In [1]:

```python
import nltk
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import re
import unicodedata2
import math
import string
import tokenize
import sklearn
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.tokenize import wordpunct_tokenize
from nltk.probability import FreqDist
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import RegexpTokenizer
from nltk.tokenize import sent_tokenize
from sklearn import metrics
from sklearn.metrics.pairwise import linear_kernel
from sklearn.metrics.pairwise import polynomial_kernel
from sklearn.naive_bayes import MultinomialNB
from string import digits
from xml.dom import minidom
from unidecode import unidecode
from nltk.stem.snowball import SnowballStemmer
from string import punctuation
from nltk.classify.scikitlearn import SklearnClassifier
from sklearn.naive_bayes import MultinomialNB,BernoulliNB
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.svm import SVC

stop_words = stopwords.words('spanish')
```

```
newStopWords = ['dr','dra','etc','bn','ud','u','ag','si','no','rt','q','m','bb','tan','aun','cr','tal','segun','w','lab
stop_words.extend(newStopWords)
data=pd.read_csv("ScoreV0_int.csv")
sbEsp = SnowballStemmer('spanish')
data.head()
```

Out[1]:

	Tweet	Sentiment
0	Comparto nuestro aliado estrategico para la Es...	5
1	@FREDYGUERRAHERR @consigliererojo @ClaudiaLope...	3
2	@FREDYGUERRAHERR @adelve3 @ClaudiaLopez @Trans...	1
3	@luciabastidasu @TransMilenio Pero esto esta p...	2
4	Lo del bus de TM y la marihuana no esta nada b...	2

In [2]:

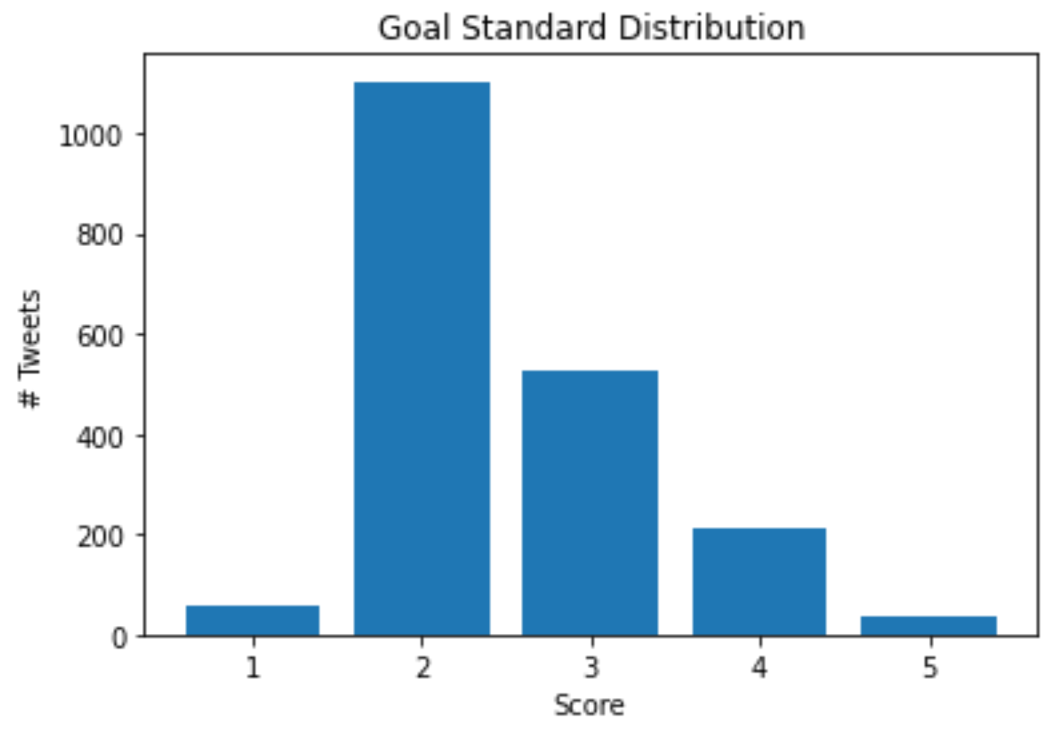
```
data.Sentiment.value_counts()
```

Out[2]:

```
2    1103
3     528
4     210
1      57
5      38
Name: Sentiment, dtype: int64
```

In [3]:

```
Sentiment_count=data.groupby('Sentiment').count()
fig, ax = plt.subplots()
plt.bar(Sentiment_count.index.values, Sentiment_count['Tweet'])
ax.set(xlabel='Score', ylabel='# Tweets',
       title='Goal Standard Distribution')
fig.savefig("distGoldStandard.png")
plt.show()
```



In [4]:

```
def strip_links(text):
    text = text.lower()
    link_regex    = re.compile('((https?):((//)|(\\\\))+([\w\d:#@%/;$()~_?\+-=\\\.&](#!)?)*)', re.DOTALL)
    links         = re.findall(link_regex, text)
    for link in links:
        text = text.replace(link[0], ', ')
    return text

# Eliminación de Hashtags y menciones
def strip_all_entities(text):
    entity_prefixes = ['@','#']
```

```python
    for separator in  string.punctuation:
        if separator not in entity_prefixes :
            text = text.replace(separator,' ')
    words = []
    for word in text.split():
        word = word.strip()
        if word:
            if word[0] not in entity_prefixes:
                words.append(word)
    return ' '.join(words)

# Eliminación de puntuacion, numeros y conversión del texto a minúsculas
def remove_punctuations(text):
    for punctuation in string.punctuation:
        text = text.replace(punctuation, '')
    for digits in string.digits:
        text = text.replace(digits,'')
    text = text.lower()
    return text

def remove_punct(strin):
    strin = strin.translate(str.maketrans('','',string.punctuation));
    strin = strin.translate(str.maketrans('','',string.digits));
    return strin;

#Normalizar: eliminar diéresis, acentos, y otros caracteres similares.
def normunicode_data(strin):
    #print(strin)
    return unicodedata2.normalize('NFKD', strin).encode('ASCII', 'ignore').decode("utf-8").lower()

def proc_str(strin):
    return remove_punct(normunicode_data(strin));

def tok_cln(text):
    return set(nltk.wordpunct_tokenize(text)).difference(stop_words) #(este es el original)

def preprocessing(text):
    text= text.apply(strip_links)
    text= text.apply(strip_all_entities)
    text= text.apply(remove_punctuations)
    text = text.apply(normunicode_data)
    return text
```

```
data.Tweet=preprocessing(data["Tweet"])
data.head()
```

Out[4]:

	Tweet	Sentiment
0	comparto nuestro aliado estrategico para la es...	5
1	si se permite esto se acaba la seguridad en bo...	3
2	que hpta desorden de ciudad ya aqui todos hace...	1
3	pero esto esta prohibido normas de convivencia...	2
4	lo del bus de tm y la marihuana no esta nada b...	2

In [5]:

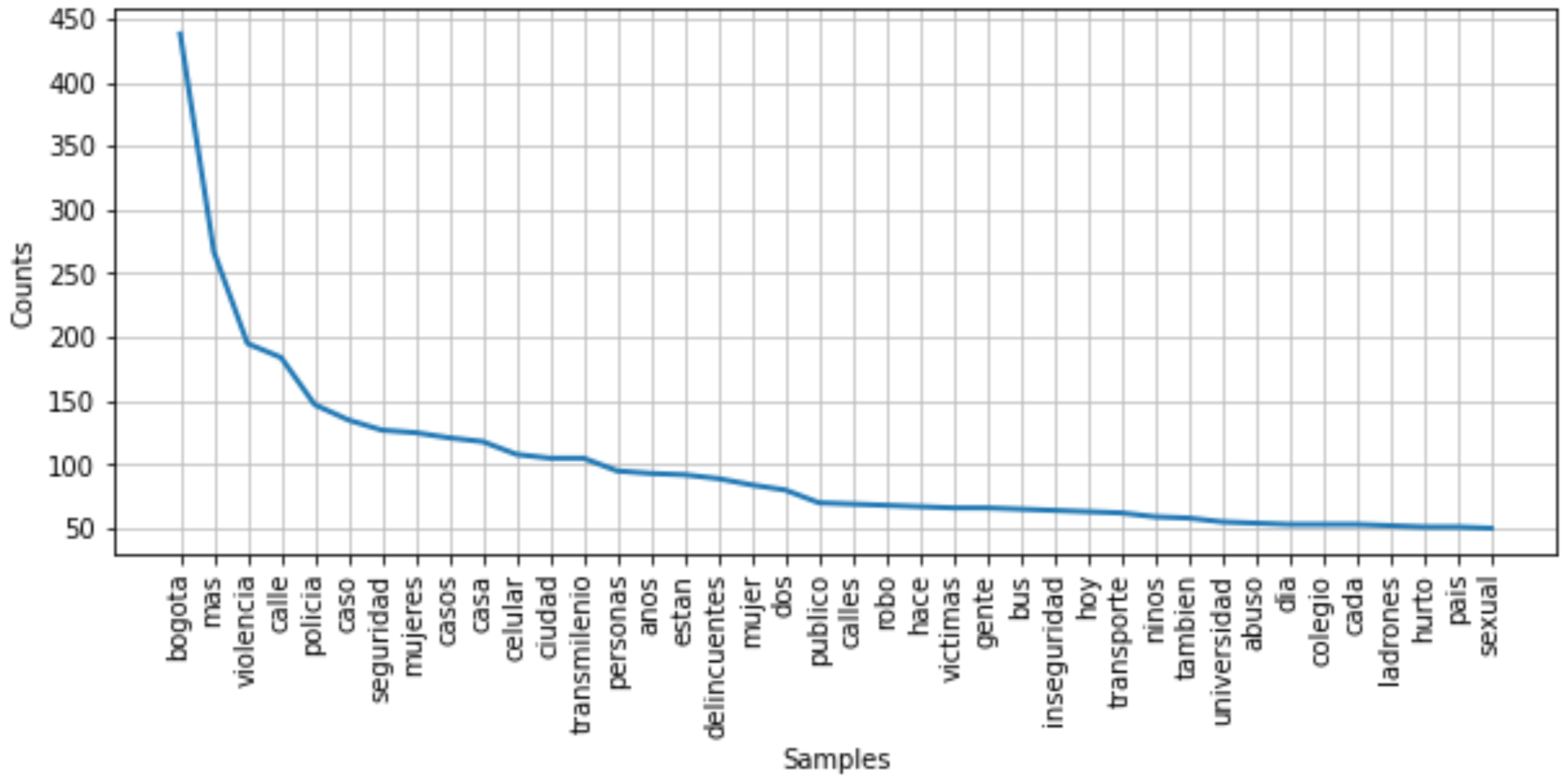
```
def stemm_data(strin):
    stemmer = SnowballStemmer("spanish");
    return stemmer.stem(strin)
```

In [6]:

```
def proc_string(strin,setData):
    resp = set([]);
    for data in tok_cln(proc_str(strin)):
        #tm  = stemm_data(data)
        tm = data
        resp.add(tm)
        if tm in setData:
            setData[tm].add(data)
        else:
            setData[tm] = set([data])
    return ', '.join(resp);

def freq_dist_tok(strin):
    token_clear = strin.apply(tok_cln)
    out = [item for t in token_clear for item in t]
    fig = plt.figure(figsize = (10,4))
    plt.gcf().subplots_adjust(bottom=0.15)
    fdist = FreqDist(out)
    fdist.plot(40,cumulative=False)
    return fig.savefig('freqDist.png', bbox_inches = "tight")

freq_dist_tok(data.Tweet)
```



In [7]:

```
def df2tdm(df,titleColumn,setData):
    listData = [];
    for idx in data.index:
        listData.append(proc_string(data[titleColumn][idx],setData));
    return listData;

def getDictionary_BOW(dfpp):
    setData = {}
    stinProc = df2tdm(dfpp,'Tweet',setData);
    cv =  CountVectorizer();
    cv_fit = cv.fit_transform(stinProc);
    cvCount =  CountVectorizer(cv.vocabulary_);
    # computes the vectorial representation of the CIE10
    cv_fitCount = cvCount.fit_transform(stinProc);
    features = cvCount.get_feature_names();
    return cv,cv_fitCount,features,setData

def getDictionary_TFIDF(dfpp):
    setData = {}
    stinProc = df2tdm(dfpp,'Tweet',setData);
    cv =  TfidfVectorizer();
    cv_fit = cv.fit_transform(stinProc);
    cvCount =  TfidfVectorizer(cv.vocabulary_);
    # computes the vectorial representation of the CIE10
    cv_fitCount = cvCount.fit_transform(stinProc);
    features = cvCount.get_feature_names();
    return cv,cv_fitCount,features,setData

dfpp = data['Sentiment']
BOW_cvQuery_fit,BOW_cv_fitCount,BOW_features,BOW_origTerms = getDictionary_BOW(dfpp)
TFIDF_cvQuery_fit,TFIDF_cv_fitCount,TFIDF_features,TFIDF_origTerms = getDictionary_TFIDF(dfpp)
```

In [8]:

```
from sklearn.model_selection import LeaveOneOut
from sklearn import model_selection
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import make_scorer, accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import KFold

#kf = KFold(n_splits=30) # Define the split - into folds
kf = KFold(n_splits=30, random_state=None, shuffle=False)
def kfold_metrics(model,X,y):
    accuracy_metr=[]
    precision_metr=[]
    f1_metr=[]
    recall_metr=[]
    for train_index, test_index in kf.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]
        clf = model.fit(X_train, y_train)
        predicted= clf.predict(X_test)
        acc=metrics.accuracy_score(y_test, predicted)
        prec=metrics.precision_score(y_test, predicted,average='weighted', zero_division=1)
        f1sc=metrics.f1_score(y_test, predicted, average='weighted')
        rec=metrics.recall_score(y_test, predicted,average='weighted',zero_division=1)
        accuracy_metr.append(acc)
        precision_metr.append(prec)
        f1_metr.append(f1sc)
        recall_metr.append(rec)
    return accuracy_metr,precision_metr,f1_metr,recall_metr
```

In [9]:

```
acc_BOW_MNB,prec_BOW_MNB,f1_BOW_MNB,recall_BOW_MNB=kfold_metrics(MultinomialNB(),BOW_cv_fitCount,data['Sentiment']);
acc_TFIDF_MNB,prec_TFIDF_MNB,f1_TFIDF_MNB,recall_TFIDF_MNB=kfold_metrics(MultinomialNB(),TFIDF_cv_fitCount,data['Sentime
```

In [10]:

```
acc_BOW_LR,prec_BOW_LR,f1_BOW_LR,recall_BOW_LR=kfold_metrics(LogisticRegression(solver = 'liblinear', multi_class = 'ovr
acc_TFIDF_LR,prec_TFIDF_LR,f1_TFIDF_LR,recall_TFIDF_LR=kfold_metrics(LogisticRegression(solver = 'liblinear', multi_clas
```

In [11]:

```
acc_BOW_BNB,prec_BOW_BNB,f1_BOW_BNB,recall_BOW_BNB=kfold_metrics(BernoulliNB(),BOW_cv_fitCount,data['Sentiment']);
acc_TFIDF_BNB,prec_TFIDF_BNB,f1_TFIDF_BNB,recall_TFIDF_BNB=kfold_metrics(BernoulliNB(),TFIDF_cv_fitCount,data['Sentiment
```

In [12]:

```
acc_BOW_SGD,prec_BOW_SGD,f1_BOW_SGD,recall_BOW_SGD=kfold_metrics(SGDClassifier(loss="log", max_iter=7),BOW_cv_fitCount,d
acc_TFIDF_SGD,prec_TFIDF_SGD,f1_TFIDF_SGD,recall_TFIDF_SGD=kfold_metrics(SGDClassifier(loss="log", max_iter=7),TFIDF_cv_
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/sklearn/linear_model/_stocha
stic_gradient.py:554: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
```

In [82]:

```
import plotly.express as px
import plotly.graph_objects as go
```

In [83]:

```
import os

if not os.path.exists("images"):
    os.mkdir("images")
```

In [84]:

```
def plots_metrics(metric_BOW_SGD, metric_TFIDF_SGD, metric_BOW_BNB, metric_TFIDF_BNB, metric_BOW_LR, metric_TFIDF_LR,
                  metric_BOW_MNB, metric_TFIDF_MNB, measure):
    fig = go.Figure()
    fig.add_trace(go.Violin(y=metric_BOW_SGD, scalegroup='BOW', name='BOW SGD',
                         side='negative',pointpos=-1.4,line_color='green', x0='SGD Classifier'))

    fig.add_trace(go.Violin(y=metric_TFIDF_SGD, scalegroup='TFIDF', name='TFIDF SGD',
```

```python
fig.add_trace(go.Violin(y=metric_TFIDF_SGD, scalegroup='TFIDF', name='TFIDF SGD',
                    side='positive', pointpos=1.4,line_color='lightseagreen', x0='SGD Classifier'))

fig.add_trace(go.Violin(y=metric_BOW_BNB, scalegroup='BOW', name='BOW BNB',
                    pointpos=-1.4,side='negative',line_color='red', x0='Bernoulli NB'))

fig.add_trace(go.Violin(y=metric_TFIDF_BNB, scalegroup='TFIDF', name='TFIDF BNB',
                    pointpos=1.4,side='positive',line_color='darkmagenta', x0='Bernoulli NB'))

fig.add_trace(go.Violin(y=metric_BOW_LR, scalegroup='BOW', name='BOW LR',
                    pointpos=-1.4,side='negative',line_color='blue', x0='Logistic Regresion'))

fig.add_trace(go.Violin(y=metric_TFIDF_LR, scalegroup='TFIDF', name='TFIDF LR',
                    pointpos=1.4,side='positive',line_color='darkblue', x0='Logistic Regresion'))

fig.add_trace(go.Violin(y=metric_BOW_MNB, scalegroup='BOW', name='BOW MNB',
                    pointpos=-1.4,side='negative',line_color='black', x0='Multinomial NB'))

fig.add_trace(go.Violin(y=metric_TFIDF_MNB, name='TFIDF MNB',
                    pointpos=1.4,side='positive',line_color='gray', x0='Multinomial NB'))

fig.update_traces(meanline_visible=True,
              points='all', # show all points
              jitter=0.05,  # add some jitter on points for better visibility
              scalemode='count')

fig.update_layout(
title_text=measure,
violingap=0, violingroupgap=0.6, violinmode='overlay')
fig.write_image("images/"+measure+".png")
fig.show()
```

In [85]:

```
plots_metrics(acc_BOW_SGD, acc_TFIDF_SGD, acc_BOW_BNB, acc_TFIDF_BNB, acc_BOW_LR, acc_TFIDF_LR,
                 acc_BOW_MNB, acc_TFIDF_MNB, "Accuracy")
```

In [86]:

```
plots_metrics(prec_BOW_SGD, prec_TFIDF_SGD, prec_BOW_BNB, prec_TFIDF_BNB, prec_BOW_LR, prec_TFIDF_LR,
                 prec_BOW_MNB, prec_TFIDF_MNB, "Precision")
```

In [87]:

```
plots_metrics(f1_BOW_SGD, f1_TFIDF_SGD, f1_BOW_BNB, f1_TFIDF_BNB, f1_BOW_LR, f1_TFIDF_LR,
                 f1_BOW_MNB, f1_TFIDF_MNB, "F1")
```

In [88]:

```
plots_metrics(recall_BOW_SGD, recall_TFIDF_SGD, recall_BOW_BNB, recall_TFIDF_BNB, recall_BOW_LR, recall_TFIDF_LR,
                 recall_BOW_MNB, recall_TFIDF_MNB, "Recall")
```

In [ ]:

```

```

```
In [1]:
```

```
Out[1]:
```

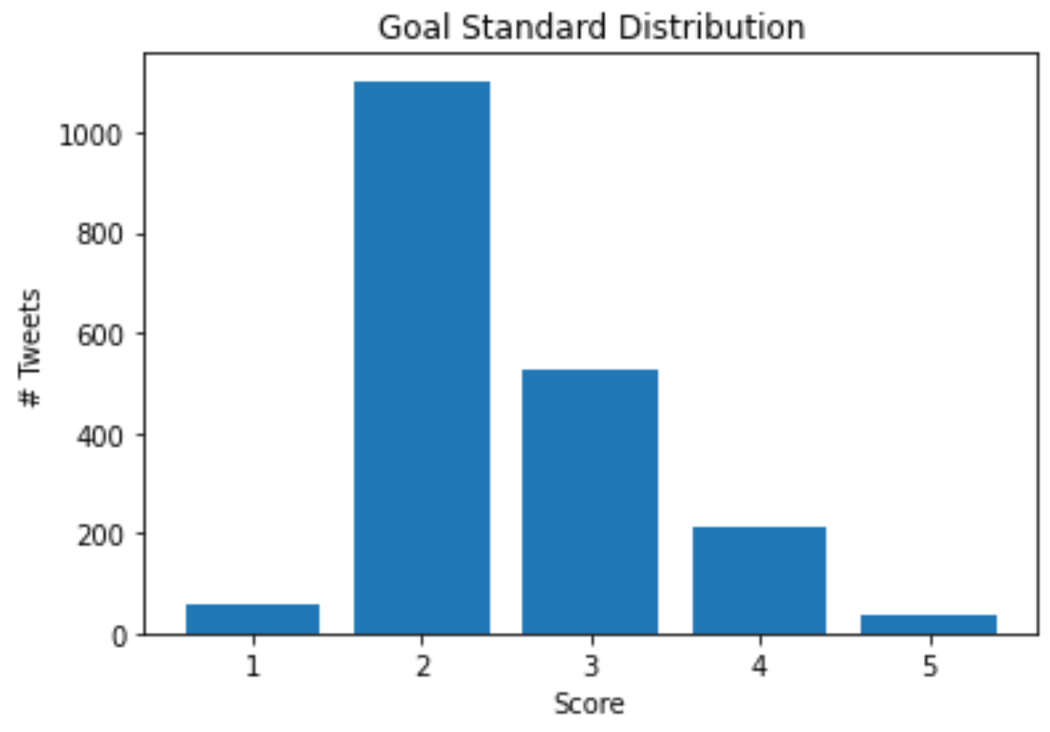
	Tweet	Sentiment
0	Comparto nuestro aliado estrategico para la Es...	5
1	@FREDYGUERRAHERR @consiglierojo @ClaudiaLope...	3
2	@FREDYGUERRAHERR @adelve3 @ClaudiaLopez @Trans...	1
3	@luciabastidasu @TransMilenio Pero esto esta p...	2
4	Lo del bus de TM y la marihuana no esta nada b...	2

```
In [2]:
```

```
Out[2]:

2    1103
3     528
4     210
1      57
5      38
Name: Sentiment, dtype: int64
```

```
In [3]:
```



```
In [4]:
```

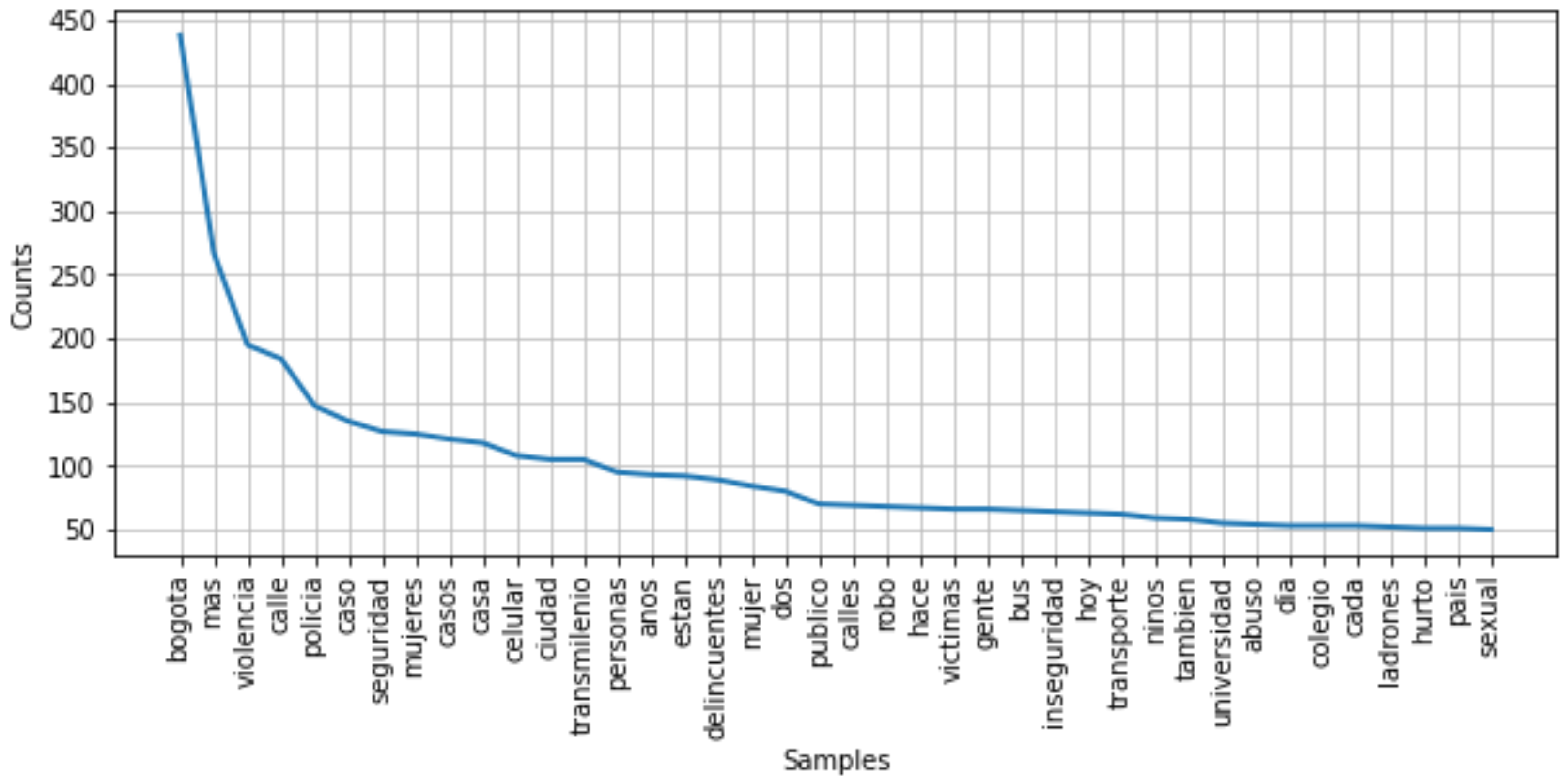
```
Out[4]:
```

	Tweet	Sentiment
0	comparto nuestro aliado estrategico para la es...	5
1	si se permite esto se acaba la seguridad en bo...	3
2	que hpta desorden de ciudad ya aqui todos hace...	1
3	pero esto esta prohibido normas de convivencia...	2
4	lo del bus de tm y la marihuana no esta nada b...	2

```
In [5]:
```

In [6]:

```
```



In [7]:

```
```

In [8]:

```
```

In [9]:

```
```

In [10]:

```
```

In [11]:

```
```

In [12]:

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/sklearn/linear_model/_stocha
stic_gradient.py:554: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
```

In [82]:

In [83]:

In [84]:

```
In [85]:
```

In [86]:

```
```

```
In [87]:
```

In [88]:

```
```

In [ ]:

```
```