



# Sistema de Classificação de SMSs para Identificação de Spam utilizando o Classificador Ingênuo de Bayes

---

# Integrantes da Equipe

---

Amanda Cristina Fernandes M. de Lima (acfml)

Maria Letícia do N. Gaspar (mlng)

Mariana Melo dos Santos (mms11)

Victória Barbosa Cesar Figueiredo(vbcf)

# Classificador ingênuo de Bayes

---

Trata-se de uma solução simples para problemas de classificação

## O que é?

É um algoritmo de aprendizagem de máquina (Machine Learning) que realiza previsões baseadas em estatísticas.

## Como funciona?

Já como diz o nome, ele analisa a base de dados de forma “ingênua”, ou seja, assume que os atributos dos dados são independentes entre si. ele também assume que eles sejam todos igualmente importantes para o resultado

# a Matemática do CIB

- **O cálculo é baseado no Teorema de Bayes**
- **Independência entre atributos**
- **probabilidade da predição dentro de uma hipótese**

The diagram illustrates Bayes' Theorem with the following components:

- Likelihood**: Points to the term  $P(x|c)$  in the numerator.
- Class Prior Probability**: Points to the term  $P(c)$  in the numerator.
- Posterior Probability**: Points to the term  $P(c|x)$  on the left side of the equation.
- Predictor Prior Probability**: Points to the term  $P(x)$  in the denominator.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$
$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

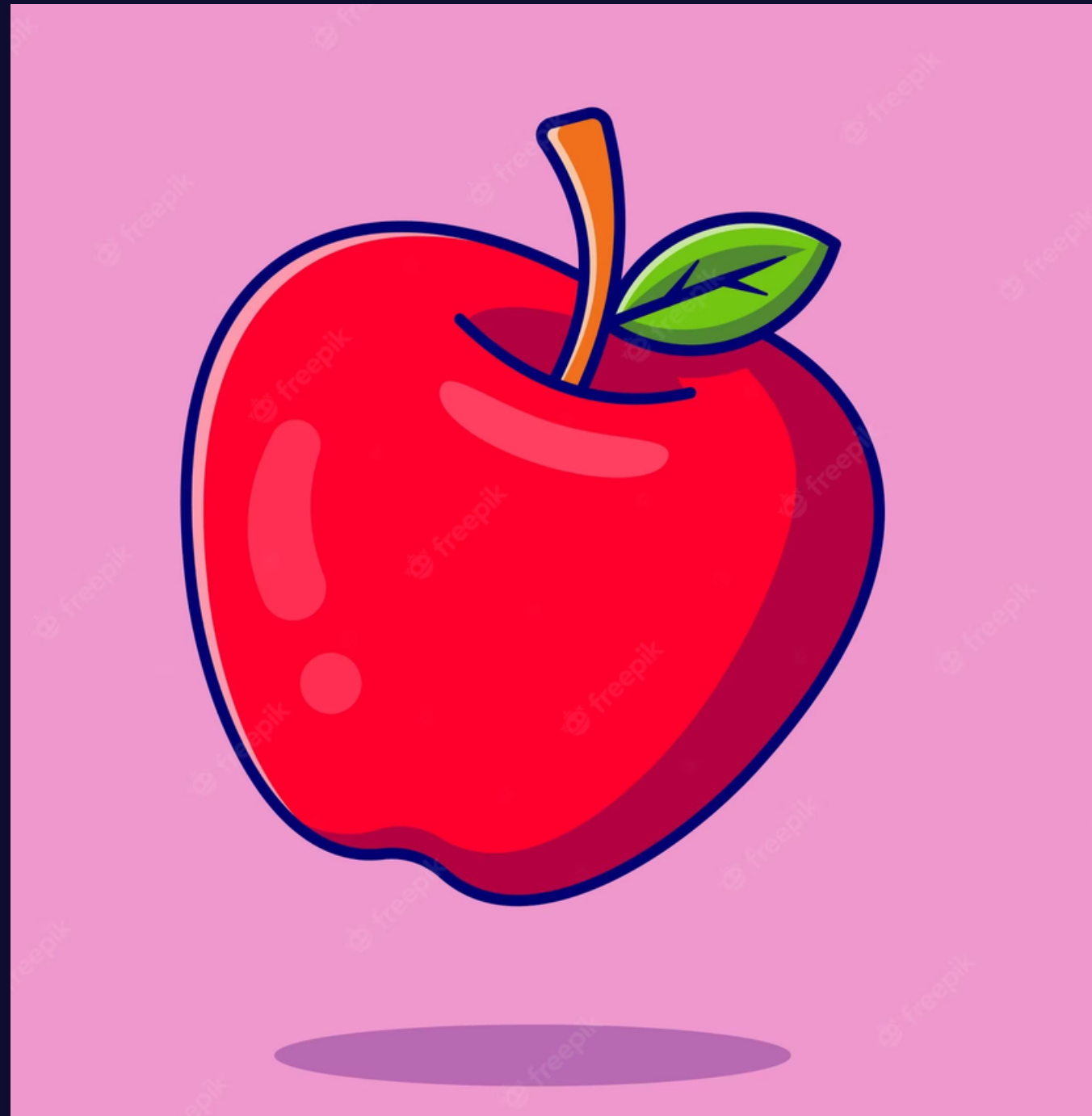
# ClB na prática

- > define-se uma tabela de probabilidades das previsões, com base nas variáveis de saída**
- > a previsão de maior probabilidade é escolhida como a classificação em spam ou não-spam.**

# Matemática do ClB

## Classes

- **Vermelha**
- **redonda**
- **diâmetro de 6 cm**




## Predição

- **Maçã**
- **Não é maçã**

# Base de dados


- Spambase – UCI Machine Learning


 UCI MACHINE LEARNING · UPDATED 7 YEARS AGO

▲ 1277

New Notebook


Download (216 kB)





## SMS Spam Collection Dataset

Collection of SMS messages tagged as spam or legitimate



Data Card

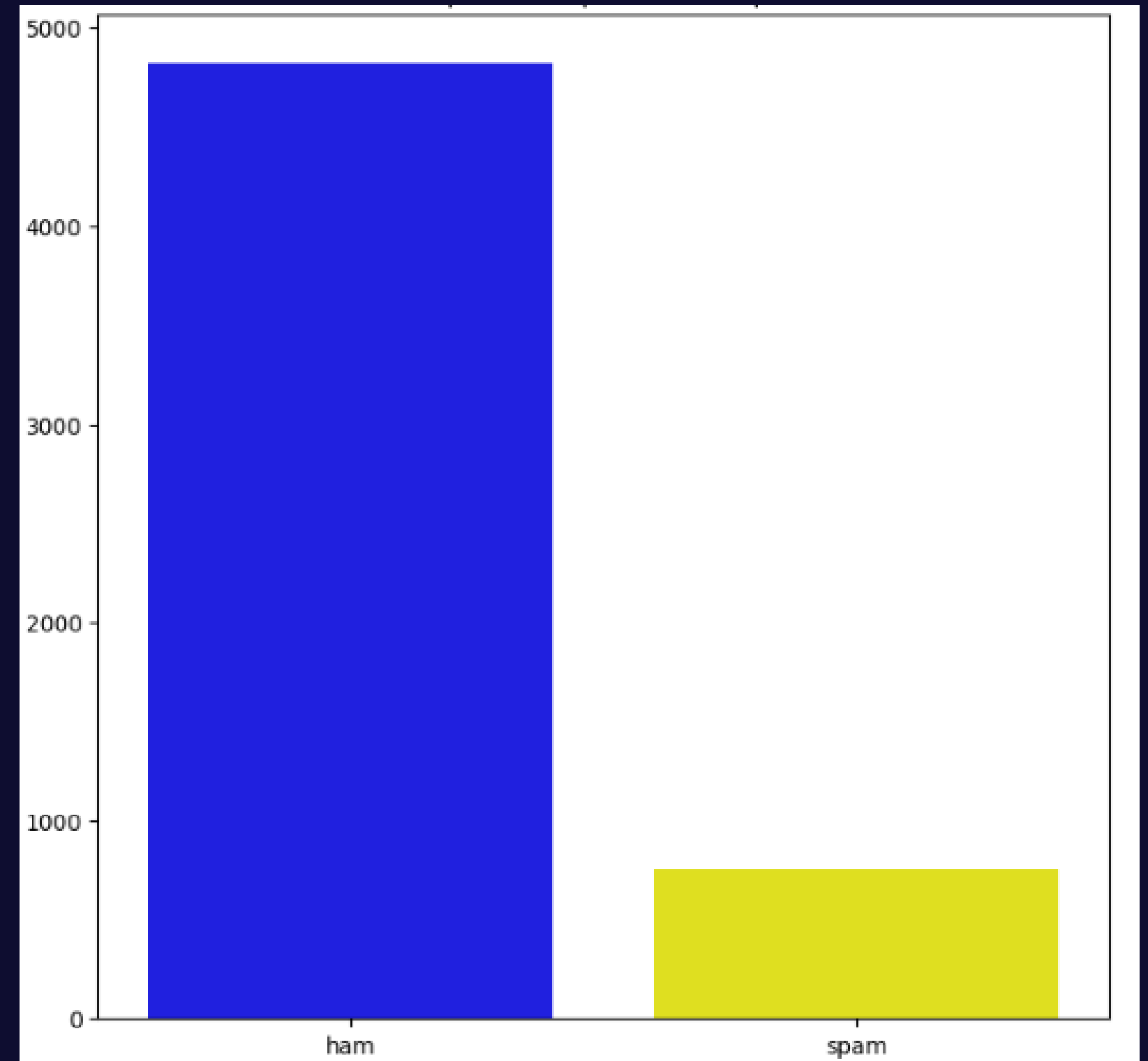
Code (1034)

Discussion (7)

# Análise Exploratória dos Dados

## Visão geral do dataset

- 5.572 mensagens de texto
- 87% ham e 13% spam
- Tabela com 5 colunas
- Descarte de colunas irrelevantes
- Agrupamento de dados
- Análise por categoria





## ▼ 1. Análise Exploratória dos Dados

- visão geral do dataset
- tamanho da mensagem

```
#importar database  
mensagens = pd.read_csv('spam.csv', encoding = 'latin-1')  
mensagens.sample(5)
```



	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
3240	ham	Ok i've sent u da latest version of da project.	NaN	NaN	NaN
4131	ham	Hi baby ive just got back from work and i was ...	NaN	NaN	NaN
5387	ham	I will be gentle baby! Soon you will be taking...	NaN	NaN	NaN
1962	spam	it to 80488. Your 500 free text messages are v...	NaN	NaN	NaN
403	ham	None of that's happening til you get here though	NaN	NaN	NaN

```
#apagar as últimas três colunas
mensagens.drop(columns = ['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], inplace = True)

#renomear as colunas
mensagens = mensagens.rename(columns = {'v1': 'categoria', 'v2': 'mensagem'})

mensagens.sample(5)
```



categoria

mensagem

2638	ham	Am in gobi arts college
1642	ham	Sleeping nt feeling well
4574	ham	Not directly behind... Abt 4 rows behind I_...
5269	spam	If you don't, your prize will go to another cu...
2990	spam	HOT LIVE FANTASIES call now 08707509020 Just 2...

```
[ ] #agrupamento por categoria (spam or not spam)
    mensagens.groupby('categoria').describe()
```

mensagem					
	count	unique	top	freq	
categoria					
ham	4825	4516		Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...		4

```
▶ #cria uma nova coluna que indica por dados numéricos (0,1) se é ou não spam - uso nos modelos de classificação
mensagens['spam'] = mensagens['categoria'].apply(lambda x:1 if x == 'spam' else 0)

#cria uma nova coluna que indica a quantidade de caracteres
mensagens['qtd_caracteres'] = mensagens['mensagem'].apply(len)

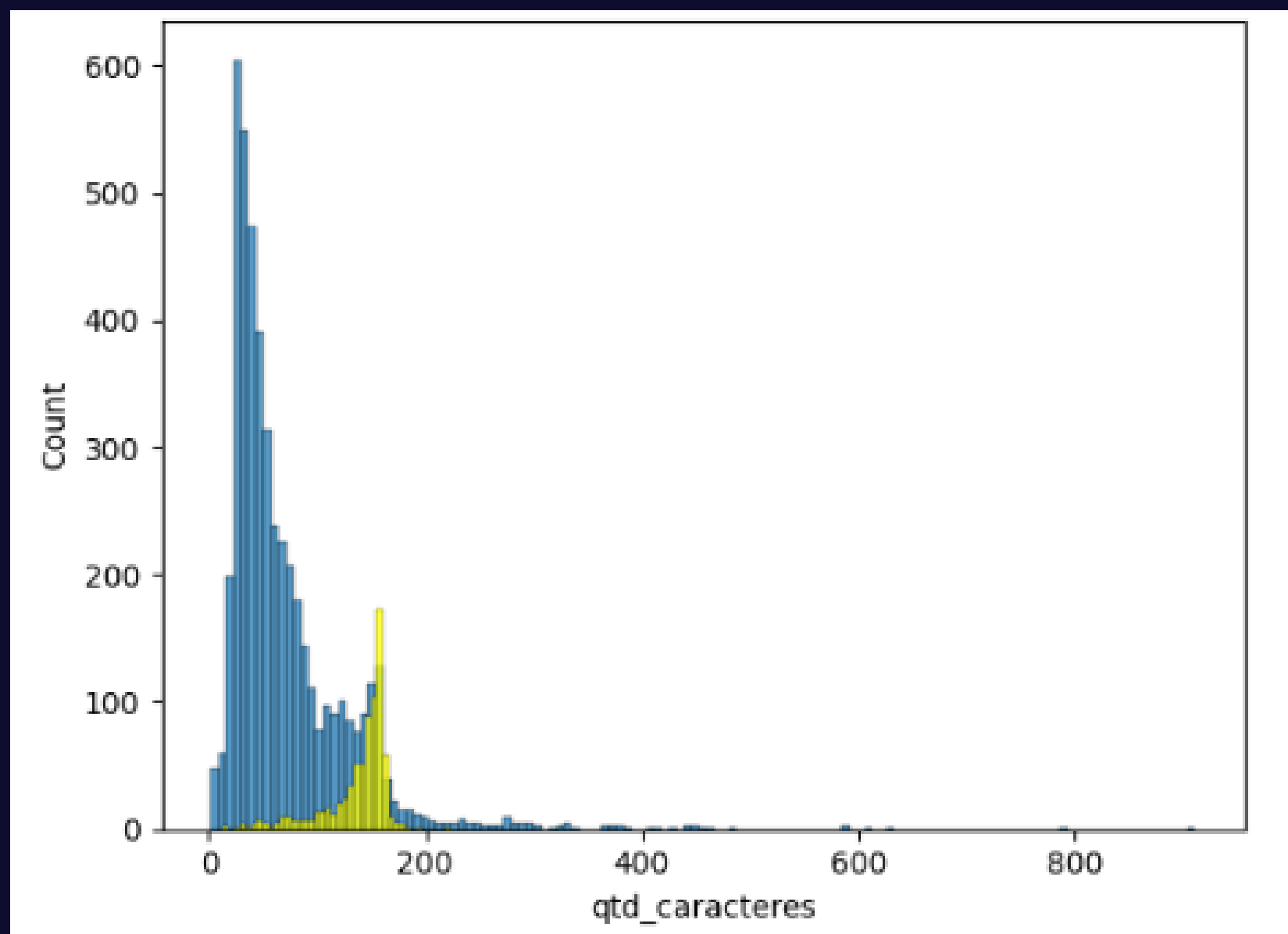
#cria uma nova coluna que indica a quantidade de palavras
mensagens['qtd_palavras'] = mensagens['mensagem'].apply(lambda x:len(nltk.word_tokenize(x)))

mensagens.sample(5)
```

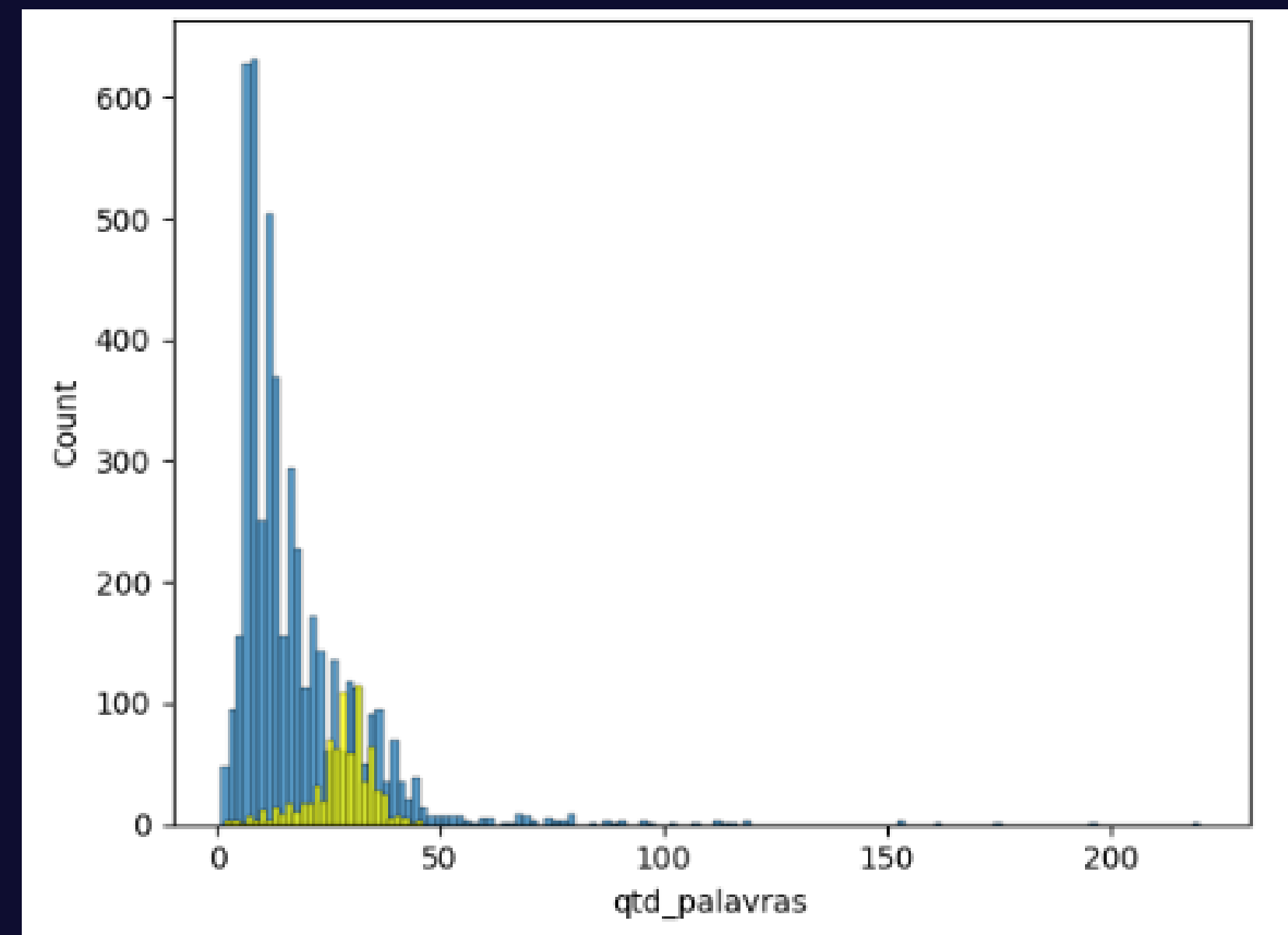


	categoria	mensagem	spam	qtd_caracteres	qtd_palavras
4678	ham	Sry da..jst nw only i came to home..	0	36	11
4091	ham	I remain unconvinced that this isn't an elabor...	0	70	13
1070	ham	alright, I'll make sure the car is back tonight	0	47	11
3213	ham	Babe, have you got enough money to pick up bre...	0	105	26
3627	ham	Meeting u is my work. . . Tel me when shall i ...	0	65	17

# Frequência de caracteres e palavras por SMS



● Ham ● Spam



● Ham ● Spam

# Pré-processamento dos dados

---

Etapas:

- Conversão para letras minúsculas
- Remoção de Pontuações
- Remoção de StopWords
- Tokenização
- Vetorização
- Ponderação TF-IDF

```
[ ] import string
    from nltk.corpus import stopwords
```

```
▶ string.punctuation
```

```
▶ '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
[ ] stopwords.words('english')[100:105]
```

```
['here', 'there', 'when', 'where', 'why']
```

```
[ ] def processamento(texto):
    Npunct = [char for char in texto if char not in string.punctuation] #remove as pontuações
    Npunct = ''.join(Npunct)
    sms_limpo = [word.lower() for word in Npunct.split() if word.lower() not in stopwords.words('english')] #remove stopwords
    return sms_limpo
```

```

▶ #cria uma nova coluna com a mensagem sem stopwords e pontuação (apenas os tokens)
mensagens['nova_mensagem'] = mensagens['mensagem'].apply(processamento)

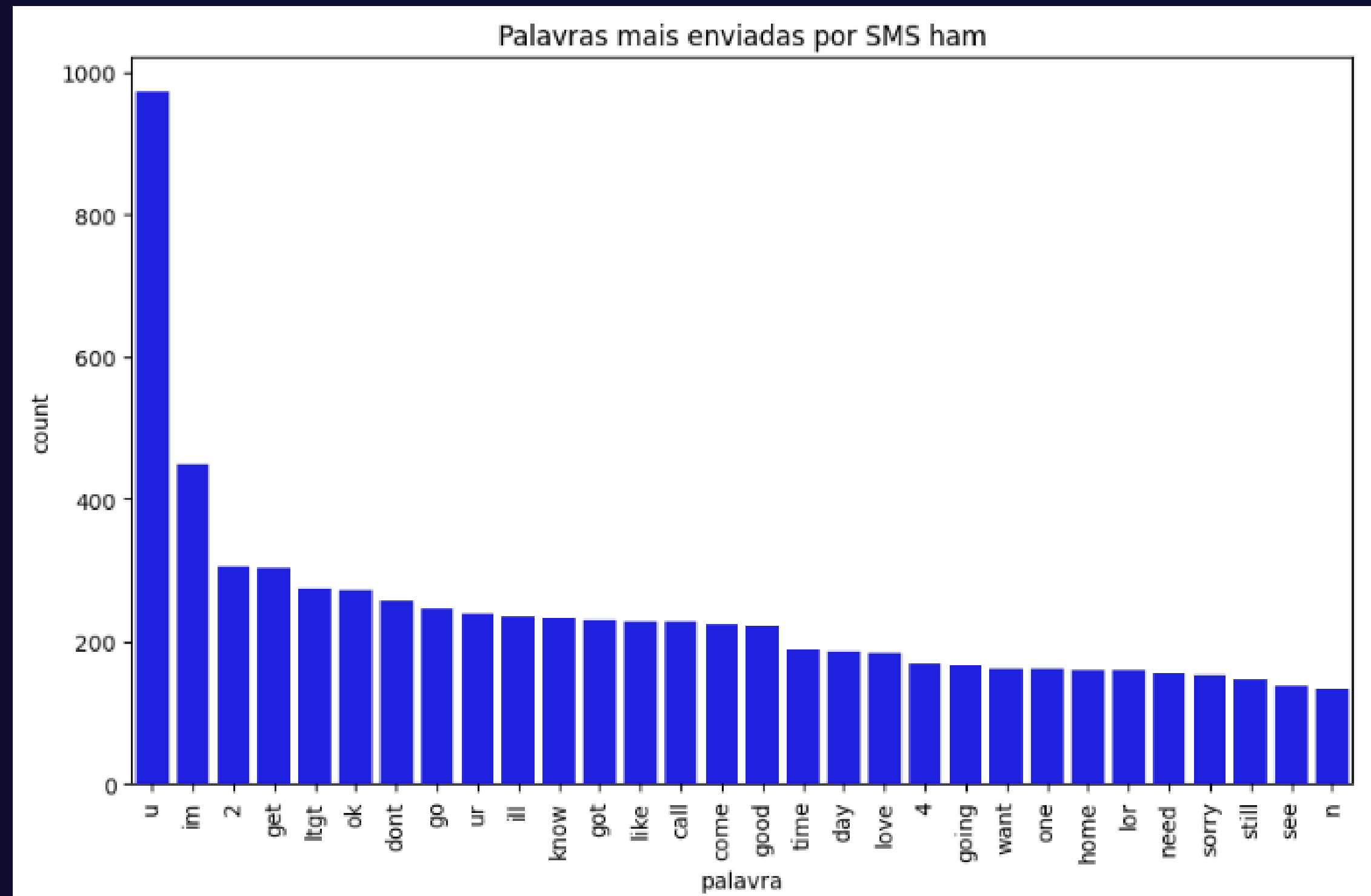
mensagens.sample(5)

```

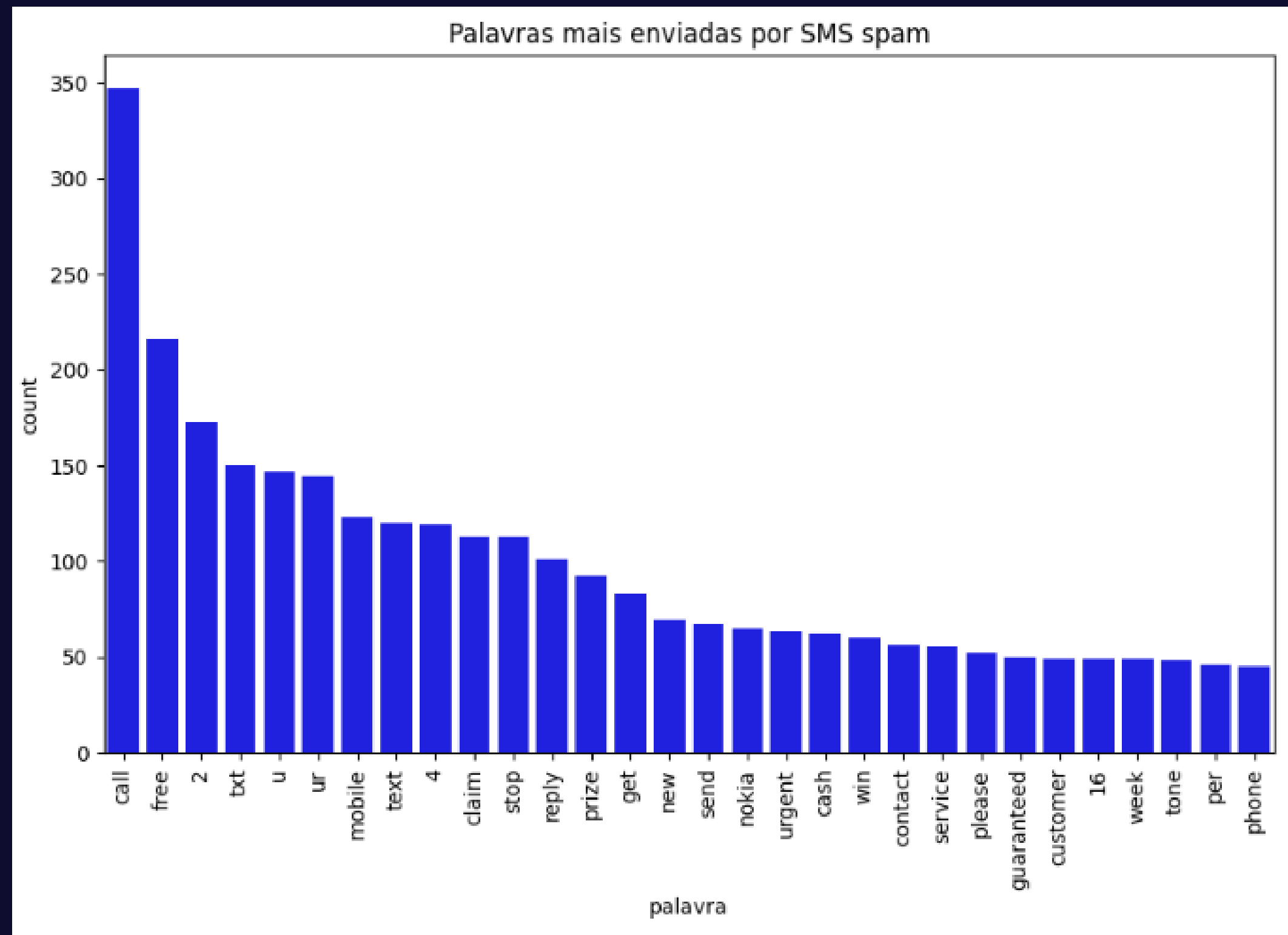
	categoria	mensagem	spam	qtd_caracteres	qtd_palavras	nova_mensagem
3523	ham	Yeah that'd pretty much be the best case scenario	0	49	10	[yeah, thatd, pretty, much, best, case, scenario]
3444	ham	wiskey Brandy Rum Gin Beer Vodka Scotch Shampa...	0	91	17	[wiskey, brandy, rum, gin, beer, vodka, scotch...
1022	ham	We still on for tonight?	0	24	6	[still, tonight]
5306	ham	Ill be at yours in about 3 mins but look out f...	0	51	13	[ill, 3, mins, look]
4303	ham	Good good, billy mates all gone. Just been jog...	0	77	18	[good, good, billy, mates, gone, jogging, enjo...



# Palavras mais enviadas por SMSs Ham



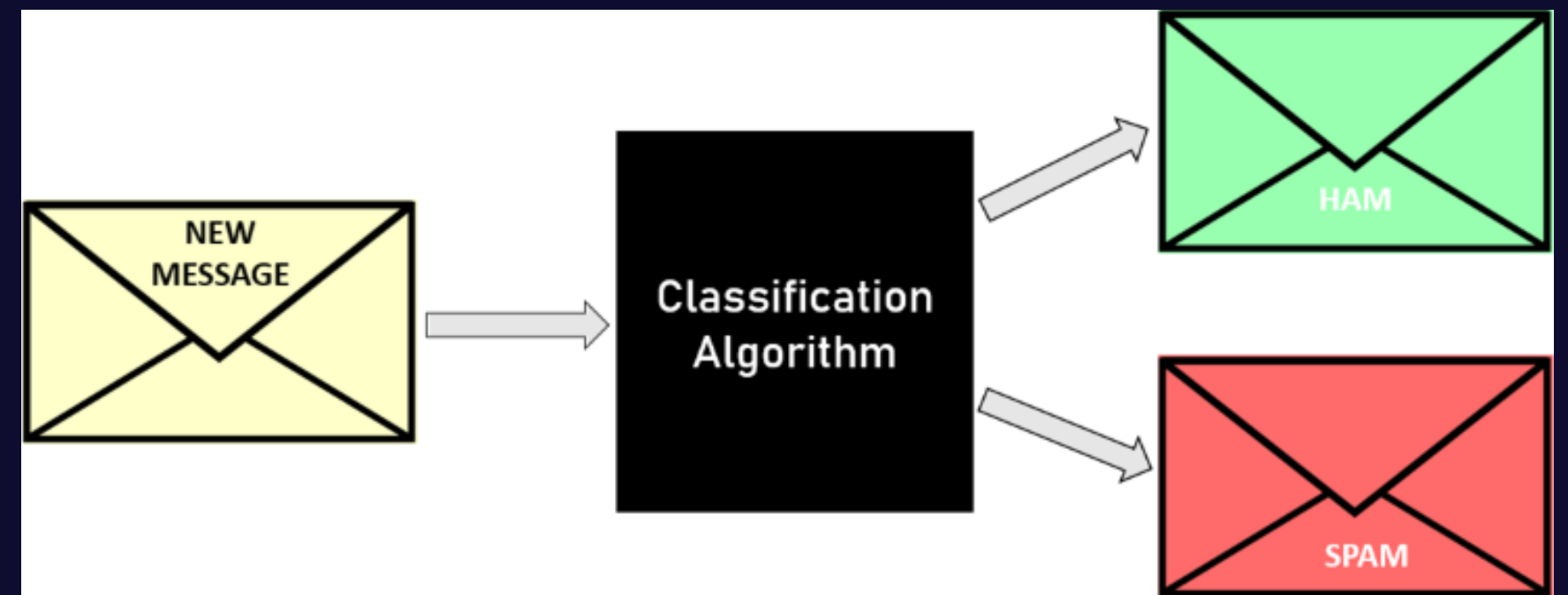
# Palavras mais enviadas por SMS Spam



# Modelo de Machine Learning

## Naive Bayes

- MultinomialNB
- Teorema de Bayes
- Independência entre eventos



# Pipeline

- Bag of words
- Tfidf
- Classificador: MultinomialNB

```
X = mensagens['mensagem'] #recursos
y = mensagens['spam'] #rótulos

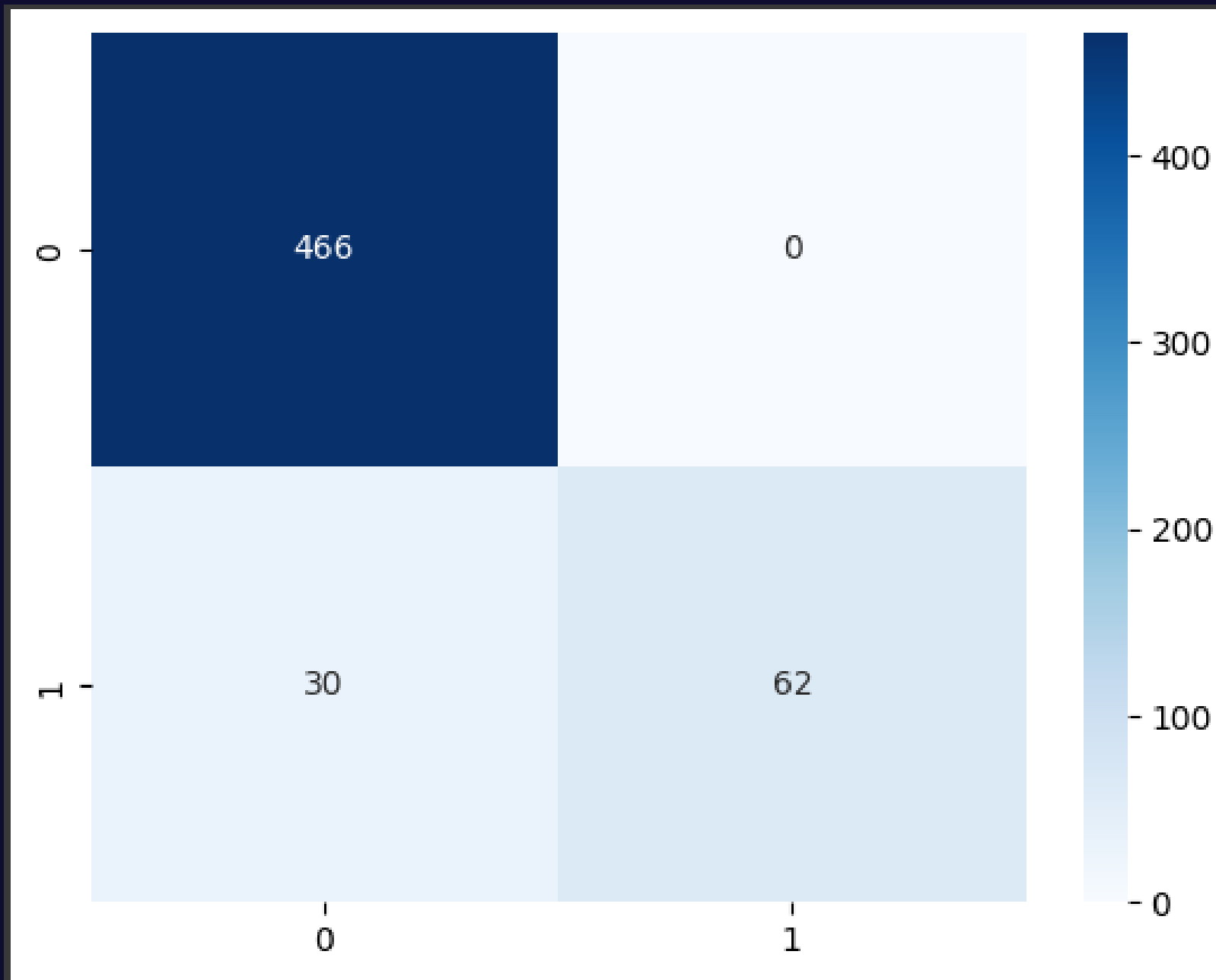
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1)

#bow - bag of words - frequência das palavras no sms
#tfidf - leva mais em consideração palavras com menor frequência - mais informativas
#classifier - escolhe o classificador multinomial do naive bayes
pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=processamento)),
    ('tfidf', TfidfTransformer()),
    ('classifier', MultinomialNB())
])
pipeline.fit(X_train, y_train) #treina o modelo
predictions = pipeline.predict(X_test) #testa o modelo - indica no conjunto de teste quem é spam
print(classification_report(y_test, predictions))
```

# Resultado do Modelo

	Precision	Recall	f1-Score	Support
Ham	0.95	1.00	0.98	477
Spam	1.00	0.72	0.83	84
Accuracy			0.96	588
Macro AVG	0.98	0.86	0.91	588
Wighted AVG	0.96	0.96	0.96	588

```
▶ sns.heatmap(confusion_matrix(y_test,predictions), annot=True, fmt = '.0f', cmap = 'Blues')  
#análise do conjunto teste  
#(0,0) - verdadeiro positivo - sms que não são spam que foram classificados corretamente  
#(0,1) - falso negativo - sms que são spam que foram classificados como não spam  
#(1,0) - falso positivo - sms que não são spam que foram classificados como spam  
#(1,1) - verdadeiro negativo - sms que são spam que foram classificados corretamente
```



## 4. Testando SMSs

```
[ ] #teste sms não spam
X_test = ["hey let's play soccer"]
pipeline.fit(X_train, y_train)
predictions = pipeline.predict(X_test)
print(predictions)
```

[0]

```
▶ #teste sms spam
X_test = ["Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005."]
pipeline.fit(X_train, y_train)
predictions = pipeline.predict(X_test)
print(predictions)
```

[1]

Obrigada!