionar o melhor número de variáveis para treinar

o modelo. Nesse processo é realizada uma etapa de treino e validação do classificador para várias quantidades de *features*, ou seja, vários *datasets* de treino de diferentes dimensões, até que encontrar a melhor combinação. Para avaliar qual o melhor conjunto, foi gerado um gráfico com a variação do número de estimadores e as respectivas curvas de acurácia no treino e na validação para cada combinação. E o conjunto de *features* que apresentou melhor resultado é composto pelas 25 primeiras.

O *RandomForest* foi utilizado para fazer a seleção de *features* por ser um classificador bom e com custo computacional mais baixo do que os classificadores baseados em *boosting*. Porém, classificadores baseados em *boosting*, como o *Light Gradient Boosting Machine* (LGBM), demonstraram um acerto maior nos casos de teste. Então, o classificador final, utilizado para prever se uma notícia da base é verdadeira ou falsa, consiste de um LGBM. Para encontrar o melhor valor de estimadores para esse método, foi gerado um gráfico com a variação do número de estimadores (de 1 a 50) e as respectivas curvas de acurácia no treino e na validação para cada classificador, e o melhor número de estimadores encontrado foi de 35, utilizando um *dataset* com as 25 primeiras *features* com maior *mutual information*. A comparação entre os acertos nos casos de teste para o *RandomForest* e o LGBM pode ser vista nas matrizes de confusão nas Figuras 4a e 4b.



(a) *RandomForest* **(b) *Light Gradient Boosting Machine***

Figura 4. Matrizes de confusão dos modelos.

Um dos grandes desafios da construção desse método foi trabalhar com uma base de dados bastante desbalanceada. Dessa forma, não foi possível se basear apenas na acurácia para validar se o classificador obtido é bom. Nesse caso é melhor avaliar métricas como F1, precisão e revocação. Essas métricas e a AUC podem ser vistas na Tabela 1.

Modelo	Revocação	F1	Precisão	AUC
RandomForest	0.8889	0.5652	0.4143	0.8580
LGBM	0.8065	0.6919	0.6058	0.8625

Tabela 1. Comparação dos modelos.

4. Caracterização dos Dados

Antes de realizar a análise de explicabilidade sobre as decisões do classificador criado, foi feita uma caracterização inicial dos dados e um estudo breve da linguagem utilizada nos títulos das notícias para que fosse possível compreender melhor os dados que estamos trabalhando.

Primeiramente, foi feita uma análise das palavras usadas nos títulos de notícias verdadeiras e falsas. A fim de tentar capturar algum padrão, foi gerada uma nuvem de palavras com as palavras mais frequentes entre títulos de notícias de sites de baixa (Figura 5) de alta (Figura 6) credibilidade. Vale ressaltar que para essa análise foram excluídas palavras irrelevantes (*stopwords*), como artigos.

Figura 5. Palavras mais frequentes em títulos de notícias falsas.

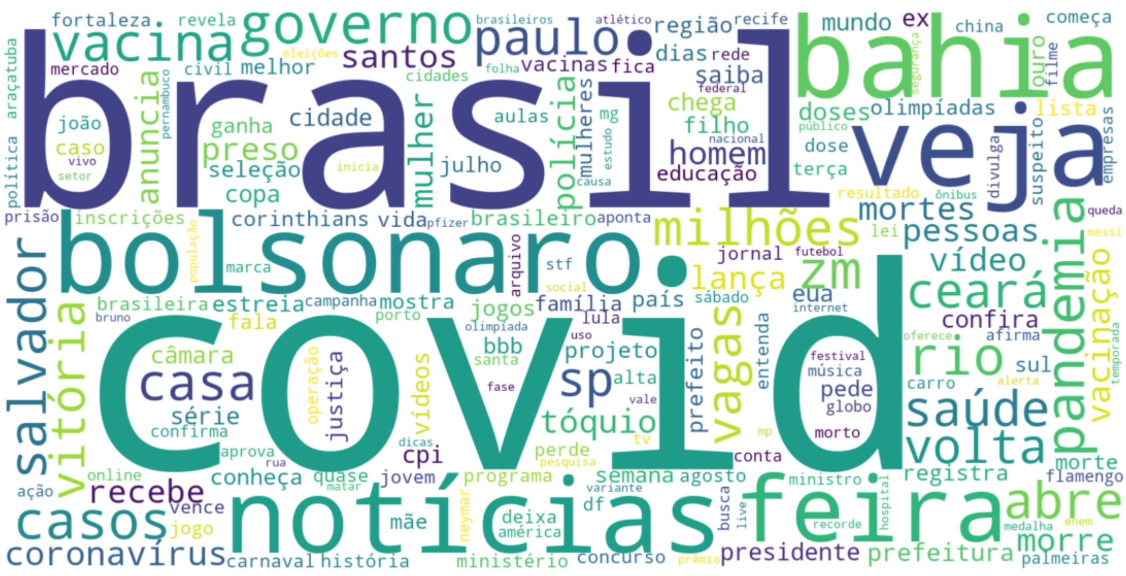


Figura 6. Palavras mais frequentes em títulos de notícias verdadeiras.

Note que na Figura 5 temos muitas palavras referentes a política, como nomes de partidos, nome de políticos e cargos. E o uso da palavra "ivermectina" nos títulos é muito mais comum do que "vacinas" ou "vacinação". Já na Figura 6 nós temos a importância dada a palavras relacionadas a pandemia do COVID-19, como "vacinação", é maior do que menções a políticos. Isso é um indício de que apenas com os títulos já conseguimos distinguir bem notícias falsas de verdadeiras.

Nós também exploramos os títulos através de uma perspectiva léxica. Primeiramente, para analisar as diferenças entre o vocabulário usado nós títulos de notícias falsas e verdadeiras, foi necessário computar o vocabulário comum entre as classes. Obtivemos os seguintes tamanhos para cada vocabulário: notícias falsas = 6.151, notícias verdadeiras = 14.852 e interseção = 3.205. A partir disso, foi calculada uma medida de associação entre as palavras no vocabulário comum e cada uma das duas classes, a fim de explorar quais palavras são mais fortemente associadas a cada classe de notícias (verdadeira ou falsa). A medida utilizada foi o *Pointwise Mutual Information*, que é definido como:

$$\text{PMI}(c, w) = log\frac{p(c, w)}{p(c)p(w)}$$

onde c é a classe (notícia verdadeira ou falsa) e w é a palavra. As probabilidades foram estimadas a partir da proporção de títulos de notícias verdadeiras e falsas, e a correspondente proporção de palavras, que pode ser obtida computando o *document frequency*, que mede em quantos documentos da coleção a palavra apareceu pelo menos uma vez.

Os resultados da associatividade das palavras estão apresentados nas seguintes nuvens de palavras, no qual quanto maior a palavra maior seu nível de associação à classe.

Figura 7. Palavras mais associadas à títulos de notícias falsas.

Figura 8. Palavras mais associadas à títulos de notícias verdadeiras.

As palavras mais associadas a títulos de notícias de sites de baixa credibilidade são relacionadas ao contexto político, como “comunista”, “comunismo”, “partido”, “conservador” e “governadores”. A palavra “urgente” também é muito associada a títulos de notícias falsas, e isso é bastante interessante, podendo indicar a maior presença de sensacionalismo nesse tipo de notícia. Note também que a palavra “ivermectina” é muito associada a essa classe. Enquanto que, para os sites de alta credibilidade, nós temos uma variedade maior de notícias, devido a isso as palavras mais fortemente associadas aos títulos dessas notícias fogem do contexto político que é muito associado aos sites de baixa credibilidade. Nesse caso, temos muitas palavras associadas à futebol.

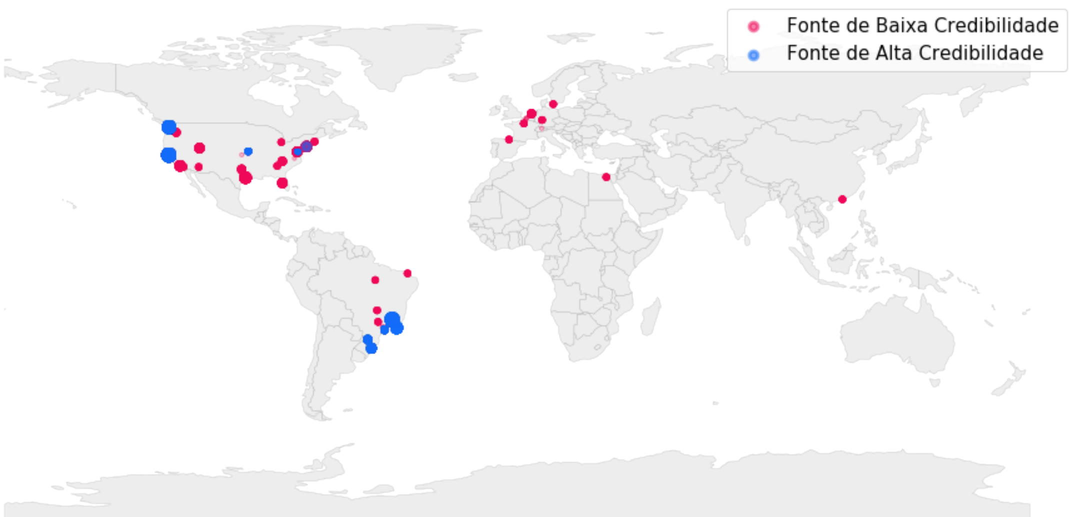


Figura 9. Mapa com as localizações das fontes de notícias.

Por fim, também foi plotado um mapa com as localizações das fontes de notícias

presentes na base de dados, usando os dados sobre a latitude e longitude do IP dos subdomínios dos sites de notícias, que foram obtidos na extração das *features* de fonte. Na Figura 9, podemos ver que com o aumento da latitude nós temos mais fontes de baixa credibilidade. Somente através dessa caracterização inicial, já podemos imaginar que essa *feature* será importante para a classificação.

5. Explicabilidade

Nesta seção, nós exploramos as decisões tomadas pelo modelo de detecção de notícias falsas criado (com o *Light Gradient Boosting Machine*). Para isso, nós usamos o SHAP (Shapley Additive exPlanations)[13], que é um método criado por Lundberg and Lee [6] para explicar predições. Esse método segue uma abordagem baseada em teoria dos jogos para explicar a saída de qualquer modelo de aprendizado de máquina. Ele conecta a alocação de crédito ideal com explicações locais usando os valores clássicos de Shapley da teoria dos jogos [21]. O objetivo do SHAP é explicar a predição de uma instância baseado na computação da contribuição de cada *feature* para essa predição.

Como estamos trabalhando com um classificador baseado em árvores, foi usado o TreeSHAP que é uma variante do SHAP para modelos de aprendizado de máquina baseados em árvores que foi proposta por Lundberg *et. al* [22].

Inicialmente, utilizamos o SHAP para explicar predições individuais, baseando-se na distribuição marginal, que nos dá a probabilidade de vários valores das variáveis no *dataset* sem considerar o valor das outras instâncias. Assim, podemos visualizar os atributos usados no treinamento do modelo como valores de Shapley, o valor de cada *feature* é representado como a força que implica no crescimento ou decrescimento do valor predito pelo modelo. Ou seja, a força que a *feature* tem para fazer com que a classe predita seja 0 (notícia verdadeira) ou 1 (notícia falsa). A predição inicia de um valor de Shapley como *baseline*, que será a média do valor de Shapley para cada predição. Então, a visualização mostra o quanto o valor de Shapley é distante da média, para uma instância. Nas Figuras 10 e 11 nós temos um exemplo dessa visualização para uma instância de notícia falsa e uma de notícia verdadeira, respectivamente.

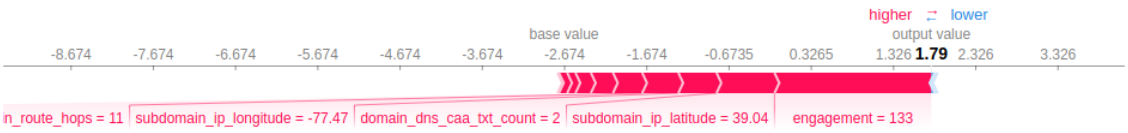


Figura 10. Força das *features* para uma instância da classe de notícia falsa.

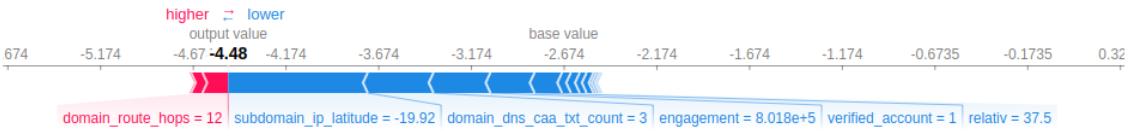


Figura 11. Força das *features* para uma instância da classe de notícia verdadeira.

Nessas visualizações podemos perceber que as *features* de latitude do IP do subdomínio da fonte de notícias e engajamento na página do Facebook são importantes tanto

[13] https://shap.readthedocs.io/en/latest/index.html

para aumentar o valor de SHAP, quanto para diminuir. Na caracterização dos dados, tivemos a intuição de que valores mais altos de latitude ajudam a prever instâncias de notícia falsa, e na Figura 10 podemos validar isso para uma instância, pois ela apresenta explicações apenas para uma predição individual. No entanto, valores de Shapley podem ser combinados em explicações globais. Ao avaliar o SHAP para cada instância da base de dados é formada uma matriz de valores de Shapley, possuindo uma linha por instância e uma coluna por *feature*. E então, podemos interpretar o modelo inteiro. Primeiramente, foi feita uma análise de importância das *features*. De forma que, atributos com valores absolutos de Shapley altos são importantes, então as *features* são ordenadas de forma decrescente pela importância, como pode ser visto na Figura 12.

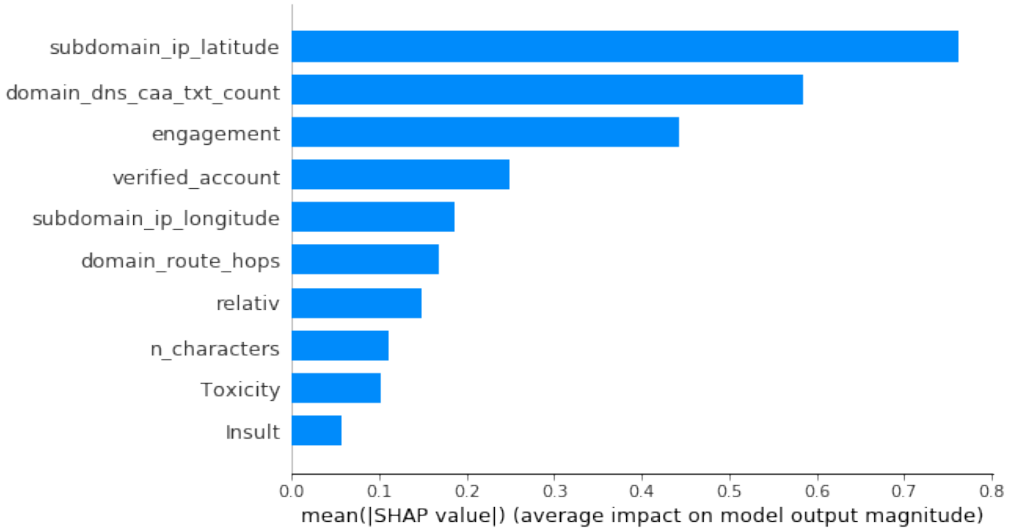


Figura 12. *Features* ordenadas pela importância para a predição.

No gráfico da Figura 12 nós temos as 10 primeiras *features* mais importantes para a classificação de notícias. Note que nossas suspeitas sobre a *feature* de latitude do IP do subdomínio da notícia foram confirmadas, essa *feature* realmente foi muito importante para diferenciar notícias de sites de alta e de baixa credibilidade. No gráfico, podemos perceber também que existem *features* de todas as três categorias (conteúdo, fonte e ambiente) entre as 10 mais importantes. A *feature* "domain_dns_caa_txt_count" (número de registros CAA ou TXT associados com o DNS table do domínio da URL da notícia), que é a segunda mais importante, nos surpreendeu muito positivamente, pois essa é uma *feature* criada pelo nosso grupo de pesquisa e que não foi encontrada na literatura.

A visualização de importância das *features* é útil, porém esse gráfico não contém nenhuma informação além disso. Uma visualização mais informativa seria o *summary plot*, Figura 13, que combina a importância das *features* com o impacto das *features* na saída do modelo. Nessa visualização, cada ponto é um valor de Shapley para uma *feature* e uma instância. A posição no eixo x é determinada pelo valor de Shapley. Cada um dos lados da linha vertical representa uma classe, para o lado esquerdo temos notícia verdadeira e para o direito notícia falsa. E a cor representa o valor da *feature* para uma instância.

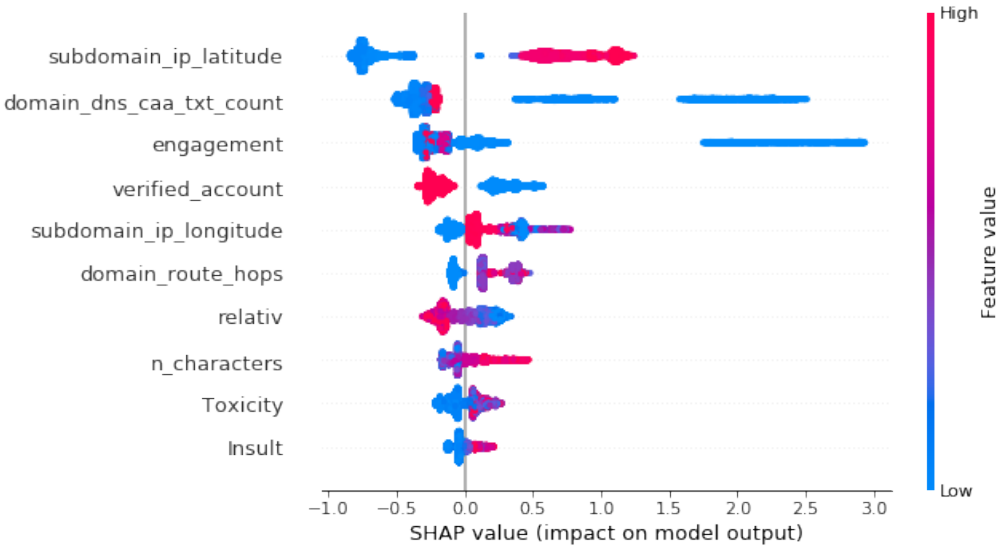


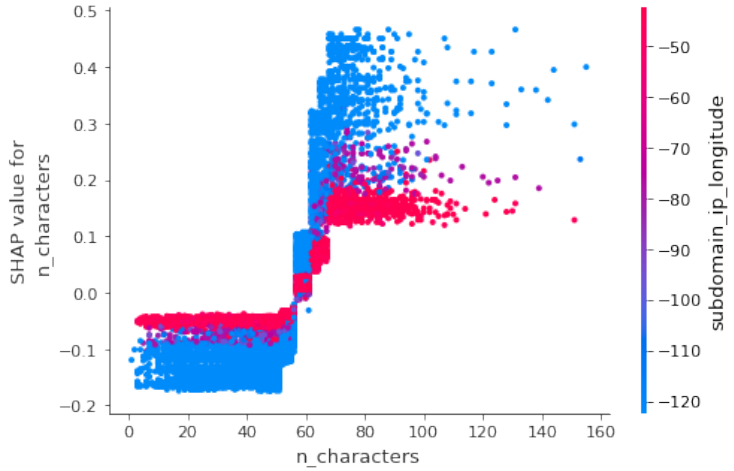
Figura 13. ***Summary Plot.***

No gráfico da Figura 13 podemos notar alguns padrões interessantes, por exemplo, que quanto mais alto for o valor da latitude mais essa *feature* ajuda a prever o valor da saída como sendo da classe de notícias falsas. Essa observação segue a intuição que foi explorada na seção de caracterização, em que é possível ver um mapa com a localização do IP dos subdomínios dos sites de notícias (Figura 9). A longitude do IP dos subdomínios também é uma *feature* importante, pois se relembrarmos o mapa na Figura 9, temos que todos as fontes de notícias verdadeiras estão localizadas na América, enquanto que existem fontes de notícias falsas tanto na América quanto na Europa, África e Ásia. Sendo assim, quanto maior a longitude mais esse valor ajuda a prever notícias falsas. Também é interessante que contas verificadas, das páginas dos jornais no Facebook, influenciam mais na predição de notícias verdadeiras, enquanto que contas não verificadas ajudam na predição de notícias falsas. Por fim, a *feature* de toxicidade dos títulos de notícias, mostra que títulos mais tóxicos influenciam mais na predição de notícias falsas.

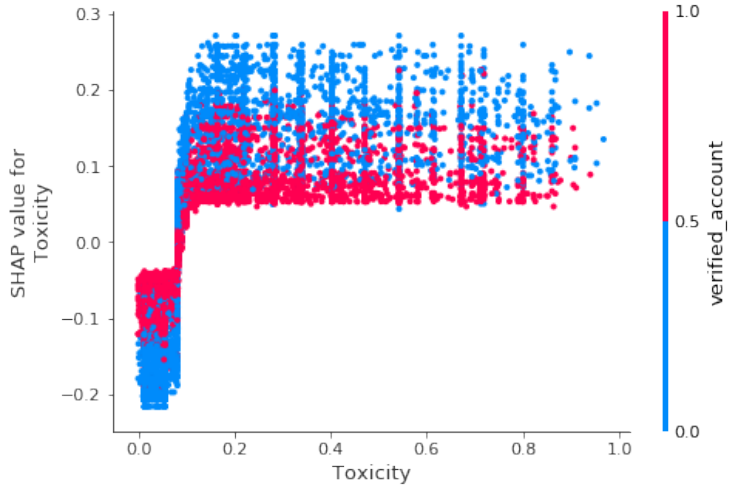
Com o *summary plot* nós temos indicações das relações entre o valor da *feature* e o impacto dela na predição. Mas para ver a forma exata dessa relação, podemos utilizar os *plots* de dependência. Com essa análise de dependência nós temos o efeito de uma única *feature* em relação ao restante da base de dados. Esse tipo de visualização pode ser melhorado utilizando uma segunda *feature* para destacar as interações.

Nas Figuras 14a e 14b, temos duas visualizações de dependência de exemplo. Em 14a, avaliamos a *feature* de número de caracteres no título em relação ao restante da base, e é usada a *feature* de longitude do IP do subdomínio da fonte de notícia para colorir as interações. Note que com o aumento do número de caracteres, o valor de SHAP aumenta, indicando que a *feature* influencia na predição da classe de notícias falsas. Porém, títulos com muitos caracteres mas que foram disseminados por fontes de notícias localizadas em uma longitude mais alta (por volta de -50), contribuem um pouco menos para a predição de notícias falsas do que os que foram disseminados por fontes de notícias localizadas em uma longitude mais baixa (como -120). Como podemos perceber na Figura 9, na América do Norte temos mais domínios de baixa credibilidade do que na América do Sul,

e a diferença entre as longitudes desses locais é a mesma expressa no gráfico 14a.



(a) Número de caracteres no título.



(b) Toxicidade do título.

Figura 14. Visualização de dependência.

Já no gráfico 14b, a *feature* de toxicidade do título está sendo avaliada usando a *feature* de conta do Facebook verificada para colorir as iterações. Note que com o aumento da toxicidade, o valor de SHAP aumenta, indicando que a *feature* ajuda mais na previsão da classe de notícias falsas. Entretanto, títulos com toxicidade alta que foram disseminados por fontes de notícias que possuem uma conta verificada no Facebook, contribuem um pouco menos para a predição de notícias falsas do que os disseminados por contas não verificadas. Isso é interessante, pois como podemos perceber na Figura 13, contas verificadas ajudam a prever a classe de notícias verdadeiras. Foram feitas outras análises de dependências, utilizando as outras *features* que estão dentre as 10 mais importantes. No entanto, esses gráficos foram os que se mostraram mais visualmente interessantes e portanto, foram os únicos listados nesse relatório.

6. Conclusão e Trabalhos futuros

Nesse trabalho nós estudamos as principais características que diferenciam notícias falsas de verdadeiras, fazendo uma análise de explicação das decisões do modelo de aprendizado de máquina criado para detecção de notícias falsas. Foi feita uma extensa revisão da literatura, e identificamos um grande número de *features* para serem implementadas para a tarefa de interesse. Foram propostas também *features* novas, que se encaixam na categoria de *features* extraídas da fonte de notícias, que apareceram entre as 10 primeiras mais importantes para as predições do modelo. A abordagem seguida revelou o quão difícil é detectar notícias falsas, necessitando uma etapa de implementação extensa de *features* para só depois conseguir validar a importância das mesmas.

Nossas descobertas sugerem que as *features* computadas realmente são relevantes para a tarefa de detecção de notícias falsas. Encontramos *features* de todas as três categorias (conteúdo, fonte e ambiente) dentre as 10 primeiras mais importantes. E a performance da detecção, utilizando as 25 *features* com maior informação mútua, atingiu um valor de AUC de 0,86. Então, compreendemos que com as *features* computadas foi possível mapear muito bem o que distingue uma notícia falsa de uma verdadeira. Além disso, não encontramos trabalhos nessa área que levem em conta apenas notícias disseminadas no Brasil, então acreditamos que esse seja um bom ponto de partida para estudos nesse campo.

Foram computadas 103 *features* de conteúdo, porém poucas delas apareceram entre as mais importantes para a predição. Até o momento, essas *features* foram calculadas utilizando como entrada apenas o título das notícias. Um possível trabalho futuro seria computar essas *features* sobre o corpo das notícias, porém isso envolve um custo computacional muito grande para a extração do conteúdo de cada URL.

Uma limitação desse trabalho foi a coleta dos dados sobre a propagação dessas URLs em redes sociais. Devido a instabilidade encontrada na API do Twitter, não foi possível computar 17 das *features* implementadas para a categoria de ambiente. Isso é um problema pois ao comparar com outros trabalhos relacionados, o nosso possui uma defasagem sobre a cobertura das *features* nessa categoria. Então, como trabalhos futuros também pretendemos encontrar uma forma de contornar esse problema, seja pelo Twitter mesmo, ou fazendo uma coleta de outra rede social.

Por fim, também acreditamos que seria interessante ampliar o conjunto de *features* atual e agrupar as notícias presentes na base de dados por categorias, a fim de construir um detector de notícias falsas específico por categoria. Como foi descoberto por Reis *et al.* [1], certos tipos de notícias falsas tendem a ser identificados por modelos com combinações específicas de *features*.

Referências

[1] Reis, Julio CS, André Correia, Fabrício Murai, Adriano Veloso e Fabrício Benevenuto: *Explainable machine learning for fake news detection*. Em *Proceedings of the 10th ACM conference on web science*, páginas 17–26, 2019.

[2] Vijjali, Rutvik, Prathyush Potluri, Siddharth Kumar e Sundeep Teki: *Two stage transformer model for covid-19 fake news detection and fact checking*. arXiv preprint arXiv:2011.13253, 2020.

[3] Paka, William Scott, Rachit Bansal, Abhay Kaushik, Shubhashis Sengupta e Tanmoy Chakraborty: *Cross-SEAN: A cross-stitch semi-supervised neural attention model for COVID-19 fake news detection*. Applied Soft Computing, 107:107393, 2021.

[4] Bang, Yejin, Etsuko Ishii, Samuel Cahyawijaya, Ziwei Ji e Pascale Fung: *Model Generalization on COVID-19 Fake News Detection*. arXiv preprint arXiv:2101.03841, 2021.

[5] Lai, Vivian e Chenhao Tan: *On human predictions with explanations and predictions of machine learning models: A case study on deception detection*. Em *Proceedings of the Conference on Fairness, Accountability, and Transparency*, páginas 29–38, 2019.

[6] Lundberg, Scott e Su In Lee: *A unified approach to interpreting model predictions*. arXiv preprint arXiv:1705.07874, 2017.

[7] Golbeck, Jennifer, Matthew Mauriello, Brooke Auxier, Keval H Bhanushali, Christopher Bonk, Mohamed Amine Bouzaghrane, Cody Buntain, Riya Chanduka, Paul Cheakalos, Jennine B Everett *et al.*: *Fake news vs satire: A dataset and analysis*. Em *Proceedings of the 10th ACM Conference on Web Science*, páginas 17–21, 2018.

[8] Lazer, David MJ, Matthew A Baum, Yochai Benkler, Adam J Berinsky, Kelly M Greenhill, Filippo Menczer, Miriam J Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild *et al.*: *The science of fake news*. Science, 359(6380):1094–1096, 2018.

[9] Vosoughi, Soroush, Deb Roy e Sinan Aral: *The spread of true and false news online*. Science, 359(6380):1146–1151, 2018.

[10] Shu, Kai, Amy Sliva, Suhang Wang, Jiliang Tang e Huan Liu: *Fake news detection on social media: A data mining perspective*. ACM SIGKDD explorations newsletter, 19(1):22–36, 2017.

[11] Reis, Julio CS, André Correia, Fabrício Murai, Adriano Veloso e Fabrício Benevenuto: *Supervised learning for fake news detection*. IEEE Intelligent Systems, 34(2):76–81, 2019.

[12] Conroy, Nadia K, Victoria L Rubin e Yimin Chen: *Automatic deception detection: Methods for finding fake news*. Proceedings of the Association for Information Science and Technology, 52(1):1–4, 2015.

[13] Tacchini, Eugenio, Gabriele Ballarin, Marco L Della Vedova, Stefano Moret e Luca de Alfaro: *Some like it hoax: Automated fake news detection in social networks*. arXiv preprint arXiv:1704.07506, 2017.

[14] Volkova, Svitlana, Kyle Shaffer, Jin Yea Jang e Nathan Hodas: *Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter*. Em *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, páginas 647–653, 2017.

[15] Wang, William Yang: *"liar, liar pants on fire": A new benchmark dataset for fake news detection*. arXiv preprint arXiv:1705.00648, 2017.

[16] Kim, Jooyeon, Behzad Tabibian, Alice Oh, Bernhard Schölkopf e Manuel Gomez-Rodriguez: *Leveraging the crowd to detect and reduce the spread of fake news and*

misinformation. Em *Proceedings of the eleventh ACM international conference on web search and data mining*, páginas 324–332, 2018.

[17] Pérez-Rosas, Verónica, Bennett Kleinberg, Alexandra Lefevre e Rada Mihalcea: *Automatic detection of fake news*. arXiv preprint arXiv:1708.07104, 2017.

[18] Li, Yaliang, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan e Jiawei Han: *On the discovery of evolving truth*. Em *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, páginas 675–684, 2015.

[19] Ciampaglia, Giovanni Luca, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer e Alessandro Flammini: *Computational fact checking from knowledge networks*. PloS one, 10(6):e0128193, 2015.

[20] Ayoub, Jackie, X Jessie Yang e Feng Zhou: *Combat COVID-19 infodemic using explainable natural language processing models*. Information Processing & Management, 58(4):102569, 2021.

[21] Shapley, Lloyd S: *A value for n-person games*. Classics in game theory, 69, 1997.

[22] Lundberg, Scott M, Gabriel G Erion e Su In Lee: *Consistent individualized feature attribution for tree ensembles*. arXiv preprint arXiv:1802.03888, 2018.