LIME Implementation

Introdução

De forma a estudar a razão de modelos de ML classificarem certas instâncias de uma determinada classe, foi-me proposto analisar como nove modelos usados para detetar mensagens fake (fakenews). Para isso foi usado um dataset com várias mensagens retiradas de grupos do WhatsApp, relativamente ao período de eleições presidenciais de 2018, utilizando apenas mensagens que são virais e não midiáticas.

Para perceber como eram as mensagens de ambas as classes (fake e real), decidi comparar antes de aplicar qualquer tipo de pré-processamento, tendo chegado à conclusão que mensagens fake têm um uso excessivo de emojis e repetitivo, usam a linguagem no modo Imperativo e seguem uma estrutura em cadeia. Por outro lado, as mensagens real, assemelha-se a conversas do quotidiano, usam emojis de forma mais natural e esporádica e não usam sinais de pontuação excessivamente.

Pré-Processamento

Nesta etapa, com o objetivo de melhorar o desempenho dos modelos, removi emojis dos textos e os sinais de pontuação separei, de modo a poder utilizar como tokens. Eliminei stopwords usando a biblioteca NLTK e também removi palavras da língua portuguesa que são consideradas stopwords, que não estão definidas na bibliotecas, manualmente pré-definidas, eliminando também a expressão de riso “kkk” e ainda converti as palavras aos seus lemas. Como existiam URL nas mensagens, decidi colocar no domínio delas apenas em vez do link completo. Para tornar o treino menos demorado, foi usada apenas a primeira centena de palavras de cada frase.

Realizei a divisão de conjunto de treino e conjunto de teste em 80%-20%, com estratificação, para cada conjunto ter em conta o balanceamento de classes.

A vetorização utilizada foi TF-IDF para ungiramas, digramas e trigramas, criando um vetor de features mais rico em contexto, tentando ajudar os modelos a realizar distinções mais subtis do que apenas contar palavras isoladas. Para aprender o vocabulário utilizei o conjunto de treino e posteriormente o vocabulário aprendido foi usado para converter os textos de treinos e de teste nas suas respetivas matrizes numéricas. As dimensões das matrizes numéricas para o treino e teste são respetivamente: (3670,53470) e (918,53470).

Modelos

Os modelos treinados foram:

- Lineares: Logistic Regression, Multinomial Naive Bayes, Linear SVM e Stochastic Gradient Descent
- Não-lineares e de ensembles: SVM, Random Forest, Gradient Boosting e MLP.
- Baseado em distância: KNN

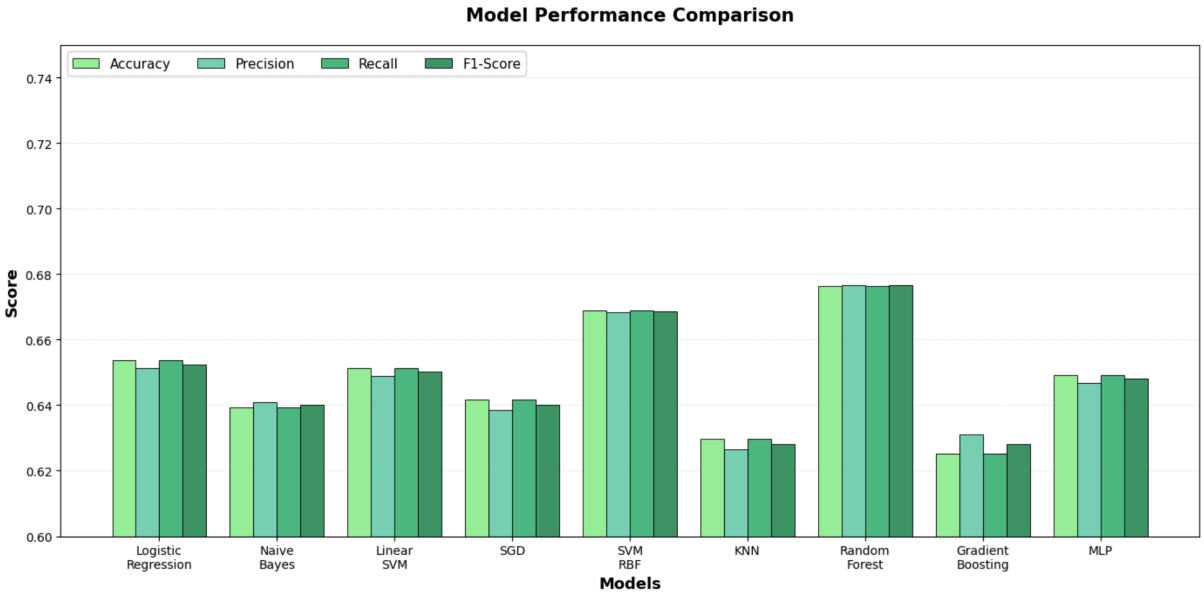

Figura 1 - Métricas de avaliação para cada modelo

Experiência 1 - Gerar explicações LIME para cada modelo e comparar as palavras mais comuns para frases real ou fake

Apliquei o lime de forma local para o conjunto de teste e depois para o conjunto de treino (para uma frase específica), para todos os modelos treinados. A frase escolhida corresponde à instância 125 do conjunto de teste com o pré-processamento: “quando alguém rua apresentar receita pedir trocadinho comprar remédio , passagem ônibus criançar colo pedir comprar leite , reflitar bem valer pena”.

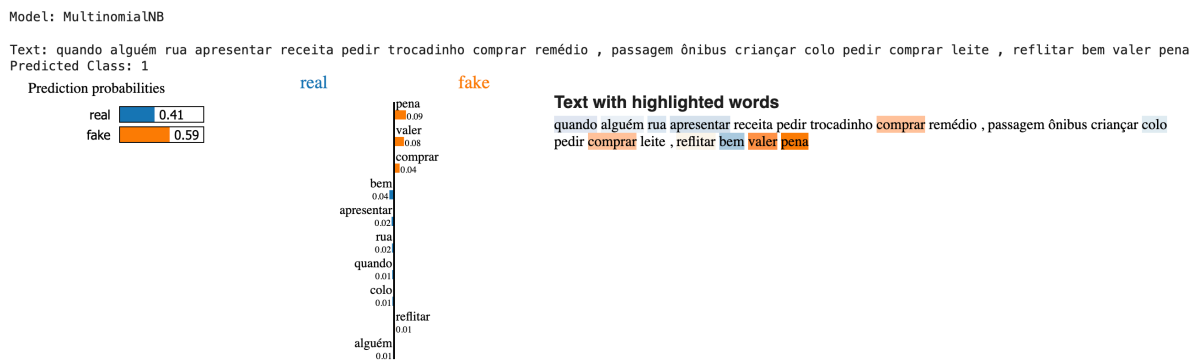

Figura 2 - Explicação gerada pelo LIME para a instância 125

index	word	avg_weight	LogisticRegression	MultinomialNB	LinearSVC	SGDClassifier	SVM	KNN	RandomForest	GradientBoosting	MLP
0	pena	0.13389546891046875	fake	fake	fake	fake	fake	fake	fake	fake	fake
1	valer	0.10542864987082276	fake	fake	fake	fake	fake	fake	fake	real	fake
2	bem	0.05611887814204143	real	real	real	real	real	real	real	real	real
3	comprar	0.04328869561456643	fake	fake	fake	fake	fake	real	fake	real	fake
4	pedir	0.03705322731541429	fake	-	fake	fake	fake	fake	fake	fake	fake
5	rua	0.030863153441764917	real	real	real	real	real	-	real	-	-
8	colo	0.025644932593609455	real	real	real	real	real	-	real	real	real
9	apresentar	0.025485496868307773	real	real	real	real	real	real	-	-	real
6	quando	0.024525232656508727	real	real	real	real	real	fake	real	real	real
7	alguém	0.021553085735319583	real	real	real	-	real	real	real	real	fake
10	reflitar	0.02002646086146859	-	fake	-	fake	-	fake	-	fake	fake
11	passagem	0.019831840016617145	-	-	-	-	-	real	real	real	-

Tabela 1 - Peso médio atribuído a cada palavra da instância 125 e classe prevista pelos modelos

De acordo com a tabela 1, as palavras mais influentes da instância 125 é “pena” e “valer”, em que ambas são, para maior parte dos modelos, palavras-chave para a frase ser considerada fake, respetivamente. Isto é um indicador de que todos os modelos aprenderam a palavra “pena” como palavra que determina a frase como sendo fake. Existe uma clara diferença em como os modelos usam as palavras, principalmente em modelos lineares e não-lineares, em que os primeiros associam o maior peso a features isoladas. Existe consistência entre a maior parte para as palavras que foram associadas à classe fake “pena”, ”comprar”, ”pedir” e “reflitar”, que também se verifica na classe real em que as palavras “bem”, “rua”, “colo”, “apresentar” e “quando” são associadas a essa classe pela maioria dos modelos.

A presença de um elevado peso na palavra “pena”, representa uma enorme disparidade em relação às restantes o que pode indicar um risco de sensibilidade dos modelos para esta instância, no caso da palavra ser removida da frase a previsão de todos os modelos, mudaria drasticamente.

Ao analisar as palavras que os modelos receberam após o pré-processamento, nenhuma contém stopwords e está de acordo com o que foi realizado no pré-processamento pelo que foi bem implementado.

Experiência 2 - Abordagem Global (com base em explicações locais)

De modo a tentar perceber as expressões textuais que os modelos aprenderam, decidi avaliar a interpretabilidade global usando 300 instâncias (para não ser tão pesado computacionalmente) provenientes do conjunto de teste. Comecei por gerar as explicações locais para cada instância, obtendo os valores dos pesos que o LIME associa a cada palavra, tendo de seguida agregar os dados obtidos num dataframe

(Model, Word, Weight, Sample_idx, Dominant_class) para posteriormente calcular o peso absoluto médio de cada palavra para o modelo, o peso médio e o número de ocorrências (frequência com que a palavra apareceu nas dez principais explicações).

word	avg_weight	n_occurrences
tocantim	-0.205240	2
antagonista	-0.192396	4
oi	-0.154682	2
recebi	-0.122134	1
carreato	-0.114175	1
extensa	-0.102979	1
manhã	-0.101484	2
ficha	-0.100519	1
super	-0.100491	1
que	-0.096774	2

word	avg_weight	n_occurrences
opinionstage	0.315092	1
03	0.221504	1
onlinar	0.202118	1
27	0.162448	1
publicar	0.146243	3
mentira	0.134562	1
olher	0.130915	1
registrar	0.130124	3
absurdo	0.120286	2
erraddad	0.113999	1

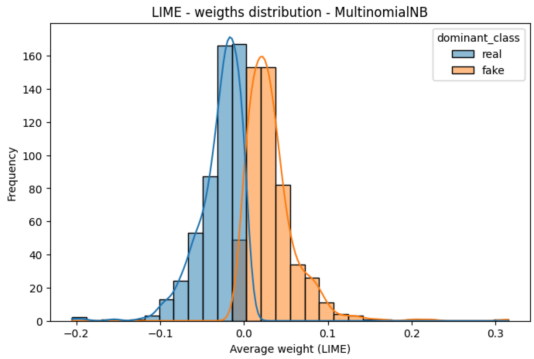

Figura 2 - Dataframe com as o número de ocorrência de cada palavra e o seu peso médio determinado pelo LIME para todos os modelos com base nas 300 instâncias fornecidas

index	model	total_abs_weight	max_word	max_abs_weight	concentration_ratio
4	MLP	77.02503697556905	oantagonista	0.38659249216608005	0.005019049744678476
2	LinearSVC	46.43121706685967	opinionstage	0.28408367176614	0.006118376594717889
7	SGDClassifier	59.86133447671266	opinionstage	0.43742756355914236	0.007307347345043085
3	LogisticRegression	21.280702555823762	plantaogospel	0.17301408978465133	0.008130092948331896
6	RandomForest	37.83761696203113	whatsapp	0.36658996870113936	0.009688505728809534
1	KNN	58.18626570303039	recebi	0.5668506671906596	0.009742001146520346
5	MultinomialNB	32.29949419871754	opinionstage	0.31590357553868376	0.009780449613084864
0	GradientBoosting	24.28640036847268	opinionstage	0.3434296573231383	0.014140821699083924

Tabela 2 - Proporção de Concentração de Peso ordenada do modelo mais estáve para o mais instável

Ao verificar a proporção de concentração, os modelos lineares e o MLP demonstram ser os mais robustos com o poder predicativo mais distribuído (menor proporção de concentração). Esta métrica é importante porque consegue-nos dar uma visão geral, de caso a palavra mais influente (a que o modelo atribui mais peso para classificar) seja retirada da frase, se a previsão mudaria drasticamente. No exemplo do MLP, que é o modelo mais estável, significa que a decisão baseia-se num vasto conjunto de palavras e não colapsaria caso “oantagonista” fosse removida. Além disso a palavra mais dominante “opinionstage” aparece na maioria dos modelos (quatro em nove). Esta palavra ter sido a mais dominante pode representar um pré-processamento com falha, e uma vez que não foi processada, pode ser por essa razão que é uma feature predicativa forte. Verifiquei e aparece na linha de índice 249 do conjunto de teste, correspondendo: “www.opinionstage.com", que pertence à classe fake.

Para estes 300 exemplos o modelo não aprendeu a semântica de notícias falsas, mas se o texto mencionar a palavra “opinionstage”, é quase garantidamente da classe fake, tendo ocorrido overfitting. Isto aconteceu porque a função que processa os URL converte para o seu domínio, podendo ser a justificação do porquê do Gradiente Boosting e o Multinomial NB obtiveram uma alta proporção de concentração de peso. O mesmo acontece para a palavra “oantagonista”, que vem de um URL, aparecendo já em mais textos (índices 85, 90, 185, 268, 338, 340, 513, e 689) em que todos pertencem à classe real.

Para evitar que os modelos façam overfitting, como nos casos anteriores, em vez de substituir os URLs pelo seu domínio, removi-os por completo e além disso defini um número mínimo de frequência no corpus para as n-gramas, que só serão vetorizadas se aparecem menos de cinco vezes no corpus, permitindo ao modelo concentrar-se em palavras mais frequentes e não apenas em casos específicos. Ainda defini mais stopwords que encontrei no corpus que não têm valor semântico relevante e adicionei palavras que estavam escritas incorretamente (apenas as que verifiquei).

A performance dos modelos diminuiu, mas representa um sinal positivo, pois o modelo está a aprender aspetos textuais mais comuns em vez de aspetos específicos, deste modo reduz a possibilidade de ocorrer overfitting. Estes resultados confirmam que os modelos estavam a aprender atalhos que influenciavam a previsão, como por exemplo os domínios “opinionstage” e “oantagonista” que tinham um peso elevado. Deste modo, pretendo chegar a um equilíbrio entre accuracy e estabilidade verificando essas melhorias na tabela 3, em que foi recalculado o concentration_ratio.

	model	total_abs_weight	max_word	max_abs_weight	concentration_ratio
4	MLP	62.55042272856489	forma	0.4264081012421859	0.006817029887912464
2	LinearSVC	57.07124336489585	Supremo	0.434015344151085	0.007604799169629534
7	SGDClassifier	76.52011151959348	Supremo	0.5848388239466319	0.0076429426504021755
3	LogisticRegression	31.119580139140567	tocantim	0.2544307339867768	0.008175905100556522
6	RandomForest	42.126139488878316	só	0.3609601130554374	0.008568554285652834
1	KNN	63.21832870470849	história	0.597275884969318	0.009447827824730727
5	MultinomialNB	33.90225836169564	tocantim	0.3397912597706723	0.010022673302336237
8	SVM	37.88049293397098	só	0.4144037779704297	0.010939767301676138
0	GradientBoosting	15.554885139107585	padre	0.23553777745032609	0.015142366873423237

Tabela 3 - Proporção de Concentração de Peso ordenada do modelo mais estáve para o mais instável após atualização do pré-processamento

Modelos baseados em árvores não são adequados para este tipo de tarefas, obtendo valores elevados para uma palavra e mais apto a overfitting, uma vez que as matrizes têm alta dimensionalidade e são esparsas em dados de texto. Após a alteração no pré-processamento a proporção de concentração diminuiu em todos os modelos mais eficazes, reduzindo o risco de overfitting, pelo que o poder predicativo está distribuído de uma forma mais “saudável”, significando que caso a palavra mais influente for removida o modelo não irá colapsar. As palavras mais dominantes “Supremo” (refere-se ao Supremo Tribunal Federal) e “tocantim” (estado do Brasil -

Tocantins) tornaram-se palavras diferenciadas para uma frase ser considerada como real.

Análise das palavras aprendidas pelos modelos

Com o objetivo de verificar se os modelos aprendiam elementos textuais semelhantes ou diferentes, utilizando as 300 instâncias e os resultados obtidos, construí uma tabela de consenso global de vocabulário que resume o que foi aprendido por todos os modelos, revelando as estratégias e elementos textuais que usam para distinguir entre mensagens fake e real. Analisando as tabelas 4 e 5, consigo deduzir que a classe real é identificada através de elementos de contexto institucional e geográfico ("Supremo" - Supremo Tribunal Federal; "tocantim" - Estado do Brasil), existindo consenso em maior parte dos modelos (mais de metade dos modelos identifica as mesmas palavras), pelo que estas palavras têm grande influência para a previsão desta classe. Os modelos usam termos de confirmação e ações ("tocar", "confirmar" e "forma") sugerindo que o texto real usa uma linguagem mais descritiva, no entanto a palavra "psl" referente ao Partido Social Liberal, tem que ser verificada ou excluída porque pode levar a um bias indesejável. Por outro lado, as mensagens com cariz falso, são identificadas através de uma combinação de ações de publicação, linguagem de opinião forte havendo um consenso maior nos modelos para palavras desta classe. A palavra "mentira" pode induzir um risco semântico, tendo alto peso absoluto média mas com baixo número de ocorrências, indicando possível overfitting.

========== TOP 10 GLOBAL CONSENSUS (FAKE) ==========

	word	class	model_consensus_count	avg_abs_weight	total_n_occurrences
63	publicar	FAKE	7	0.3581298886520707	21
75	só	FAKE	6	0.3365393539014148	6
7	absurdo	FAKE	5	0.2617106504695721	10
48	olher	FAKE	5	0.2624371383832186	5
21	dado	FAKE	4	0.3768933160152749	4
12	atacar	FAKE	4	0.33024677730427754	4
44	mentira	FAKE	4	0.4024671211573265	4
3	Folha	FAKE	3	0.3869233554288832	3
58	pretender	FAKE	3	0.4267063733386483	3
23	dever	FAKE	3	0.344453820553378	3

Tabela 4 - Consenso de todos os modelos e o número total de ocorrências das palavras nas 300 instâncias para a classe Fake

========== TOP 10 GLOBAL CONSENSUS (REAL) ==========

	word	class	model_consensus_count	avg_abs_weight	total_n_occurrences
76	tocantim	REAL	7	0.32291186881391437	14
5	Supremo	REAL	6	0.3293209241452179	12
77	tocar	REAL	4	0.3126779219096561	4
61	psl	REAL	4	0.15339070423603918	4
29	forma	REAL	4	0.33543018347900266	4
19	confirmar	REAL	4	0.26123274559614884	4
2	30	REAL	3	0.2380907701745566	3
47	oi	REAL	3	0.16854046069857445	5
46	nosso	REAL	3	0.26858669861958345	6
36	já	REAL	3	0.26766784338343746	3

Tabela 5 - Consenso de todos os modelos e o número total de ocorrências das palavras nas 300 instâncias para a classe Real

Os modelos classificam uma frase como real quando ela tem indicadores de contexto institucional, geográfico ou relato factual, enquanto uma frase fake quando demonstra linguagem de opinião ou ataque pessoal.

Os termos “tocantim” e “Supremo” não são spam, estando ligadas a contexto político, a palavra “absurdo” está relacionado com a presença de emojis e reticências e a palavra “publicar” está ligada à plataforma do Facebook e possivelmente a contextos de opinião, confirmando que a feature chave não é a palavra em si mas a sua ligação à divulgação de conteúdo, representado na figura 3.

Palavra-Chave	Índice de Recurso	Índice da Ocorrência	Contexto de Ocorrência
tocantim	1174	204	*soldado bolsonaro tocantim ! !....*
		233	*montoya tocantim :.....*
		357	*Ibope tocantim : vicentinho , 34 %...*
		362	*o infindável familiocracia político tocantim : negócio pai filho , irmão , marido esposo...*
		497	*cidade tocantim confirmar ato nacional : # ptnão dia 21 outubro ....*
		716	*candidato psl 17 bolsonaro tocantim....*
		763	*candidato federal desistir : adir gentil (poder) , desistir candidatura deputado federal dedicar i...*
		895	*o governador candidato reeleição , mauro carler , receber final tarde de este segundo – feira , 24 ....*
Supremo	94	43	*" fechar Supremo Tribunal federal " , dizer deputado pt...*
		238	*" fechar Supremo Tribunal federal " , dizer deputado pt –...*
publicar	1009	126	*publicar facebook gen...*
		236	*o extensa ficha criminal Fernando haddad publicar Justiça...*
		259	** publicar facebook gen cupertino (excelente) *...*
		322	*em o último domingo caetano veloso publicar artigo folha criticar escritor olavo Carvalho , apoiador...*
absurdo	100	115	*absurdo !....*
		183	*que absurdo....*
		703	*absurdo ....*

Figura 3 - Frases que contêm as palavras mais influentes no top 10 de consenso dos modelos

Consenso entre modelos (real e fake)

Analisando com maior profundidade o consenso entre os modelos, decidi comparar as três palavras com maior peso absoluto aprendidas pelos modelos para ambas as classes “real” e “fake”. Os modelos que obtiveram melhores métricas (MLP e Multinomial NB) aprenderam para a classe real a mesma palavra com maior peso absoluto (e quase o mesmo peso atribuído) e para a classe fake aprenderam duas palavras, pelo que sempre entram em consenso na palavra com maior peso. Se compararmos os quatro melhores modelos (incluir o Linear SVM e o SGD Classifier) também estão em concordância para maior parte das três palavras com maior peso para a classe real, o mesmo não acontecendo para a outra classe.

KNN	histório, propagandar, tocantim	0.5973, 0.4358, 0.3908	-0.5973, -0.4358, -0.3908
LinearSVC	Supremo, forma, tocantim	0.4336, 0.3870, 0.3396	-0.4336, -0.3870, -0.3396
LogisticRegression	tocantim, Supremo, oi	0.2560, 0.1554, 0.1536	-0.2560, -0.1554, -0.1536
MLP	tocantim, Supremo, confirmar	0.3260, 0.2988, 0.2887	-0.3260, -0.2988, -0.2887
MultinomialNB	tocantim, carreato, válir	0.3396, 0.2098, 0.2082	-0.3396, -0.2098, -0.2082
RandomForest	tocantim, que, você	0.3459, 0.2952, 0.2638	-0.3459, -0.2952, -0.2638
SGDClassifier	Supremo, forma, pegar	0.5794, 0.4921, 0.4753	-0.5794, -0.4921, -0.4753
SVM	Supremo, psl, confirmar	0.1692, 0.1679, 0.1565	-0.1692, -0.1679, -0.1565

Tabela 6 - Top 3 palavras que os modelos atribuem mais peso para a classe real

GradientBoosting	padre, olher, urgente	0.2332, 0.1888, 0.1767	0.2332, 0.1888, 0.1767
KNN	saber, Wilson, achar	0.4210, 0.4174, 0.4130	0.4210, 0.4174, 0.4130
LinearSVC	atacar, dado, pretender	0.3917, 0.3828, 0.3701	0.3917, 0.3828, 0.3701
LogisticRegression	petista, olher, 15	0.1995, 0.1977, 0.1814	0.1995, 0.1977, 0.1814
MLP	destruição, rouanet, olher	0.3134, 0.3006, 0.2914	0.3134, 0.3006, 0.2914
MultinomialNB	destruição, olher, absurdo	0.2212, 0.2086, 0.2026	0.2212, 0.2086, 0.2026
RandomForest	publicar, só, juíza	0.4084, 0.3672, 0.3642	0.4084, 0.3672, 0.3642
SGDClassifier	só, mentira, publicar	0.5579, 0.5491, 0.5451	0.5579, 0.5491, 0.5451
SVM	só, 15, dever	0.3982, 0.3427, 0.3085	0.3982, 0.3427, 0.3085

Tabela 7 - Top 3 palavras que os modelos atribuem mais peso para a classe fake

Esta análise permite concluir que as palavras “tocantim”, “Supremo” ,“forma”, “destruição” e “olher” ajudam a distinguir as classes em questão, pelo que a correlação entre a palavra identificada e aprendida e a classe não é um artefacto do algoritmo (conclusão que não reflete a verdade dos dados e é criada como consequência indesejada e artificial), mas sim uma característica inerente aos dados (neste caso os 300 exemplos). Isto ajuda bastante a simplificar a interpretação do que os modelos estão a aprender, porque em vez de entender cada lógica para modelos diferentes (no caso de aprenderem palavras diferentes) foco-me nas poucas palavras que são consistentemente importante.

Esta informação é bastante relevante e um ponto de partida para distinguir causalidade de correlação (a palavra ser um artefacto do modelo), e além de saber quais palavras são importantes ajuda a sabermos o porquê delas o serem. De modo a

explorar a causalidade mais um pouco devo implementar métodos que testam contrafactuais e a estabilidade das features. O métodos contrafactuais simulam a ausência da palavra na frase ou caso ela fosse substituída para alterar a saída. Na remoção uso uma instância classificada por exemplo como “real” e removo a palavra de maior peso e se a previsão mudar para “fake” é uma evidência de que a palavra é causal para a classificação “real” nessa instância, na substituição é a mesma coisa, substituo a palavra por outra da classe oposta com alto peso e se a previsão mudar então verifica-se a causalidade.

Verificação de Estabilidade das Explicações LIME

Antes de aplicar explicações contrafactuais, é necessário que a explicação do LIME seja robusta o suficiente para ser usada como base para inferências de causalidade e de forma a verificar a estabilidade das explicações geradas pelo LIME para os diferentes modelos, que é essencial para fornecer mais confiança nelas, uma vez que o LIME pode variar ligeiramente em cada execução. Para isso, é necessário medir quão consistentes são as palavras mais importantes que o LIME identifica para uma determinada instância, quando se executa várias vezes.

Para medir a estabilidade utilizei o índice de Jaccard comparar o conjunto das cinco features mais importantes em cada par das vinte execuções para a instância 125, medindo a similaridade entre todos os pares possíveis de conjuntos de palavras extraídos das vinte execuções. Os resultados desta etapa estão apresentados e descritos abaixo.

	Model	Avg. Jaccard Index (Stability)
0	LogisticRegression	1.000000
1	LinearSVC	1.000000
2	RandomForest	1.000000
3	MLP	1.000000
4	SGDClassifier	0.826316
5	MultinomialNB	0.824561
6	SVM	0.804762
7	KNN	0.493546
8	GradientBoosting	0.484712

Tabela 8 - Análise do ranking da estabilidade utilizando o *Average Jaccard Índex* das explicações LIME para os modelos treinados

O MLP e a LinearSVM obtiveram uma estabilidade perfeita, na qual as cinco palavras mais importantes foram exatamente as mesmas em todas as vinte execuções do LIME sugerindo que a fronteira de decisão local nesses modelos é estável e bem definida ao redor da instância 125. O SGDClassifier, SVM e Logistic Regression também obtiveram valores elevados demonstrando que existe consistência para as palavras mais importantes da instância nesses modelos.

Depois de ter analisado a estabilidade das explicações LIME para uma instância específica, decidi realizar uma análise global, mas antes disso necessitei de otimizar o tamanho da vizinhança para poder obter os melhores resultados possíveis. Para isso testei diferentes números de vizinhos (500, 1000 e 2000) e calculei para cada modelo o Índice de Jaccard médio, guardando também as top 5 palavras para cada tamanho da vizinhança, possibilitando uma visualização mais detalhada dos resultados obtidos apresentados na tabela abaixo.

	N_Samples	Avg. Jaccard Index (Stability)	Top_Features_Sample	Model
17	2000	0.518519	{pedir, pena, passagem, comprar, valer}	GradientBoosting
16	1000	0.501587	{quando, pena, receita, comprar, valer}	GradientBoosting
15	500	0.494180	{rua, pena, remédio, comprar, valer}	GradientBoosting
11	2000	0.718519	{rua, pena, bem, apresentar, comprar}	KNN
10	1000	0.504586	{pena, apresentar, alguém, reflitar, comprar}	KNN
9	500	0.354497	{pedir, pena, leite, bem, apresentar}	KNN
2	2000	1.000000	{quando, pena, bem, comprar, valer}	LogisticRegression
0	500	0.933333	{quando, pena, bem, comprar, valer}	LogisticRegression
1	1000	0.933333	{quando, pena, bem, comprar, valer}	LogisticRegression
18	500	1.000000	{quando, pedir, pena, comprar, valer}	MLP
19	1000	1.000000	{quando, pedir, pena, comprar, valer}	MLP
20	2000	1.000000	{quando, pedir, pena, comprar, valer}	MLP
4	1000	0.881481	{pena, bem, apresentar, comprar, valer}	MultinomialNB
5	2000	0.881481	{pena, bem, apresentar, comprar, valer}	MultinomialNB
3	500	0.822222	{pena, bem, apresentar, comprar, valer}	MultinomialNB
13	1000	1.000000	{pedir, pena, bem, comprar, valer}	RandomForest
14	2000	1.000000	{pedir, pena, bem, comprar, valer}	RandomForest
12	500	0.933333	{pedir, pena, bem, comprar, valer}	RandomForest
8	2000	1.000000	{rua, pena, apresentar, comprar, valer}	SVM
7	1000	0.933333	{rua, pena, apresentar, comprar, valer}	SVM
6	500	0.868783	{rua, pena, apresentar, comprar, valer}	SVM

Tabela 9 - Índice de Jaccard médio e top 5 palavras para cada modelo de acordo com os diferentes tamanhos da vizinhança

De seguida, prossegui para uma abordagem global, em que utilizei as 300 instâncias que tinham sido usadas anteriormente, e apliquei o mesmo procedimento que para a instância 125, pelo que utilizei o Índice de Jaccard Médio para os modelos e obtendo os valores representados na tabela 9.

	Model	LIME Stability (Avg. Jaccard)	F1_Score
6	RandomForest	1.000000	0.676603
4	SVM	0.804762	0.668500
0	LogisticRegression	1.000000	0.651309
2	LinearSVC	1.000000	0.648905
8	MLP	1.000000	0.646778
1	MultinomialNB	0.824561	0.640901
3	SGDClassifier	0.826316	0.638630
7	GradientBoosting	0.484712	0.631113
5	KNN	0.493546	0.626472

Tabela 10 - Análise do ranking da estabilidade utilizando o *Average Jaccard Índex* das explicações LIME para os modelos treinados e o F1-score correspondente

Dado estes resultados, é necessário ver as métricas de avaliação do modelo para fazer a escolha certa dos modelos a utilizar para depois fazer unificação com explicações contrafactuais, focando-me na sua performance e na sua interpretabilidade.

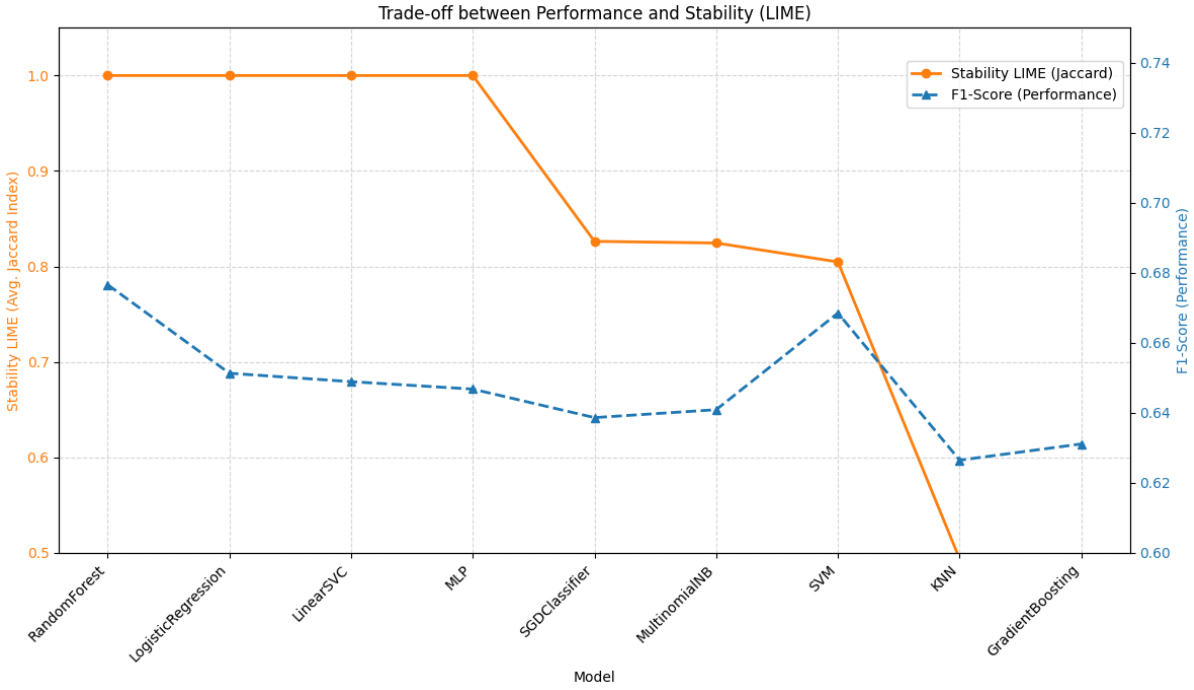

Figura 4 - F1-score e Estabilidade das explicações LIME para os diferentes modelos

========== FIDELITY ==========

Para finalizar a minha análise e reforçar a confiança nas explicações do LIME, prossegui para o uso da fidelidade que serve como medida quantitativa para a explicação gerada em relação à representação fiel do comportamento dos modelos. Para esta métrica utilizei pontuações da fidelidade (maior pontuação, maior a garantia de que as palavras identificadas são as que o modelo usou de facto para prever) obtidas através de três outras métricas distintas. As palavras que forma utilizadas para calcular estas métricas foram as mais importantes obtidas na análise anterior das 300 instâncias.

Para calcular a fidelidade utilizei três métricas que se baseiam na diferença absoluta na probabilidade de previsão após uma perturbação no texto:

- *Area Over Perturbation Curve* (AOPC): mede a queda de confiança ao remover as palavras mais importantes e é obtida através da remoção das top-k palavras importantes e medindo a mudança na probabilidade.

$$\text{AOPC} = \frac{1}{K}\sum_{k=1}^{K}|P(\hat{y}|x) - P(\hat{y}|x_{/F_k})|$$

- Comprehensiveness - mede a importância das features identificadas.

$$\text{Comp} = |P(\hat{y}|x) - P(\hat{y}|x_{/F_K})|$$

- Sufficiency - mede se apenas as palavras importantes são suficientes mantendo apenas as top-k palavras, isto medindo se a previsão se mantém.

$$\text{Suff} = |P(\hat{y}|x) - P(\hat{y}|x_{F_K})|$$

A fórmula que utilizei para calcular a pontuação da fidelidade foi a seguinte:

Fidelity_Score = k1 x AOPC + k2 x Comprehensiveness - k3 x Sufficiency

Em que *k1*, *k2* e *k3* são os pesos que se podem atribuir a cada métrica, neste caso tomaram valor de 1.

Após calcular a pontuação de fidelidade para todos os modelos, para facilitar a sua comparação utilizei gráficos como os que se encontram na figura 5 e na tabela 11. Quanto maior os valores de AOPC e Comprehensiveness melhor e o caso oposto para a Sufficiency, sendo possível através destas métricas avaliar a fidelidade. Faço isso para as 300 instâncias do conjunto de mensagens de teste e por fim faço um ranking deles todos com base na fidelidade.

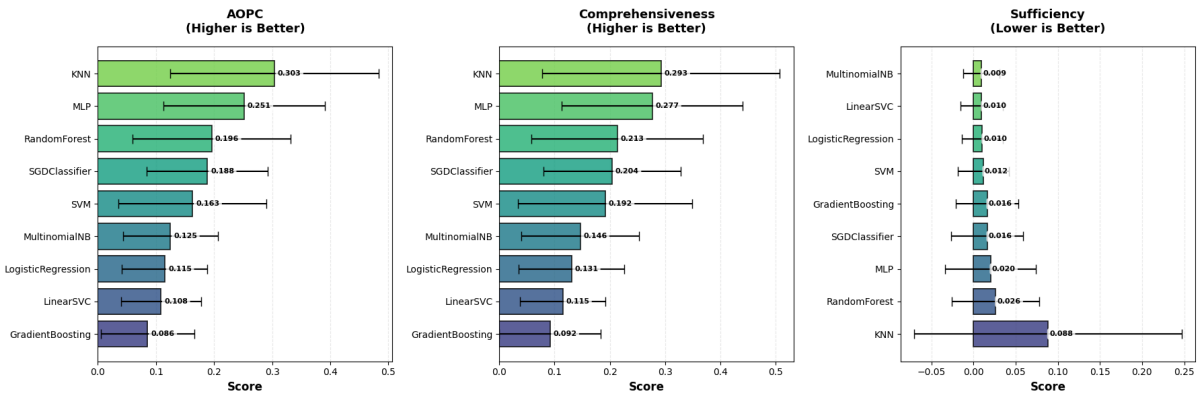

Figura 5 - Avaliação da Qualidade das Explicações LIME através das Componentes da Fidelidade

	Model	Fidelity_Score	AOPC_mean	Comprehensiveness_mean	Sufficiency_mean
8	MLP	0.847889	0.251355	0.276528	0.020163
5	KNN	0.666667	0.303470	0.292528	0.088329
3	SGDClassifier	0.647088	0.188030	0.204237	0.016310
6	RandomForest	0.632290	0.195892	0.213453	0.026300
4	SVM	0.608101	0.162864	0.191752	0.011503
1	MultinomialNB	0.483600	0.124751	0.146389	0.009318
0	LogisticRegression	0.438109	0.115160	0.130640	0.010418
2	LinearSVC	0.404473	0.108394	0.114815	0.009703
7	GradientBoosting	0.304089	0.085699	0.091932	0.016250

Tabela 11 - *Ranking* de Fidelidade das Explicações LIME com Base nas Métricas Agregadas (300 instâncias)

Analisando os resultados obtidos o melhor modelo de longe foi *MLP*, que também obteve dos melhores desempenhos e tem um *trade-off* de estabilidade e *f1-score* bastante alto como se encontra na tabela 10. O KNN obteve uma pontuação razoavelmente boa de fidelidade, no entanto foi o pior modelos quando calculada a estabilidade, isto pode ser proveniente do alto espaço de características que é criado com a vetorização TF-IDF, em que o vizinho mais próximo se torna instável, pelo que a distância perde o significado (os pontos tendem a ficar todos próximos uns dos outros), uma vez que a cada execução do LIME os vizinhos podem mudar ligeiramente, o que leva à instabilidade das explicações LIME. Como a *Random Forest* obteve o melhor *trade-off* entre a estabilidade e o desempenho e também uma pontuação razoavelmente boa de fidelidade, para a próxima etapa de testar a causalidade os modelos que vou considerar são: *MLP* e *Random Forest*.

========== INFERÊNCIAS CAUSAIS ==========