Start coding or ge nerate with AI.

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

data = pd.read\_csv('/content/News.csv',index\_col=0) data.head()

 title text subject date class

1. Donald Trump Sends Out Embarrassing New Year’...
2. Drunk Bragging Trump Staffer Started Russian ...
3. Sheriff David Clarke Becomes An

Internet Joke...

1. Trump Is So Obsessed He Even Has

Obama’s Name...

1. Pope Francis Just Called Out Donald Trump Dur...

Donald Trump just couldn t wish

all Americans ...

House Intelligence Committee Chairman Devin Nu...

On Friday, it was revealed that

former Milwauk...

On Christmas day, Donald Trump

announced that ...

Pope Francis used his annual Christmas Day mes...

News December 31,

2017

0

News December 31,

0

2017

News December 30,

0

2017

News December 29,

0

2017

News December 25,

0

2017

Next steps:

Generate code with data

 View recommended plots

data.shape

(44919, 5)

data = data.drop(["title", "subject","date"], axis = 1)

data.isnull().sum()

text 0

class 0

dtype: int64

# Shuffling

data = data.sample(frac=1)

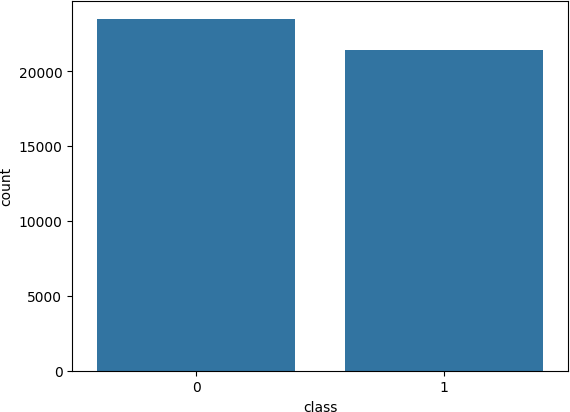
data.reset\_index(inplace=True)

data.drop(["index"], axis=1, inplace=True)

sns.countplot(data=data,

x='class',

order=data['class'].value\_counts().index)

<Axes: xlabel='class', ylabel='count'>

from tqdm import tqdm import re

import nltk

nltk.download('punkt')

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.stem.porter import PorterStemmer from wordcloud import WordCloud

[nltk\_data] Downloading package punkt to /root/nltk\_data... [nltk\_data] Unzipping tokenizers/punkt.zip.

[nltk\_data] Downloading package stopwords to /root/nltk\_data... [nltk\_data] Unzipping corpora/stopwords.zip.

def preprocess\_text(text\_data): preprocessed\_text = []

for sentence in tqdm(text\_data):

sentence = re.sub(r'[^\w\s]', '', sentence)

preprocessed\_text.append(' '.join(token.lower()

for token in str(sentence).split()

if token not in stopwords.words('english')))

return preprocessed\_text

preprocessed\_review = preprocess\_text(data['text'].values) data['text'] = preprocessed\_review

100%|██████████| 44919/44919 [35:50<00:00, 20.89it/s]

# Real

consolidated = ' '.join(

word for word in data['text'][data['class'] == 1].astype(str)) wordCloud = WordCloud(width=1600,

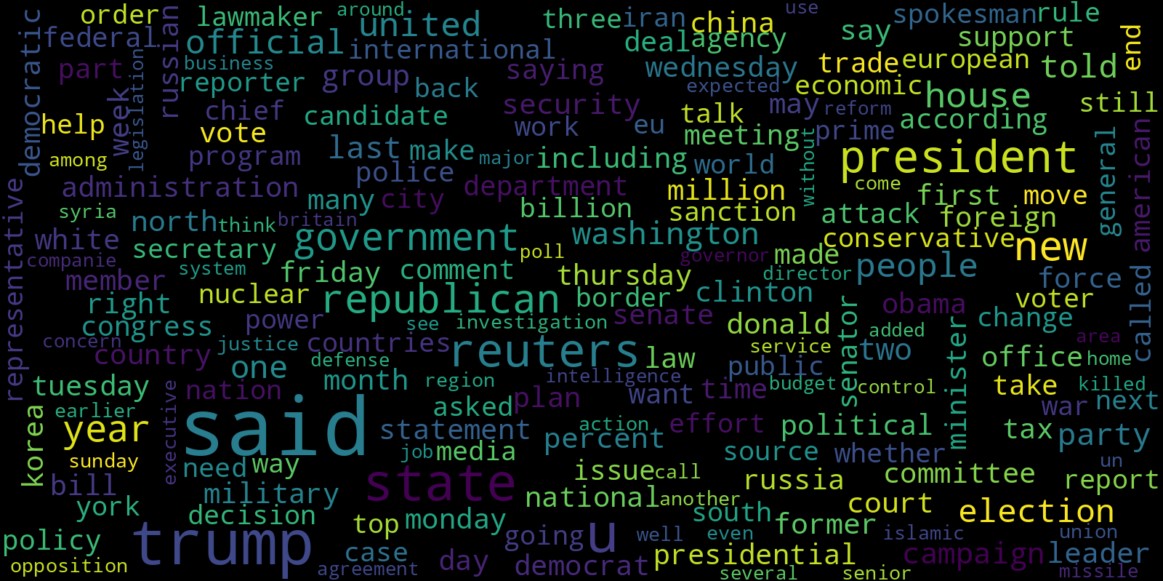
height=800,

random\_state=21, max\_font\_size=110,

collocations=False) plt.figure(figsize=(15, 10))

plt.imshow(wordCloud.generate(consolidated), interpolation='bilinear') plt.axis('off')

plt.show()



from sklearn.feature\_extraction.text import CountVectorizer

def get\_top\_n\_words(corpus, n=None):

vec = CountVectorizer().fit(corpus) bag\_of\_words = vec.transform(corpus) sum\_words = bag\_of\_words.sum(axis=0)

words\_freq = [(word, sum\_words[0, idx])

for word, idx in vec.vocabulary\_.items()] words\_freq = sorted(words\_freq, key=lambda x: x[1],

reverse=True) return words\_freq[:n]

common\_words = get\_top\_n\_words(data['text'], 20)

df1 = pd.DataFrame(common\_words, columns=['Review', 'count'])

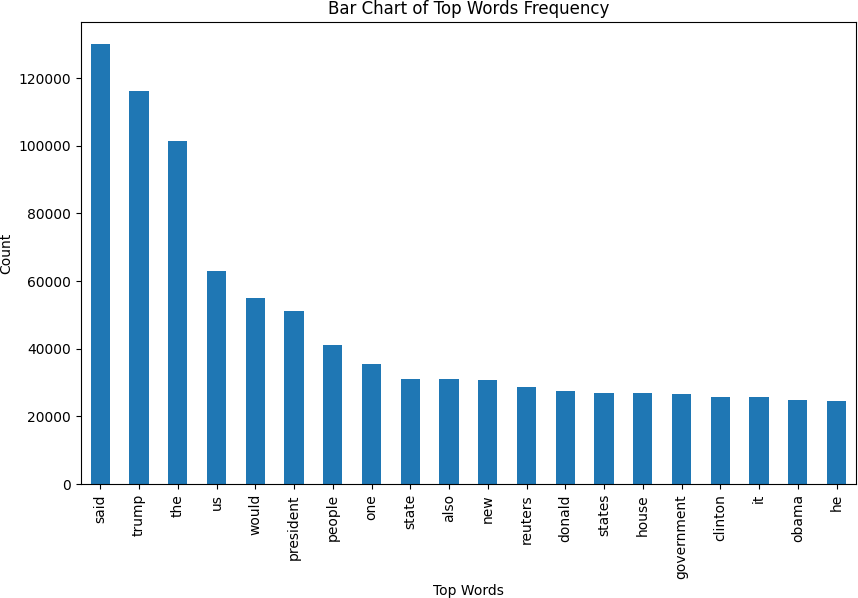
df1.groupby('Review').sum()['count'].sort\_values(ascending=False).plot( kind='bar',

figsize=(10, 6),

xlabel="Top Words", ylabel="Count",

title="Bar Chart of Top Words Frequency"

)

<Axes: title={'center': 'Bar Chart of Top Words Frequency'}, xlabel='Top Words', ylabel='Count'>

from sklearn.model\_selection import train\_test\_split from sklearn.metrics import accuracy\_score

from sklearn.linear\_model import LogisticRegression

x\_train, x\_test, y\_train, y\_test = train\_test\_split(data['text'],

data['class'], test\_size=0.25)

from sklearn.feature\_extraction.text import TfidfVectorizer vectorization = TfidfVectorizer()

x\_train = vectorization.fit\_transform(x\_train) x\_test = vectorization.transform(x\_test)

from sklearn.linear\_model import LogisticRegression model = LogisticRegression()

model.fit(x\_train, y\_train)

# testing the model

print(accuracy\_score(y\_train, model.predict(x\_train))) print(accuracy\_score(y\_test, model.predict(x\_test)))

0.9936180949271276

0.9896705253784506

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier() model.fit(x\_train, y\_train)

# testing the model

print(accuracy\_score(y\_train, model.predict(x\_train))) print(accuracy\_score(y\_test, model.predict(x\_test)))

0.9999703167205913

0.9963490650044523

# Confusion matrix of Results from Decision Tree classification from sklearn import metrics

cm = metrics.confusion\_matrix(y\_test, model.predict(x\_test))

cm\_display = metrics.ConfusionMatrixDisplay(confusion\_matrix=cm,

display\_labels=[False, True])

cm\_display.plot() plt.show()

