UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE TECNOLOGIA

**Mineração de Texto para identificar spam em comentários de vídeos do Youtube**

Curso de Bacharelado em Sistemas de Informação

**Alunos**:

André Estevam ra 166348

Karen Malzoni ra 177493

Kaulitz Guimarães ra 188530

Pedro Artico Rodrigues ra 185545

**Professora:** Dra. Ana Estela Antunes da Silva

**04/2018**

**Sumário**

[**Introdução**](#_l8ktse3hx0md) **3**

[**2. Definição de mineração de dados**](#_ctterj5ss5vo) **4**

[**3. Tarefas da mineração de dados**](#_grmw0t7rqsw5) **4**

[**4. Processamento de Língua Natural**](#_lzahlzgybuu2) **5**

[4.1 Tokenização](#_59npshmojxnu) 6

[4.2 Stopwords](#_e4d3b8alow3i) 6

[**5. Descrição do problema**](#_a75ylvuxn8q3) **6**

[5.1 Atributos do problema](#_njv7epvy83g1) 7

[5.2 Pré-Processamento](#_q4wtcxjjssws) 8

[5.3. Tarefa aplicada](#_lv0i8nf1x1tg) 9

[5.3.1. Logistic Regression - Modelo de classificação](#_j8opktsfbqma) 10

[**6. Resultado**](#_9xol72rh5kj3) **11**

[**7. Conclusão**](#_d0wscax6zcvj) **13**

[**8. Referências Bibliográficas**](#_v193sltqw8yq) **14**

[**9. Anexos**](#_d1s37aogu0cc) **16**

[Anexo A](#_y18wek5dteqf) 16

[Anexo B](#_chyyxzlc0jhm) 19

[Anexo C](#_y71zd1lxwj17) 22

# Introdução

O YouTube é, atualmente, a maior plataforma de compartilhamento de vídeos na internet, permitindo às pessoas exibir conteúdo gratuitamente. Ele revolucionou a sociedade no sentido de tirar das mãos das gravadoras e emissoras o poder de escolha sobre o que seria exibido às massas. Permitiu também o acesso à informação àqueles que têm este acesso restrito ou limitado por seu regime de governo. A plataforma permite também que usuários comentem os vídeos postados, possibilitando assim uma maior interação entre quem produz e quem consome aquele conteúdo.

Quase cinco bilhões de vídeos são assistidos todos os dias por usuários do mundo inteiro (DANNY, 2018), o que faz do YouTube um ótimo veículo de propaganda. Contudo, esta grande quantidade de alvos fáceis atrai também olhares de pessoas mal intencionadas, que deixam nos vídeos, comentários com links maliciosos, como vírus e spam

Este trabalho objetiva a identificação de comentários maliciosos no Youtube, através da aplicação da tarefa de classificação, pertencente à mineração de dados. Por meio disso, será possível agir rapidamente, excluindo esses comentários e evitando que usuários desavisados tenham seus dados roubados.

# 2. Definição de mineração de dados

A Mineração de dados (MD) compreende um processo orientado a extração de conhecimento implícito, previamente desconhecido, e útil, a partir de grandes repositórios de dados (WITTEN; FRANK, 2005). Essa área é formada por ferramentas e técnicas que utilizam [algoritmos](https://pt.wikipedia.org/wiki/Algoritmo) de aprendizagem ou classificação para explorar um conjunto de dados, extraindo ou evidenciando padrões nestes dados e auxiliando na descoberta de conhecimento. (J. HAN, M. KAMBER, J.PEI, 2011).

Segundo Fayyad (et al., 1996), a mineração de dados é “um passo no processo de Descoberta de Conhecimento que consiste na realização da análise dos dados e na aplicação de algoritmos de descoberta que, sob certas limitações computacionais, produzem um conjunto de padrões de certos dados”.

Existem muitas ferramentas utilizadas para executar algoritmos de mineração, e as mesmas contribuem para que os especialistas concentrem esforços apenas em partes mais significativas dos dados (SELIYA, KHOSHGOFTAAR, 2007). Algumas ferramentas a serem citadas, são: Weka,Oracle Data Mining (ODM), SAS Text Miner, KNIME, dentre outras (CAMILO, SILVA, 2009).

Algoritmos utilizados pela MD, em diferentes tarefas: Apriori, k-means, k-medoids, MTS, CART, Algoritmo Genético Simples,CHC, Redes de Kohonen, Rede Bam, dentre outros.

# 3. Tarefas da mineração de dados

A Mineração de Dados é classificada pela sua capacidade em realizar determinadas tarefas (LAROSE, 2005), tais como:

**Descrição:** pode ser definida como a tarefa usada para descrever padrões e tendências dos dados. Esta tarefa oferece uma possível interpretação para os resultados obtidos, sendo muito utilizada em conjunto com técnicas de análise exploratória de dados, para comprovar a influência de certas variáveis no resultado obtido.

**Classificação:** visa identificar a qual classe um determinado registro pertence. O modelo deve analisar os registros fornecidos, com cada registro contendo a indicação da classe pertencente, com o intuito de “aprender” como classificar um novo registro (HAN, KAMBER, 2006). Esta tarefa pode ser usada em diversos casos, como por exemplo:

• Determinar quando uma transação de cartão de crédito pode ser uma fraude;

• Diagnosticar onde uma determinada doença pode estar presente;

• Identificar quando uma pessoa pode ser uma ameaça para a segurança.

**Estimação:** esta tarefa é usada em registros identificados por valores numéricos (ao invés de categóricos). Desse modo, é possível estimar o valor de uma variável analisando-se os valores das demais. (GONZÁLEZ-ARANDA, MENASALVAS, RUIZ, SEGOVIA, 2008) Por exemplo: um conjunto de registros contendo os valores mensais gastos por diversos tipos de consumidores e de acordo com os hábitos de cada um. Após ter analisado os dados, o modelo é capaz de dizer qual será o valor gasto por um novo consumidor.

**Predição:** esta tarefa visa descobrir o valor futuro de um determinado atributo.

Alguns exemplos de predição:

• Predizer o valor de uma ação três meses adiante;

• Predizer o percentual que será aumentado de tráfego na rede se a velocidade aumentar;

**Agrupamento:** têm o objetivo de identificar e aproximar os registros similares. Um cluster é uma coleção de registros similares entre si, porém diferentes dos outros registros nos demais agrupamentos. O agrupamento não necessita que os registros sejam previamente categorizados. As aplicações das tarefas de agrupamento são as mais variadas possíveis: pesquisa de mercado, reconhecimento de padrões, processamento de imagens, análise de dados, entre outras (OLIVEIRA, CARVALHO, 2008). Geralmente a tarefa de agrupamento é combinada com outras tarefas, além de serem usadas na fase de preparação dos dados.

**Associação:** consiste em identificar quais atributos estão relacionados. Apresentam a forma: SE X ENTÃO Y. É uma das tarefas que apresenta bons resultados obtidos, principalmente na análise da "Cestas de Compras", em que identifica-se quais produtos são levados juntos pelos consumidores. Alguns exemplos:

• Determinar os casos onde um novo medicamento pode apresentar efeitos colaterais;

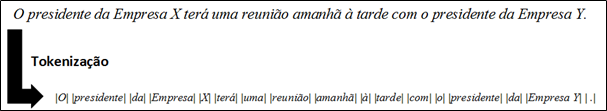
• Identificar os usuários de planos que respondem bem a oferta de novos serviços.

# 4. Processamento de Língua Natural

O Processamento de Língua Natural (PLN) consiste no estudo e desenvolvimento de modelos computacionais capazes de compreenderem alguma linguagem natural e realizarem tarefas que dependem deste tipo de informação (Russell & Norvig, 1995). O PLN abrange diversos processos, dentre os quais podemos citar a tokenização, a remoção de *stopwords* e *bag of words*.

## 4.1 Tokenização

A tokenização (aportuguesamento da palavra inglesa “*tokenization*”) objetiva a separação de palavras em unidades (*tokens*) (Manning, Raghavan, & Schütze, 2008) de acordo com regras previamente estabelecidas. Em geral, o caractere de espaço em branco é a regra usada para marcar os limites do *token* (Branco & Silva, 2006). A tokenização pode ser exemplificada pela Figura 1, que mostra a realização desse processo na frase “O presidente da Empresa X terá uma reunião amanhã à tarde com o presidente da Empresa Y”.

****

## 4.2 Stopwords

Na computação, *stopwords* são palavras filtradas antes ou depois do processamento do texto. Embora essa lista de palavras se refira às palavras mais comuns em um idioma, não há uma “lista universal” de *stopwords* e, na verdade, nem todas as ferramentas usam essa lista.

Algumas ferramentas evitam especificamente remover essas *stopwords* para oferecer suporte à pesquisa de frases. Qualquer grupo de palavras pode ser escolhido, tais como preposições e artigos.

# 5. Descrição do problema

Vídeos do Youtube recebem comentários maliciosos diariamente, contendo links que redirecionam o usuário para sites com vírus específicos para a captura de dados pessoais, como login e senha. Ao obter os dados pessoais do usuário, os criminosos virtuais podem acessar contas bancárias, redes sociais, emails, lista de contatos, ou quaisquer outras informações.

Os criminosos virtuais, em geral, utilizam videoclipes para disseminar os comentários maliciosos, e apresentam informações, como: notícias falsas sobre a morte do cantor, supostas fotos contendo nudez explícita, ofertas e falsas possibilidades de ganhar prêmios.

A partir disso, considera-se importante detectar e distinguir quais comentários são Spam, para que os mesmos possam ser deletados, evitando assim que usuários desavisados sejam enganados por esse tipo de armadilha.

## 5.1 Atributos do problema

O conjunto de dados foi retirado da base de dados da Universidade da Califórnia, no repositório **Center for Machine Learning and Intelligent Systems** (2017). Cada conjunto de dados refere-se a artistas diferentes (Shakira, PSY, LMFao, Eminem e Katy Perry).

Os atributos, antes do pré-processamento, estavam dispostos da seguinte forma:

|  |  |  |
| --- | --- | --- |
| **Nome** | **Domínio** | **Presença de ruído** |
| id | textual | não |
| author | textual | não |
| date | date | sim |
| content | textual | não |
| class | binário | não |



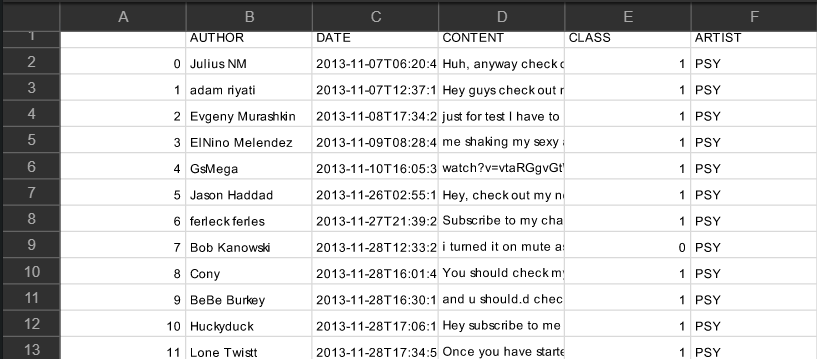
Conjunto de dados antes do Pré-Processamento

## 5.2 Pré-Processamento

O pré-processamento foi realizado a partir da biblioteca **pandas[[1]](#footnote-0)**, pertencente à linguagem *python,* através de um *script* (Anexo A). Com relação ao atributo date, que apresentava ruídos relativos à falta de dados, foi utilizada a operação ***fillna*** para preenchê-los automaticamente. O atributo id (do comentário) foi excluído, pois mostrava-se desnecessário. Além disso, os cinco conjuntos de dados foram integrados em apenas um.

Após o pré-processamento os atributos ficaram dispostos da seguinte forma:

|  |  |
| --- | --- |
| **Nome** | **Domínio** |
| author | textual |
| date | date |
| content | textual |
| class | binário |
| artist | textual |

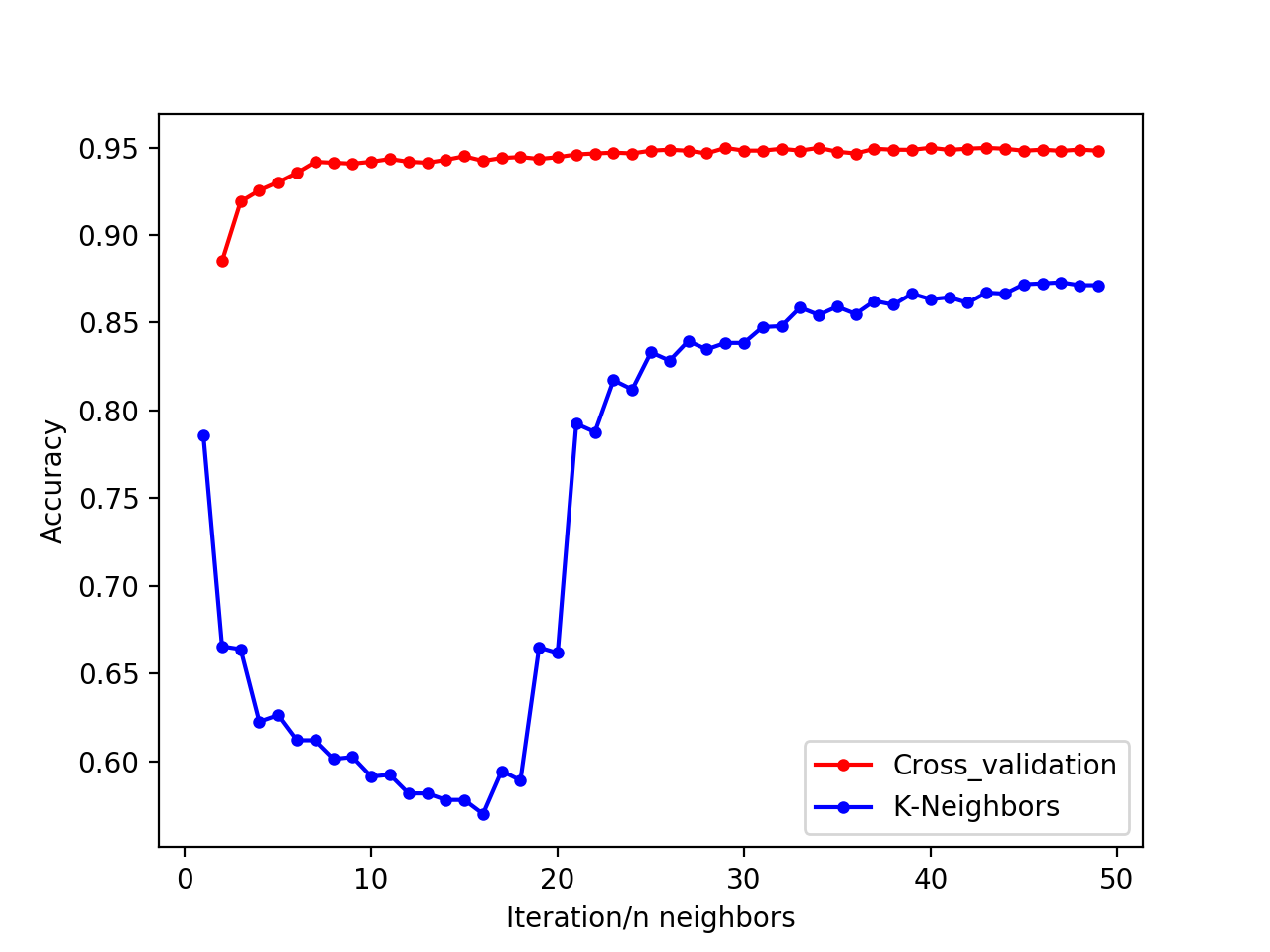
****

Conjunto de dados após o processamento

## 5.3. Tarefa aplicada

A tarefa aplicada foi a classificação, sendo atribuído o valor 1 para comentário malicioso (spam) e 0 para comentário normal (não-spam). A escolha do algoritmo foi feita através do método *cross validation[[2]](#footnote-1)*. Além disso, foram usados a acurácia e o número de vizinhos para testar e avaliar a qualidade dos algoritmos *logistic regression* e *K nearest neighbors (KNN)*, respectivamente.

Foi utilizado um script em *python* (Anexo B)juntamente à biblioteca **pandas**, para a importação do conjunto de dados de teste (artista Katy Perry), além da biblioteca *skit-learn***[[3]](#footnote-2)***,* aplicada para a realização de *cross validation*

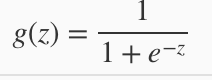
.

Resultado do teste

## 5.3.1. *Logistic Regression* - Modelo de classificação

Logistic regression é um modelo que utiliza uma classificação binária e constrói **N** classificadores binários diferentes. Para cada classificador de ordem **i**, este é considerado a classe positiva e os demais, negativa (RIFKIN, 2008). Este algoritmo apresenta uma boa performance em separação linear de classes.

O modelo faz as previsões da probabilidade de uma determinada amostra pertencente a uma classe, utilizando a função *sigmoid function*, apresentada na figura abaixo(RASCHKA,2015).



Fórmula da função sigmóide

Onde z é o somatório dos valores de entrada (*input values)* de cada amostra, multiplicado pelo peso (*weight)* correspondente a cada valor (**z = w0 + x1 \* w1 + … + xm \* wm**) (RASCHKA,2015).

A forma da função sigmoid remete à um s, conforme o gráfico abaixo:

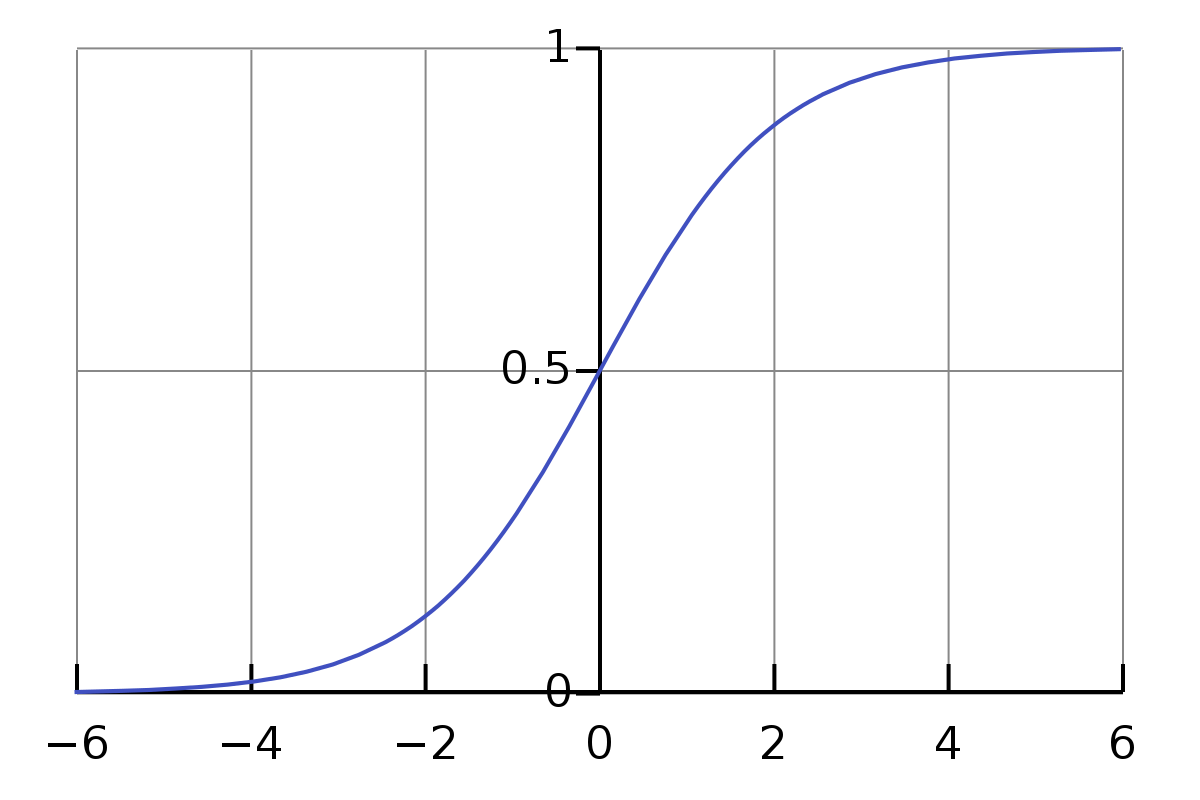


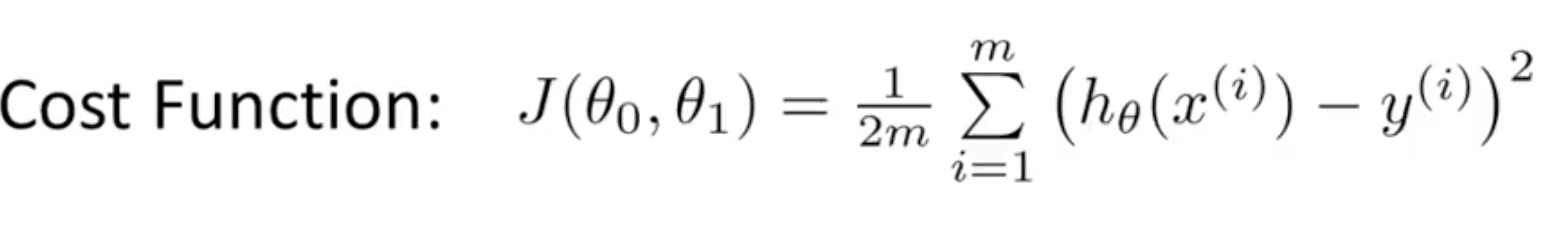
Gráfico da função da sigmóide.

É possível observar que "g(z)" se aproxima de 1 quando tende ao infinito (z→∞), a medida em que o valor de "e^-z" cresce. Assim como "g(z)" se aproxima de 0 quando tende a menos infinito, (z→-∞) a medida em que o denominador da função cresce [RASCHKA,2015].

O algoritmo *logistic regression,*  além de prever a classe a qual as amostras pertencem, também é capaz de estimar a probabilidade.

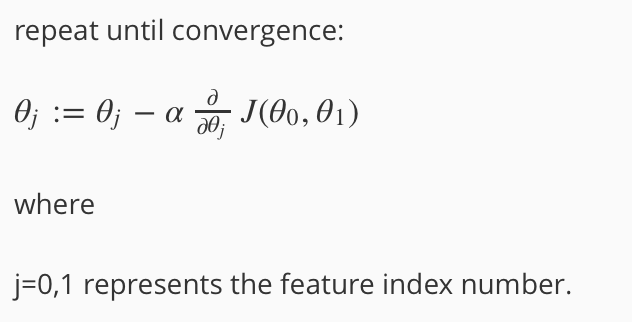
Por exemplo, ao aplicar a *sigmoid function* em uma amostra cuja classe possui rótulo 1, e seu valor de saída é 0.8, dizemos que a probabilidade dessa amostra pertencer à classe 1 é de 80%, sendo assim, a probabilidade da mesma amostra pertencer à classe de rótulo 0 é 20% (RASCHKA,2015).

O peso (também chamado de theta ou parâmetro) é responsável por fazer a generalização[[4]](#footnote-3) do algoritmo. Primeiramente é calculado a média do erro, através da *cost function*, como é mostrado na figura 3 (NG, 2008).



Fórmula *cost function*

Em seguida, utiliza-se uma fórmula iterativa, chamada *gradient descent*, para minimizar o erro ao máximo, conforme a figura abaixo (NG, 2008).



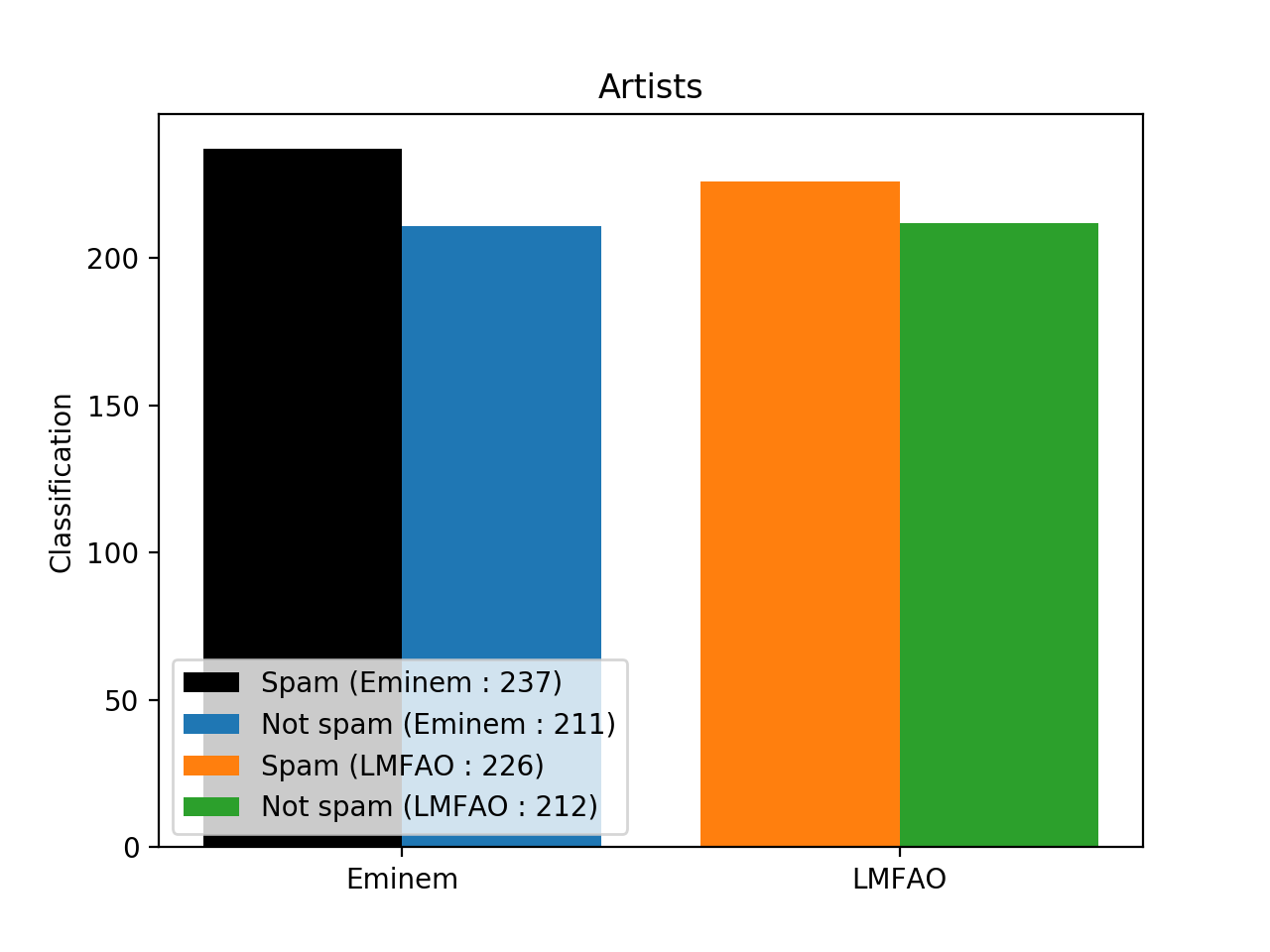
Fórmula *gradient descent*

# 6. Resultado

# 

Um script em python (Anexo C) foi feito para treinar e testar o algoritmo escolhido. Esta etapa foi dividida em treinamento e realização de testes.

Após o treinamento, foram feitos os testes para as duas partes selecionadas, cada uma pertencente a um artista. Foi gerado um gráfico utilizando a biblioteca *matplotlib***[[5]](#footnote-4)** para a visualização do resultado do teste (classes preditas pelo algoritmo).

 Gráfico do resultado do teste.

Também foi obtida a acurácia de cada subconjunto testado.

|  |  |
| --- | --- |
| Subconjunto (Artista) | Acurácia (arredondada) |
| Eminem | 0.98 |
| LMFAO | 0.92 |

Após a análise dos dos dados obtidos pelo teste, é possível concluir que este é um bom classificador.

# 7. Conclusão

Através deste trabalho, foi possível verificar os problemas de segurança enfrentados pelos usuários do Youtube no quesito Spam. Os usuários estão vulneráveis a ameaças de todos os tipos, realizadas tanto por humanos, quanto por SpamBots.

Esse problema pode ser contido, ou até mesmo solucionado, por meio da aplicação da mineração de dados. Essa importante área da ciência da computação oferece diversas tarefas que podem ser aplicadas de diferentes maneiras a um conjunto de dados. Podemos destacar a classificação, como sendo a possível solução para a problemática de Spams.

A tarefa de classificação é capaz de rotular registros de um conjunto de dados como **Spam**, ou **não-Spam** por meio de um classificador. O classificador deve ser treinado e testado, para que se possa saber qual a eficiência apresentada por ele.

Através do treinamento e testes realizados no conjunto de dados deste trabalho, pôde-se constatar que a cada 886 registros, 52,26% apresentam Spam, ou seja, mais da metade dos registros eram comentários maliciosos. A confiabilidade dos resultados é alta, pois o classificador apresentou mais de 90% de acertos.

Logo, pode-se concluir que são necessárias medidas para que este problema seja contido. O Youtube poderia usar bots capazes de classificar e remover esses comentários, visando aumentar a confiabilidade do site, bem como a segurança de seus usuários.

# 8. Referências Bibliográficas

WITTEN, I. H.; FRANK, E. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2005.

J. HAN; M. KAMBER; J. PEI – Data Mining: Concepts and Techniques (2011).

FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P. Advances in knowledge discovery and data mining. In: FAYYAD, U. M. et al. (Ed.). Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996. cap. From Data Mining to Knowledge Discovery: An Overview, p. 1–34. ISBN 0-262-56097-6

CIOS, K. J; PEDRYCZ, W; SWINIARSKI, R. W; KURGAN, L. A. Data Mining - A Knowledge Discovery Approach. Springer, 2007.

SELIYA, N; KHOSHGOFTAAR, T. M. Software Quality Modeling With Limited Apriori Defect Data, chapter Chapter 1, p. 1–16. Idea Group Publishing, 2007.

Wang, J, editor. Encyclopedia of Data Warehousing and Mining. Idea Group Reference, 2005.

LAROSE, D. T. Discovering Knowledge in Data: An Introduction to Data Mining. John Wiley and Sons, Inc, 2005.

OLIVEIRA, R. R; CARVALHO, C. L. Algoritmos de agrupamento e suas aplicações. Technical report, Universidade Federal de Goiás, 2008.

CAMILO, C.; SIlVA, C. Mineração de Dados: Conceitos, Tarefas, Métodos e Ferramentas. Technical report, Universidade Federal de Goiás, 2009.

HAN, J; KAMBER, M. Data Mining: Concepts and Techniques. Elsevier, 2006

GONZÁLEZ-ARANDA, P; MENASALVAS, E; RUIZ, S. M. C; SEGOVIA, J. Towards a methodology for data mining project development: The importance of abstraction. In: STUDIES IN COMPUTATIONAL INTELLIGENCE, p. 165U178. Springer-Verlag, ˝ 2008.

DIAS, M. M. Parâmetros na escolha de técnicas e ferramentas de mineração de dados. Departamento de Informática, Universidade Estadual de Maringá, 2002.

Russell, S., & Norvig, P. (1995). *Artificial Intelligence - a modern approach.* Prentice Hall.

Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.

Branco, A., & Silva, J. R. (2006). *A Suite of Shallow Processing Tools for Portuguese: LX-Suite.* University of Lisbon, Department of Informatics.

RASCHKA,S. Python machine learning, p. 456. Packt Publishing, 2015.

NG, Andrew. Lecture 3 | Machine Learning. Disponível em <https://www.youtube.com/watch?v=HZ4cvaztQEs&list=PLA89DCFA6ADACE599&index=2> . Acesso em : 02 mai. 2018.

DANNY. *37 Mind Blowing YouTube Facts, Figures and Statistics – 2018.* Abril de 2018, sl. Disponível em <<https://merchdope.com/youtube-statistics/>>. Acesso em 8 Abr. 2018.

# 9. Anexos

## Anexo A

*'''*

*Script para transformar conjunto de dados em bag of words, aplicar o algoritmo e visualizar*

*resultado da classificação.*

*Segue os mesmos passos do script de teste, com a diferença de não ter necessidade de aplicar*

*cross validation.*

*Bibliotecas usadas :*

*Pandas = biblioteca de Data Science utilizada na manipulação de arquivos em csv e xls.*

*SkLearn = Biblioteca utilizada para a aplicação de Machine Learning;*

*Numpy = Biblioteca para trabalhar com operações matemáticas com mais eficiências*

*'''*

**import** pandas **as** pd

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.feature\_extraction.text **import** CountVectorizer

**from** sklearn.feature\_extraction.text **import** TfidfTransformer

**from** sklearn **import** metrics

**import** numpy **as** np

**from** matplotlib **import** pyplot **as** plt

df = pd.read\_csv(**"merged.csv"**,index\_col=**"ARTIST"**,parse\_dates=[**"DATE"**])

lr = LogisticRegression()

count\_vect = CountVectorizer()

tfidf\_transformer = TfidfTransformer()

*# Nota : 1 = Eminem, 2 = LMFA) conj. = conjunto de dados*

**'''**

**Separa em três partes o conjunto de dados :**

**1 - training\_data = conj. a ser treinado**

**2 - predict\_data1 = conj. a ser testado**

**3 - predict\_data2 = onj. a ser testado**

**'''**

traing\_data,predict\_data1 = [x **for** \_, x **in** df.groupby(df.index == **'EMINEM'**)]

traing\_data,predict\_data2 = [x **for** \_, x **in** df.groupby(df.index == **'LMFAO'**)]

*# Processo de transformar o conj. para treinar em bag of words*

count\_bag\_train\_c = count\_vect.fit\_transform(traing\_data.CONTENT)

count\_bag\_train\_t = tfidf\_transformer.fit\_transform(count\_bag\_train\_c)

*# Processo de transformar o conj. para testar em bag of words*

count\_vect2 = CountVectorizer(vocabulary=count\_vect.get\_feature\_names())

count\_bag\_pred1 = count\_vect2.fit\_transform(predict\_data1.CONTENT)

count\_bag\_pred2 = count\_vect2.fit\_transform(predict\_data2.CONTENT)

count\_bag\_pred1 = tfidf\_transformer.fit\_transform(count\_bag\_pred1)

count\_bag\_pred2 = tfidf\_transformer.fit\_transform(count\_bag\_pred2)

*#Treina o algoritmo com o conj. para treinar e o vetor com os rótulos*

*#contidos na coluna "CLASS" do conj. inicial*

lr.fit(count\_bag\_train\_t,traing\_data.CLASS)

*#Faz o teste do algortimos com os conj. 1 e 2*

y\_pred1 = lr.predict(count\_bag\_pred1)

y\_pred2 = lr.predict(count\_bag\_pred2)

y\_pred1 = list(y\_pred1)

y\_pred2 = list(y\_pred2)

*# Calcula e mostra a porcentagem da precisão de cada con. de teste,*

*# utiliza a classe metrics.*

print(**"Eminem : "**,(metrics.accuracy\_score(predict\_data1.CLASS,y\_pred1)\* 100),**'%'** )

print(**"LMFAO : "** , (metrics.accuracy\_score(predict\_data2.CLASS,y\_pred2)\* 100),**'%'** )

*# Mostra um gráfico de barras com os resultados, onde y corresponde aos artistas e x ao*

*# número de spam/não spam de cada um deles.*

artists = [ **"Eminem"**,**"LMFAO"**]

xpos = np.arange( len( artists ) )

plt.xticks( xpos , artists ) *# replace the elemenst in the legend*

plt.bar( xpos[0]-0.2 , [y\_pred1.count(1)] ,width = 0.4, color =**'black'** , label =(**"Spam ("** + **"Eminem : "** + str(y\_pred1.count(1)) + **")"**))

plt.bar( xpos[0]+0.2 , [y\_pred1.count(0)],width = 0.4, label=(**"Not spam ("** + **"Eminem : "** + str(y\_pred1.count(0)) + **")"**))

plt.bar( xpos[1]-0.2 , [y\_pred2.count(1)] ,width = 0.4 , label =(**"Spam ("** + **"LMFAO : "** + str(y\_pred2.count(1)) + **")"**))

plt.bar( xpos[1]+0.2 , [y\_pred2.count(0)],width = 0.4, label=(**"Not spam ("** + **"LMFAO : "** + str(y\_pred2.count(0)) + **")"**))

plt.ylabel(**"Classification"**)

plt.title(**'Artists'**)

plt.legend(loc = **'lower left'**)

*#plt.show()*

# 

## Anexo B

*'''*

*Esse script foi utilizado para testar 2 algoritmos, "Logistic Regression"*

*e " K Nearest Neighbor". Consiste em :*

*1 - Importar o conjunto de dados para aplicar a tarefa;*

*2 - Utilizar o método "bag of words" para deixar o conjunto de dados*

*própria para aplicar o algoritmo;*

*3- Limpar alguns atributos do conjunto de dados;*

*4- Salvar a o conjunto de dados por segurança e para análises futuras;*

*5- Testar os algoritmos e*

*6- Apresentar o resultado do teste em forma de gráfico.*

*Biblioteca(s) utilizada(s) :*

*Pandas = biblioteca de Data Science utilizada na manipulação de arquivos em csv e xls.*

*SkLearn = Biblioteca utilizada para a aplicação de Machine Learning;*

*MatplotLib = Biblioteca para a programação de gráficos.*

*'''*

**import** pandas **as** pd

**from** sklearn.model\_selection **import** cross\_val\_score

**from** sklearn.neighbors **import** KNeighborsClassifier

**import** matplotlib.pyplot **as** plt

**from** sklearn.feature\_extraction.text **import** TfidfTransformer

**from** sklearn.feature\_extraction.text **import** CountVectorizer

**from** sklearn.linear\_model **import** LogisticRegression

**'''**

**PRÉ PROCESSAMENTO**

**'''**

*# Importa o conjunto de dados.*

df = pd.read\_csv(**"Youtube02-KatyPerry.csv"**,index\_col=**"AUTHOR"**,parse\_dates=[**"DATE"**])

*# objeto da classe CountVectorizer, esta transforma o conjunto de dados.*

*# ou parte dele em un bag of words .*

count\_vect = CountVectorizer()

*# Faz o "encaixe" dos dados contidos na coluna "CONTENT" do conjunto de dados.*

X\_count= count\_vect.fit\_transform(df.CONTENT)

*# objeto da classe TfidfTransformer (term frequency–inverse document frequency), esta limpa o "bag of words",*

*# tirando as palavras mais "comuns" e e sem muita relevância.*

tfidf\_transformer = TfidfTransformer()

*#Faz o "encaixe" dos da matriz contida no "bag of words".*

X\_tfidf = tfidf\_transformer.fit\_transform(X\_count)

*# Salva o "bag of words" em um arquivo csv*

d = pd.DataFrame(X\_count.A,columns=count\_vect.get\_feature\_names())

d.to\_csv(**"newD.csv"**)

**'''**

**TESTE DE ALGORITMOS PARA A TAREFA DE CLASSIFICAÇÃO**

**'''**

*# Número de iterações que determina o estratégia de separação do cross-validation splitting strategy*

cross\_validation\_range = list(range(2,50))

*# Lista que armazena os valores de cada iteração do cross-validation*

result\_LR = []

*# Objeto da classe Logistic Regression, que é responsável por aplicar o algoritmo de logistic regression*

*#na matriz formada anteriormente.*

lr = LogisticRegression()

*#Laço que itera o cross validation-scores com os valores determinados anteriormente,*

*#utilizando a precisão como métrica e armazena o resultado na lista.*

**for** i **in** cross\_validation\_range:

scores = cross\_val\_score(lr, X\_tfidf,df.CLASS, cv =i, scoring=**'accuracy'**)

result\_LR.append(scores.mean())

*# Número de iterações que determina o estratégia de separação do cross-validation splitting strategy*

k\_range = list(range(1,50))

*# Lista que armazena os valores de cada iteração do cross-validation*

k\_scores = []

*#Laço que itera o número de vizinhos com os valores determinados anteriormente,*

*#utilizando a precisão como métrica e armazena o resultado na lista.*

**for** i **in** k\_range:

knn = KNeighborsClassifier ( n\_neighbors=i )

scores = cross\_val\_score(knn, X\_tfidf,df.CLASS, cv =20, scoring=**'accuracy'**)

k\_scores.append(scores.mean())

**'''**

**VISUALIZAÇÃO DO RESULTADO DO TESTE**

**'''**

*#Plota o resultado da execução do logistic regression*

plt.plot(cross\_validation\_range,result\_LR,color=**'red'**,marker = **'.'**,markersize = 7,label=**"Logistic Regression"**)

*#Plota o resultado da execução do k nearest neighbor*

plt.plot(k\_range,k\_scores,color=**'blue'**,marker = **'.'**,markersize = 7,label=**"K-Neighbors"**)

*# Coloca legenda no gráfico*

plt.xlabel(**'Iteration/n neighbors'**)

plt.ylabel(**'Accuracy'**)

*#Habilita a legenda*

plt.legend()

*#Mostra o gráfico*

plt.show()

## Anexo C

*'''*

*Script para transformar conjunto de dados em bag of words, aplicar o algoritmo e visualizar*

*resultado da classificação.*

*Segue os mesmos passos do script de teste, com a diferença de não ter necessidade de aplicar*

*cross validation.*

*Bibliotecas usadas :*

*Pandas = biblioteca de Data Science utilizada na manipulação de arquivos em csv e xls.*

*SkLearn = Biblioteca utilizada para a aplicação de Machine Learning;*

*Numpy = Biblioteca para trabalhar com operações matemáticas com mais eficiências*

*'''*

**import** pandas **as** pd

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.feature\_extraction.text **import** CountVectorizer

**from** sklearn.feature\_extraction.text **import** TfidfTransformer

**from** sklearn **import** metrics

**import** numpy **as** np

**from** matplotlib **import** pyplot **as** plt

df = pd.read\_csv(**"merged.csv"**,index\_col=**"ARTIST"**,parse\_dates=[**"DATE"**])

lr = LogisticRegression()

count\_vect = CountVectorizer()

tfidf\_transformer = TfidfTransformer()

*# Nota : 1 = Eminem, 2 = LMFAO conj. = conjunto de dados*

**'''**

**Separa em três partes o conjunto de dados :**

**1 - training\_data = conj. a ser treinado**

**2 - predict\_data1 = conj. a ser testado**

**3 - predict\_data2 = onj. a ser testado**

**'''**

traing\_data,predict\_data1 = [x **for** \_, x **in** df.groupby(df.index == **'EMINEM'**)]

traing\_data,predict\_data2 = [x **for** \_, x **in** df.groupby(df.index == **'LMFAO'**)]

*# Processo de transformar o conj. para treinar em bag of words*

count\_bag\_train\_c = count\_vect.fit\_transform(traing\_data.CONTENT)

count\_bag\_train\_t = tfidf\_transformer.fit\_transform(count\_bag\_train\_c)

*# Processo de transformar o conj. para testar em bag of words*

count\_vect2 = CountVectorizer(vocabulary=count\_vect.get\_feature\_names())

count\_bag\_pred1 = count\_vect2.fit\_transform(predict\_data1.CONTENT)

count\_bag\_pred2 = count\_vect2.fit\_transform(predict\_data2.CONTENT)

count\_bag\_pred1 = tfidf\_transformer.fit\_transform(count\_bag\_pred1)

count\_bag\_pred2 = tfidf\_transformer.fit\_transform(count\_bag\_pred2)

*#Treina o algoritmo com o conj. para treinar e o vetor com os rótulos*

*#contidos na coluna "CLASS" do conj. inicial*

lr.fit(count\_bag\_train\_t,traing\_data.CLASS)

*#Faz o teste do algoritmos com os conj. 1 e 2*

y\_pred1 = lr.predict(count\_bag\_pred1)

y\_pred2 = lr.predict(count\_bag\_pred2)

y\_pred1 = list(y\_pred1)

y\_pred2 = list(y\_pred2)

*# Calcula e mostra a porcentagem da precisão de cada con. de teste,*

*# utiliza a classe metrics.*

print(**"Eminem : "**,(metrics.accuracy\_score(predict\_data1.CLASS,y\_pred1)\* 100),**'%'** )

print(**"LMFAO : "** , (metrics.accuracy\_score(predict\_data2.CLASS,y\_pred2)\* 100),**'%'** )

*# Mostra um gráfico de barras com os resultados, onde y corresponde aos artistas e x ao*

*# número de spam/não spam de cada um deles.*

artists = [ **"Eminem"**,**"LMFAO"**]

xpos = np.arange( len( artists ) )

plt.xticks( xpos , artists ) *# replace the elements in the legend*

plt.bar( xpos[0]-0.2 , [y\_pred1.count(1)] ,width = 0.4, color =**'black'** , label =(**"Spam ("** + **"Eminem : "** + str(y\_pred1.count(1)) + **")"**))

plt.bar( xpos[0]+0.2 , [y\_pred1.count(0)],width = 0.4, label=(**"Not spam ("** + **"Eminem : "** + str(y\_pred1.count(0)) + **")"**))

plt.bar( xpos[1]-0.2 , [y\_pred2.count(1)] ,width = 0.4 , label =(**"Spam ("** + **"LMFAO : "** + str(y\_pred2.count(1)) + **")"**))

plt.bar( xpos[1]+0.2 , [y\_pred2.count(0)],width = 0.4, label=(**"Not spam ("** + **"LMFAO : "** + str(y\_pred2.count(0)) + **")"**))

plt.ylabel(**"Classification"**)

plt.title(**'Artists'**)

plt.legend(loc = **'lower left'**)

plt.show()

1. Biblioteca de Data Science utilizada na manipulação de arquivos em csv e xls. [↑](#footnote-ref-0)
2. Técnica para avaliar modelos preditivos através da divisão do conjunto original em um conjunto de treinamento e um conjunto de teste. [↑](#footnote-ref-1)
3. Biblioteca para a aplicação de técnicas de *machine learning*. [↑](#footnote-ref-2)
4. Capacidade do modelo de gerar uma predição satisfatória, ou seja, quão bom o modelo é na predição de uma classe. [↑](#footnote-ref-3)
5. Biblioteca para programar gráficos em *python*. [↑](#footnote-ref-4)