Nanodegree Engenheiro de Machine Learning

Projeto final

João Paulo Lopes de Souza

07 de julho de 2019

Detecção automática de fake news no Brasil

1. Definição

1.1 Visão geral do projeto

As pessoas assumem cada vez mais um papel ativo na disseminação da informação, consequência da popularização das redes sociais, fóruns e blogs na internet. Nessa quadra, são cada vez mais compartilhadas notícias e opiniões. Sendo que muitas delas são falsas e em determinados momentos podem ter alcance maior que as fontes reais[1]. Existem diferentes tipos de conteúdo enganoso, como explica Monteiro, 2018 [1] : “como notícias falsas, conhecidas popularmente como fake news, notícias satíricas, revisões falsas, boatos, etc.” As notícias falsas, que serão o foco neste trabalho, utilizam fatos forjados ou reais retirados do seu contexto original para desinformar o público. Isso normalmente acontece para beneficiar alguém.
Essa desinformação tem produzido enormes problemas em todo o mundo. Em especial para as democracias que tem suas eleições perturbadas para moldar os votos no sentido favorável à propagação de enganações.
Para tentar enfrentar esse problema da desinformação observamos grandes esforços em duas áreas em especial: 1. As agências de checagem de informação e 2. Sistemas automatizados para detecção de conteúdo falso. A primeira frente, as agências, realizam uma checagem manual para as notícias que surgem. Contudo, devido a sua natureza e a grande quantidade de notícias que surgem a cada dia, essa frente mostra-se necessária, porém ineficiente. A segunda frente, os sistemas automatizados, têm tentado realizar essa checagem de forma automática com o uso de machine learning. As iniciativas de checagem automática no Brasil são bastante tímidas e carecem de maior esforço de pesquisa.

1.2 Descrição do problema

O presente trabalho pretende estudar como realizar detecção automática de notícias falsas, em especial, aquelas relacionadas à política que influenciam diretamente na democracia de diversos países e influenciam a vida de muitas pessoas. Nesse sentido, deve-se utilizar características

[1] <https://www.buzzfeed.com/br/alexandrearagao/noticias-falsas-lava-jato-facebook>

verbais extraídos dos textos das notícias, realizar o processamento de linguagem natural - PLN e aplicar técnicas de machine learning para classificar as notícias em reais ou falsas.

1.3 MÉTRICAS

Serão utilizadas as mesmas métricas usadas no artigo [2] para avaliar o desempenho do modelo comparando-o com o benchmark: precision, recall, f1-mesure e accuracy. Além dessas métricas será utilizada a matriz de confusão.

1.3.1 MATRIZ DE CONFUSÃO

De acordo com Vasconcelos [5], é uma matriz 2x2, podendo ser maior dependendo da quantidade de classes que se deseja prever. É uma métrica voltada para analisar em mais detalhes os modelos de classificação.
Como exemplo, imagine um modelo que pretende classificar fotos como sendo um gato ou não. O dataset é composto de 100 fotos, sendo metade das fotos de gato e a outra metade não são gatos. Após treinamento do modelo obtivemos a seguinte matriz de confusão:

		Classe esperada	
		Gato	Não é gato
Classe prevista	Gato	25 Verdadeiro Positivo	10 Falso Positivo
	Não é gato	25 Falso Negativo	40 Verdadeiro Negativo

Figura 1: Matriz de confusão do modelo exemplo

A seguir serão detalhados os conceitos de verdadeiros positivos e negativos e falsos positivos e negativos com ajuda do exemplo acima .

- **Verdadeiros positivos (true positive):** são os casos que o modelo classificou como positivos e realmente eram positivos. No exemplo temos 25 casos foram previstos como gato e estavam corretos;
- **Falso positivos (false positive):** são casos que o modelo classificou como positivos, mas na realidade estavam errados. Como exemplo temos que 10 casos foram previstos erroneamente como não gato, quando eles eram gatos na realidade;
- **Verdadeiros negativos(true negative):** casos que o modelo classificou como negativos e estava certo. Do exemplo temos que 40 casos foram previstos como não é gato e estavam corretos;
- **Falso negativos(false negative):** casos que foram classificados como negativos, mas estavam errados. Como exemplo 25 casos foram previstos como não é gato, mas na realidade esses casos eram gatos.

1.3.2 ACCURACY

Métrica que define a taxa de classificação correta do total de casos. Segue a fórmula da acurácia:

Accuracy = $total\ de\ ocorrências\ classificadas\ corretamente \div total\ de\ ocorrências$;

1.3.3 PRECISION

Para Koehrsen [6], precision expressa a proporção de casos que nosso modelo previu como relevantes e eles realmente são. Segue a fórmula da precisão:

Precision = $número\ de\ verdadeiros\ positivos\ \div (número\ de\ verdadeiros\ positivos\ +\ falsos\ positivos)$;

1.3.4 RECALL

Segundo Koehrsen [6], recall é a habilidade que nosso tem de encontrar todos os casos relevantes com o conjunto de dados. Abaixo listamos a sua fórmula:

Recall = $número\ de\ verdadeiros\ positivos\ \div (número\ de\ verdadeiros\ positivos\ +\ falsos\ negativos)$

1.3.5 F1-MESURE

É uma combinação das duas métricas anteriores, recall e precision. Segundo Koehrsen [6], é uma média harmônica que leva em conta as duas métricas. Segue sua fórmula:

F1 = $2 \times ((precision\ \times\ recall)\ \div\ (precision\ +\ recall))$

2. ANÁLISE

2.1 Exploração dos dados

Utilizaremos o conjunto de dados reunidos por pesquisadores da USP [2] e chamado de Córpus Fake.Br [3]. Esse é o primeiro dataset detalhado sobre essa temática no Brasil.
O conjunto de dados Córpus Fake.Br possui 7200 notícias com seus respectivos metadados que foram coletadas no período de janeiro de 2016 a janeiro de 2018. Sendo 3600 notícias verdadeiras e 3600 notícias falsas com assuntos relacionados. Essas notícias possuem as seguintes categorias: política, TV e celebridades, sociedade e notícias cotidianas, ciência e tecnologia, economia e religião. O foco no presente trabalho serão as notícias da primeira categoria, política.
O conjunto de dados está organizado em duas pastas, uma contendo as notícias com os textos completos e outra com os textos truncados (em número de palavras). Esse corte foi realizado para adequar os tamanhos dos textos do par real/fake, sendo o texto maior limitado ao tamanho do menor. Existe um arquivo contendo o texto das notícias reais, outro arquivo com as notícias falsas e dois arquivos com os respectivos metadados.
São várias as características presentes nos metadados que serão listadas abaixo. Vale destacar que essas características foram extraídas dos textos sem cortes.
Zhou et. al [4] explica o significado de algumas características usadas:

- número de pausas = $número\ total\ de\ símbolos\ de\ pontuação \div número\ total\ de\ sentenças$
- tamanho médio das sentenças = $número\ total\ de\ palavras\ \div\ número\ total\ de\ sentenças$
- tamanho médio das palavras = $número\ total\ de\ caracteres\ \div\ número\ total\ de\ palavras$
- emotividade = $\frac{(número\ de\ adjetivos + número\ de\ adverbios)}{(número\ de\ substantivos + número\ de\ verbos)}$
- diversidade = $número\ de\ palavras\ distintas\ \div\ número\ total\ de\ palavras$

O dataset é composto por campos textuais e numéricos. Os textuais são:

- texto normalizado, é o texto da notícia que foi normalizada ou truncada;
- autor, é o autor da notícia;
- link, é a URL fonte da notícia;
- categoria, é a categoria do tipo de notícia.

Ele possui também diversos campos numéricos: número de tokens(número de palavras), número de palavras sem pontuação, número de tipos, número de links dentro das notícias, número de palavras em caixa alta, número de verbos, número de verbos no subjuntivo ou imperativo, número de substantivos, número de adjetivos, número de advérbios, número de verbos auxiliares, número de pronomes pessoais no singular nas primeiras e segunda pessoas, número de pronomes pessoais no plural da primeira pessoa, número de pronomes, pausas, número de caracteres, tamanho médio das sentenças, tamanho médio das palavras, percentagem de erros na escrita, emotividade e diversidade. Segue a listagem desses campos e seus respectivos tipos:

```
df_tratado.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7200 entries, 0 to 7199
Data columns (total 19 columns):
seq                                             7200 non-null int64
texto normalizado                               7200 non-null object
author                                          7200 non-null object
link                                            7200 non-null object
category                                        7200 non-null object
number of tokens                                7200 non-null int64
number of types                                 7200 non-null int64
number of links inside the news                 7200 non-null object
number of verbs                                 7200 non-null int64
number of subjuntive and imperative verbs       7200 non-null int64
number of nouns                                 7200 non-null int64
number of adjectives                            7200 non-null int64
number of adverbs                               7200 non-null int64
number of modal verbs (mainly auxiliary verbs)  7200 non-null int64
number of pronouns                              7200 non-null int64
percentage of news with speeling errors         7200 non-null float64
emotiveness                                     7200 non-null float64
diversity                                       7200 non-null float64
fake                                            7200 non-null int64
dtypes: float64(3), int64(11), object(5)
memory usage: 1.0+ MB
```

Figura 2 - Lista dos campos do dataset

O último campo criado foi fake que possui valor 1 para as fake news e 0 para os casos de notícias reais.

Segue uma amostra dos dados utilizados:

Coluna 1 a 12:

```
df_completo.head(3)
```

texto normalizado	fake	author	link	category	number of tokens	number of types	number of links inside the news	number of verbs	number of subjuntive and imperative verbs	number of nouns	number of adjectives
O Podemos decidiu expulsar o deputado federal...	0	Naira Trindade	http://politica.estadao.com.br/blogs/coluna-do...	politica	168	107	None	24	2	43	5
Bolsonaro é um liberal completo, diz president...	0	Marco Rodrigo Almeida	http://www1.folha.uol.com.br/poder/2018/01/194...	politica	1028	474	None	135	2	237	56
Ministro do STF libera Andrea Neves de prisão ...	0	Fernando Zuba , Pedro Ângelo E Renan Ramalho	https://g1.globo.com/mg/minas-gerais/noticia/s...	politica	540	232	None	69	0	146	10

Figura 3 - Valores dos campos dos dataset (colunas de 1 a 12)

Colunas 5 a 19:

```
df_completo.head(3)
```

link	category	number of tokens	number of types	number of links inside the news	number of verbs	number of subjuntive and imperative verbs	number of nouns	number of adjectives	number of adverbs	number of modal verbs (mainly auxiliary verbs)	number of pronouns	percentage of news with speeling errors	emotiveness	diversity
ogs/coluna-do...	politica	168	107	None	24	2	43	5	4	3	7	0.000000	0.134328	0.722973
r/2018/01/194...	politica	1028	474	None	135	2	237	56	45	14	63	0.001156	0.271505	0.547977
erais/noticia/s...	politica	540	232	None	69	0	146	10	20	7	19	0.000000	0.139535	0.487395

Figura 4 - Valores dos campos dos dataset (colunas de 5 a 19)

Estatísticas básicas sobre o dataset:

```
df_tratado_filtrado.describe()
```

	number of tokens	number of verbs	number of subjuntive and imperative verbs	number of nouns	number of adjectives	number of adverbs	number of modal verbs (mainly auxiliary verbs)	number of pronouns	percentage of news with speeling errors	emotiveness	diversity
count	7200.000000	7200.00000	7200.000000	7200.000000	7200.000000	7200.000000	7200.000000	7200.000000	7200.000000	7200.000000	7200.000000
mean	742.763333	100.76750	4.063194	182.920278	33.154444	30.014444	13.869028	38.758056	0.003065	0.209758	6.472975
std	774.155454	108.03151	5.492677	186.507322	39.702327	37.742105	15.303212	51.759915	0.007780	0.070239	62.426813
min	11.000000	1.00000	0.000000	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.232210
25%	183.000000	26.00000	1.000000	46.000000	7.000000	7.000000	3.000000	8.000000	0.000000	0.162500	0.471261
50%	416.000000	57.00000	2.000000	106.000000	18.000000	15.000000	8.000000	19.000000	0.000579	0.204082	0.578047
75%	1064.250000	141.00000	5.000000	265.000000	45.000000	39.000000	19.000000	49.000000	0.003799	0.251055	0.678938
max	8634.000000	1148.00000	65.000000	2170.000000	532.000000	497.000000	181.000000	720.000000	0.382353	0.750000	875.000000

Figura 5 - Estatísticas dos campos dos dataset

2.2 Visualização exploratória

Um característica muito importante dos dados trabalhados é o tamanho médio substancialmente maior das notícias reais comparadas às fake, conforme podemos verificar no gráfico abaixo:

```
sns.boxplot(x="fake", y="number of tokens", data=df_tratado,palette='rainbow')
<matplotlib.axes._subplots.AxesSubplot at 0x12ef94dbc88>
```

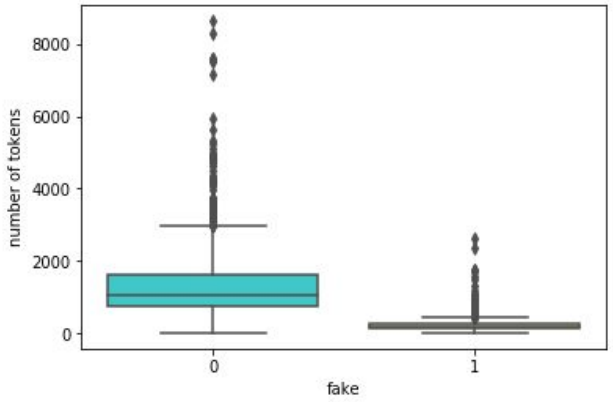

Figura 6 - Faixas de valores do campo number of tokens

Levando em consideração essa característica para evitar um modelo enviesado precisamos trabalhar com os textos das notícias truncados ou normalizados.
Outra característica importante dos dados e realçado por Monteiro et. al.[2] nos resultados da sua pesquisa é a Emotividade que conseguiu incrementar um pouco a qualidade do modelo. Pode-se verificar no gráfico abaixo que a emotividade possui faixas ligeiramente diferentes para as notícias reais e fake:

```
sns.boxplot(x="fake", y="emotiveness", data=df_tratado,palette='rainbow')
<matplotlib.axes._subplots.AxesSubplot at 0x1692b6f45f8>
```

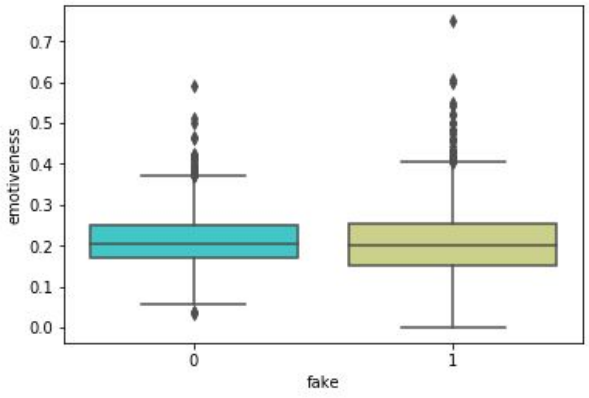

Figura 7 - Faixas de valores do campo emotiveness

2.3 Algoritmos e técnicas

O problema estudado neste trabalho é um típico caso de classificação de textos usando processamento de linguagem natural (NLP - Natural Language Processing), cujo modelo tentará prever se as notícias são reais ou fake.

Um método muito reconhecido para tratar de problemas com textos é Bag of Words - BOW. A seguir serão detalhados esse conceito e os classificadores testados.

Para Bedi[11], classificação de textos é um processo automatizado de classificar textos em categorias pré-definidas. E ainda segundo o mesmo autor, NLP é um campo da Inteligência Artificial que foca em apoiar computadores a entenderem e interpretar linguagem humana.

2.3.1. Bag of Words - BOW

BOW é uma abordagem para se extrair características do texto e utilizá-las em algoritmos de machine learning, conforme explica D'Souza[12]. Ela trabalha dividindo o texto em tokens e descobrindo a frequência de cada token no texto. Após a divisão em tokens é criado um vetor esparso, cujas colunas são os tokens e cada linha representa um texto.
Duas das principais abordagens para transformar o texto em vetor:

- **CountVectorizer** - faz uma contagem das ocorrências dos tokens;
- **TF-IDF Vectorizer** - TF-IDF é uma medida estatística usada para avaliar a importância de um token para um documento em uma coleção. A importância aumenta proporcionalmente ao número de vezes que o termo aparece no documento, mas é decrementado pela frequência que ele aparece na coleção.

2.3.2 Classificadores

Para resolver o problema de classificação em questão trabalhou-se com diversos algoritmos do sckit learning: Naive Bayes[2], Linear Regression[3], Linear SVC[4]. Bem como se utilizou o XGBoost.

2.3.2.1 Naive Bayes

Segundo Gandhi [7], são modelos classificadores probabilísticos que se baseiam no teorema de Bayes. Usando este teorema pode-se encontrar a probabilidade de A acontecer, dado que B ocorreu. O pressuposto aqui é que as variáveis são independentes. Esses algoritmos, apesar de sua simplicidade são muito eficientes e rápidos.

2.3.2.2 Linear Regression

Para Swaminathan[8], modelos de regressão linear são usados para encontrar relações lineares entre um alvo e uma ou mais variáveis. A idéia principal é encontrar uma linha que melhor descreve os dados. Essa linha é aquela cujo erro total de previsão é o menor possível.

2.3.2.3 Linear SVC

Pupale [9] explica que Support Vector Machine - SVM são modelos lineares para problemas de classificação e regressão. Sua idéia básica é criar um hiperplano que separa os dados em classes.

---

[2] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[4] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

O algoritmo encontra os pontos mais próximos da linha a partir de cada classe, esse pontos são chamados de support vectors. Depois calcula-se a distância entre esses pontos e a linha. Nomeamos essa distância de margin. A meta será maximizar essa margem para que a fronteira que separa as classes seja a mais ampla possível.

2.3.2.4 XGBoost

De acordo com Brownlee[10], XGBoost - eXtreme Gradient Boosting é uma implementação de Gradient Boosting Machines usando árvores de decisão. Gradient boosting é um método de ensemble com modelos sendo incluídos sequencialmente, cujos novos modelos são criados para predizer os erros dos modelos anteriores. Além do mais é chamado gradiente porque usa o algoritmo do gradient descent para minimizar os erros. No geral, XGBoost é muito rápido quando comparado com outras implementações de gradient boosting. A performance dos seus modelos é reconhecida por ganhar diversas competições na plataforma Kaggle[5].

2.4 BENCHMARK

Será utilizado como modelo de referência o resultado obtido na pesquisa da USP [2]. O melhor e mais eficiente resultado foi obtido utilizando a técnica de bag of words e a feature emotiveness aplicando o algoritmo SVM Linear SVC. Os autores obtiveram acurácia de 89%, conforme tabela abaixo.

Features	Precision		Recall		F-Measure		Accuracy
	Fake	True	Fake	True	Fake	True	
Part of speech (POS) tags	0.76	0.74	0.73	0.77	0.74	0.76	0.75
Semantic classes	0.73	0.73	0.73	0.73	0.73	0.73	0.73
Bag of words	0.88	0.89	0.89	0.88	0.88	0.88	0.88
POS tags + semantic classes + bag of words	0.88	0.89	0.89	0.88	0.89	0.89	0.88
Pausality	0.52	0.52	0.58	0.46	0.55	0.49	0.52
Emotiveness	0.57	0.56	0.53	0.61	0.55	0.58	0.56
Uncertainty	0.51	0.51	0.46	0.57	0.48	0.54	0.51
Non-immediacy	0.53	0.51	0.16	0.86	0.24	0.64	0.51
Pausality + emotiveness + uncertainty + non-immediacy	0.57	0.56	0.53	0.60	0.55	0.58	0.57
Bag of words + emotiveness	0.88	0.89	0.89	0.88	0.89	0.89	0.89
All the features	0.89	0.89	0.89	0.89	0.89	0.89	0.89

Figura 8 - Benchmark - resultados

3. METODOLOGIA

3.1 Pré-processamento de dados

Quando se trabalha com classificação de textos a primeira tarefa crucial é realizar o pré-processamento, que consistiu em:

- tokenizar o texto em partes menores chamados tokens;

[5] https://www.kaggle.com/

- transformar em minúsculas;
- retirar os números;
- retirar a pontuação;
- retirar os stop words;
- realizar o processo de stemmer, que consiste em obter o radical de cada palavra;

Segue o trecho de código que realiza esse pré-processamento:

```
def limpeza_coluna_texto(texto,coluna):

    with open('var/stopwords.txt') as f:

        cachedStopWords = f.read()

        stemmer = nltk.stem.SnowballStemmer('portuguese')

        translator = str.maketrans({key:' ' for key in string.punctuation})

    frase_processada = list()

    for frase in texto[coluna]:

        result = ''

        ## Retirar pontuação, os números e transformar em minúscula

        result = re.sub('[0-9]', '' , frase.translate(translator)).lower()

        ## Retirar os stop words e realiza stemização ou obteção do radical

        result = ' '.join([stemmer.stem(word) for word in result.split() if word not in cachedStopWords])

        frase_processada.append(''.join(result))

    return frase_processada
```

3.2 Implementação

A implementação compreendeu inicialmente no tratamento dos arquivos do corpus [3], cujas notícias fake e reais e seus metadados estão separados em diversos arquivos conforme descrito na seção 1.1.1. Em função disso, foi necessário processamento inicial para juntar todos esses arquivos em um único. Isso foi feito no notebook do jupyter **Fake News Prepara Arquivo Dados.ipynb**.

Em seguida, utilizando outro notebook do jupyter **Fake News Classifier.ipynb**, foi realizado pré-processamento nos dados textuais para retirar os ruídos, conforme descrito na seção 3.1.

Posteriormente, pretendia-se filtrar os registros para utilizar apenas as notícias da categoria política esperando que isso melhorasse as métricas do modelo, mas isso não aconteceu. Dessa forma, resolveu-se manter os dados de todas as categorias. O código desse filtro encontra-se abaixo:

```
df_politica = df_completo.loc[df_completo['category']=='politica']
```

Foram criadas duas novas features sobre o texto normalizado, número de palavras ou tokens e tamanho de toda a sentença, conforme trecho abaixo:

df_completo['words'] = df_completo['texto normalizado'].apply(lambda x: len(x.split(' ')))

df_completo['length'] = df_completo['texto normalizado'].apply(lambda x: len(x))

Essas novas features, porém, não foram relevantes para melhorar a pontuação do modelo. Suas faixas não possuem diferenças entre as notícias fake e reais, conforme gráficos abaixo.

```
sns.boxplot(x="fake", y="words", data=df_completo,palette='rainbow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b038da1710>
```

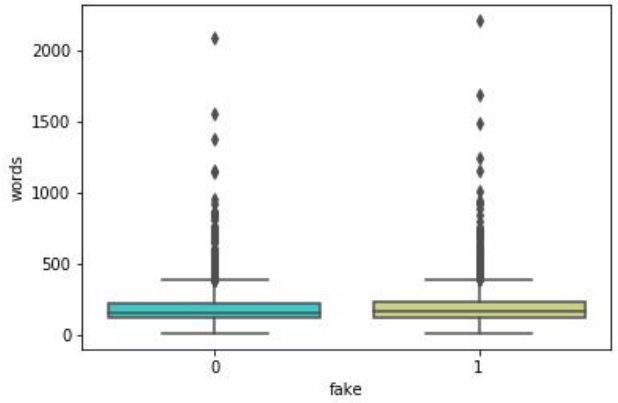

Figura 9 - Gráfico de distribuição da feature words

```
sns.boxplot(x="fake", y="length", data=df_completo,palette='rainbow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b038d967b8>
```

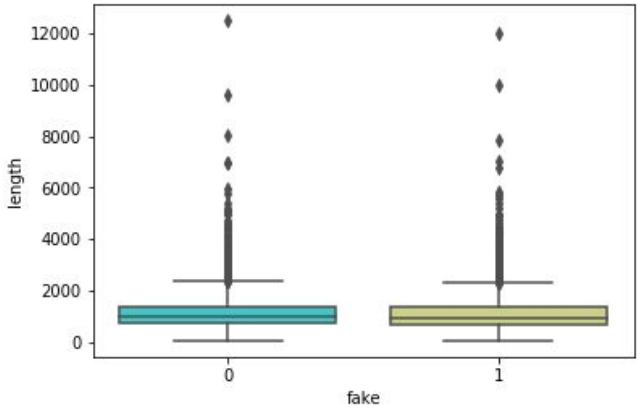

Figura 10 - Gráfico de distribuição da feature length

Na sequência, os dados foram divididos em dados de treino e testes, sendo 80% na primeira e 20% na última, conforme trecho de código a seguir:

```
treino, teste, classe_treino, classe_teste = train_test_split(df_completo[features],
                                                              df_completo['fake'],
                                                              test_size=0.20,
                                                              random_state = 42,shuffle = True, stratify = df_completo['fake'])
```

Construiu-se um pipeline que uniu características textuais e numéricos, conforme trecho de código abaixo. Para o texto das notícias foi incluído um pipeline com o **CountVectorizer** e **TFIDFTransformer**. Para os dados numéricos foi incluído um pipeline com a seleção da coluna e depois esse valor foi normalizado.

```
regressao_logistica = LogisticRegression(solver = "lbfgs")

text_pipe = Pipeline([
        ('selector', TextSelector(key='texto preprocessado')),
        ('cv', CountVectorizer(ngram_range=(1,2))),
        ('tfidf', TfidfTransformer())
    ])
emotiveness_pipe =  Pipeline([
        ('selector', NumberSelector(key='emotiveness')),
        ('standard', MinMaxScaler())
    ])
feats = FeatureUnion([('text', text_pipe),
            ('emotiveness', emotiveness_pipe)
            ])
pipe = Pipeline([
   ('features',feats),
   ('classificador', regressao_logistica)
])
classificar_texto(pipe,treino, classe_treino,0)
```

Em seguida, os dados de treino foram divididos utilizando o cross validation k-fold[6] com 5 partes e realizou-se o treinamento e validação dos resultados, para isso criou-se a função **classificar_texto**:

```
def classificar_texto(pipe, X_treino, y_treino,isXGB):
   kf = StratifiedKFold(n_splits=5,shuffle=True,random_state = 42)
   all_classes_tests = []
   all_predictions = []
   for train_index, test_index in kf.split(X_treino, y_treino):
      treino_kf = X_treino.iloc[train_index]
```

[6] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

```
        classe_treino_kf = y_treino.iloc[train_index]
        teste_kf = X_treino.iloc[test_index]
        classe_teste_kf = y_treino.iloc[test_index]
        y_pred = []
        predictions = []

        if(isXGB == 1):
            pipe.fit(treino_kf, classe_treino_kf)
            y_pred = pipe.predict(teste_kf)
            predictions = [round(value) for value in y_pred]
        else:
            pipe.fit(treino_kf, classe_treino_kf)
            predictions = pipe.predict(teste_kf)

        for predict in predictions:
            all_predictions.append(predict)
        for clas_teste in classe_teste_kf:
            all_classes_tests.append(clas_teste)
    # evaluate predictions
    print ("Accuracy:", accuracy_score(all_classes_tests, all_predictions))
    print ("Precision:", precision_score(all_classes_tests, all_predictions))
    print (classification_report(all_classes_tests, all_predictions))
    print (confusion_matrix(all_classes_tests, all_predictions))
```

Foram avaliados alguns dos algoritmos usados no trabalho [2], Naive Bayes, Linear Regression e Linear SVC, e também o classificador XGBoost que não foi abordado no referido trabalho. Todos esses classificadores foram utilizados na sua configuração padrão.

Após essa avaliação, foi selecionado o classificador com melhor pontuação, em seguida ele foi submetido a um grid search cv para testar quais seus parâmetros ótimos. Segue o trecho de código:

```
hyperparameters = { 'classificador__C':[0.01,1,10,100,1000],
                    'classificador__tol':[0.0001,0.001,0.01],
```

```
                'classificador__loss':['hinge','squared_hinge']                }

clf = GridSearchCV(pipe, hyperparameters, cv=5)

clf.fit(treino,classe_treino)

print(clf.best_params_)
```

Por fim, utilizou-se esses parâmetros ótimos para treinar seu pipeline e realizar a avaliação da previsão sobre os dados de testes, que ainda não foram explorados. Isso para verificar o quão bem o modelo generaliza quando apresentado a novos dados. Segue o trecho de código:

```
svc = LinearSVC(random_state=0, C=10, loss='hinge', tol=0.0001)

pipe = Pipeline([

    ('features',feats),

    ('classificador', svc)

])

classificar_texto(pipe,treino, classe_treino,0)

preds = pipe.predict(teste)
```

3.3 Refinamento

Conforme definido na seção 2.4, o modelo utilizado na pesquisa [2] obteve o melhor resultado usando BOW + Emotiveness. Os autores ressaltam o resultado obtido somente com BOW de 88% de acurácia.

Dessa forma, partimos testando inicialmente só o BOW e, em seguida, usamos BOW + Emotiveness.

Para otimizar o BOW avaliamos usar unigramas (1 token) e bigramas (combinação de 2 tokens) .

Utilizou-se o método GridSearch para obter os parâmetros ótimos do melhor preditor, conforme código na seção anterior.

4. RESULTADOS

4.1 Modelo de avaliação e validação

Segundo descrito na seção 3.2, o modelo foi validado usando a estratégia de cross validation k-fold, definida abaixo. Utilizou-se uma divisão em 5 partes, permitindo assim que o modelo fosse treinado com diferentes entradas. As métricas apresentadas pelo modelo final foram próximas daqueles destacados no benchmark e seus parâmetros finais foram obtidos usando uma busca exaustiva, grid search cv. O modelo final após treinado e validado ainda foi testado usando o conjunto de testes que ainda não foi utilizado. Os resultados foram próximos aos exibidos durante treinamento e validação.

Dessa forma, nos permite dizer que o modelo não apresenta overfitting para os dados treinados, ele generaliza bem.

4.1.1 Cross validation

Para Seif[13], Cross validation é uma técnica estatística para testar a performance de modelos de machine learning. Este método levará mais tempo uma vez que o modelo será treinado e validado diversas vezes. Ele ajuda a evitar tanto o viés de seleção, que ocorre quando selecionamos uma parte particular para treinar, quanto o overfitting.
A técnica de cross validation usada foi k-fold que consiste em dividir aleatoriamente o conjunto de treino em k partes de tamanhos iguais. Na sequência treinamos o modelo k vezes, em cada um desses treinamentos selecionamos uma parte para validar o treinamento e as demais k-1 para treinar, conforme figura abaixo.

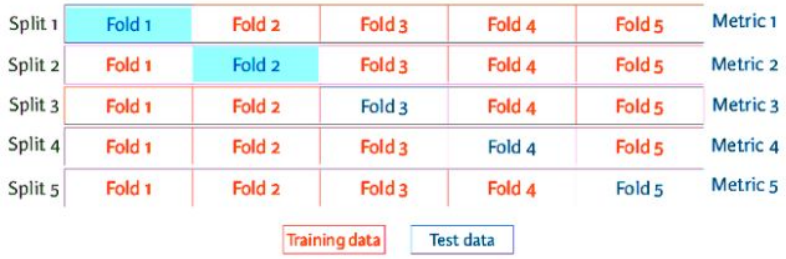

Figura 11 - Exemplo de cross validation k-fold

4.2 Justificativa

Uma hipótese que planejamos testar era se o classificador XGBoost poderia produzir melhores resultados que o Linear SVC usado no benchmark. Posteriormente verificamos que os resultados desse classificador foram muito bons com acurácia de 89% e precisão de 88%, conforme figura abaixo. Praticamente o mesmo que o benchmark, contudo, o seu tempo de execução ainda é bastante alto se comparado ao Linear SVC.

```
Accuracy: 0.8880208333333334
Precision: 0.8964171692089393
              precision    recall  f1-score   support

           0       0.88      0.90      0.89      2880
           1       0.90      0.88      0.89      2880

   micro avg       0.89      0.89      0.89      5760
   macro avg       0.89      0.89      0.89      5760
weighted avg       0.89      0.89      0.89      5760

[[2588  292]
 [ 353 2527]]
```

Figura 12 - Resultados do XGBoost

Foram testados os classificadores Naive Bayes e Linear Regression, que obtiveram os respectivos acurácia de 86% e precisão de 88% para o primeiro e acurácia de 89% e precisão de 88% para o último, conforme figura abaixo.

```
Accuracy: 0.8657986111111111
Precision: 0.888888888888888
              precision    recall  f1-score   support

           0       0.85      0.90      0.87      2880
           1       0.89      0.84      0.86      2880

   micro avg       0.87      0.87      0.87      5760
   macro avg       0.87      0.87      0.87      5760
weighted avg       0.87      0.87      0.87      5760

[[2579  301]
 [ 472 2408]]
```

Figura 13 - Resultados do Naive Bayes

```
Accuracy: 0.8925347222222222
Precision: 0.8878216123499142
              precision    recall  f1-score   support

           0       0.90      0.89      0.89      2880
           1       0.89      0.90      0.89      2880

   micro avg       0.89      0.89      0.89      5760
   macro avg       0.89      0.89      0.89      5760
weighted avg       0.89      0.89      0.89      5760

[[2553  327]
 [ 292 2588]]
```

Figura 14 - Resultados do Linear Regression

O melhor resultado aconteceu quando utilizamos um pipeline contendo somente texto (countVectorizer e tfidfTrasnform - levando em conta unigramas e bigramas) e o classificador Linear SVC com parâmetros obtidos com o grid search: C=10, loss='hinge' e tol=0.0001, detalhes abaixo.

LinearSVC(C=10, class_weight=None, dual=True, fit_intercept=True,
intercept_scaling=1, loss='hinge', max_iter=1000, multi_class='ovr',
penalty='l2', random_state=0, tol=0.0001, verbose=0)

Essa configuração obteve o resultado que superou o benchmark, alcançando acurácia de 91% e precisão de 91% com os dados de treinamento e acurácia de 90% e precisão de 90% com dados de testes, conforme figuras abaixo.

```
Accuracy: 0.9138888888888889
Precision: 0.916491963661775
              precision    recall  f1-score   support

           0       0.91      0.92      0.91      2880
           1       0.92      0.91      0.91      2880

   micro avg       0.91      0.91      0.91      5760
   macro avg       0.91      0.91      0.91      5760
weighted avg       0.91      0.91      0.91      5760

[[2641  239]
 [ 257 2623]]
```

Figura 15 - Resultado do Linear SVC com dados de treinamento

```
Accuracy: 0.9020833333333333
Precision: 0.9037656903765691
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       720
           1       0.90      0.90      0.90       720

   micro avg       0.90      0.90      0.90      1440
   macro avg       0.90      0.90      0.90      1440
weighted avg       0.90      0.90      0.90      1440

[[651  69]
 [ 72 648]]
```

Figura 16 - Resultado do Linear SVC com dados de testes

Com base nesses resultados e considerando:

- o modelo foi treinado usando diversas partes diferentes do conjunto de dados;
- o modelo treinado realizou a previsão em um conjunto de dados totalmente novo.

Pode-se dizer que o modelo é bem robusto e generaliza bem para novos dados. Entretanto, como o modelo é fortemente baseado no contexto do conjunto textual de treinamento - BOW e levando em conta também o resultado apresentado por Monteiro[1], que no seu trabalho gerou uma nova base para testes contendo 57 notícias reais e 57 fake coletadas em um período diferente do período usado para gerar o corpus Fake.Br [3] e obteve uma redução considerável de 16% na pontuação do modelo. Acreditamos que o modelo é sensível aos assuntos presentes no seu contexto de treinamento, para resolver isso deve-se realizar nova coleta de notícias que capture o novo contexto e treinar novamente o modelo de tempos em tempos. Por exemplo, o corpus Fake.Br [3] foi gerado com notícias coletadas entre janeiro/2016 a janeiro/2018, mas o assunto sobre o vazamento de conversas da lava jato divulgadas pelo Intercept Brasil[7] só ocorreu em 2019, dessa forma, é bem provável que o modelo tenha dificuldades para tratar esse novo assunto.

5. CONCLUSÃO

5.1 Foma livre de visualização

O gráfico abaixo exibe as 30 principais features utilizadas para tomada de decisão pelo modelo. Vale ressaltar que quase todas as features são unigramas, possuindo uma feature bigrama entre as 15 features mais importantes. Daí a necessidade de utilizar o modelo de BOW com unigramas e bigramas.

[7] https://theintercept.com/2019/06/09/editorial-chats-telegram-lava-jato-moro/

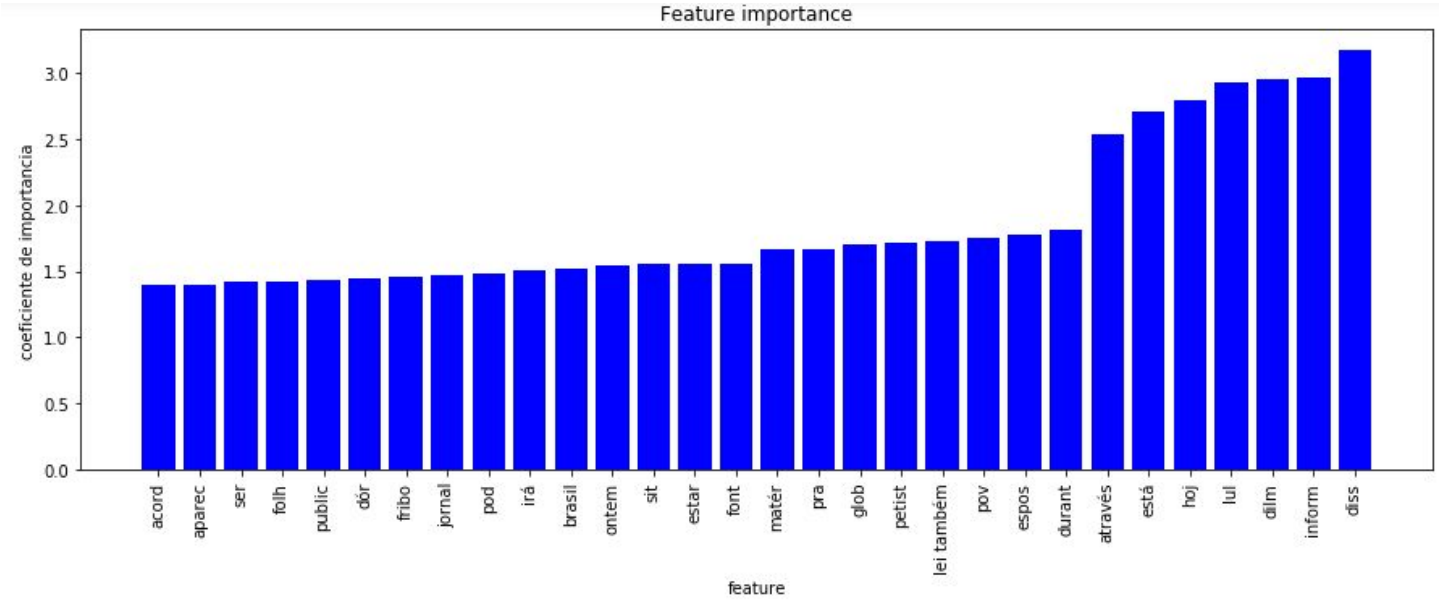

Figura 17 - Feature Importance para o Linear SVC

5.2 Reflexão

No início, a grande dificuldade foi reunir todas as informações que estavam espalhadas em diversos arquivos em um único arquivo.

Em seguida, foi realizada uma análise exploratória de dados - EDA, que mostrou uma discrepância muito grande da feature number of tokens, uma vez que as notícias reais tem tamanhos bem maiores que aquelas fake, detalhado na seção 2.2. Então, esse campo foi descartado porque acreditamos que pode enviesar o modelo, tanto é que seus resultados alcançam uma acurácia muito alta, maior que 95%.

```
Accuracy: 0.9625
Precision: 0.937007874015748
              precision    recall  f1-score   support

           0       0.99      0.93      0.96      2880
           1       0.94      0.99      0.96      2880

   micro avg       0.96      0.96      0.96      5760
   macro avg       0.96      0.96      0.96      5760
weighted avg       0.96      0.96      0.96      5760

[[2688  192]
 [  24 2856]]
```

Figura 18 - Resultados do Linear SVC com a feature number of tokens

Prosseguindo, foram criadas novas características de número de tokens e tamanho da sentença sobre os textos normalizados, porém esses não provocaram melhorias nos resultados. Isso em função da semelhança das faixas desses dois valores tanto para as notícias fake quanto para as reais.

Na sequência, construiu-se então um pipeline para realizar algumas combinações de testes, BOW com unigramas e bigramas e também com uso ou não da feature numérica Emotiveness e diversos classificadores.

Por fim, tentamos otimizar o classificador que apresentou melhores resultados, alterando seus hyper parâmetros.

5.3 Melhorias

Acreditamos que existem três caminhos para explorar melhoria no modelo de classificação em questão.

Primeiramente, o dataset utilizado possui muitas métricas numéricas e não conseguimos testar todas elas, decidimos utilizar aquela que apresentou bons resultados no benchmark. Então, uma possibilidade de análise que pode melhorar o resultado seria trabalhar as demais métricas.

Outra possibilidade de melhoria seria utilizar outro método para representar os dados textuais, como por exemplo, word2vec embeddings[8].

E por fim, pode-se analisar outros classificadores que utilizam Deep Learning[9] Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) e Hierarchical Attention Network (HAN).

Acreditamos ainda que existe uma outra linha de pesquisa em campo similar aos das notícias falsas deste trabalho que seria analisar mensagens de aplicativos de comunicação eletrônica como whats up e telegram. Uma vez que nesse meio circulam notícias sem nenhuma fonte e que tem um grande poder de compartilhamento. Possíveis fontes para se montar um corpus seriam os sites brasileiros boatos[10] e e-farsas[11].

[8] https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c
[9] https://medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f
[10] https://www.boatos.org
[11] http://www.e-farsas.com

REFERÊNCIAS

1. Monteiro, R.A. (2018). Detecção Automática de Notícias Falsas. Trabalho de Conclusão de Curso. Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo. São Carlos-SP, November, 40p.<https://drive.google.com/file/d/1dgazD5ZfFwY8aVPJUur4xE1kHwzSU8ET/view>. Acesso em 30/05/2019.

2. Monteiro, R.A.; Santos, R.L.S.; Pardo, T.A.S.; Almeida, T.A.; Ruiz, E.E.S.; Vale, O.A. (2018). Contributions to the Study of Fake News in Portuguese: New Corpus and Automatic Detection Results. In the Proceedings of the 13th International Conference on the Computational Processing of Portuguese (PROPOR) (LNAI 11122), pp. 324-334. September, 24-26. Canela-RS/Brazil. <http://conteudo.icmc.usp.br/pessoas/taspardo/PROPOR2018-MonteiroEtAl.pdf>
Acesso em 30/05/2019.

3. Fake.br-Corpus. <https://github.com/roneysco/Fake.br-Corpus>. Acesso em 30/05/2019.

4. Zhou, L., Burgoon, J., Twitchell, D., Qin, T., Nunamaker Jr., J.: A comparison of classification methods for predicting deception in computer-mediated communication. Journal of Management Information Systems 20(4), 139{165 (2004)

5. Vasconcelos, P.: Como saber se seu modelo de Machine Learning está funcionando mesmo. Medium. 2018.
<https://paulovasconcellos.com.br/como-saber-se-seu-modelo-de-machine-learning-est%C3%A1-funcionando-mesmo-a5892f6468b> Acesso em 07/07/2019.

6. Koehrsen, W.: Beyond Accuracy: Precision and Recall. Medium. Towards Data Science. <https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c> Acesso em 07/07/2019.

7. Gandhi, R.: Naive Bayes Classifier. Medium. Towards Data Science.<https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c> Acesso em 07/07/2019.

8. Swaminathan, S.: Linear Regression — Detailed View. Medium. Towards Data Science.<https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86> Acesso em 07/07/2019.

9. Pupale, R.: Support Vector Machines(SVM) — An Overview. Medium. Towards Data Science. <https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989> Acesso em 07/07/2019.

10. Brownlee, J.: A Gentle Introduction to XGBoost for Applied Machine Learning. Machine Learning Mastery. <https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/> Acesso em 07/07/2019.

11. Bedi, G.: A guide to Text Classification(NLP) using SVM and Naive Bayes with Python. Medium.<https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34 > Acesso em 14/07/2019.

12. D'Souza, J.: An Introduction to Bag-of-Words in NLP. Medium.<https://medium.com/greyatom/an-introduction-to-bag-of-words-in-nlp-ac967d43b428> Acesso em 14/07/2019.

13. Seif, G.: Why and How to do Cross Validation for Machine Learning. Medium. Towards Data Science.<https://towardsdatascience.com/why-and-how-to-do-cross-validation-for-machine-learning-d5bd7e60c189> Acesso em 15/07/2019.