INTRODUCTION

**Introduction and motivation**

Nowadays on the Internet there are a lot of sources that generate immense amounts of daily news. In addition, the demand for information by users has been growing continuously, so it is crucial that the news is classified to allow users to access the information of interest quickly and effectively.

This way, the machine learning model for automated news classification could be used to identify topics of untracked news and/or make individual suggestions based on the user’s prior interests. Thus, our aim is to build models that take as input news headline and short description and output news category

Objective

**Data and features**

**1) DATA**

DATASET are contains almost 125,000 news from the past 5 years obtained from HuffPost .

News in these dataset belong to 31 different topics (labels). Each news record consists of several attributes from which we are using only ‘Category’, ‘Headline’ and ‘Short description’ in our analysis. In addition, we combine data attributes ‘Headline’ and ‘Short description’ into the single attribute ‘Text’ as the input data for classification.

The data preprocessing consisted in combining some raw data categories that are very close (for example, "Arts" and "Arts and Culture", "Education" and "College" etc).

**2)Features**

First, using the preprocessed news descriptions we created the dictionary of words. The total number of unique words is around 40,000. Then, we extracted the following word features for classification task:

**Word binary and word count features**:

For binary and count features we used first

5,000 most common words to define the

dictionary and then, encoded the news

descriptions as vectors - either as vectors of 0

and 1 for binary features or of word counts in

the description.

* + - **Word level TF-IDF scores:**
      * For TF-IDF method we decided to extend the dictionary to the first 10,250 most frequent words. Moreover, we combined the text from all the news belonging to that category and treated it as the one document. Thus, our corpus of documents consisted of 25 documents (one for each news category) from which we learn TF-IDF representation and then, we apply it both to train and dev set samples.
    - **Word embeddings:** 
      * Word embeddings are a family of NLP techniques aiming at mapping the semantic meaning into a geometric space [3]. To learn the word embeddings from the data we applied an Embedding layer of Keras [4]. Also, we considered only 30,000 most common words in the dataset and we truncated each example to a maximum length of 50 words.

**Context :**

This dataset contains around 150k news headlines from the year 2016 to 2021. The model trained on this dataset could be used to identify tags for untracked news articles or to identify the type of language used in different news articles.

**Content :**

Each news headline have been a corresponding category. Categories and corresponding article around counts are as follows:

1. **Politics :30000**
2. **Welness : 17200**
3. **Entertainment : 16300**
4. **Travels : 98000**
5. **Style and Beauti : 9600**

**…etc**

BAKGROUND

* **Algorithms:**
  + In the first part of our work we experimented with traditional machine learning techniques: Naive Bayes, multinomial logistic regression, kernel SVM and Random Forest.
  + Naive Bayes With binary features we applied multivariate Bernoulli model and with count features - multinomial event model. For each example, we classify as yˆ = arg maxy P(y) Qn i=1 P(xi |y), where we use MAP estimation for P(y) and P(xi |y) while also applying Laplace smoothing .
  + Multinomial Logistic Regression We use the cross-entropy loss with L2 regularization . The regularized cost function is J(θ) = − Pm i=1 PK k=1 y (i) k log ˆy (i) k + λ Pn l=1 ||θl ||2 2
  + Kernel SVM We use a multi-class SVM with a "one-vs-rest" approach and an RBF kernel K(x, z) = exp −γ||x − z||2 . Optimal parameter C and kernel parameter γ were optimized by 3-fold cross-validated grid-search over a parameter grid.
  + Random Forest We used the Gini measure G(Xm) = P k pmk(1 − pmk), where pmk is the proportion of class k samples in node m . We regularized each tree in terms of maximum depth.
  + In the second part of our work, we focused on building the neural network models: with word embedding featur es provided by the Embedding layer of
  + Keras we trained several neural network models with one or two convolutional layers (CNN) and/or recurrent (LSTM) layer (RNN ).
  + CNN This a class of deep, feed-forward artificial neural networks that excel at learning the spatial structure in the input data by learning the set of filters applied to the data.
  + RNN This is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit temporal dynamic behavior for a time sequence

**Modules uses:**

mpl\_toolkits

sklearn

nltk

matplotlib

numpy

os

pandas

**HARDWARE AND SOFTWARE REQUIREMENTS**

* **HARDWARE :**

**LAPTOP OR DESKTOP**

* **SOFTWARE REQUIREMENTS :**

SYSTEMESOFTWARE(ANY ONE) :

WINDOW 10

LINUX

MAC

APPLYCATON SOFTWARE :

PYTHON (DATA SCIENCE AND ML)

VS CODE OR PYCHARM

JUPYTER NOTEBOOK

CODEING AND OUTPUT

**Import module** In [1]:

from mpl\_toolkits.mplot3d import Axes3D

from sklearn.preprocessing import StandardScaler

from sklearn.feature\_extraction.text import CountVectorizer , TfidfVectorizer

from nltk.tokenize import casual\_tokenize, word\_tokenize

from sklearn.model\_selection import train\_test\_split

from sklearn.utils import class\_weight

from sklearn.metrics import confusion\_matrix

from sklearn import metrics

import matplotlib.pyplot as plt # plotting

import numpy as np # linear algebra

import os # accessing directory structure

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

# Load data[¶](#Load-data)

In [2]:

def load\_data(file\_path) :

df = pd.read\_json(file\_path, lines = True)

df['category'] = pd.Categorical(df['category'])

return df[["headline", "category"]]

dataframe = load\_data('data.json')

dataframe[:]

Out[2]:

|  | **headline** | **category** |
| --- | --- | --- |
| **0** | There Were 2 Mass Shootings In Texas Last Week... | CRIME |
| **1** | Will Smith Joins Diplo And Nicky Jam For The 2... | ENTERTAINMENT |
| **2** | Hugh Grant Marries For The First Time At Age 57 | ENTERTAINMENT |
| **3** | Jim Carrey Blasts 'Castrato' Adam Schiff And D... | ENTERTAINMENT |
| **4** | Julianna Margulies Uses Donald Trump Poop Bags... | ENTERTAINMENT |
| **...** | ... | ... |
| **200848** | RIM CEO Thorsten Heins' 'Significant' Plans Fo... | TECH |
| **200849** | Maria Sharapova Stunned By Victoria Azarenka I... | SPORTS |
| **200850** | Giants Over Patriots, Jets Over Colts Among M... | SPORTS |
| **200851** | Aldon Smith Arrested: 49ers Linebacker Busted ... | SPORTS |
| **200852** | Dwight Howard Rips Teammates After Magic Loss ... | SPORTS |

200853 rows × 2 columns

In [3]:

dataframe.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 200853 entries, 0 to 200852

Data columns (total 2 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 headline 200853 non-null object

1 category 200853 non-null category

dtypes: category(1), object(1)

memory usage: 1.7+ MB

In [4]:

dataframe.isnull().sum()

headline 0 Out[4]:

category 0

dtype: int64

In [5]:

print("We have a total of {} categories :".format(dataframe['category'].nunique()))

dataframe['category'].value\_counts()

We have a total of 41 categories :

Out[5]:

POLITICS 32739

WELLNESS 17827

ENTERTAINMENT 16058

TRAVEL 9887

STYLE & BEAUTY 9649

PARENTING 8677

HEALTHY LIVING 6694

QUEER VOICES 6314

FOOD & DRINK 6226

BUSINESS 5937

COMEDY 5175

SPORTS 4884

BLACK VOICES 4528

HOME & LIVING 4195

PARENTS 3955

THE WORLDPOST 3664

WEDDINGS 3651

WOMEN 3490

IMPACT 3459

DIVORCE 3426

CRIME 3405

MEDIA 2815

WEIRD NEWS 2670

GREEN 2622

WORLDPOST 2579

RELIGION 2556

STYLE 2254

SCIENCE 2178

WORLD NEWS 2177

TASTE 2096

TECH 2082

MONEY 1707

ARTS 1509

FIFTY 1401

GOOD NEWS 1398

ARTS & CULTURE 1339

ENVIRONMENT 1323

COLLEGE 1144

LATINO VOICES 1129

CULTURE & ARTS 1030

EDUCATION 1004

Name: category, dtype: int64

# Clasification data[¶](#Clasification-data)

#### explore\_data[¶](" \l "explore_data) In [6]:

def explore\_data(df, number\_of\_bins=30, text\_only=False):

stats = get\_key\_statistics\_about\_data(df)

print("Number of samples:", stats["n\_samples"])

print("Number of classes:", stats["n\_classes"])

print("Number of empty headlines:", stats["n\_empty\_headlines"])

print("Number of missing headlines:", stats["n\_missing\_headlines"])

print("Number of missing classes:", stats["n\_missing\_classes"])

print("Samples per class: Min:", stats["min\_n\_samples\_per\_class"], "Median:", stats["median\_n\_samples\_per\_class"],

"Max:", stats["max\_n\_samples\_per\_class"])

print("Words per sample: Min:", stats["min\_n\_words\_per\_sample"], "Median:", stats["median\_n\_words\_per\_sample"],

"Max:", stats["max\_n\_words\_per\_sample"])

print("Samples / Median number of words per sample:", stats["n\_samples"] / stats["median\_n\_words\_per\_sample"])

print()

display(df["category"].value\_counts().to\_frame())

if not text\_only:

plot\_class\_distribution(df["category"])

plot\_word\_count\_distribution(df["headline"], bins=number\_of\_bins)

plot\_frequency\_distribution\_of\_ngrams(df["headline"])

#### key\_statistics Analysis[¶](#key_statistics-Analysis)

In [7]:

def get\_key\_statistics\_about\_data(df):

ret = dict()

ret["n\_samples"] = len(df)

ret["n\_classes"] = len(df["category"].unique())

ret["n\_missing\_headlines"] = df["headline"].isnull().sum()

ret["n\_empty\_headlines"] = len(df[df["headline"] == ""])

ret["n\_missing\_classes"] = df["category"].isnull().sum()

number\_of\_words\_per\_sample = df["headline"].apply(lambda x: len(x.split()))

ret["n\_words\_per\_sample"] = number\_of\_words\_per\_sample

ret["min\_n\_words\_per\_sample"] = number\_of\_words\_per\_sample.min()

ret["median\_n\_words\_per\_sample"] = number\_of\_words\_per\_sample.median()

ret["max\_n\_words\_per\_sample"] = number\_of\_words\_per\_sample.max()

number\_of\_samples\_per\_class = df["category"].value\_counts()

ret["min\_n\_samples\_per\_class"] = number\_of\_samples\_per\_class.min()

ret["median\_n\_samples\_per\_class"] = number\_of\_samples\_per\_class.median()

ret["max\_n\_samples\_per\_class"] = number\_of\_samples\_per\_class.max()

return ret

#### Class distribution[¶](#Class-distribution)

In [8]:

def plot\_class\_distribution(labels):

counts = labels.value\_counts()

plt.figure(figsize=(16, 8))

plt.bar(counts.index, counts, width=0.8, color='b')

plt.xlabel('Class')

plt.ylabel('Number of samples')

plt.title('Class distribution')

plt.xticks(counts.index, counts.index, rotation=90)

plt.show()

#### Word count distribution[¶](#Word-count-distribution)

In [9]:

def plot\_word\_count\_distribution(sample\_texts, bins=30):

plt.figure(figsize=(16, 8))

plt.hist([len(s.split()) for s in sample\_texts], bins)

plt.xlabel('Word count')

plt.ylabel('Number of samples')

plt.title('Word count distribution')

plt.show()

#### Frequency distribution of n-grams[¶](#Frequency-distribution-of-n-grams)

In [10]:

def plot\_frequency\_distribution\_of\_ngrams(sample\_texts,ngram\_range=(1, 2),num\_ngrams=50):

kwargs = {

'ngram\_range': (1, 1),

'dtype': 'int32',

'strip\_accents': 'unicode',

'decode\_error': 'replace',

'analyzer': 'word', # Split text into word tokens.

}

vectorizer = CountVectorizer(\*\*kwargs)

# This creates a vocabulary (dict, where keys are n-grams and values are

# idxices). This also converts every text to an array the length of

# vocabulary, where every element idxicates the count of the n-gram

# corresponding at that idxex in vocabulary.

vectorized\_texts = vectorizer.fit\_transform(sample\_texts)

# This is the list of all n-grams in the index order from the vocabulary.

all\_ngrams = list(vectorizer.get\_feature\_names())

num\_ngrams = min(num\_ngrams, len(all\_ngrams))

# ngrams = all\_ngrams[:num\_ngrams]

# Add up the counts per n-gram ie. column-wise

all\_counts = vectorized\_texts.sum(axis=0).tolist()[0]

# Sort n-grams and counts by frequency and get top `num\_ngrams` ngrams.

all\_counts, all\_ngrams = zip(\*[(c, n) for c, n in sorted(

zip(all\_counts, all\_ngrams), reverse=True)])

ngrams = list(all\_ngrams)[:num\_ngrams]

counts = list(all\_counts)[:num\_ngrams]

idx = np.arange(num\_ngrams)

plt.figure(figsize=(16, 8))

plt.bar(idx, counts, width=0.8, color='b')

plt.xlabel('N-grams')

plt.ylabel('Frequencies')

plt.title('Frequency distribution of n-grams')

plt.xticks(idx, ngrams, rotation=45)

plt.show()

#### Explore All Data[¶](#Explore-All-Data)

In [11]:

explore\_data(dataframe)

Number of samples: 200853 Out[6,7,8,9,10,11]:

Number of classes: 41

Number of empty headlines: 6

Number of missing headlines: 0

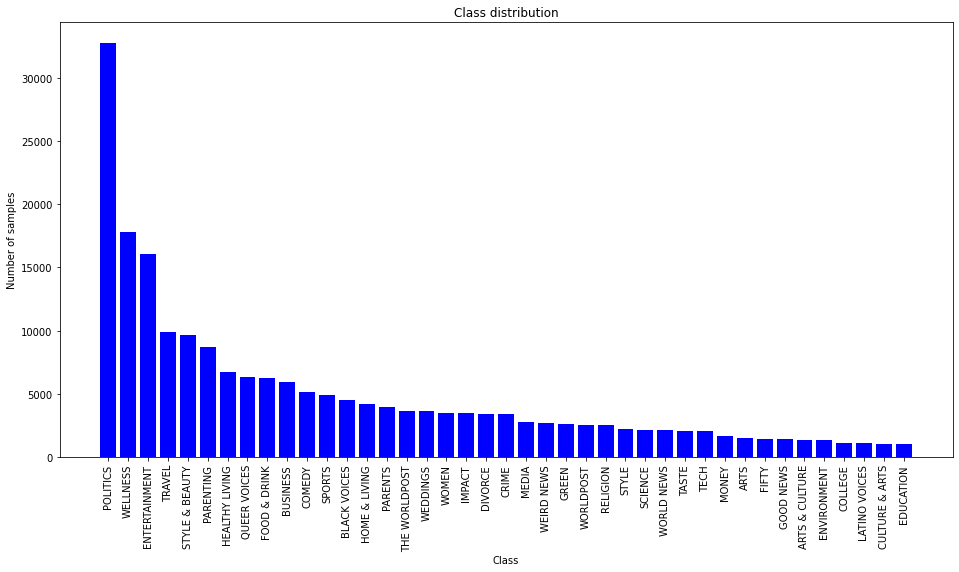
Number of missing classes: 0

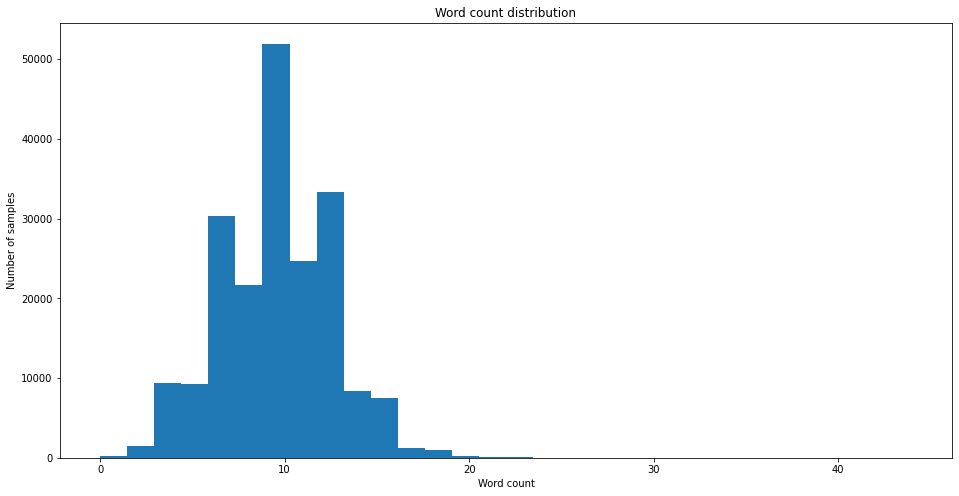
Samples per class: Min: 1004 Median: 3405.0 Max: 32739

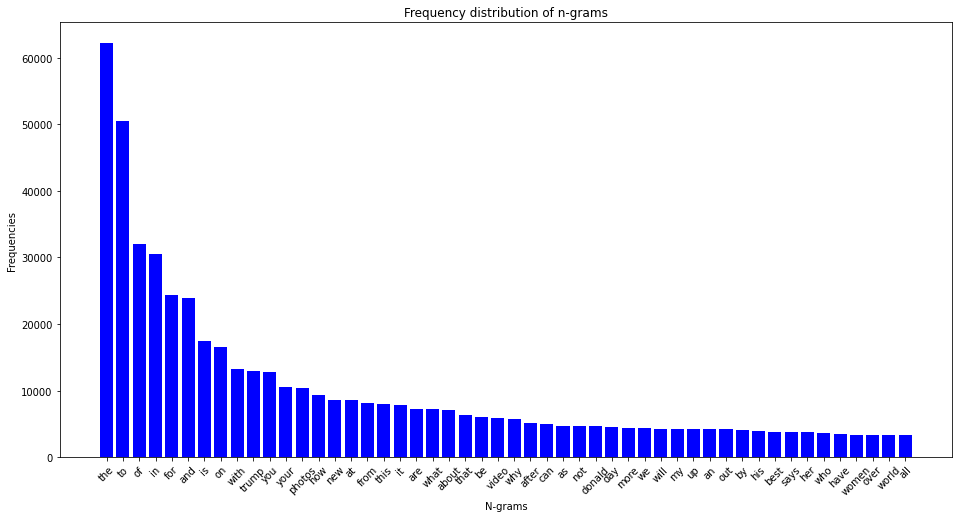
Words per sample: Min: 0 Median: 10.0 Max: 44

Samples / Median number of words per sample: 20085.3

|  | **category** |
| --- | --- |
| **POLITICS** | 32739 |
| **WELLNESS** | 17827 |
| **ENTERTAINMENT** | 16058 |
| **TRAVEL** | 9887 |
| **STYLE & BEAUTY** | 9649 |
| **PARENTING** | 8677 |
| **HEALTHY LIVING** | 6694 |
| **QUEER VOICES** | 6314 |
| **FOOD & DRINK** | 6226 |
| **BUSINESS** | 5937 |
| **COMEDY** | 5175 |
| **SPORTS** | 4884 |
| **BLACK VOICES** | 4528 |
| **HOME & LIVING** | 4195 |
| **PARENTS** | 3955 |
| **THE WORLDPOST** | 3664 |
| **WEDDINGS** | 3651 |
| **WOMEN** | 3490 |
| **IMPACT** | 3459 |
| **DIVORCE** | 3426 |
| **CRIME** | 3405 |
| **MEDIA** | 2815 |
| **WEIRD NEWS** | 2670 |
| **GREEN** | 2622 |
| **WORLDPOST** | 2579 |
| **RELIGION** | 2556 |
| **STYLE** | 2254 |
| **SCIENCE** | 2178 |
| **WORLD NEWS** | 2177 |
| **TASTE** | 2096 |
| **TECH** | 2082 |
| **MONEY** | 1707 |
| **ARTS** | 1509 |
| **FIFTY** | 1401 |
| **GOOD NEWS** | 1398 |
| **ARTS & CULTURE** | 1339 |
| **ENVIRONMENT** | 1323 |
| **COLLEGE** | 1144 |
| **LATINO VOICES** | 1129 |
| **CULTURE & ARTS** | 1030 |
| **EDUCATION** | 1004 |







# FIFTY category[¶](" \l "FIFTY-category)

In [12]:

display(dataframe[dataframe["category"] == "FIFTY"])

Out[12]:

|  | **headline** | **category** |
| --- | --- | --- |
| **35278** | Love, Facebook and Infidelity | FIFTY |
| **38400** | Boomers Were Time's "Man of the Year" Fifty Ye... | FIFTY |
| **38986** | Be Grateful At The Holidays For Sprinkles Of H... | FIFTY |
| **39037** | A No Bullsh-t Holiday Letter | FIFTY |
| **39629** | How Our Vocabulary Gives Away Our Age | FIFTY |
| **...** | ... | ... |
| **124796** | Middle-aged and Invisible at Coachella | FIFTY |
| **124835** | How A Dinner Party Changed My Outlook On Aging | FIFTY |
| **124836** | What Kind Of Inheritance Do You Really Owe You... | FIFTY |
| **124848** | Eight Factors To Consider When Choosing Your O... | FIFTY |
| **124916** | 4 Stunning Spring Dresses For Boomer Women | FIFTY |

1401 rows × 2 columns

# IMPACT category[¶](#IMPACT-category)

In [13]:

display(dataframe[dataframe["category"] == "IMPACT"]) Out[13]:

|  | **headline** | **category** |
| --- | --- | --- |
| **12** | With Its Way Of Life At Risk, This Remote Oyst... | IMPACT |
| **66** | Monsanto And Bayer Are Set To Merge. Here's Wh... | IMPACT |
| **125** | You're Going To Use That Self-Checkout Machine... | IMPACT |
| **193** | Machines Don't Always Steal Our Jobs. Increasi... | IMPACT |
| **286** | She Started A Suicide Prevention Site At Age 1... | IMPACT |
| **...** | ... | ... |
| **200762** | Texana Hollis, 101-Year-Old Evicted Detroit Wo... | IMPACT |
| **200763** | Malaria's Defeat, Africa's Future | IMPACT |
| **200825** | Tinker and Change the World | IMPACT |
| **200826** | Pregnant and Displaced: Double the Danger | IMPACT |
| **200827** | Tom Brady Helps Mentor, Tom Martinez, Find A K... | IMPACT |

3459 rows × 2 columns

# WOMEN category[¶](#WOMEN-category)

In [14]:

display(dataframe[dataframe["category"] == "WOMEN"]) Out[14]:

|  | **headline** | **category** |
| --- | --- | --- |
| **35** | Morgan Freeman Dropped From Marketing Campaign... | WOMEN |
| **67** | The Joy Of Watching Harvey Weinstein’s Perp Walk | WOMEN |
| **68** | The 20 Funniest Tweets From Women This Week | WOMEN |
| **79** | Morgan Freeman Accused Of Inappropriate Behavi... | WOMEN |
| **274** | What Do You Say To Sexist Passengers? This Fem... | WOMEN |
| **...** | ... | ... |
| **124915** | 7 Ways to Cook up Chemistry Through Conversation | WOMEN |
| **124955** | The Power of the Purse: An Untapped Opportunity | WOMEN |
| **124967** | 'It's Complicated': How I Learned to Fend off ... | WOMEN |
| **124981** | Millennial Women Don't Lack Confidence -- They... | WOMEN |
| **124984** | Why I Thought I'd Never Live To See 33 | WOMEN |

3490 rows × 2 columns

# WELLNESS category[¶](#WELLNESS-category)

In [15]:

display(dataframe[dataframe["category"] == "WELLNESS"])

Out[15]:

|  | **headline** | **category** |
| --- | --- | --- |
| **124989** | Why Overeating Doesn't Make You Fat | WELLNESS |
| **124990** | 14 Habits Of People With A Healthy Relationshi... | WELLNESS |
| **124993** | 5 Things That Could Be Stealing Your Joy | WELLNESS |
| **124994** | Moments Make a Life | WELLNESS |
| **124996** | Fat Facts | WELLNESS |
| **...** | ... | ... |
| **200797** | Shoveling Snow? How to Protect Your Back (And ... | WELLNESS |
| **200799** | 7 Reasons Working Too Much Is Bad For Your Health | WELLNESS |
| **200800** | The Sleep Library: 11 Soothing Books For Bedtime | WELLNESS |
| **200802** | The Benefits of Caring for a Pet | WELLNESS |
| **200805** | This Is Only the Beginning: Surprising Advice ... | WELLNESS |

17827 rows × 2 columns

# Assigning numeric values to the categories[¶](#Assigning-numeric-values-to-the-categor)

In [16]:

def codify\_labels(df):

label = dict( zip( df['category'].cat.codes, df['category'] ) )

category\_code = dict( zip( df['category'], df['category'].cat.codes ) )

df['category'] = df.category.cat.codes

return label, category\_code

label, category\_code = codify\_labels(dataframe)

display(label)

Out[16]:

{6: 'CRIME',

10: 'ENTERTAINMENT',

39: 'WORLD NEWS',

18: 'IMPACT',

24: 'POLITICS',

36: 'WEIRD NEWS',

2: 'BLACK VOICES',

38: 'WOMEN',

5: 'COMEDY',

25: 'QUEER VOICES',

28: 'SPORTS',

3: 'BUSINESS',

34: 'TRAVEL',

20: 'MEDIA',

32: 'TECH',

26: 'RELIGION',

27: 'SCIENCE',

19: 'LATINO VOICES',

9: 'EDUCATION',

4: 'COLLEGE',

23: 'PARENTS',

1: 'ARTS & CULTURE',

29: 'STYLE',

15: 'GREEN',

31: 'TASTE',

16: 'HEALTHY LIVING',

33: 'THE WORLDPOST',

14: 'GOOD NEWS',

40: 'WORLDPOST',

12: 'FIFTY',

0: 'ARTS',

37: 'WELLNESS',

22: 'PARENTING',

17: 'HOME & LIVING',

30: 'STYLE & BEAUTY',

8: 'DIVORCE',

35: 'WEDDINGS',

13: 'FOOD & DRINK',

21: 'MONEY',

11: 'ENVIRONMENT',

7: 'CULTURE & ARTS'}

In [17]:

categories = dataframe["category"].unique()

display(categories)

array([ 6, 10, 39, 18, 24, 36, 2, 38, 5, 25, 28, 3, 34, 20, 32, 26, 27,

19, 9, 4, 23, 1, 29, 15, 31, 16, 33, 14, 40, 12, 0, 37, 22, 17,

30, 8, 35, 13, 21, 11, 7], dtype=int8)

# Splitting the data into training, validation, and test data[¶](#Splitting-the-data-into-training,-valid)

In [18]:

def split\_dataset(df, seed=42, percentage\_train=0.8, percentage\_validation=0.15, percentage\_test=0.05):

assert percentage\_train + percentage\_validation + percentage\_test == 1

dataset\_size = len(df)

headlines\_train, headlines\_rest, categories\_train, categories\_rest = train\_test\_split(df["headline"], df["category"], random\_state=seed, test\_size=0.2)

headlines\_validation, headlines\_test, categories\_validation, categories\_test = train\_test\_split(headlines\_rest, categories\_rest, random\_state=seed, test\_size=0.25)

train\_df = pd.concat([categories\_train, headlines\_train], axis=1)

train\_df.columns = ["category", "headline"]

validation\_df = pd.concat([categories\_validation, headlines\_validation], axis=1)

validation\_df.columns = ["category", "headline"]

test\_df = pd.concat([categories\_test, headlines\_test], axis=1)

test\_df.columns = ["category", "headline"]

return train\_df, validation\_df, test\_df

train\_df, validation\_df, test\_df = split\_dataset(dataframe)

display(train\_df[:])

|  | **category** | **headline** |
| --- | --- | --- |
| **66880** | 3 | Martin Shkreli Wants To Be The Only One To Own... |
| **68387** | 23 | Panel Recommends Depression Screening For Preg... |
| **100514** | 15 | 200 Yellowstone Bison Allegedly Sent To Slaughter |
| **197218** | 30 | Abercrombie & Fitch Employees Forced To Do Pus... |
| **144020** | 37 | Life's Path Is a Maze, Not a Straight Line |
| **...** | ... | ... |
| **119879** | 16 | Overwhelmed? 7 Strategies for Restoring Balance |
| **103694** | 26 | Single Dad Of 5 Can't Afford Christmas Present... |
| **131932** | 30 | Pharrell's Hat Has A Twitter, Because Of Course |
| **146867** | 17 | Easy Storage Solutions From A To Z |
| **121958** | 14 | It Takes a Long Time to Be an Overnight Success |

160682 rows × 2 columns

# Computing class weights[¶](#Computing-class-weights)

In [19]:

class\_weights = dict(zip(train\_df["category"].unique(), class\_weight.compute\_class\_weight('balanced', train\_df["category"].unique(), train\_df["category"])))

c:\python\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass classes=[ 3 23 15 30 37 32 20 25 24 2 18 10 16 6 8 34 28 5 29 17 35 27 22 33

12 13 31 4 26 11 38 36 21 7 39 40 14 19 9 0 1], y=66880 3

68387 23

100514 15

197218 30

144020 37

..

119879 16

103694 26

131932 30

146867 17

121958 14

Name: category, Length: 160682, dtype: int8 as keyword args. From version 1.0 (renaming of 0.25) passing these as positional arguments will result in an error

warnings.warn(f"Pass {args\_msg} as keyword args. From version "

In [20]:

display({label[category\_code]: weight for category\_code, weight in class\_weights.items()})

{'BUSINESS': 0.8137610404343246,

'PARENTS': 1.2355211761449267,

'GREEN': 1.8600252352784563,

'STYLE & BEAUTY': 0.5112946080537126,

'WELLNESS': 0.2730871138409663,

'TECH': 2.382415301356661,

'MEDIA': 1.7257037299567184,

'QUEER VOICES': 0.7797598827560102,

'POLITICS': 0.14926959324820824,

'BLACK VOICES': 1.0748966458397442,

'IMPACT': 1.4184122948721345,

'ENTERTAINMENT': 0.30335731641239316,

'HEALTHY LIVING': 0.7322632979693026,

'CRIME': 1.4225310964543403,

'DIVORCE': 1.4355579379969623,

'TRAVEL': 0.4976600851722803,

'SPORTS': 1.0132040255252603,

'COMEDY': 0.9542423108672284,

'STYLE': 2.1869827961672472,

'HOME & LIVING': 1.1709211744044539,

'WEDDINGS': 1.3603169631140948,

'SCIENCE': 2.2575306283016747,

'PARENTING': 0.5660128784996689,

'THE WORLDPOST': 1.3334716470676105,

'FIFTY': 3.46207877273119,

'FOOD & DRINK': 0.7899764504599289,

'TASTE': 2.392596563328271,

'COLLEGE': 4.3496927533093315,

'RELIGION': 1.9043115504041337,

'ENVIRONMENT': 3.739573636194377,

'WOMEN': 1.4168738867432058,

'WEIRD NEWS': 1.815226109648776,

'MONEY': 2.8965803183530725,

'CULTURE & ARTS': 4.850338082588746,

'WORLD NEWS': 2.239470383275261,

'WORLDPOST': 1.914544782966149,

'GOOD NEWS': 3.4929350897787055,

'LATINO VOICES': 4.287826226183487,

'EDUCATION': 4.80278574844572,

'ARTS': 3.2362288775654067,

'ARTS & CULTURE': 3.7112435328898745}

End of the code

CONCLUSION

* We have built a number of models to predict the category of news from its headline and short description - using methods both from traditional ML and deep learning. Our best model (ensemble of four NN models) achieves on the dev set 68.85% accuracy, if considering top 1 label, and 88.72%, if considering top 3 labels predicted by the model. It is interesting how this news dataset is extremely hard to classify for even the most complex models. We attribute this to the subjectivity in category assignment in the data. However, in the future work we may also try to apply character-level language models based on multi-layer LSTM or learn embeddings for the whole news descriptions (as in doc2vec).

**REFERENCES AND BIBLIOGRAPHY**

* [1] The Huffington Post. <https://www.huffingtonpost.com/>.
* [2] Keras: The Python Deep Learning library. <https://keras.io/>.
* [3] Scikit-learn 0.20.0 documentation. ”1.9 Naive Bayes”. **https://scikit-learn.org/ stable/modules/naive\_bayes.html.**
* [4] Scikit-learn 0.20.0 documentation. ”1.1.11 Logistic Regression”. **https://scikit-learn. org/stable/modules/linear\_model.html#logistic-regression**.
* [5] Scikit-learn 0.20.0 documentation. ”1.4 Support Vector Machines”. **https://scikit-learn. org/stable/modules/svm.html.**
* [6] Scikit-learn 0.20.0 documentation. ”1.10 Decision Trees”. **https://scikit-learn.org/ stable/modules/tree.html.**
* [7] Jürgen Schmidhuber’s page on recurrent neural networks. **http://people.idsia.ch/ ~juergen/rnn.html**.
* [8] Scikit-learn 0.20.1 documentation, t-distributed Stochastic Neighbor Embedding. **https: //scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.**
* **[9]** T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013
* **[10]** J. Pennington, R. Socher, and C. D. Manning. Glove: Global Vectors for Word Representation. EMNLP, 14:1532–1543, 2014.

……….etc.