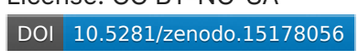
Remixing Writing Pedagogies with Writing Code and Data through Exploratory Data Analysis

CCCC 2025 | Computer Love | Session 2151 | Room 350

- **Time**: 04/10/2025 4:45PM - 6:00PM EDT
- **Abstract**: What is our discipline's role in teaching and advocating for critical approaches to writing data? The facilitator and attendees will work through a computational notebook together that conducts exploratory data analysis. This coding and data process is meant to facilitate discussion to learn more about what writing and technical communication can bring to data practices.

  The goal will be to learn more about the problems and potential for writing and technical and professional communication (TPC) scholars and teachers to teach coding and data practices, such as machine learning and data modeling, with a humanities-driven, advocacy approach.
- Who Should Come?: Novices and experts are all welcome and encouraged to attend. No previous experience coding is required, since this ELE is meant to facilitate a discussion about the role that writing and TPC plays pedagogically beyond using (or not using) generative AI tools.
- Accompanying Google Slides
- License: CC BY-NC-SA
- 

Citation

- Lindgren, C. A. (10 Apr. 2025). Remixing Writing Pedagogies with Writing Code and Data through Exploratory Data Analysis. NCTE CCCC 2025. Baltimore, MD, United States. https://doi.org/10.5281/zenodo.15177209

Chapter 4. Text Classification with Logistic Regression

Our first machine-learning challenge is to create a prediction model that categorizes news articles with the appropriate categories from a set of 31 categories: politics, entertainment, etc. The model uses logistic regression techniques in Python on a dataset with headlines, short descriptions, and URLs.

NOTE: Be sure to watch and read the materials posted in the Canvas module before and while you work through this notebook.

Learning Objectives

1. Import and run EDA techniques to understand the potential limits and affordances of the dataset with our ML goal in mind.
2. Learn about the basic mechanics of logistic regression (LR).
3. Apply LR to this text classification goal of categorizing the news genre of articles based on potential "features" in the data, such as the article's headline, short description, and URL.

Sources

- Notebook modified from Ganesan's LR example exercise: Text Classification with Logistic Regression
- Dataset: **HuffPost News Headlines & Categories** in ../data/news_category_dataset.json

Import Libraries

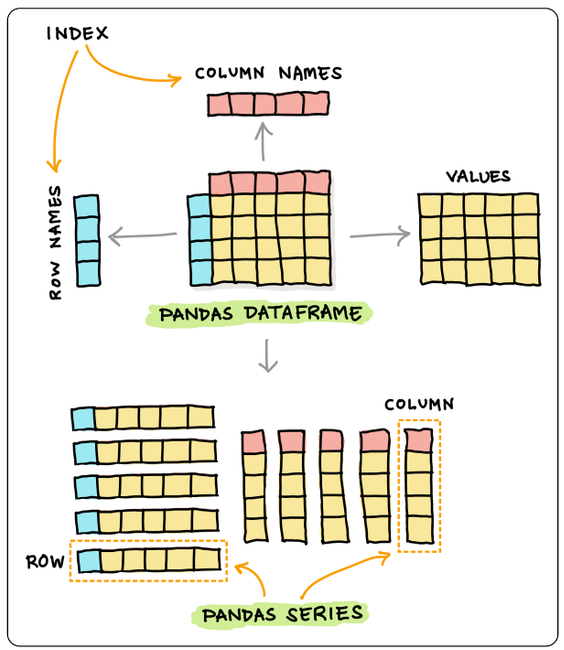
In [3]:
```
# You may need to install some of the libraries below
# If so, uncomment any of the below commands
# %pip install pandas==2.1.4
# %pip install matplotlib==3.10.0
# %pip install scipy==1.13.1
# %pip install seaborn==0.13.0
# %pip install wquantiles==0.6
# %pip install statsmodels==0.14.4
# %pip install scikit-learn==1.6.0
# %pip install mplcyberpunk==0.7.6
```

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
import logging
# Custom utility functions
from utils import _reciprocal_rank,compute_accuracy,compute_mrr_at_k,collect_preds,extract_features,get_top_k_predictions,train_model,roc_curve

%config InlineBackend.figure_formats = ['svg']
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
```

0. Refresher on pandas' dataframes

We can use the pandas library to read in, review, and revise the data set.

`pandas`, i.e., **Panel data** or **Dataframe**, organizes information in rows for *What is observed** and columns for *properties of those observations*, as well as an *index* column to make it easier to reference each row uniquely.



Original image src: Nantasenamat

Pandas makes it so much easier to work with indexed tabular data. The main types of data structures that pandas work with are *Series* and *DataFrames*.
Pandas Series: One-dimensional array akin to a `column` *OR* a `row` in a spreadsheet. They are essentially a special kind of array list with special methods and functions.

Pandas DataFrame: Two-dimensional array akin to the combination of `columns` *AND* `rows` in a spreadsheet. Like Series, the DataFrame has special methods and functions that we can use to explore and analyze data. Also, we will focus a lot more on DFs, rather than Series. But, you should know about Series, because you sometimes create new Series (columns or rows) to add to your existing DataFrame. It's pretty rad.

Indices: Notice the color-coding going in the figure: red, blue, and yellow. The red and blue represent the indices of the rows and columns, while the yellow squares represent the values in the dataset.

Recall how pandas' "panel data" helps us more easily transform multiple types of structured data into what's called a *DataFrame*. Dataframe's are two-dimensional tabular data that are mutable (transformable) with labeled axes (rows and columns). See chapter 3 to refresh your memory, if needed. You can also reference pandas in general, or its Dataframe datatype.

1. Import the Data

In [5]:
```
# Imports the noted JSON file as a pandas DataFrame based on the path below
df = pd.read_json("./data/news_category_dataset.json", lines=True)
```

2. Exploratory Data Analysis

`.shape` returns a `tuple` e.g., `(value, value)`, with info about the basic 2 dimensions of panel data:

1. the number of rows, and
2. the number of columns.

In [6]:
```
df.shape
```

Out[6]:
```
(124989, 6)
```

The pandas DataFrame `.info()` method returns a more comprehensive description of the data set.

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 124989 entries, 0 to 124988
Data columns (total 6 columns):
 #   Column             Non-Null Count   Dtype         
---  ------             --------------   -----         
 0   short_description  124989 non-null  object        
 1   headline           124989 non-null  object        
 2   date               124989 non-null  datetime64[ns]
 3   link               124989 non-null  object        
 4   authors            124989 non-null  object        
 5   category           124989 non-null  object        
dtypes: datetime64[ns](1), object(5)
memory usage: 5.7+ MB
```

Looks like there are no `null` values in the dataset.

Let's dig deeper into the columns and values.

2.1 Review particular columns

Be sure to review the data per column. The goals of this modeling may be set for you, but you will be asked to perform similar EDA work on your final project, so you can better understand the modeling possibilities and boundaries with your data. So, the below includes a series of exercises for you to understand this dataset before you conduct the actual modeling work.

In [8]:
```
# List the columns for reference
```

```
df.columns
```

Out[8]:
```
Index(['short_description', 'headline', 'date', 'link', 'authors', 'category'], dtype='object')
```

2.2 Describe and examine the `headline` column

We can use `.describe()` on a Series, so we can consider any potential quirks.

Let's check out the `headline` column, since that's an important column for training our model.

What are the summary stats for the `headline`?

In [9]:
```
df.headline.describe()
```

Out[9]:
```
count              124989
unique             124560
top       Sunday Roundup
freq                   90
Name: headline, dtype: object
```

Some initial observations:

- Noticing how there is a decently sized difference between the `count` and `unique` values
- `top`: How "Sunday Roundup" headline appears 90 times (`freq`) in the `headline` column (`Name: headline, dtype: object`).

Let's review rows with the "Sunday Roundup" value in the headline column by using our new skills with pandas. Below I ...

1. `df.loc[]`: query the dataframe with the location method
2. `[df.headline]`: Specifiy what slice of the data I want to isolate.
3. `.str.contains('Sunday Roundup')`: Since `headline` values are Strings, and I want to isolate any rows with the "Sunday Roundup" headline, search the isolated Series, `headline`, with the `.str.contains()` method. It takes a string as its main parameter.

In [10]:
```
df.loc[df.headline.str.contains('Sunday Roundup')].sample(10)
```

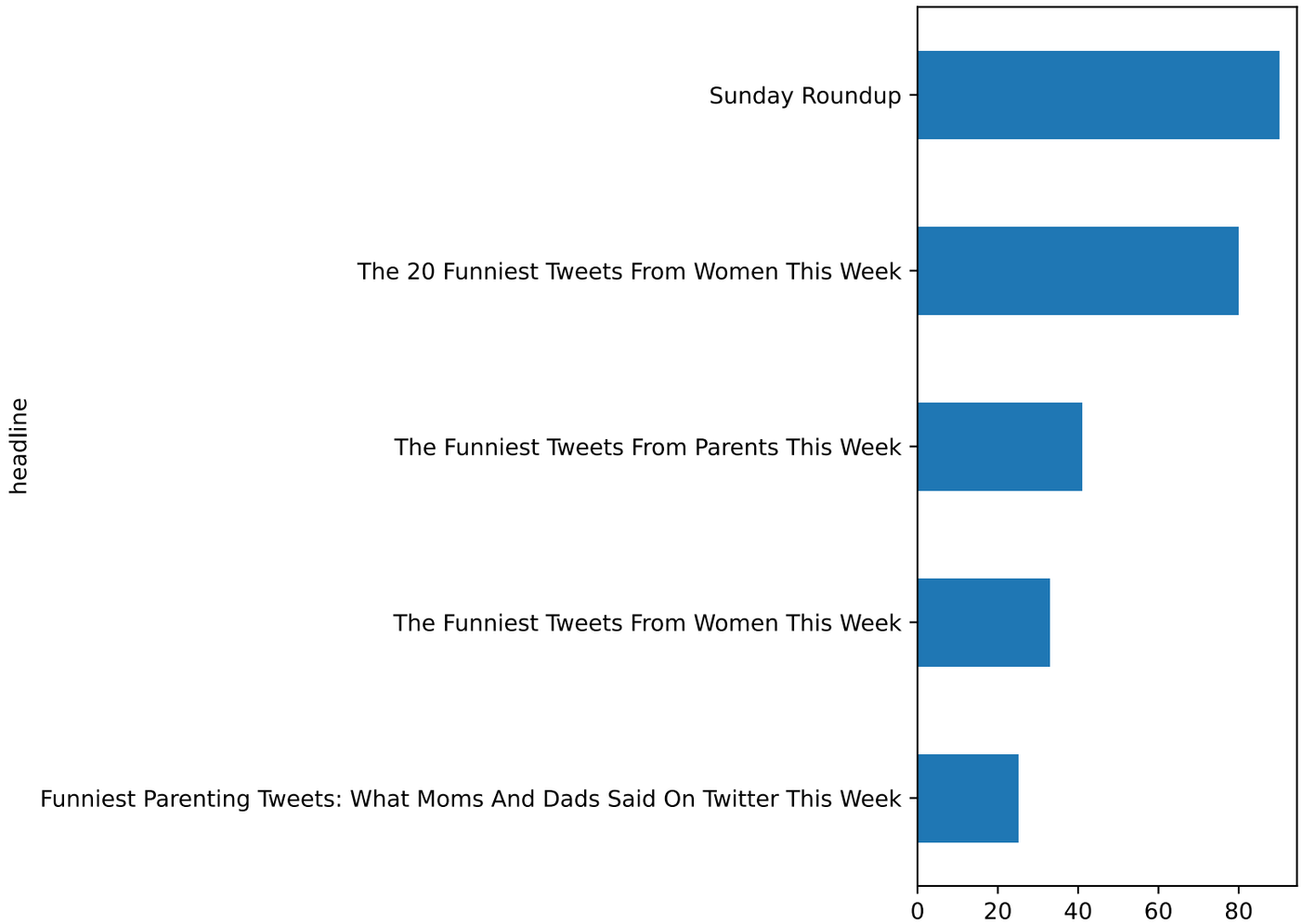
Out[10]:

	short_description	headline	date	link	authors	category
107742	We don't know who will be victorious in Tuesda...	Sunday Roundup	2014-11-02	https://www.huffingtonpost.com/entry/sunday-ro...	Arianna Huffington, Contributor	POLITICS
116983	This week, President Obama's $3.7 billion requ...	Sunday Roundup	2014-07-20	https://www.huffingtonpost.com/entry/sunday-ro...	Arianna Huffington, Contributor	POLITICS
82168	This week we saw how dissimilar appeals to our...	Sunday Roundup	2015-08-23	https://www.huffingtonpost.com/entry/sunday-ro...	Arianna Huffington, Contributor	POLITICS
96792	This week proved that while the arc of the mor...	Sunday Roundup	2015-03-08	https://www.huffingtonpost.com/entry/sunday-ro...	Arianna Huffington, Contributor	POLITICS
92626	This week, the White House revealed it really ...	Sunday Roundup	2015-04-26	https://www.huffingtonpost.com/entry/sunday-ro...	Arianna Huffington, Contributor	POLITICS
62390	This week the nation got to experience March M...	Sunday Roundup	2016-04-03	https://www.huffingtonpost.com/entry/sunday-ro...	Arianna Huffington, Contributor	POLITICS
88321	This week, the nation waited in breathless ant...	Sunday Roundup	2015-06-14	https://www.huffingtonpost.com/entry/sunday-ro...	Arianna Huffington, Contributor	POLITICS
113253	This week got off to a horrifying start as a 9...	Sunday Roundup	2014-08-31	https://www.huffingtonpost.com/entry/sunday-ro...	Arianna Huffington, Contributor	POLITICS
92027	This week, the nation's eyes were on Baltimore...	Sunday Roundup	2015-05-03	https://www.huffingtonpost.com/entry/sunday-ro...	Arianna Huffington, Contributor	POLITICS
124201	This week, Mika Brzezinski and I hosted our se...	Sunday Roundup	2014-04-27	https://www.huffingtonpost.com/entry/sunday-ro...	Arianna Huffington, Contributor	POLITICS

Let's review the top 5 repeating headlines, in case there's somethign worth noting.

In [11]:
```
df.groupby(['headline'])['headline'].count().sort_values(ascending=True)[-5:].plot(
  kind='barh',
  figsize=(3,7)
)
```

Out[11]:
```
<Axes: ylabel='headline'>
```



2.3 How long are the headlines? (What's the distribution?)

Also curious about the distribution of the length of those headlines, because that can tell us how much semantic info may be available in the headlines. Specifically, **if a large sum of headlines are only 3-4 words, that may put a cap on how much contextual info each categorical label will actually supply the LR model**.

So, let's add a new column to the pandas DataFrame, `df`, by `apply()`ing the `len` (length) method to the `headline` column. Below, I do so with the `apply()`

method and assign the values per row to a new Series (column) that I call `headline_length`.

In [12]:
```
df['headline_char_length'] = df.headline.apply(len)
df.head()
```

Out[12]:

	short_description	headline	date	link	authors	category	headline_char_length
0	She left her husband. He killed their children...	There Were 2 Mass Shootings In Texas Last Week...	2018-05-26	https://www.huffingtonpost.com/entry/texas-ama...	Melissa Jeltsen	CRIME	64
1	Of course it has a song.	Will Smith Joins Diplo And Nicky Jam For The 2...	2018-05-26	https://www.huffingtonpost.com/entry/will-smit...	Andy McDonald	ENTERTAINMENT	75
2	The actor and his longtime girlfriend Anna Ebe...	Hugh Grant Marries For The First Time At Age 57	2018-05-26	https://www.huffingtonpost.com/entry/hugh-gran...	Ron Dicker	ENTERTAINMENT	47
3	The actor gives Dems an ass-kicking for not fi...	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	2018-05-26	https://www.huffingtonpost.com/entry/jim-carre...	Ron Dicker	ENTERTAINMENT	69
4	The "Dietland" actress said using the bags is ...	Julianna Margulies Uses Donald Trump Poop Bags...	2018-05-26	https://www.huffingtonpost.com/entry/julianna-...	Ron Dicker	ENTERTAINMENT	71

Ok, we counted characters in the headline to capture one angle of headline length. Now, let's add an approximate word length Series as a column in the DataFrame.

Use `apply()`, again, but use a little Python built-in method magic to create a simple function that:

1. Assigns a variable to the row's headline String value: `lambda hl:`
2. Splits the `hl` String value into a List array delimited by spaces: `hl.split(' ')`
3. Returns the length of the List array as an Integer: `len()`

BOOM! We got a great approximation of the number of words in a new column.

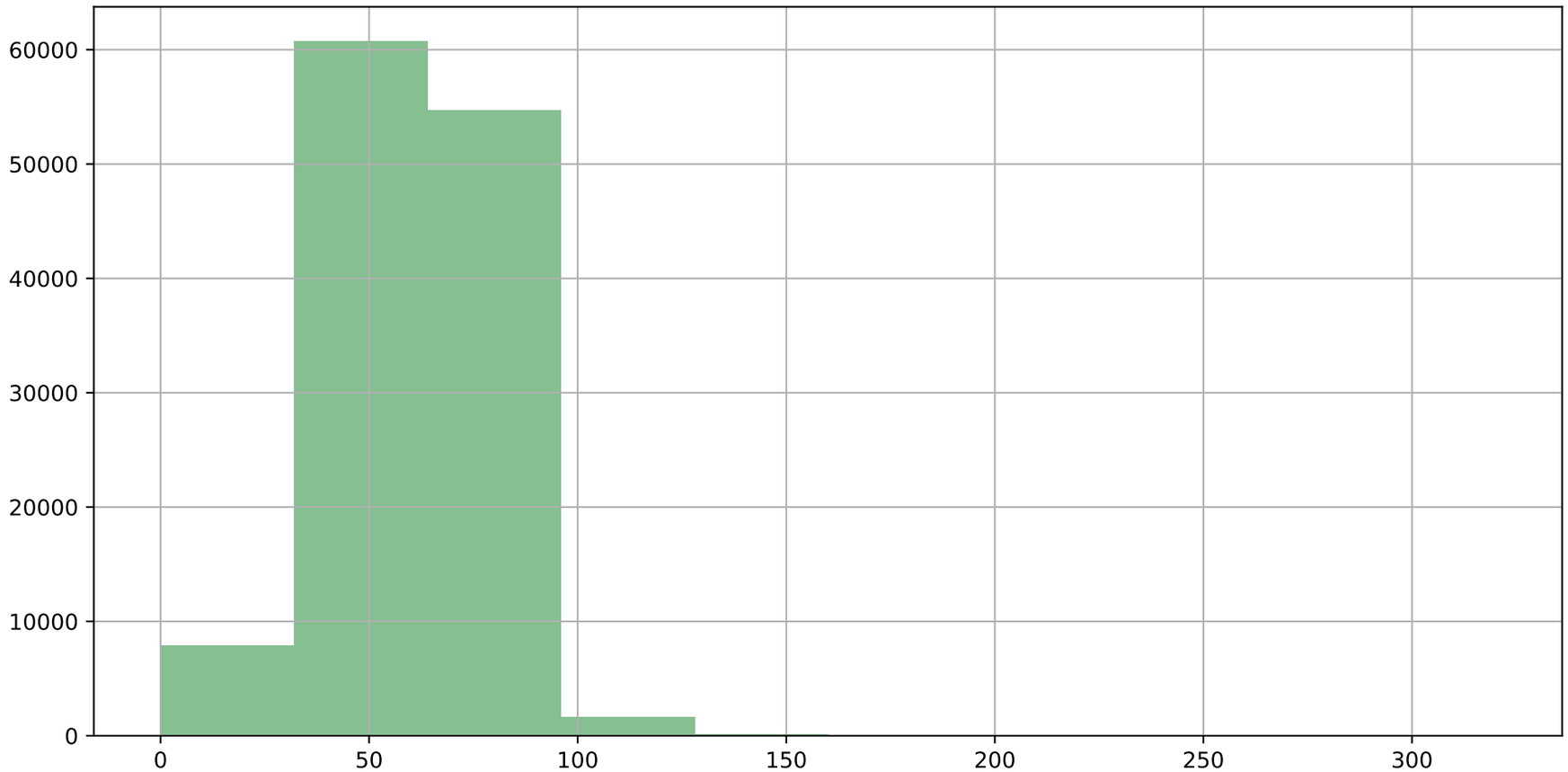
In [13]:
```
df['headline_word_length'] = df.headline.apply(lambda hl: len(hl.split(' ')) )
df.head()
```

Out[13]:

	short_description	headline	date	link	authors	category	headline_char_length	headline_word_length
0	She left her husband. He killed their children...	There Were 2 Mass Shootings In Texas Last Week...	2018-05-26	https://www.huffingtonpost.com/entry/texas-ama...	Melissa Jeltsen	CRIME	64	14
1	Of course it has a song.	Will Smith Joins Diplo And Nicky Jam For The 2...	2018-05-26	https://www.huffingtonpost.com/entry/will-smit...	Andy McDonald	ENTERTAINMENT	75	14
2	The actor and his longtime girlfriend Anna Ebe...	Hugh Grant Marries For The First Time At Age 57	2018-05-26	https://www.huffingtonpost.com/entry/hugh-gran...	Ron Dicker	ENTERTAINMENT	47	10
3	The actor gives Dems an ass-kicking for not fi...	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	2018-05-26	https://www.huffingtonpost.com/entry/jim-carre...	Ron Dicker	ENTERTAINMENT	69	11
4	The "Dietland" actress said using the bags is ...	Julianna Margulies Uses Donald Trump Poop Bags...	2018-05-26	https://www.huffingtonpost.com/entry/julianna-...	Ron Dicker	ENTERTAINMENT	71	13

In [14]:
```
'''
  .hist() -- Creates a histogram chart with a dataframe's Series
  Histograms place a metric in bins -- dates in this case -- to understand the distribution of the data. In this case, the distribution of the
'''
df.headline_char_length.hist(
  figsize=(12,6),
  color='#86bf91',
)
```

Out[14]:
```
<Axes: >
```

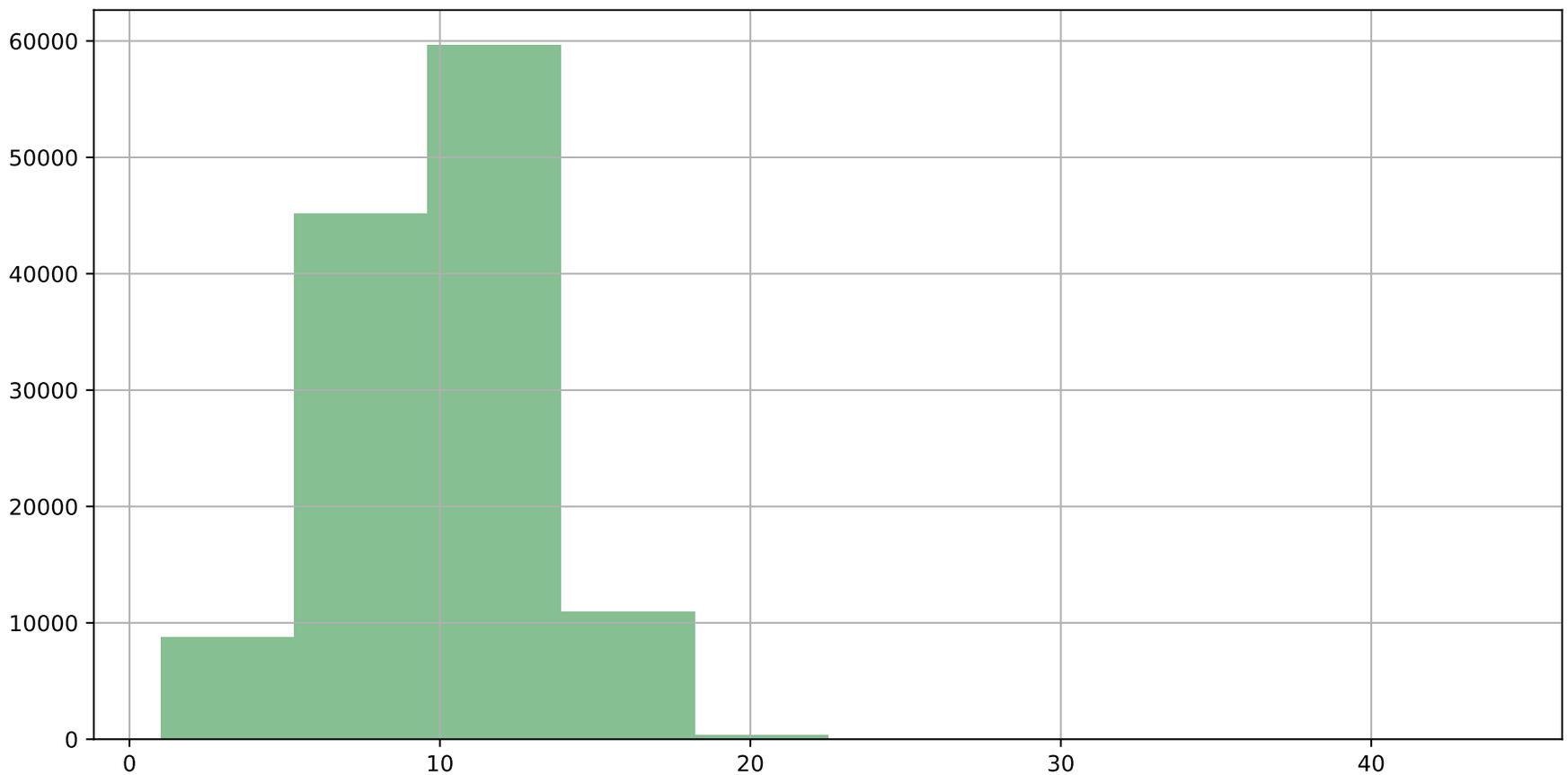


Looks like there's some variance with some headlines that are over 100 characters long and as much as ~300.

Let's use `.describe()` on this new column as a table of values to review.

In [15]:
```
'''
  .hist() -- Creates a histogram chart with a dataframe's Series
  Histograms place a metric in bins -- dates in this case -- to understand the distribution of the data. In this case, the distribution of the
'''
df.headline_word_length.hist(
  figsize=(12,6),
  color='#86bf91',
)
```

Out[15]: <Axes: >



In [16]:
```
df.headline_char_length.describe()
```

Out[16]:
```
count    124989.000000
mean         60.023194
std          17.274685
min           0.000000
25%          49.000000
50%          62.000000
75%          71.000000
max         320.000000
Name: headline_char_length, dtype: float64
```

In [17]:
```
df.headline_word_length.describe()
```

Out[17]:
```
count    124989.000000
mean          9.863868
std           2.886560
min           1.000000
25%           8.000000
50%          10.000000
75%          12.000000
max          44.000000
Name: headline_word_length, dtype: float64
```

2.3.1 Exercise -- Observations about the headlines column

- ***ENTER YOUR FIRST OBSERVATION HERE***
  - Question: How might this impact our model's output?
  - ***YOUR RESPONSE HERE***
- ***ENTER YOUR SECOND OBSERVATION HERE***
  - Question: How might this impact our model's output?
  - ***YOUR RESPONSE HERE***
- ***ENTER MORE OBSERVATIONS***
  - Question: How might this impact our model's output?
  - ***YOUR RESPONSE HERE***

2.4 What are the range and distribution of dates?

Since we have a `date` column, and the datatype is in a standardized format, we can quickly plot the date range in a histogram figure.

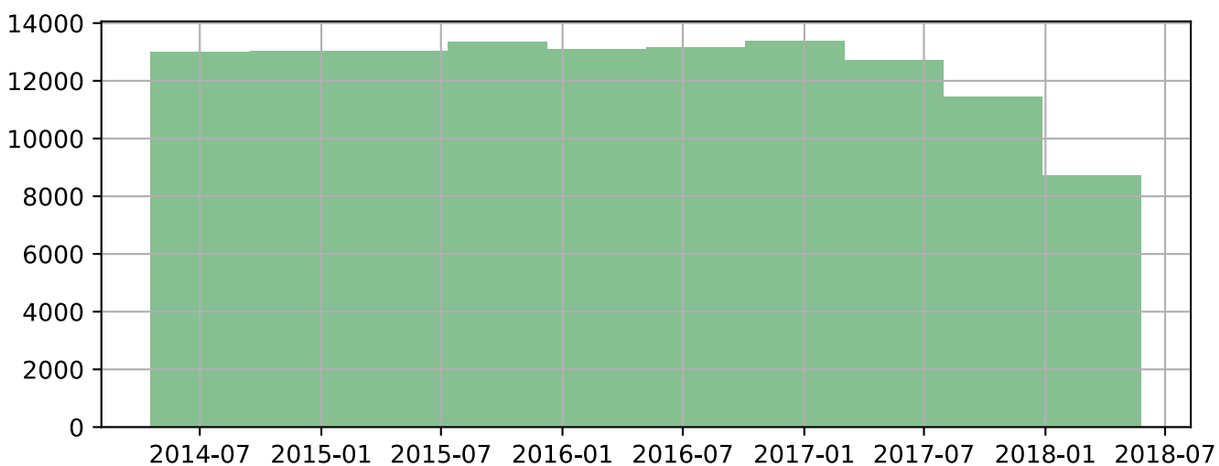
Let's jot down some observations in our notebook.

NOTE: Be sure to respond to the questions and add at least one more potential question, observation, and potential explanation for it

2.4.1 *EXERCISE* -- Notes on the distribution of the data based on the `date` column

In [18]:
```
'''
  .hist() -- Creates a histogram chart with a dataframe's Series
  Histograms place a metric in bins -- dates in this case -- to understand the distribution of the data. In this case, the distribution of the
'''
df.date.hist(
  figsize=(8,3),
  color='#86bf91',
)
```

Out[18]: <Axes: >



- Articles' publishing dates range between July 2014 and July 2018
  - Question: How might this impact the model's output?
  - ***YOUR RESPONSE HERE***
- Fewer 2018 articles compared to the rest of the dataset.
  - Question: How might this impact the model's output?

- ***YOUR RESPONSE HERE***

2.7 Describe and examine the `category` column of news genres in the data

Use *dot notation* and a *column name* with `sample()` to sample values in that column (Series).

Remember: The dataframe stays the same because we are not altering `df`.

In [19]:
```
df.category.sample(5)
```

Out[19]:
```
42740          POLITICS
56263     THE WORLDPOST
104391    ENTERTAINMENT
24967             CRIME
121039            FIFTY
Name: category, dtype: object
```

Now, let's read and understand the news genre categories (politics, entertainment, etc.) in the dataset, as well as their distribution.

We can use the `len()` and `set()` functions in Python to isolate the unique number of values in a column.

`len()` by itself would provide a length of *all values combined*, even if the value is repeated. By using `set()` first on the column, we can tell Python to reduce the column to unique set of values. Then, it will count that unique set with `len()`.

In [20]:
```
len(
  set(df['category'].values)
)
```

Out[20]:
```
31
```

In [21]:
```
set(
  df['category'].values
)
```

Out[21]:
```
{'ARTS',
 'ARTS & CULTURE',
 'BLACK VOICES',
 'BUSINESS',
 'COLLEGE',
 'COMEDY',
 'CRIME',
 'EDUCATION',
 'ENTERTAINMENT',
 'FIFTY',
 'GOOD NEWS',
 'GREEN',
 'HEALTHY LIVING',
 'IMPACT',
 'LATINO VOICES',
 'MEDIA',
 'PARENTS',
 'POLITICS',
 'QUEER VOICES',
 'RELIGION',
 'SCIENCE',
 'SPORTS',
 'STYLE',
 'TASTE',
 'TECH',
 'THE WORLDPOST',
 'TRAVEL',
 'WEIRD NEWS',
 'WOMEN',
 'WORLD NEWS',
 'WORLDPOST'}
```

2.7.1 `category` by count

Let's review the distribution of categories, since this could impact the model.

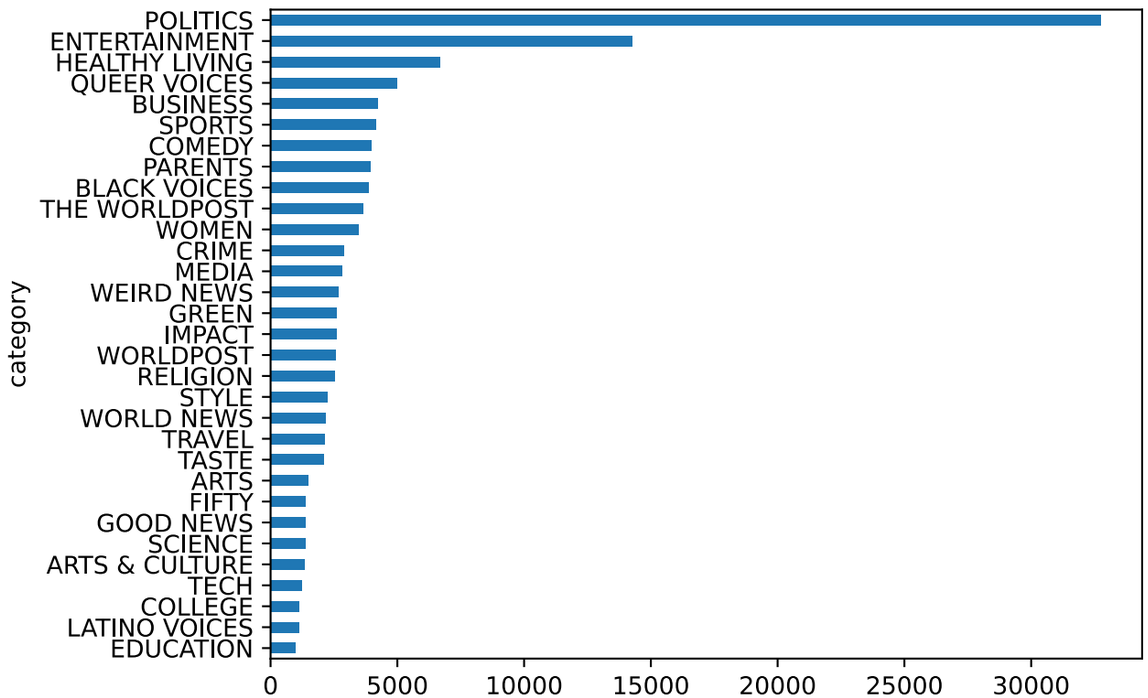
In the code below, we tell Python to:

1. Select the `category` column,
2. Count the values of each value in the column with `value_counts()`,

3. Plot the results with `plot` and provide the value `bar` for its parameter `kind`

In [22]:
```
df['category'].value_counts().sort_values(ascending=True).plot(
  kind='barh'
)
```

Out[22]: <Axes: ylabel='category'>



2.7.2 *EXERCISE* -- Notes on the distribution of the data based on the `category` column

- ***ENTER YOUR FIRST OBSERVATION HERE***
  - Question: How might this impact our model's output?
  - ***YOUR RESPONSE HERE***
- ***ENTER YOUR SECOND OBSERVATION HERE***
  - Question: How might this impact our model's output?

    - ***YOUR RESPONSE HERE***
- ***ENTER MORE OBSERVATIONS***
    - Question: How might this impact our model's output?
    - ***YOUR RESPONSE HERE***

2.8 Describe and examine the `short_description` column

In [23]:
```
df.short_description.describe()
```

Out[23]:
```
count     124989
unique    103905
top
freq       19590
Name: short_description, dtype: object
```

In [24]:
```
df['sd_char_length'] = df.short_description.apply(len)
df[['short_description','sd_char_length']].sample(5)
```

Out[24]:

	short_description	sd_char_length
91240	In major cities, car-hailing apps like Uber, L...	295
83273	The word "alien" will no longer be used to ref...	103
105089	Andre Allen (Rock) is a hip New York-based sta...	201
41362	Trump’s unsubstantiated claims renew fundament...	115
108430		0

In [25]:
```
df['sd_word_length'] = df.short_description.apply(lambda sd: len(sd.split(' ')) )
df.head()
```

Out[25]:

	short_description	headline	date	link	authors	category	headline_char_length	headline_word_length	sd_char_length	sd_
0	She left her husband. He killed their children...	There Were 2 Mass Shootings In Texas Last Week...	2018-05-26	https://www.huffingtonpost.com/entry/texas-ama...	Melissa Jeltsen	CRIME	64	14	76	
1	Of course it has a song.	Will Smith Joins Diplo And Nicky Jam For The 2...	2018-05-26	https://www.huffingtonpost.com/entry/will-smit...	Andy McDonald	ENTERTAINMENT	75	14	24	
2	The actor and his longtime girlfriend Anna Ebe...	Hugh Grant Marries For The First Time At Age 57	2018-05-26	https://www.huffingtonpost.com/entry/hugh-gran...	Ron Dicker	ENTERTAINMENT	47	10	87	
3	The actor gives Dems an ass-kicking for not fi...	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	2018-05-26	https://www.huffingtonpost.com/entry/jim-carre...	Ron Dicker	ENTERTAINMENT	69	11	86	
4	The "Dietland" actress said using the bags is ...	Julianna Margulies Uses Donald Trump Poop Bags...	2018-05-26	https://www.huffingtonpost.com/entry/julianna-...	Ron Dicker	ENTERTAINMENT	71	13	87	

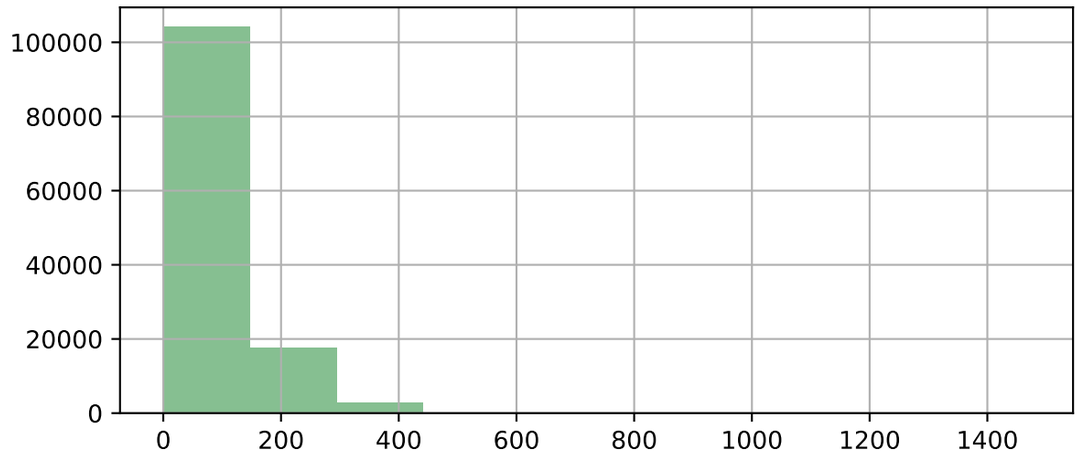
In [26]:
```
# Print out a short comparison report
print(
  '# Short Desc CHAR Length',
  '\n', df.sd_char_length.describe(),
```

```
  '\n\n# Short Desc WORD Length',
  '\n', df.sd_word_length.describe(),
)
```

```
# Short Desc CHAR Length
 count    124989.000000
mean         92.415373
std          84.832972
min           0.000000
25%          33.000000
50%          76.000000
75%         123.000000
max        1472.000000
Name: sd_char_length, dtype: float64

# Short Desc WORD Length
 count    124989.000000
mean         15.935050
std          14.447419
min           1.000000
25%           6.000000
50%          13.000000
75%          21.000000
max         243.000000
Name: sd_word_length, dtype: float64
```

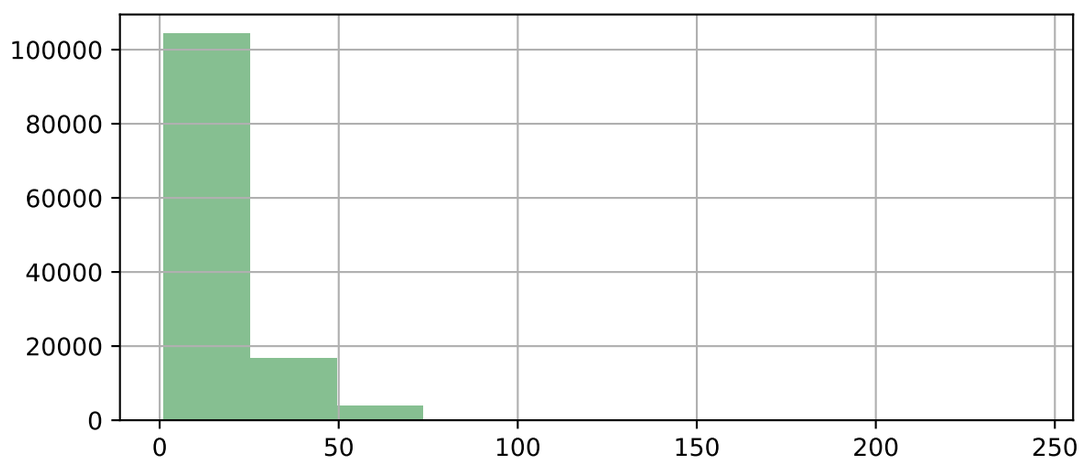
In [27]:
```
df.sd_char_length.hist(
  figsize=(7,3),
  color='#86bf91',
)
```

Out[27]: <Axes: >



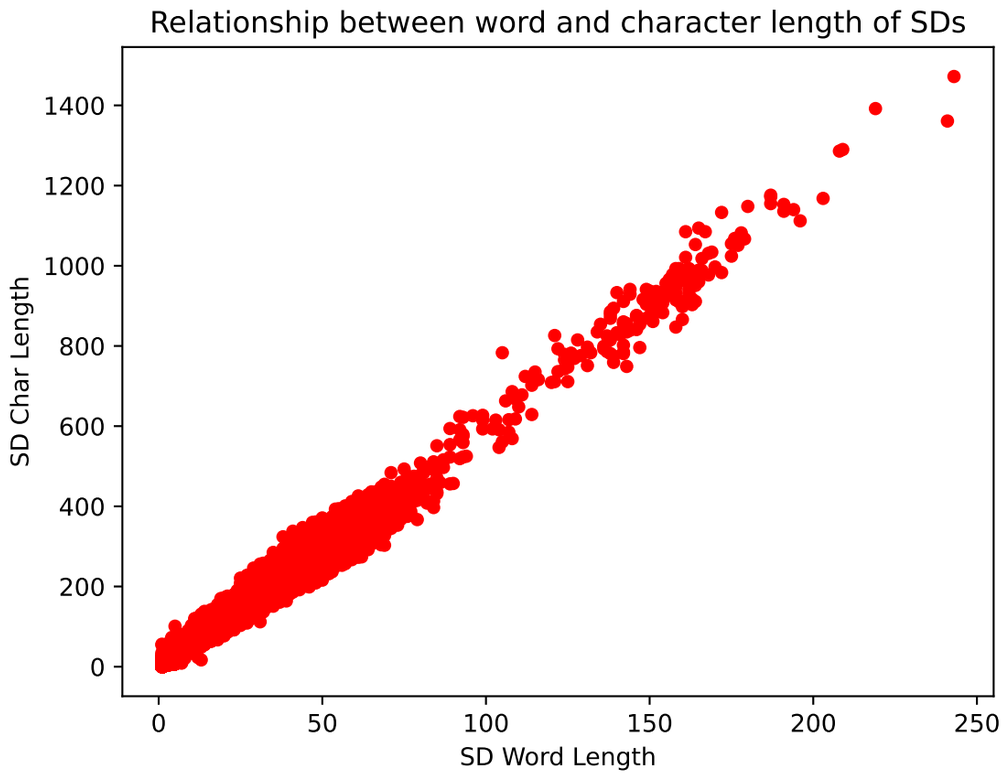
In [28]:
```
df.sd_word_length.hist(
  figsize=(7,3),
  color='#86bf91',
)
```

Out[28]: <Axes: >



In [29]:
```
# Scatter plot using pandas
ax = df.plot(
```

```
  kind='scatter',
  x='sd_word_length',
  y='sd_char_length',
  color='red',
  title='Relationship between word and character length of SDs'
)

# Customizing plot elements
ax.set_xlabel("SD Word Length")
ax.set_ylabel("SD Char Length")

plt.show()
```



2.8.1 *EXERCISE* -- Notes on the short description columns

- ***ENTER YOUR FIRST OBSERVATION HERE***
  - Question: How might this impact our model's output?
  - ***YOUR RESPONSE HERE***
- ***ENTER YOUR SECOND OBSERVATION HERE***
  - Question: How might this impact our model's output?
  - ***YOUR RESPONSE HERE***
- ***ENTER MORE OBSERVATIONS***
  - Question: How might this impact our model's output?
  - ***YOUR RESPONSE HERE***

3. Process the Data for Classification

Some of the fields will be useful to use for the classification task.

1. `headline`: Since we're writing a model to classify headlines, this column is integral to our goal.
2. `short_description`: This column includes a short descrtiption of the story, which will provide potentially helpful contextual information to make a better model for our genre classifier. We'll see!
3. `tokenized_url`: This column includes information too! Different parts of URLs often capture how people have organized and classified information. So, the URL may also provide potentially helpful contextual information to make a better model. Again, we'll see!

The code below creates 3 new columns for that task.

There's a lot to unpack below, but it basically

1. normalizes the URL for each article via the `tokenize_url()` function, which strips the URL down to the portion where the URL conveys article-specific information. Then,
2. it creates 3 new columns that are combined with the available data.

In [30]:
```
def tokenize_url(url:str):
    url=url.replace("https://www.huffingtonpost.com/entry/","")
    url=re.sub("(\W|_)+"," ",url)
    return url

df['tokenized_url']=df['link'].apply(lambda x:tokenize_url(x))
```

```
#just the description
df['text_desc'] = df['short_description']

#description + headline
df['text_desc_headline'] = df['short_description'] + ' '+ df['headline']

#description + headline + tokenized url
df['text_desc_headline_url'] = df['short_description'] + ' '+ df['headline']+" " + df['tokenized_url']

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 124989 entries, 0 to 124988
Data columns (total 14 columns):
 #   Column                  Non-Null Count   Dtype         
---  ------                  --------------   -----         
 0   short_description       124989 non-null  object        
 1   headline                124989 non-null  object        
 2   date                    124989 non-null  datetime64[ns]
 3   link                    124989 non-null  object        
 4   authors                 124989 non-null  object        
 5   category                124989 non-null  object        
 6   headline_char_length    124989 non-null  int64         
 7   headline_word_length    124989 non-null  int64         
 8   sd_char_length          124989 non-null  int64         
 9   sd_word_length          124989 non-null  int64         
 10  tokenized_url           124989 non-null  object        
 11  text_desc               124989 non-null  object        
 12  text_desc_headline      124989 non-null  object        
 13  text_desc_headline_url  124989 non-null  object        
dtypes: datetime64[ns](1), int64(4), object(9)
memory usage: 13.4+ MB
```

3.1 EXERCISE -- Notes on the on the `text_desc_headline_url` column

Let's sample the new row with the URL info to see what it includes.

In [31]:
```
# Randomly .sample() some of the text_desc_headline_url column
# Execute multiple times to get different random sample (n)
df['text_desc_headline_url'].sample(5).values.tolist()
```

Out[31]:
```
['"Unreported rapes count. Reported rapes count. End of story." Abigail Breslin Had A Powerful Response To Commenter Who Said Only Reported Rapes \'Count\' abigail breslin just revealed why she never reported her sexual assaulted us 58fdf901e4b018a9ce5cd0ae',
 'He entered the race in June. Rick Perry Ends His 2016 Presidential Campaign rick perry ends presidential campaign us 55f3427ee4b063ecbfa46be7',
 'By Terry Gaspard MSW, LICSW While many couples see remarriage as a second chance at happiness, the statistics tell a different 10 Rules For A Successful Second Marriage 10 rules for a successful second marriage us 58011b00e4b06f314afeb31b',
 'WASHINGTON (AP) — American employers added a strong 292,000 jobs in December, suggesting that the U.S. economy is so far U.S. Economy Adds 292,000 Jobs In December, Unemployment Steady At 5% obama jobs report us 568fbbfbe4b0cad15e64580c',
 'By reducing immigration levels, the government is cutting off its nose to spite its face. Restricting Legal Immigration Only Harms -- Not Protects -- American Workers restricting legal immigration only harms not protects us 58aa4216e4b0b0e1e0e20d00']
```

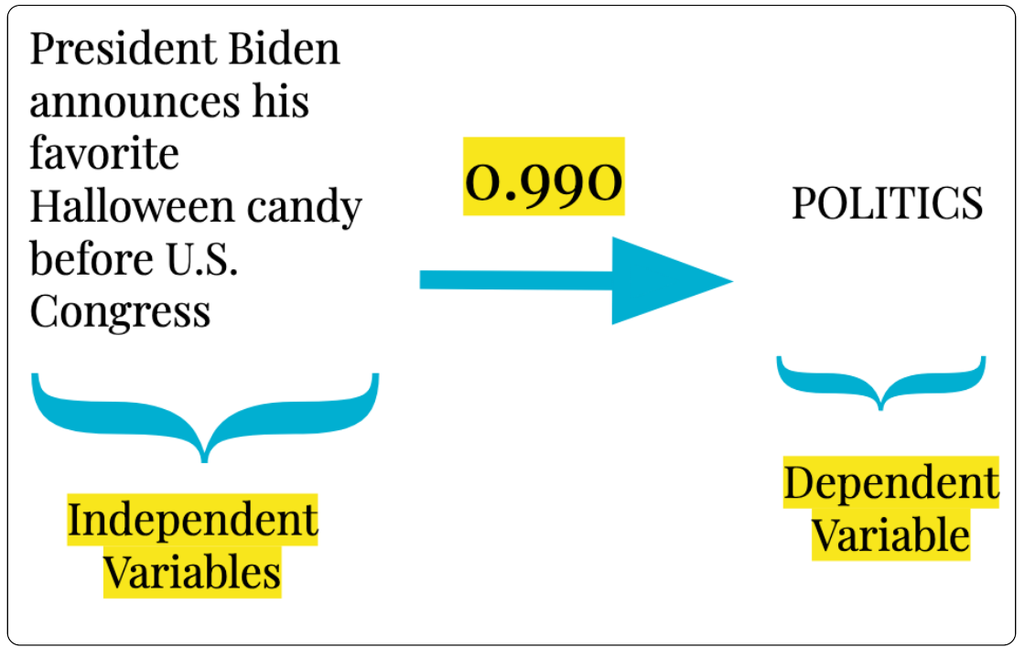
- ***ENTER YOUR OBSERVATION HERE***
  - Question: How might this impact our model's output?
  - ***YOUR RESPONSE HERE***
- ***ENTER MORE OBSERVATIONS***
  - Question: How might this impact our model's output?
  - ***YOUR RESPONSE HERE***

4. Train a Logistic Regression Model

4.1 What's a Logistic Regression Model?

Logistic regression returns a probability that something may or may not happen or "be". We can use this probability as a single value to assess the likelihood of the event/being, such as "This X headline is of Y news genre".

- **Binary Classification Thresholds**:
  Uses binary (dichotomous or discrete) data to predict probability of binary decision: Yes or No? Typically, this is called a *classification threshold*.



Original image src: Lindgren

- **"Predictors"**:
  1 or more independent variables: features, i.e., words from a corpus of headlines
- **"Criterion"**:
  Variables used to determine a binary decision about the output (dependent variable (news genre of article)
- **Computation Method**:
  Uses probability, or likelihood, scores to output predictions based on the available features in the input as compared with the trained models features

New genres present us with an interesting case, where inputs may interact with the features scored in the trained model in interesting ways.

In this lesson, our LR model uses those probability estimates as a binary category. To do so, we must decide what's called a "classification threshold" or "decision threshold". Any value above that threshold indicates a headline is about POLITICS, and any value below the threshold indicates that the headline is not POLITICS, but some other news genre.

So, let's say our LR model returns a value of 0.990 for a particular headline's news genre as being "POLITICS". This probability score is very likely to accurately predict that this headline is indeed an article about POLITICS. Conversely, another headline with a prediction score of 0.005 on that same logistic regression model is very likely not about POLITICS. Yet, what about a headline with a prediction score of 0.6?

The default decision threshold in the scikit-learn code library that we will use is 0.5. But, this library also enables us to "tune" the LR model based on our problem-dependency / context, as well as take the best/top probability score to predict the news genre of the input headline.

4.2 How to Train a Drago... Erm ... a LR Model!

4.2.1 *From Words to Numbers!* - Transform a corpus of words into matrix of integers with `CountVectorizer()`

Before you review the many functions defined in section 4.2 below, functions that will automate the modeling process, I want you to understand a very important first step: how we get from word label data to numbers for the model. After all, this is "Writing & Digital Media" **:-)**

So, first things first, we're working with text, so strings and characters. But, a logistic regression model requires numbers. So, we need to extract the features of our corpus and transform them into a data type that can be quantified.

`CountVectorizer()` to the rescue!

To better understand how scikit-learn's built-in functions transform the textual data for us, I provide us with a tiny tiny case of a list of sentences, which transforms our textural data into a **sparse matrix** of values that indicate the frequency of times the word (feature/token) is present in a unit of content (document).

In this tiny case below, our documents are pretty short strings in a list. Those could be much much bigger, and really it should be for this type of modeling. But, for example purposes, let's see how the matrix logs how many instances a word shows up in each document.

Each row is the document, (i.e., sentence, in this case), while each column is the feature (word/token). So the values at the intersection of the row and column in the matrix report the raw frequency count (integer).

In [32]:
```
# Tiny weeny bitty small corpus :-)
# 3 documents (strings) in a list
list_test_text_1 = [
  'Hello my name is chris we are here to do python python python python',
  'chris this is my python notebook you know python',
  'The python notebook is a chore chris no chris really it is a chore chore chore'
]

# Instantiate an sklearn CountVectorizer (CV) objects
test_cv_object_binary = CountVectorizer(binary=True) #binary values
test_cv_object_counts = CountVectorizer(binary=False) #regular frequency counts
# Instantiate an sklearn TfidfVectorizer objects
test_tfidf_vectorizer_normed = TfidfVectorizer() # normed tfidf scores
test_tfidf_vectorizer = TfidfVectorizer(norm=None, smooth_idf=False) #not normed tfidf scores

# Use the vectorizers' fit_transform() function to conduct the transformation of words into particular quantified values
test_count_matrix_binary = test_cv_object_binary.fit_transform(list_test_text_1)
test_count_matrix_counts = test_cv_object_counts.fit_transform(list_test_text_1)
test_tfidf_matrix_normed = test_tfidf_vectorizer_normed.fit_transform(list_test_text_1)
test_tfidf_matrix = test_tfidf_vectorizer.fit_transform(list_test_text_1)
```

In [33]:
```
# Note the specialized data type for sklearn
print(
  'Binary Data Type:',type(test_count_matrix_binary),
  '\nCounts Data Type::',type(test_count_matrix_counts),
  '\nTFIDF Data Type:',type(test_tfidf_matrix),
)
```

```
Binary Data Type: <class 'scipy.sparse._csr.csr_matrix'>
Counts Data Type:: <class 'scipy.sparse._csr.csr_matrix'>
TFIDF Data Type: <class 'scipy.sparse._csr.csr_matrix'>
```

If I try to print out the sklearn cv matrix, I'll get some basic data structural info.

In [34]:
```
print(test_count_matrix_binary.get_shape())
```

```
(3, 20)
```

The matrix has 3 rows (documents/sentences) and 20 columns (unique words/tokens) with values that are dependent on the type of summary: binary, counts, or tfidf.

Let's check out the now transformed/quantified data by converting the returned sparse matrices as a more recognizable data type: a pandas dataframe.

In [35]:
```
def convert_sparse_matrix_to_df(matrix, vectorized_object):
  # Convert the matrix into a more basic Python array
  count_array = matrix.toarray()

  # Use that array as the data in the DF, then nicely use CV or TFIDF vector object's `get_feature_names_out()` method to provide the column na
  sparse_matrix_as_df = pd.DataFrame(
    data=count_array,
    columns=vectorized_object.get_feature_names_out()
  )

  return sparse_matrix_as_df
```

In [36]:
```
# Binary
convert_sparse_matrix_to_df(test_count_matrix_binary, test_cv_object_binary)
```

Out[36]:

	are	chore	chris	do	hello	here	is	it	know	my	name	no	notebook	python	really	the	this	to	we	you
0	1	0	1	1	1	1	1	0	0	1	1	0	0	1	0	0	0	1	1	0
1	0	0	1	0	0	0	1	0	1	1	0	0	1	1	0	0	1	0	0	1
2	0	1	1	0	0	0	1	1	0	0	0	1	1	1	1	1	0	0	0	0

In [37]:
```
# Counts
convert_sparse_matrix_to_df(test_count_matrix_counts, test_cv_object_counts)
```

Out[37]:

	are	chore	chris	do	hello	here	is	it	know	my	name	no	notebook	python	really	the	this	to	we	you
0	1	0	1	1	1	1	1	0	0	1	1	0	0	4	0	0	0	1	1	0
1	0	0	1	0	0	0	1	0	1	1	0	0	1	2	0	0	1	0	0	1
2	0	4	2	0	0	0	2	1	0	0	0	1	1	1	1	1	0	0	0	0

In [38]:
```
# Not Normed TF-IDF
convert_sparse_matrix_to_df(test_tfidf_matrix, test_tfidf_vectorizer)
```

Out[38]:

	are	chore	chris	do	hello	here	is	it	know	my	name	no	notebook	python	really	the	
0	2.098612	0.000000	1.0	2.098612	2.098612	2.098612	1.0	0.000000	0.000000	1.405465	2.098612	0.000000	0.000000	4.0	0.000000	0.000000	0.0
1	0.000000	0.000000	1.0	0.000000	0.000000	0.000000	1.0	0.000000	2.098612	1.405465	0.000000	0.000000	1.405465	2.0	0.000000	0.000000	2.0
2	0.000000	8.394449	2.0	0.000000	0.000000	0.000000	2.0	2.098612	0.000000	0.000000	0.000000	2.098612	1.405465	1.0	2.098612	2.098612	0.0

In [39]:
```
# Normed TF-IDF
convert_sparse_matrix_to_df(test_tfidf_matrix_normed, test_tfidf_vectorizer_normed)
```

Out[39]:

	are	chore	chris	do	hello	here	is	it	know	my	name	no	notebook	python	really	
0	0.268634	0.000000	0.158660	0.268634	0.268634	0.268634	0.158660	0.000000	0.000000	0.204303	0.268634	0.000000	0.000000	0.634638	0.000000	0
1	0.000000	0.000000	0.236251	0.000000	0.000000	0.000000	0.236251	0.000000	0.400008	0.304216	0.000000	0.000000	0.304216	0.472502	0.000000	0
2	0.000000	0.821339	0.242548	0.000000	0.000000	0.000000	0.242548	0.205335	0.000000	0.000000	0.000000	0.205335	0.156162	0.121274	0.205335	0

What do we notice?

All dataframe/matrices include:

- 3 rows == number of documents
- 20 columns == Set number of terms/tokens
- **Binary** values are either 0 or 1
- **Counts** values are the *frequency* of times a word shows up in a document
- **TFIDF** values are a type of *weighted average* score per word in a document

So what is TFIDF, again?

TF-IDF measures how important a *term* is within a *document* relative to a *collection of documents/corpus*. It is comprised of 2 values: *Term Frequency/commonality* & *Inverse Document Frequency/rarity*. In our test simple case, we have 3 documents (3 lists of strings), wherein the total number of words/tokens across all docs is 36.

A term's TF-IDF score is high when it is frequently used in a document **and rarely** in other documents in the collection. Conversely, a term's TF-IDF score is lowered when it is just as frequently a document **and frequently** in other documents in the collection. Consequently, a term's commonality within a document (TF) is balanced by the term's rarity across all documents (IDF), which provides one useful score to measure the importance of a term for a document in the corpus.

Term Frequency

A term's *commonality* -- Number of times the term appears in a document compared to the total number of words in the document. -- TF = Frequency of term in a document / Total number of tokens (39)

$$TF = \frac{\text{number of times the term appears in the document}}{\text{total number of terms in the document}}$$

In [40]:
```
test_total_terms_in_collection = len( (list_test_text_1[0] +' '+ list_test_text_1[1] +' '+ list_test_text_1[2]).split() )
test_total_terms_in_collection
```

Out[40]: 39

In [41]:
```
# Term Frequency (TF) for 'python' in second document, i.e., string in the list
tf_python_d1 = 4 / test_total_terms_in_collection
tf_chore_d3 = 4 / test_total_terms_in_collection
print(
  'TF \'python\':', tf_python_d1,
```

```
  '\nTF \'chore\':', tf_chore_d3,
)
```

```
TF 'python': 0.10256410256410256
TF 'chore': 0.10256410256410256
```

Inverse Document Frequency: A term's *rarity* -- IDF of a term reflects the proportion of documents in the corpus that contain the term. Words unique to a small percentage of documents (e.g., technical jargon terms) receive higher importance values than words common across all documents (e.g., a, the, and).

$$IDF = log(\frac{\text{number of the documents in the corpus}}{\text{number of documents in the corpus contain the term}})$$

In [42]:
```
idf_d1_python = np.log10(3 / 3)
idf_d3_chore = np.log10(3 / 1)
print(
  'IDF \'python\':', idf_d1_python,
  '\nIDF \'chore\':', idf_d3_chore,
)
```

```
IDF 'python': 0.0
IDF 'chore': 0.47712125471966244
```

In [43]:
```
print(
  'TF-IDF of \'python\' in doc 1 of the collection/corpus:', (tf_python_d1 * idf_d1_python),
  '\nTF-IDF of \'chore\' in doc 3 of the collection/corpus:', (tf_chore_d3 * idf_d3_chore),
)
```

```
TF-IDF of 'python' in doc 1 of the collection/corpus: 0.0
TF-IDF of 'chore' in doc 3 of the collection/corpus: 0.04893551330458076
```

Summarize `CountVectorizer()` & `TfidfVectorizer()`

In the modeling functions below, the `extract_features()` function uses `CountVectorizer()` and `TfidfVectorizer()` to, well, extract these features from the much larger corpus.

They both nicely automate the process for us: writing and calculating at amazing speed and scale.

Can you imagine transforming the data manually?! I don't think we want to subject ourselves to even such a thought.

4.3 Modeling functions

Be sure to review the model training functions in the utils.py file, under "Model Training Functions."

4.4 LR Model 1 - Binary or Count features with `text_desc` only

Train the model based on the short description data.

FYI. Your functions can return multiple variables, if you'd like.

If you review the `train_model()` function, it returns multiple variables in a specific order. Those mirror the variables and the comma-separated variables and their desired types below.

4.4.1 Enact the Training

In [44]:
```
'''
  Parameters to configure for our train_model() function
'''

# Use the short description only to train a model
training_field = 'text_desc'
# Specify if this model should use a binary approach to the features (0 or 1) or the actual counts created by CountVectorizer()
# Let's use counts
feature_rep = 'counts'
# Tell the model function to return the top 3 'best fits' among the distributed probabilities
top_k = 3

# Train that supervised ML logistic regression model!
model_td_only,transformer_td_only,accuracy_td_only,mrr_at_k_td_only,X_train,X_test,Y_test,Y_train,preds,eval_items = train_model(
  df, # full corpus
  field=training_field,
  feature_rep=feature_rep,
  top_k=top_k
)
```

```
2025-04-09 08:49:57,377 : INFO : Starting model training...
2025-04-09 08:49:57,397 : INFO : Extracting features and creating vocabulary...
2025-04-09 08:49:58,753 : INFO : Training a Logistic Regression Model. This may take a few minutes. ...
2025-04-09 08:52:19,144 : INFO : Starting evaluation...
2025-04-09 08:52:19,204 : INFO : Done training and evaluation.
```

Model Assessment Roadmap

In the following subsections, you will examine the LR model that you just trained by:

1. Print out the overall accuracy scores for the model
2. Compare the predictive power of the model across the news genre categories, i.e., *classes*
3. Visualize the predictive power per class/category with a "confusion matrix" heatmap
4. Dig deeper into the categories by comparing some categories' performance against the EDA work on the data set used to train the model

4.4.2 Test the accuracy/performance of the model

4.4.2.1 See the accuracy and Mean Reciprocal Rank Scores

We already computed the overall accuracy and MRR scores, when we trained the model, so let's output them.

If you need to refresh yourself on these scores, check out the functions' documentation in the 4.3 section. But, here's the gist:

The **Mean Reciprocal Rank** evaluates any process that produces a list of possible *ranked* responses to a sample of queries, ordered by probability of correctness, e.g., 1 for first place, ½ for second place, ⅓ for third place and so on. (See table below for examples.)

Query	Proposed Results	Correct response	Rank	Reciprocal rank
cat	catten, cati, **cats**	cats	3	1/3
torus	torii, **tori**, toruses	tori	2	1/2
virus	**viruses**, virii, viri	viruses	1	1

(table src: wikipedia)

In [45]:
```
print(
  f"Overall Mean Average Model Accuracy = {accuracy_td_only}\nMean Reciprocal Rank = {mrr_at_k_td_only}"
)
```

```
Overall Mean Average Model Accuracy = 0.5956541218637993
Mean Reciprocal Rank = 0.4785052910053067
```

4.4.2.2 Compare Actual Labels (ground truths) with Predicted Labels (with the k threshold)

Before we test our model with headline inputs, we can test its performance with scikit-learn's `predict_proba()` associated function with the output model.

This function returns the estmated probability of each categorical label (news genre) in the test sample. With this returned data set, we can visualize the results to see what categories might be mislabeled more often than others and, of course, which categories are performing well.

Here is a description of parameter and return data:

- `X_test` : sample of the original data to test the trained model
- `Y_probability_td_only` : List of the predicted news genre category outputs (String)
- `Y_probability_a_td_only` : List of lists of the predicted news genre category based on the estimated probability score
  - First list == 1 row from X_test data set
    - Lists within that list == Each list contains estimated probability scores (Float) per (31) news genre categories

In [46]:
```
# Predict the classes on the test data
Y_predictions_td_only = model_td_only.predict(X_test)
# Predict the classes on the test data, and return the probabilities for each class
Y_probability_a_td_only = model_td_only.predict_proba(X_test)
```

4.4.3 Visualize Actual vs. Prediction with a Confusion Matrix

A confusion matrix helps you organize a direct comparison across all of your possible categories. Essentially, it asks:

1. How many times did X (actual) equal the predicted (Y)?
2. How many times did X (actual) not equal the predicted (Y)?

Another way to phrase the questions is to think of it in action. So, with this data set, there are 31 possible categories (news genres), which means we have 31 categories to test against each other—**that's 961 categories** (31 * 31 = 961)!

A confusion matrix maps what are called the True Positive, False Positive, False Negative, and True Negative prediction outcomes. In the table below, I provide a high-level description of what each of these values represent in the scheme of our LR model on predicting news genres.

True Positive (TP):	**False Positive (FP):**
• Reality: Input SCIENCE headline • LR predicted: "SCIENCE" • Outcome: LR's prediction is correct/accurate	• Reality: Input is **NOT** a SCIENCE headline • LR predicted: "SCIENCE" • Outcome: LR's prediction is incorrect/inaccurate

False Negative (FN):

- Reality: Input SCIENCE headline
- LR predicted: **NOT** "SCIENCE"
- Outcome: LR's prediction is incorrect/inaccurate

True Negative (TN):

- Reality: Input is **NOT** a SCIENCE headline
- LR predicted: **NOT** "SCIENCE"
- Outcome: LR's prediction is correct/accurate

Ok, so let's visualize the TPs, FPs, FNs, and TNs as a heatmap with annotated values for quick reference.

In [47]:
```
from sklearn.metrics import precision_recall_fscore_support,confusion_matrix
from sklearn.preprocessing import normalize

import seaborn as sns
import mplcyberpunk
```

In [48]:
```
cm_td_only = confusion_matrix(
  Y_test, #Sorted List of ground truth (correct/actual) target values
  Y_predictions_td_only #Sorted List of estimated targets as returned by a classifier
)
```

Let's be sure to **normalize the confusion matrix values**.

The current values in the confusion matrix are simply counts (Integers). But, if we want to visualize the prediction efficacy as a heatmap, we first need to "normalize" the values by converting them to a score relative to all of the other comparisons in the same column. For example, `ARTS` has results across 31 categories

In [49]:
```
# Normalize matrix by columns
cm_td_only_normed_by_column = normalize(cm_td_only, axis=1, norm='l1')
# Compare the original to the normalized
print(
  'Simple Counts of the ARTS (first) column:\n', cm_td_only[0],'\n\n',
  'Normalized Counts of the ARTS (first) column:\n',cm_td_only_normed_by_column[0],
)
```

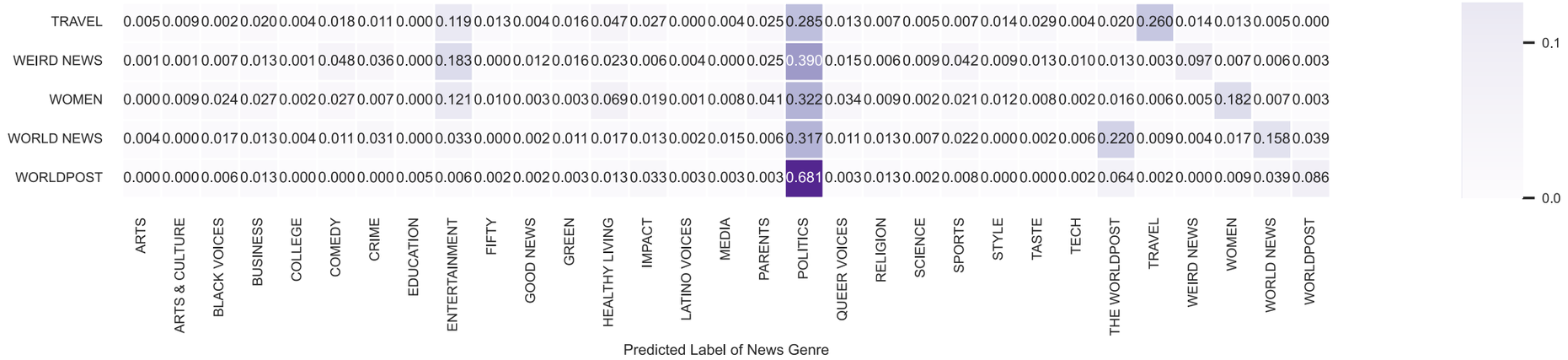
```
Simple Counts of the ARTS (first) column:
 [ 39  18   6   7   3   7   0   0  46   1   0   0   6   3   4   5   7 182
   8   5   1   2   2   3   2   0   3   0   5   0   5]

 Normalized Counts of the ARTS (first) column:
 [0.10540541 0.04864865 0.01621622 0.01891892 0.00810811 0.01891892
 0.         0.         0.12432432 0.0027027  0.         0.
 0.01621622 0.00810811 0.01081081 0.01351351 0.01891892 0.49189189
 0.02162162 0.01351351 0.0027027  0.00540541 0.00540541 0.00810811
 0.00540541 0.         0.00810811 0.         0.01351351 0.
 0.01351351]
```

4.4.3.1 Heatmap of the `text_desc` only normalized confusion matrix

In [71]:
```
plt.figure(figsize=(18,10))

sns.set_theme(
  font_scale=0.6,
)
sns.color_palette("viridis", as_cmap=True)

sns.heatmap(
  data=cm_td_only_normed_by_column, #normalized confusion matrix
  annot=True, # add normalized counts of co-occurrences between actual vs. predicted
  fmt=".3f", # round to thousandths decimal place
  linewidths=1, # style choice for row/column lines
  square=True, # make the
  cmap='Purples',
  xticklabels=model_td_only.classes_,
  yticklabels=model_td_only.classes_,
)

# Label the X and Y axes
plt.ylabel('Actual Label of News Genre')
plt.xlabel('Predicted Label of News Genre')

# Let's plug in our overall accuracy measures into the the title
all_sample_title = f"Short Description Count Model\nOverall Accuracy Score: {accuracy_td_only}\nMean Reciprocal Rank: {mrr_at_k_td_only}"
plt.title(
  all_sample_title,
  size=12
```

```
)

# Ok, output time!
plt.tight_layout()
mplcyberpunk.add_glow_effects()
plt.show()
```


Short Description Count Model
Overall Accuracy Score: 0.5956541218637993
Mean Reciprocal Rank: 0.4785052910053067
Actual Label of News Genre
ARTS 0.105 0.049 0.016 0.019 0.008 0.019 0.000 0.000 0.124 0.003 0.000 0.000 0.016 0.008 0.011 0.014 0.019 0.492 0.022 0.014 0.003 0.005 0.005 0.008 0.005 0.000 0.008 0.000 0.014 0.000 0.014
ARTS & CULTURE 0.032 0.069 0.040 0.017 0.000 0.038 0.003 0.000 0.289 0.000 0.000 0.014 0.043 0.006 0.006 0.009 0.026 0.202 0.049 0.009 0.009 0.014 0.009 0.009 0.012 0.009 0.000 0.040 0.038 0.009 0.000
BLACK VOICES 0.004 0.006 0.195 0.013 0.010 0.011 0.030 0.007 0.188 0.002 0.004 0.004 0.013 0.012 0.004 0.011 0.016 0.364 0.021 0.011 0.001 0.018 0.004 0.003 0.000 0.007 0.002 0.005 0.021 0.009 0.001
BUSINESS 0.000 0.003 0.006 0.286 0.006 0.008 0.007 0.001 0.042 0.002 0.000 0.014 0.051 0.016 0.001 0.015 0.017 0.422 0.014 0.003 0.001 0.009 0.005 0.006 0.018 0.011 0.003 0.005 0.020 0.003 0.007
COLLEGE 0.000 0.000 0.022 0.040 0.159 0.000 0.014 0.051 0.051 0.000 0.004 0.000 0.058 0.022 0.000 0.004 0.036 0.417 0.018 0.014 0.004 0.018 0.004 0.004 0.000 0.014 0.000 0.004 0.036 0.004 0.004
COMEDY 0.001 0.003 0.011 0.006 0.001 0.090 0.004 0.002 0.212 0.002 0.006 0.004 0.028 0.004 0.002 0.014 0.017 0.466 0.008 0.005 0.005 0.027 0.011 0.018 0.004 0.002 0.006 0.021 0.015 0.001 0.002
CRIME 0.001 0.001 0.029 0.007 0.004 0.004 0.243 0.001 0.069 0.000 0.003 0.007 0.020 0.011 0.003 0.003 0.010 0.469 0.014 0.007 0.000 0.013 0.001 0.001 0.001 0.029 0.010 0.023 0.004 0.009 0.000
EDUCATION 0.000 0.004 0.016 0.037 0.053 0.004 0.000 0.276 0.016 0.004 0.004 0.000 0.037 0.021 0.004 0.012 0.033 0.358 0.025 0.008 0.008 0.004 0.000 0.004 0.008 0.004 0.004 0.000 0.025 0.004 0.025
ENTERTAINMENT 0.004 0.005 0.025 0.005 0.002 0.027 0.006 0.000 0.424 0.001 0.005 0.002 0.020 0.003 0.004 0.006 0.016 0.353 0.025 0.003 0.002 0.019 0.010 0.004 0.001 0.002 0.002 0.010 0.010 0.002 0.001
FIFTY 0.007 0.005 0.017 0.024 0.002 0.015 0.002 0.000 0.037 0.105 0.000 0.005 0.093 0.017 0.007 0.002 0.105 0.362 0.037 0.005 0.002 0.005 0.005 0.020 0.007 0.002 0.029 0.012 0.054 0.015 0.000
GOOD NEWS 0.000 0.006 0.006 0.012 0.000 0.027 0.009 0.003 0.169 0.003 0.051 0.024 0.045 0.021 0.000 0.003 0.066 0.453 0.009 0.012 0.000 0.018 0.012 0.006 0.000 0.003 0.003 0.030 0.003 0.003 0.000
GREEN 0.002 0.002 0.009 0.025 0.005 0.012 0.012 0.006 0.035 0.002 0.009 0.205 0.023 0.020 0.000 0.006 0.012 0.473 0.008 0.002 0.025 0.005 0.009 0.005 0.005 0.020 0.014 0.017 0.009 0.012 0.012
HEALTHY LIVING 0.001 0.004 0.008 0.034 0.001 0.011 0.004 0.002 0.065 0.007 0.005 0.010 0.332 0.016 0.002 0.005 0.033 0.345 0.014 0.008 0.007 0.010 0.007 0.014 0.002 0.009 0.005 0.009 0.024 0.002 0.004
IMPACT 0.003 0.006 0.024 0.035 0.005 0.006 0.008 0.014 0.048 0.002 0.011 0.032 0.083 0.143 0.003 0.005 0.040 0.345 0.030 0.011 0.005 0.011 0.008 0.008 0.014 0.029 0.019 0.003 0.030 0.011 0.008
LATINO VOICES 0.000 0.004 0.039 0.025 0.011 0.014 0.004 0.007 0.193 0.011 0.000 0.007 0.011 0.018 0.139 0.007 0.018 0.379 0.029 0.021 0.000 0.007 0.007 0.007 0.004 0.007 0.000 0.000 0.021 0.007 0.004
MEDIA 0.003 0.003 0.011 0.030 0.005 0.019 0.003 0.002 0.086 0.000 0.002 0.000 0.014 0.011 0.002 0.174 0.003 0.534 0.020 0.005 0.005 0.011 0.002 0.005 0.006 0.008 0.003 0.006 0.013 0.014 0.003
PARENTS 0.000 0.002 0.019 0.013 0.002 0.025 0.005 0.007 0.133 0.009 0.014 0.005 0.069 0.023 0.001 0.002 0.280 0.246 0.028 0.007 0.003 0.013 0.007 0.013 0.007 0.007 0.007 0.014 0.033 0.001 0.001
POLITICS 0.001 0.001 0.011 0.013 0.003 0.010 0.008 0.002 0.043 0.000 0.001 0.008 0.016 0.007 0.002 0.010 0.007 0.778 0.011 0.006 0.001 0.007 0.002 0.002 0.003 0.018 0.003 0.004 0.008 0.006 0.005
QUEER VOICES 0.003 0.009 0.015 0.012 0.002 0.014 0.007 0.001 0.138 0.001 0.003 0.002 0.027 0.014 0.002 0.009 0.030 0.320 0.282 0.011 0.002 0.013 0.008 0.004 0.003 0.009 0.007 0.007 0.036 0.007 0.003
RELIGION 0.003 0.004 0.028 0.006 0.004 0.006 0.011 0.003 0.050 0.003 0.003 0.001 0.030 0.017 0.003 0.009 0.021 0.470 0.034 0.205 0.001 0.007 0.004 0.003 0.001 0.017 0.001 0.006 0.016 0.021 0.010
SCIENCE 0.003 0.009 0.015 0.009 0.000 0.034 0.019 0.003 0.062 0.003 0.006 0.046 0.087 0.006 0.000 0.000 0.022 0.471 0.015 0.000 0.121 0.009 0.003 0.006 0.009 0.012 0.003 0.019 0.003 0.000 0.003
SPORTS 0.001 0.003 0.012 0.013 0.005 0.009 0.017 0.003 0.124 0.000 0.002 0.001 0.026 0.007 0.003 0.007 0.008 0.369 0.015 0.000 0.003 0.316 0.008 0.002 0.005 0.007 0.003 0.015 0.010 0.004 0.004
STYLE 0.004 0.005 0.016 0.009 0.000 0.024 0.004 0.000 0.151 0.011 0.002 0.004 0.045 0.007 0.002 0.002 0.018 0.471 0.022 0.005 0.002 0.016 0.131 0.011 0.004 0.002 0.007 0.013 0.013 0.000 0.000
TASTE 0.000 0.006 0.008 0.026 0.002 0.026 0.004 0.000 0.121 0.004 0.006 0.010 0.078 0.004 0.000 0.000 0.036 0.197 0.018 0.008 0.002 0.014 0.014 0.346 0.006 0.002 0.024 0.026 0.004 0.004 0.002
TECH 0.000 0.000 0.026 0.092 0.007 0.046 0.010 0.003 0.114 0.000 0.007 0.003 0.046 0.013 0.003 0.013 0.026 0.297 0.007 0.003 0.007 0.007 0.020 0.020 0.127 0.033 0.010 0.033 0.023 0.007 0.000
THE WORLDPOST 0.001 0.003 0.013 0.009 0.001 0.009 0.036 0.000 0.037 0.001 0.003 0.019 0.020 0.012 0.003 0.007 0.008 0.315 0.012 0.012 0.002 0.010 0.003 0.002 0.002 0.323 0.009 0.015 0.007 0.070 0.032
0.7
0.6
0.5
0.4
0.3
0.2



4.4.4 Explore and Assess Categories/Classes with True Positive Rates, False Positive Rates, etc.

Based on the above overall accuracy score and heatmap results per category, you can begin to identify news genre categories (or classes) that may be performing well versus categories that are not performing well. You can use the following code below to explore those performance biases per news genre in the corpus/model.

As you work through the code below, take notes as you compare the overall accuracy score of the model with specific optimal threshold scores, and be sure to compare those results against the EDA work that you conducted earlier in the notebook.

Our LR model has 31 classes (news genres), so it's a "multi-class" LR model. To help us explore the predictive accuracy across each class, we must first use scikit-learn's `LabelBinarizer()` to enable us to compute the True Positive Rates (TPR) and False Positive Rates (FPR) for each class/category.

The 3 variables assigned below will help us process the training and testing data to do so.

4.4.4.1 - Create data set of binary predictions per row across all of our classes

In [51]:
```
from sklearn.preprocessing import LabelBinarizer
```

In [52]:
```
# scikit-learn's LB() will turn the prediction output on the test data into zeroes and ones, where 1 == the predicted class (news genre) and 0 =
label_binarizer = LabelBinarizer().fit(Y_test)

# This variable transforms the predictions into the 0s and 1—commonly referred to as either the "one vs all" test or "one hot" test. I actually
y_onehot_test = label_binarizer.transform(Y_test)

# (rows == X inputs, columns == Y possible predicted classes)
```

```
y_onehot_test.shape
```

Out[52]: (31248, 31)

It is always a good practice to print out your data in some manner, so you can better understand and be sure that it's what you want/need.

In [53]:
```
# Reviews the test data per row and returns a binary
print(
  'Number of Classes/News Genres:',len(y_onehot_test[0]),
  '\nFirst row in test set:',y_onehot_test[0],
  '\nPrediction (1):',Y_test[0]
)
```

```
Number of Classes/News Genres: 31
First row in test set: [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
Prediction (1): HEALTHY LIVING
```

4.4.4.2 - Use that data to visualize the TPs & FPs with an "ROC Curve" & "Area Under the Curve"

An *ROC curve* (Receiver Operating Characteristic curve) is a tool to identify the relationship between the accurate "hit rate" (TP) and accurate "false alarms" (FP).

This curve plots the False Positive (x) and True Positive (y) rates at their different classification thresholds to help us visualize the performance of the model at different potential thresholds. In effect, this visual can help you isolate the performance of some classes over others in relationship with the original EDA work that you have conducted.

You can also use ROC curve plots to see how when you lower the classification threshold, the model will classify more items as positive and increase both False Positives and True Positives.

The goal is to identify the optimal Area Under the Curve (AUC). It aggregates all of the probable classification values as a normalized value between 0 and 1 for each new input in the data set.

- 0 == The model is wrong 100% of the time
- 1 == The model is correct 100% of the time

Think of it like the spectrum between 0 and 1 below, where the random positive predictions are positioned to the right of the random negative predictions. So, if the AUC score is `0.714`, then the probability that `p` is positioned to the right of `n` is `71.4%`.

```
Ouput of the LR Model
n = Actual Negative Prediction
```

```
p = Actual Positive Prediction
____0
|n
|n
|n
|n
|n
|n
|n
|n
|n
|n
|p
|n
|n
|p
|p
|n
|p
|p
|p
|p
|p
|p
————1
```

Ok, that is a lot of info. Let's see some different visual methods to help us understand ROC curves as a means to plot the relationship between FP rates and TP rates.

4.4.4.3 - Functions to Generate FPR, TPR, ROC, and AUC Data & Visualizations

In utils.py, under the "MODEL EVALUATION FUNCTIONS" section, please review the following functions to create evaluative tests and visuals.

4.4.4.4 - Create the ROC Data Set

Now, let's compute the FP and TP rates with the first function: `roc_curve_per_category`

In [54]:
```
df_all_classes_roc_values = roc_curve_per_category(
  list_classes=model_td_only.classes_,
  label_binarizer=label_binarizer,
```

```
  y_proba=Y_probability_a_td_only,
  y_one_vs_all=y_onehot_test
)
df_all_classes_roc_values.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44025 entries, 0 to 44024
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Class        44025 non-null  object 
 1   FPR          44025 non-null  float64
 2   TPR          44025 non-null  float64
 3   FPR_Optimal  44025 non-null  float64
 4   TPR_Optimal  44025 non-null  float64
dtypes: float64(4), object(1)
memory usage: 1.7+ MB
```

Let's check out a category in the dataframe. You can change the value of the `Class` query, so you can explore what data are returned by the `roc_curve_per_category` function. You can see that the function has generated the FPR and TPR for each new headline in the testing data. I also created a repeating column that tracks the Optimal FPR and TPR to use in our upcoming ROC curve plot.

In [55]:
```
df_all_classes_roc_values.loc[df_all_classes_roc_values.Class == 'WOMEN']
```

Out[55]:

	Class	FPR	TPR	FPR_Optimal	TPR_Optimal
40471	WOMEN	0.000000	0.000000	0.1926	0.57
40472	WOMEN	0.000033	0.000000	0.1926	0.57
40473	WOMEN	0.000066	0.000000	0.1926	0.57
40474	WOMEN	0.000066	0.001129	0.1926	0.57
40475	WOMEN	0.000066	0.006772	0.1926	0.57
...	...	...	...	...	...
42076	WOMEN	0.974870	0.997743	0.1926	0.57
42077	WOMEN	0.974870	0.998871	0.1926	0.57
42078	WOMEN	0.979744	0.998871	0.1926	0.57
42079	WOMEN	0.979744	1.000000	0.1926	0.57
42080	WOMEN	1.000000	1.000000	0.1926	0.57

1610 rows × 5 columns

Ok, let's create a separate dataframe that isolates the optimal TPR and FPR values per class/category/news genre, so we can visualize them in alternative ways.

In [56]:
```
df_all_classes_optimal_roc_values = df_all_classes_roc_values.drop_duplicates(subset=['Class']).reset_index()[['Class','FPR_Optimal','TPR_Optim

df_all_classes_optimal_roc_values
```

Out[56]:

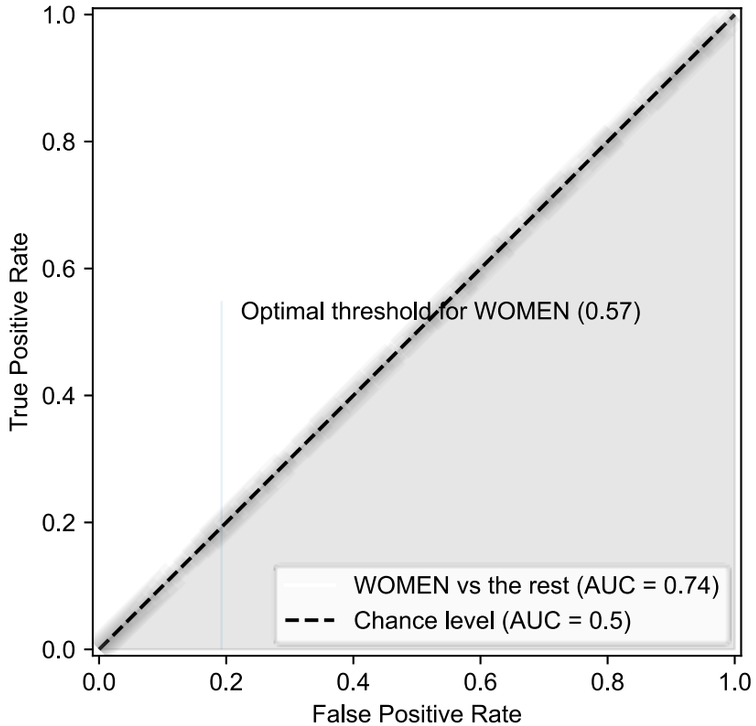
	Class	FPR_Optimal	TPR_Optimal
0	ARTS	0.2146	0.7189
1	ARTS & CULTURE	0.1482	0.5896
2	BLACK VOICES	0.1828	0.5845
3	BUSINESS	0.3023	0.7343
4	COLLEGE	0.2820	0.7138
5	COMEDY	0.4350	0.8087
6	CRIME	0.2441	0.7425
7	EDUCATION	0.0555	0.5885
8	ENTERTAINMENT	0.3533	0.8073
9	FIFTY	0.3154	0.7531
10	GOOD NEWS	0.3403	0.7553
11	GREEN	0.2807	0.7396
12	HEALTHY LIVING	0.2979	0.7693
13	IMPACT	0.3793	0.6916
14	LATINO VOICES	0.3982	0.6643
15	MEDIA	0.2994	0.6792
16	PARENTS	0.1485	0.6237
17	POLITICS	0.1957	0.6985
18	QUEER VOICES	0.3245	0.7092
19	RELIGION	0.2885	0.7308
20	SCIENCE	0.2974	0.7399
21	SPORTS	0.3033	0.7762
22	STYLE	0.2606	0.7782

	Class	FPR_Optimal	TPR_Optimal
23	TASTE	0.0936	0.6841
24	TECH	0.1329	0.5817
25	THE WORLDPOST	0.1609	0.7517
26	TRAVEL	0.2794	0.7690
27	WEIRD NEWS	0.3725	0.7808
28	WOMEN	0.1926	0.5700
29	WORLD NEWS	0.1808	0.7174
30	WORLDPOST	0.2135	0.7943

4.4.4.5 - Plotting the ROC Curve per Class

In [57]:
```
class_of_interest = "WOMEN"
plot_class_roc_curve(
  class_of_interest=class_of_interest,
  label_binarizer=label_binarizer,
  Y_one_vs_all=y_onehot_test,
  Y_prob_a=Y_probability_a_td_only,
  df_class_row=df_all_classes_optimal_roc_values.loc[df_all_classes_optimal_roc_values.Class == class_of_interest]
)
```

```
/Users/calindgr/Documents/NCSU/Conferences/2024-2025/032025-CCCC/cccc25-cpu-love/3.10.12/lib/python3.10/site-packages/matplotlib/text.py:1475: F
utureWarning: Calling float on a single element Series is deprecated and will raise a TypeError in the future. Use float(ser.iloc[0]) instead
  x = float(self.convert_xunits(x))
/Users/calindgr/Documents/NCSU/Conferences/2024-2025/032025-CCCC/cccc25-cpu-love/3.10.12/lib/python3.10/site-packages/matplotlib/text.py:1477: F
utureWarning: Calling float on a single element Series is deprecated and will raise a TypeError in the future. Use float(ser.iloc[0]) instead
  y = float(self.convert_yunits(y))
/Users/calindgr/Documents/NCSU/Conferences/2024-2025/032025-CCCC/cccc25-cpu-love/3.10.12/lib/python3.10/site-packages/matplotlib/text.py:905: Fu
tureWarning: Calling float on a single element Series is deprecated and will raise a TypeError in the future. Use float(ser.iloc[0]) instead
  x = float(self.convert_xunits(self._x))
/Users/calindgr/Documents/NCSU/Conferences/2024-2025/032025-CCCC/cccc25-cpu-love/3.10.12/lib/python3.10/site-packages/matplotlib/text.py:906: Fu
tureWarning: Calling float on a single element Series is deprecated and will raise a TypeError in the future. Use float(ser.iloc[0]) instead
  y = float(self.convert_yunits(self._y))
/Users/calindgr/Documents/NCSU/Conferences/2024-2025/032025-CCCC/cccc25-cpu-love/3.10.12/lib/python3.10/site-packages/matplotlib/text.py:762: Fu
tureWarning: Calling float on a single element Series is deprecated and will raise a TypeError in the future. Use float(ser.iloc[0]) instead
  posx = float(self.convert_xunits(x))
/Users/calindgr/Documents/NCSU/Conferences/2024-2025/032025-CCCC/cccc25-cpu-love/3.10.12/lib/python3.10/site-packages/matplotlib/text.py:763: Fu
tureWarning: Calling float on a single element Series is deprecated and will raise a TypeError in the future. Use float(ser.iloc[0]) instead
  posy = float(self.convert_yunits(y))
```



4.4.4.6 - Alternative visualizations of the ROC Curve

There are other ways to plot this curve. Here are a few, which might help us understand the "Area Under the Curve" (AUC).

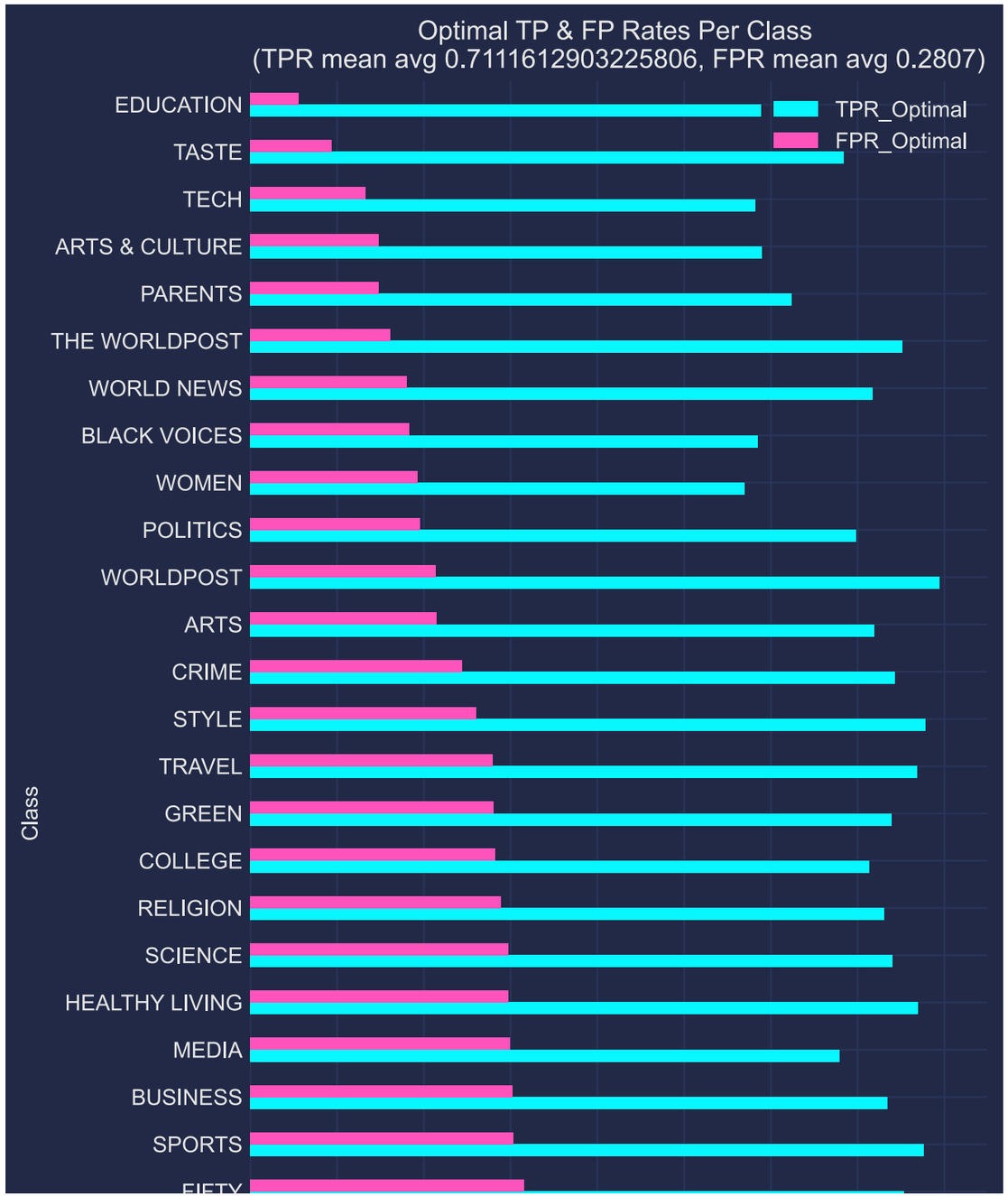
In [58]:
```
TPR_OPT_MEAN = df_all_classes_optimal_roc_values.drop_duplicates(subset=['Class'])['TPR_Optimal'].mean()
TPR_OPT_MEDIAN = df_all_classes_optimal_roc_values.drop_duplicates(subset=['Class'])['TPR_Optimal'].median()

FPR_OPT_MEAN = df_all_classes_optimal_roc_values.drop_duplicates(subset=['Class'])['FPR_Optimal'].median()
FPR_OPT_MEDIAN = df_all_classes_optimal_roc_values.drop_duplicates(subset=['Class'])['FPR_Optimal'].median()

print(
  'TPR/FPR - Estimates of Location',
  '\n----------------------------',
  '\nTPR_OPT_MEAN:',TPR_OPT_MEAN,
```

```
  '\nTPR_OPT_MEDIAN:',TPR_OPT_MEDIAN,
  '\n\nFPR_OPT_MEAN:',FPR_OPT_MEAN,
  '\nFPR_OPT_MEDIAN:',FPR_OPT_MEDIAN,
)
```

```
TPR/FPR - Estimates of Location
-------------------------
TPR_OPT_MEAN: 0.7111612903225806
TPR_OPT_MEDIAN: 0.7308

FPR_OPT_MEAN: 0.2807
FPR_OPT_MEDIAN: 0.2807
```

In [59]:
```
plt.figure()

df_all_classes_optimal_roc_values.sort_values(by='FPR_Optimal', ascending=False).plot(
  kind='barh',
  x='Class',
  y=['TPR_Optimal','FPR_Optimal'],
  figsize=(6,12),
  title='Optimal TP & FP Rates Per Class \n(TPR mean avg '+str(TPR_OPT_MEAN)+', FPR mean avg '+str(FPR_OPT_MEAN)+')'
)
```

Out[59]:
```
 <Axes: title={'center': 'Optimal TP & FP Rates Per Class \n(TPR mean avg 0.7111612903225806, FPR mean avg 0.2807)'}, ylabel='Class'>
<Figure size 640x480 with 0 Axes>
```


Optimal TP & FP Rates Per Class
(TPR mean avg 0.7111612903225806, FPR mean avg 0.2807)
TPR_Optimal
FPR_Optimal
Class
EDUCATION
TASTE
TECH
ARTS & CULTURE
PARENTS
THE WORLDPOST
WORLD NEWS
BLACK VOICES
WOMEN
POLITICS
WORLDPOST
ARTS
CRIME
STYLE
TRAVEL
GREEN
COLLEGE
RELIGION
SCIENCE
HEALTHY LIVING
MEDIA
BUSINESS
SPORTS
FIFTY

You could use the above bar chart to compare class label performances in relationship to the mean/median optimal threshold rates.

4.4.5 `text_desc` only & count-based modeling assessment

- **Accuracy**:
- **MRR**:
- **Insert Observation #1**
  - Enter thoughts about this observation based on the available assessment evidence above
- **Insert Observation #2**
  - Enter thoughts about this observation based on the available assessment evidence above
- ...

4.5 Model 2 - `tfidf` features with `text_desc_headline` - short description + headline

In [60]:
```
field='text_desc_headline'
feature_rep='tfidf'
top_k=3

model_tfidf_tdh,transformer_tfidf_tdh,accuracy_tfidf_tdh,mrr_at_k_tfidf_tdh,X_tfidf_tdh_train,X_tfidf_tdh_test,Y_tfidf_tdh_test,Y_tfidf_tdh_tra
  df,
  field=field,
  feature_rep=feature_rep,
```

```
  top_k=top_k
)

print(f"\n\nSimple Mean Average Model Accuracy = {accuracy_tfidf_tdh}\nMean Reciprocal Rank = {mrr_at_k_tfidf_tdh}\n")
```

```
2025-04-09 08:52:20,666 : INFO : Starting model training...
2025-04-09 08:52:20,691 : INFO : Extracting features and creating vocabulary...
2025-04-09 08:52:22,839 : INFO : Training a Logistic Regression Model. This may take a few minutes. ...
2025-04-09 08:53:04,092 : INFO : Starting evaluation...
2025-04-09 08:53:04,128 : INFO : Done training and evaluation.

Simple Mean Average Model Accuracy = 0.8359254992319508
Mean Reciprocal Rank = 0.7172085680150102
```

4.5.1 `text_desc`, `headline`, & TF-IDF-based modeling assessment

Now, use the same assessment techniques as in the `text_desc` only model to assess this model that uses TF-IDF + a combination of the short description and headline values as features for the model.

Write your observations and evidence below.

- **Accuracy**:
- **MRR**:
- **Insert Observation #1**
  - Enter thoughts about this observation based on the available assessment evidence above
- **Insert Observation #2**
  - Enter thoughts about this observation based on the available assessment evidence above
- ...

4.6 Model 3 - `tfidf` features with `text_desc_headline_url`: description, headline, *and* url

In [61]:
```
field='text_desc_headline_url'
feature_rep='tfidf'
top_k=3

model_tfidf_all,transformer_tfidf_all,accuracy_tfidf_all,mrr_at_k_tfidf_all,X_tfidf_all_train,X_tfidf_all_test,Y_tfidf_all_test,Y_tfidf_all_tra
```

```
  df,
  field=field,
  feature_rep=feature_rep,
  top_k=top_k
)

print("\nAccuracy={0}; MRR={1}".format(accuracy_tfidf_all,mrr_at_k_tfidf_all))
```

```
2025-04-09 08:53:04,136 : INFO : Starting model training...
2025-04-09 08:53:04,157 : INFO : Extracting features and creating vocabulary...
2025-04-09 08:53:06,874 : INFO : Training a Logistic Regression Model. This may take a few minutes. ...
2025-04-09 08:53:55,923 : INFO : Starting evaluation...
2025-04-09 08:53:55,984 : INFO : Done training and evaluation.
Accuracy=0.8672875064004096; MRR=0.7511734084314612
```

4.6.1 `text_desc`, `headline`, `URL`, & TF-IDF-based modeling assessment

Now, use the same assessment techniques as the previous models to assess this third model that uses TF-IDF + a combination of the short description, headline, and URL values as features for the model.

Write your observations and evidence below.

- **Accuracy**:
- **MRR**:
- **Insert Observation #1**
  - Enter thoughts about this observation based on the available assessment evidence above
- **Insert Observation #2**
  - Enter thoughts about this observation based on the available assessment evidence above
- ...

5. Check Predictions on Unseen Articles

Try from articles that aren't from HuffPost: CNN, Fox, MSNBC, etc.

Remember that the algorithm will return the top `k` probabilities. I've created a default of 3 for `k`. The first value in the list is the top returned predicted value.

5.1 Model 1 trained by counts method & the features from the short description only

This uses Model 1:

- Model: `model_td_only`
- Transformer: `transformer_td_only`

In [62]:
```
# https://www.cnn.com/2019/07/19/health/astronaut-exercise-iv-faint-scn/index.html
test_features = transformer_td_only.transform(
  ["Exercise in space keeps astronauts from fainting when they return to Earth, study says."]
)
get_top_k_predictions(
  model_td_only,
  test_features,
  3
)
```

Out[62]:
```
[['SCIENCE', 'WOMEN', 'GREEN']]
```

5.2 Model 2 trained by TF-IDF & the features from the short description & headline

This uses Model 2:

- Model: `model_tfidf_tdh`
- Transformer: `transformer_tfidf_tdh`

In [63]:
```
# URL: https://www.network.com/enter/url/to/story/here.html

test_features=transformer_tfidf_tdh.transform(
  ["Exercise in space keeps astronauts from fainting when they return to Earth, study says."]
)

get_top_k_predictions(
  model_tfidf_tdh,
  test_features,
  3
)
```

Out[63]:
```
[['SCIENCE', 'WEIRD NEWS', 'HEALTHY LIVING']]
```

5.3 Model 3 trained by TF-IDF & the features from the short description, headline & URL

This uses Model 3:

- Model: `model_tfidf_all`
- Transformer: `transformer_tfidf_all`

In [64]:
```
# URL: https://www.network.com/enter/url/to/story/here.html
test_features=transformer_tfidf_tdh.transform(
  ["Exercise in space keeps astronauts from fainting when they return to Earth, study says."]
)

get_top_k_predictions(
  model_tfidf_tdh,
  test_features,
  3
)
```

Out[64]:
```
[['SCIENCE', 'WEIRD NEWS', 'HEALTHY LIVING']]
```

6. Save Models for Future Use

In [65]:
```
import pickle
```

In [66]:
```
def pickle_dump(model, model_path_filename, transformer, transformer_path_filename):
  '''
    # pickle_dump()

    **Purpose**: Save a trained model and transformer.

    ## @params
    - `model`: Trained model to save as `.pkl` file.
    - `model_path_filename`: String. Model's full path and filename with `.pkl` file format.
    - `transformer`: Vectorizer transformer object.
    - `transformer_path_filename`: String. Transformer's full path and filename with `.pkl` file format.
```

```
    ## @returns
    - None. It prints out a message where it saved to verify.
  '''
  # Save both model & transformer to encode a document and the model itself to make predictions based on the weight vectors
  pickle.dump( model, open(model_path_filename, 'wb') )
  pickle.dump( transformer, open(transformer_path_filename,'wb') )
  print(f"Model saved to {model_path_filename}")
  print(f"Transformer saved to {transformer_path_filename}")
```

In [67]:
```
# Use imported `pickle` code library to save trained models & transformers—see pickle_dump() utility function.

#### 1. COUNTS - 'text_desc' as features only ####
pickle_dump(
  model=model_td_only,
  model_path_filename="./models/news_genre_model_binary_text_desc_only.pkl",
  transformer=transformer_td_only,
  transformer_path_filename="./models/news_genre_transformer_binary_text_desc_only.pkl"
)

#### 2. TF-IDF - 'text_desc_headline' as features ####
pickle_dump(
  model=model_td_only,
  model_path_filename="./models/news_genre_model_tfidf_tdh.pkl",
  transformer=transformer_td_only,
  transformer_path_filename="./models/news_genre_transformer_tfidf_tdh.pkl"
)

#### 3. TF-IDF - 'text_desc_headline_url' as features ####
pickle_dump(
  model=model_td_only,
  model_path_filename="./models/news_genre_model_tfidf_all.pkl",
  transformer=transformer_td_only,
  transformer_path_filename="./models/news_genre_transformer_tfidf_all.pkl"
)
```

```
Model saved to ./models/news_genre_model_binary_text_desc_only.pkl
Transformer saved to ./models/news_genre_transformer_binary_text_desc_only.pkl
Model saved to ./models/news_genre_model_tfidf_tdh.pkl
Transformer saved to ./models/news_genre_transformer_tfidf_tdh.pkl
Model saved to ./models/news_genre_model_tfidf_all.pkl
Transformer saved to ./models/news_genre_transformer_tfidf_all.pkl
```

7. Use Loaded Model

With pickle, we can now also load the model and use it without needing to retrain everything. A good time saver to consider for your team's final project!

In [68]:
```
# Load Model 2 for reference
# Note my variable assignments include "loaded_" so I can distinguish variable assignments, if necessary.
model_path_tdh = "./models/news_genre_model_tfidf_tdh.pkl"
transformer_path_tdh = "./models/news_genre_transformer_tfidf_tdh.pkl"

loaded_model_tfidf_tdh = pickle.load(open(model_path_tdh, 'rb'))
loaded_transformer_tfidf_tdh = pickle.load(open(transformer_path_tdh, 'rb'))
```

In [69]:
```
# URL: https://www.usatoday.com/story/travel/airline-news/2025/04/07/avelo-airlines-ice-deportation-flights/82980364007/
loaded_test_features_tfidf_tdh = loaded_transformer_tfidf_tdh.transform(
  ["Ultra low cost carrier Avelo Airlines will operate deportation flights under ICE charter"]
)

get_top_k_predictions(
  loaded_model_tfidf_tdh,
  loaded_test_features_tfidf_tdh,
  3
)
```

Out[69]:
```
[['TRAVEL', 'POLITICS', 'WORLDPOST']]
```

Conclusion

Overall, do your best to have understood this LR modeling/training process: the goals for the model in relationship to the original data set used. By documenting your observations and using Python to conduct EDA + model assessments, you can complete the following work that you will need to repeat for your final project:

1. Identify potential boundaries, biases, and limits of the data in relationship to your developing and changing goals for modeling
2. Conversely, identify potential possibilities and affordances of the data in relationship to your developing and changing goals for modeling
3. Documenting these biases, affrodances, and changes in goals
4. Use this notebook material to develop a Model Card that communicates the aforementioned material in a more concise and helpful manner.

Note that there is a next step that folks would take next, based on their analysis to "tune" their model. We won't get to that part of the process in this class. Instead, we

are focusing on the initial training step, the impact of different features available to use in the data set, and assessing the initial training based on the data.