Table of Contents

[Design 2](#_Toc24616950)

[1.1- Project Overview 2](#_Toc24616951)

[1.2- Modular System 7](#_Toc24616952)

[1.3 – Database Schema: 8](#_Toc24616953)

[1.4 – Data Structures and other resources: 12](#_Toc24616954)

[2.1 – Planned SQL queries: 12](#_Toc24616955)

[2.2 – Algorithms and Prototyping: 14](#_Toc24616956)

[News Categorisation: 14](#_Toc24616957)

[Detecting headline duplicates: 24](#_Toc24616958)

[Generating Short Headlines for wordcloud: 24](#_Toc24616959)

[Twitter Loader: 24](#_Toc24616960)

[News Loader: 24](#_Toc24616961)

[Scraping News Body: 24](#_Toc24616962)

[Summaries of News: 24](#_Toc24616963)

[UI Design & HTML: 24](#_Toc24616964)

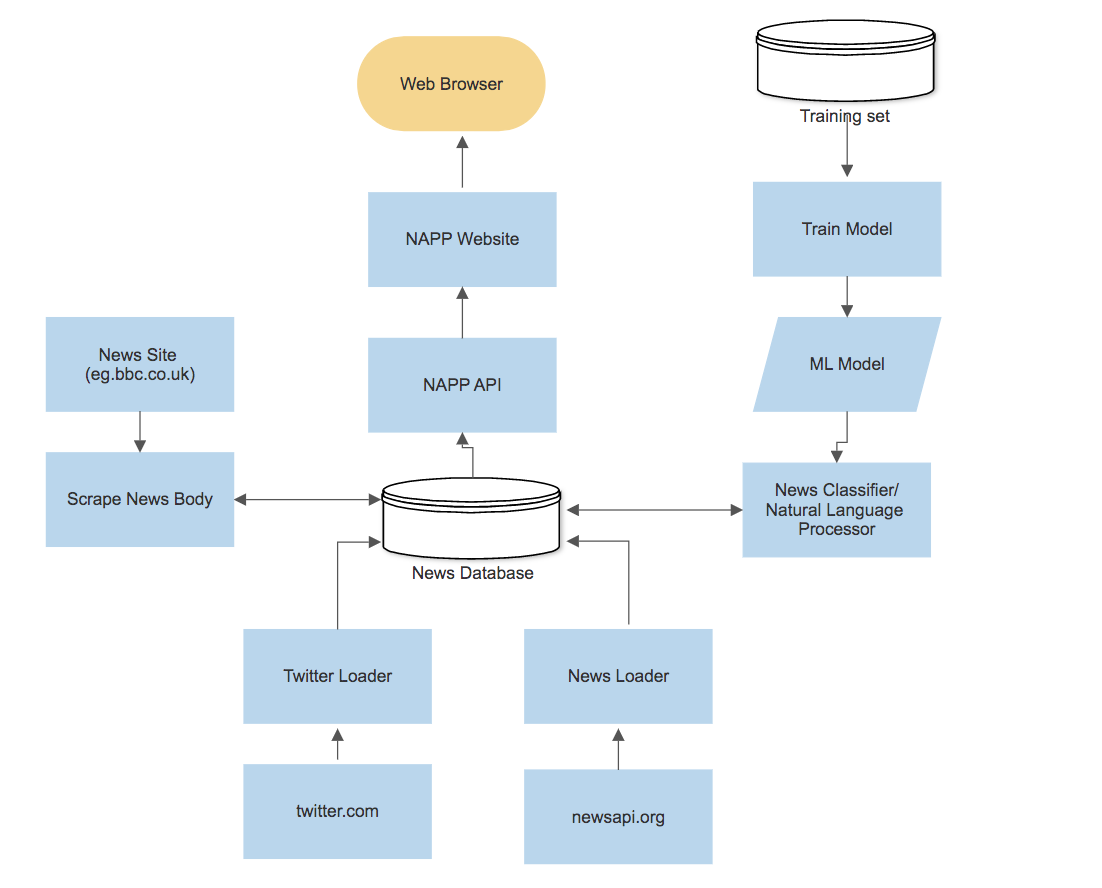
Design

# 1.1- Project Overview

To visualise the NAPP application, I created a flowchart to present all different components and the flow of data between them represented by the arrows.

Loader components such as News Loader and Tweet Loader read from data sources respectively and store news and tweets in the NAPP News Database. These components are running constantly and updating the database. The News Classifier component is being activated by a timer and processes newly added items in the database: classifies news using a pretrained machine learning model and applies natural language processing to additionally categorise news and tweets and provides summaries. The result is stored back into the Database.

Users open the website and upon user request, the NAPP website shows categorised news and summaries, by requesting them from NAPP API which in turn reads them from the database.



Independently from the rest of the processes, there is a model training flow, where I train the model. This can be done regularly or only once. I use Jupyter Notebook to train a classifier model using a training set and that model is used by the News Classifier component as explained above

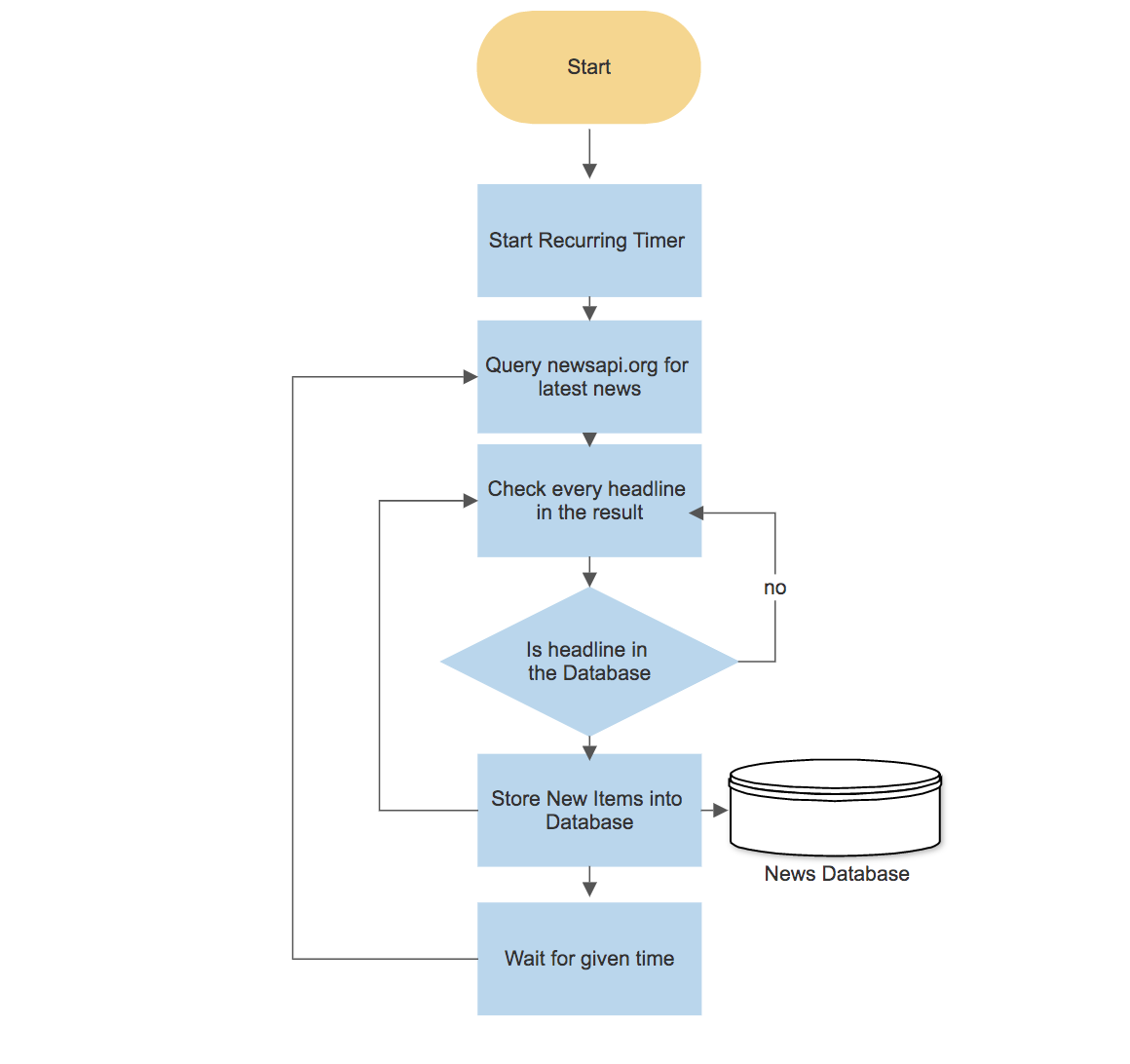
**Machine Learning model:**

Next, I created a more in-depth flowchart of how the machine learning model is trained and saved. After loading the data, I cleaned the data so it is ready to be vectorised and encoding data (turning words into numbers). Each category is given a number and each headline is given a unique number key (a vector of the same size. This step performs feature engineering that is required to train the model. The model is then tested to specify its success rate and then saved as a pickle file.



**News Loader:**

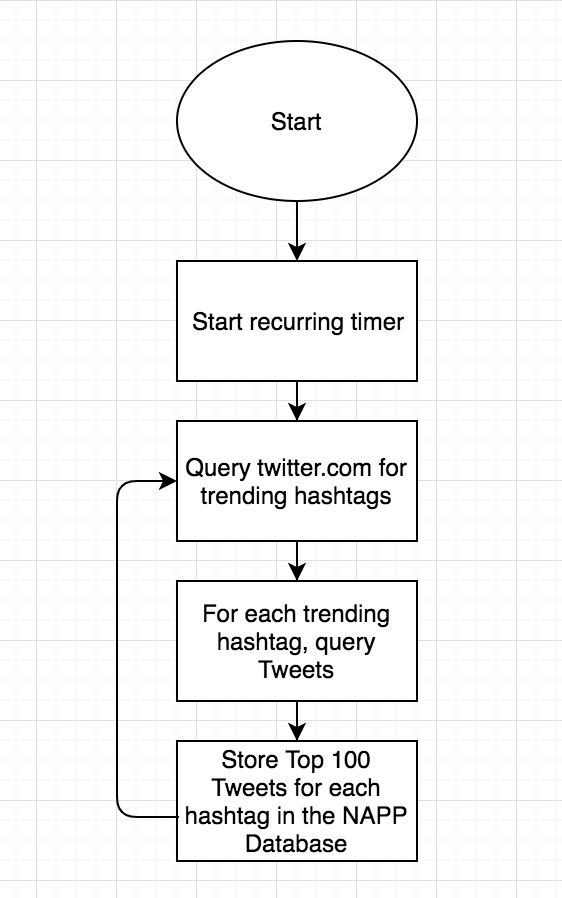
To load the news headlines, I will be checking the newsapi.org every given time (recurring timer), and query newsapi.org for the latest headlines in all countries using one filter: language = English.

For each headline in the response, I will check if it already exists in the database and will find any new items that will be saved to the database.

**Twitter Loader:**

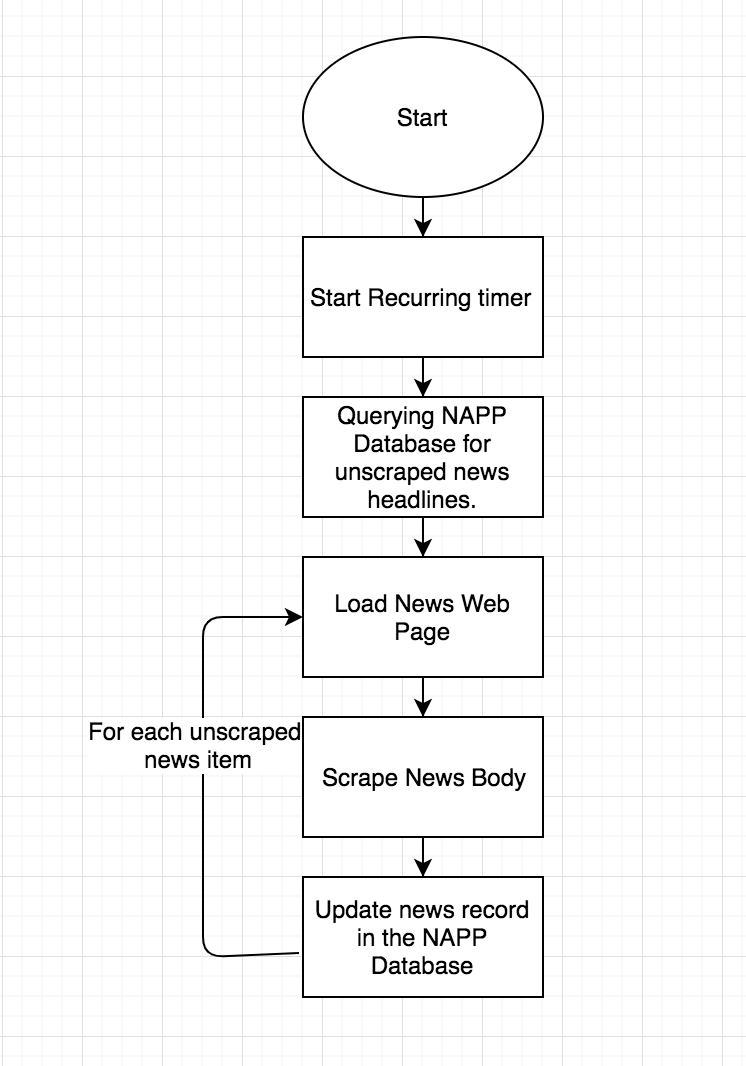
Due to the extensive amount of tweets produced per second in the world, I will have to harshly limit the types of tweets I use. I will only concentre on tweets in English, in selected countries and will only consider the most popular tweets and hashtags.

To load Tweets, I will query Twitter.com for trending hashtags and for each hashtag query tweets and store them in the database. I will only store the top 100 tweets to filter out tweets and the top 100 and the top 100 tweets will be the most popular at the moment.

****

**News Scraping:**

This diagram shows the process of how the scraping component works. It is a small application (component that works with other components to serve a purpose) that activates a recurring timer (scheduler) that activates at configurable intervals, the scraping body routine.

****To scrape the news, I will be querying the Database for the records for of news where the ScrapedAt field is NULL (see Database Schema Section). For each unscraped record, the component will load the webpage from the news URL, and will scrape the news with the help of Beautiful Soup4, updating the record in the Database. To ensure we don’t save duplicates, I will check if the tweet is already in the database.

# 1.2- Modular System

The site will contain multiple menus; the main page that will show a word cloud of the most popular news stories globally (the automatic categorisation option). It will also contain 2 option bars, one for category and one for location.

Once a headline is clicked on, a page will be opened with a short summary and the links to the news pages that were used to gather the information from. It will also display the Twitter feed underneath, showing related hashtags and comments.

Home Page – shows most popular news globally (automatically loaded when site is entered). It provides the menu bar for categories.

Customised Page – every time a category is chosen, the news headlines (event name) are categorised, and a different page is presented for each category and each location.

Summary Page – When a headline is clicked, it sends you to a summary, link and Tweeter feedback.

There are 9 possible news categories and 3 locations

**News categories:**

1. Politics
2. Health
3. Environmental
4. IT/Tech
5. Science
6. Entertainment
7. LGBT
8. World
9. UK

**Location categories:**

1. Global
2. Local (your country)
3. Custom (select country)

All the news will come from newsapi.org, where the news is filtered by the language (English) and the news is taken from many different sources.

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Processes | Storage | Outputs |
| -User input of location and category, clicking on event  (mouse clicks) | -ML model trained on sample data, and saved as pickle file  -Reading news and Tweets and storing them in the database using News Loader and Tweet Loader components.  -The News Classifier component is being activated by a timer and processes newly added items in the database. It classifies news using a pretrained machine learning model to categorise the news and applies natural language processing to shorten headlines and filter out news duplicates by using my algorithm.  -Scraping news bodies of sites for the summaries | -Training set database contains training data for ML model  -NAPP database contains all news, events and tweets. | -Outputs headlines in form of a wordcloud (when filters are applied)  -Outputs summaries once clicked on headline and outputs the source link  -Outputs filtered Twitter comments and hashtags. |

# 1.3 – Database Schema:

**News Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Filed Type | Field Size | Purpose | Example Data | Mandatory |
| NewsID | INT | 8 bytes | To uniquely identify news | 1234 | Yes  Primary Key |
| Headline | CHAR |  | To store a headline | “Trump was…” | Yes |
| Source | CHAR |  | Source of the news | BBC News | Yes |
| URL | CHAR |  | Link to the news article | bbc.co.uk/… | Yes |
| CountryCode | CHAR |  | Identifies country of the news | UK | No |
| CategoryID | INT | 8 bytes | Numeric category identifier | 2 | No  Foreign Key |
| EventID | INT | 8 bytes | To identify which event this news represents | 1234 | No  Foregin Key |
| NewsBody | CHAR |  | To keep news body for latter analysation to make summaries | “This home in Fishlake, near Doncaste…” | No |
| ScrapedAt | DATETIME | 8 bytes | To identify the date and time when the news was scraped | 2019-11-07 13:43:00-04:00 | No |
| CreatedAt | DATETIME | 8 bytes | To identify the date and time when record was added to the table | 2019-11-07 13:43:00-04:00 | Yes  Default value = utcnow |

**Tweet Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Filed Type | Field Size | Purpose | Example Data | Mandatory |
| TweetID | INT | 8 bytes | To uniquely identify tweet | 1234 | Yes  Primary Key |
| Tweet | CHAR |  | To store a tweet | “Trump was…” | Yes |
| Hashtags | CHAR |  | Comma separated list of hashtags to relevant event | #metoo | No |
| User | CHAR |  | Author of tweet | @oryna | Yes |
| URL | CHAR |  | Link to the tweet | twitter.com/… | Yes |
| PublishedAt | DATETIME |  | Identifies time of publishing | 2019-11-07 13:43:00-04:00 | No |
| CategoryID | INT | 8 bytes | Numeric category identifier | 2 | No  Foreign Key |
| EventID | INT | 8 bytes | Numeric event identifier | 1234 | No  Foreign Key |
| CreatedAt | DATETIME | 8 bytes | To identify the date and time when record was added to the table | 2019-11-07 13:43:00Z | Yes  Default value = utcnow |

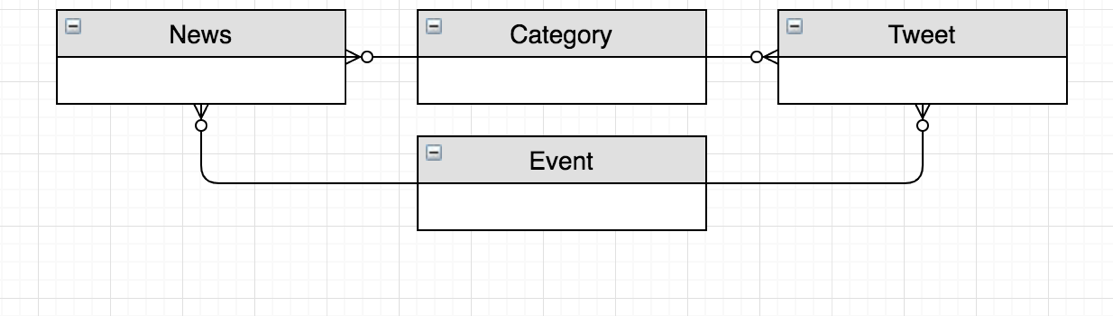
**Event Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Filed Type | Field Size | Purpose | Example Data | Mandatory |
| EventID | INT | 8 bytes | To uniquely identify event | 1234 | Yes  Primary Key |
| Name | CHAR |  | Shortened headline for the word cloud | “Trump” | Yes |
| Summary | CHAR |  | Autogenerated summary for event | ‘This home in Fishlake, near Doncaster, has been left nearly submerged by floodwater.’ | No |
| CreatedAt | DATETIME | 8 bytes | To identify the date and time when record was added to the table | 2019-11-07 13:43:00Z | Yes  Default value = utcnow |

**Category Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Filed Type | Field Size | Purpose | Example Data | Mandatory |
| CategoryID | INT | 8 bytes | To uniquely identify category | 1234 | Yes  Primary Key |
| Name | CHAR |  | Category Name | Health | Yes |

The ERD presents the entities in my database:



‘News’ Table contains information on each news item (including headline, url, etc).

The ‘Tweet’ Table contains information on each filtered Tweet and the ‘Category’ Table contains the list of categories. The ‘Event’ Table will contain information about each event.

News Loader will save News to the News Table and Tweet Loader will save Tweets to the Tweet Table. We will filter and only save Tweets that are of our interest. Our components Classifier will assign categories to the News and Tweets using pretrained model.

The fields CreatedAt (in both News and Tweet Tables) are used so that the Classifier component can query for the records added to the tables since a particular point in time (since processing news last time). I will provide a default value of utcnow when I define a table so that all inserted records will get this value automatically.

The fields Body and ScrapedAt in the News Table will be populate by the news scraping component which is going to be activated at configurable intervals and will load the news web page specified in the URL and will scrape the body, with the help of Beautiful Soup4 (a library for web scraping), storing it in the news body column. It will then update the ScrapedAt field so that we know the news has been scraped.

The URL is saved to be displayed on the summary page of each news article so that the user can visit the original site.

When NAPP identifies that several news articles from different news sources are about the same event in the real world, I will store that as a new entry in the Event table and will connect the news to that event by assigning an EventID on the news. The event name will be generated by applying NLP to the news bodies and headlines. The ‘Name’ will be using in the wordcloud to identify event.

In a similar way, NAPP will analyse Tweets to identify the event that the Tweet describes. Now I have the ability to query all hashtags that describe the event by joining the Event and Tweet tables.

Our database schema is normalised into 3rd Normal Form, for example, the category is referenced by the categoryID. The features of 3rd Normal Form include: no non-key attributes are dependent on any other non-key attributes in the table, no repeating attributes in tables and all attributes depend on the primary key and nothing else.

# 1.4 – Data Structures and other resources:

I will be using several data structures from the standard python library and additional packages

**Data Structures:**

1. Python Dictionaries
2. Python List
3. Pandas Dataframe
4. Python Datetime
5. Python Deque
6. Collections counter
7. Numpy array
8. Python regular expressions

**Frameworks and libraries:**

1. Scikit learn
2. Pandas
3. Numpy
4. sqlite3
5. Fast API
6. Beautiful Soup4
7. Schedule
8. Requests
9. Newsapi client

# 2.1 – Planned SQL queries:

I am implementing multiple databases to store my data in. The first one contains all my test headlines with given categories for my prototyping and training the machine learning model. The second NAPP Database will contain all news tweets categories and events that will be displayed on the website to the user.

To create my Database, I will use sqlite3. Here are the queries I will use to create my NAPP Database:

**Create category table:**

CREATE TABLE Category(

CategoryID INT PRIMARY KEY,

Name VARCHAR NOT NULL

)

**Create event table:**

CREATE TABLE Event(

EventID INT PRIMARY KEY AUTOINCREMENT,

Name VARCHAR NOT NULL,

Summary VARCHAR,

CreatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

)

**Create news table:**

CREATE TABLE News(

NewsID INT PRIMARY KEY AUTOINCREMENT,

Headline VARCHAR NOT NULL,

Source VARCHAR NOT NULL,

URL VARCHAR NOT NULL,

CountryCode CHAR(3) NOT NULL,

CategoryID INT,

EventID INT,

NewsBody INT,

ScrapedAt TIMESTAMP,

CreatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY(CategoryID) REFERENCES Category(CategoryID), FOREIGN KEY(EventID) REFERENCES Event(EventID)

)

**Create tweet table:**

CREATE TABLE Tweet(

TweetID INT PRIMARY KEY AUTOINCREMENT,

Tweet VARCHAR NOT NULL,

Hashtag VARCHAR NOT NULL,

URL VARCHAR NOT NULL,

User VARCHAR NOT NULL,

CategoryID INT,

EventID INT,

PublishedAt TIMESTAMP NOT NULL,

CreatedAT TIMESTAMP DEFUALT CURRENT\_TIMESTAMP,

FOREIGN KEY(CategoryID) REFERENCES Category(CategoryID), FOREIGN KEY(EventID) REFERENCES Event(EventID)

)

**Insert news:**

INSERT INTO News(Headline, Source, URL, CountryCode)

VALUES(‘Brexit Party will not stand in Tory seats’,’BBC’, ‘https://www.bbc.co.uk/news/election-2019’, ‘UK’)

**Insert category:**

INSERT INTO Category(CategoryID, Name)

VALUES(1, ‘Business’)

**Insert event:**

INSERT INTO Event(EventID, Name)

VALUES(1, ‘Brexit’)

**Insert tweet:**

INSERT INTO Tweet(Tweet, Hashtag, User, URL)

VALUES(‘Brexit Party will not stand in Tory seats’,’#brexit’, ‘@oryna’,‘https://www.bbc.co.uk/news/election-2019’)

# 2.2 – Algorithms and Prototyping:

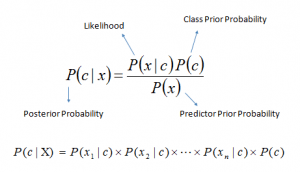
## News Categorisation:

To compete the most left branch of my project overview Dataflow diagram (Page 1) I will need to use a training set to train my model to classify news using an machine learning algorithm. I will then download the trained model and use at as a News Classifier. The algorithm I will use for this section is the Multinomial Naïve Bayes Algorithm.

**Multinomial Naïve Bayes Algorithm:**

To categorise the news into my wanted categories, I must use a machine learning algorithm.

The Naive Bayes Algorithm, which I discovered on Sci-Kit learn, is an algorithm used for classification, that is extremely fast compared to other classification algorithms. I chose to use this algorithm because of its efficiency, accuracy and suitability for text classification, making it perfect for headline categorisation. When given a labelled data set, each instance containing headline and an assigned category, the algorithm uses the Bayes theorem to work out the probability of each word being used for each category. The algorithm assumes that each trait of a class (a word) is independent and unrelated to all other features so all features independently contribute to overall probability. This algorithm often outperforms more sophisticated methods. It uses the following formula for classification:



First of all, the data fed to the algorithm must be normalised and vectorised. Normalised, means removed punctuation converting words to lowercase. Vectorised means all unique words across all headlines are given a unique integer number, each category is also given a unique number. Each headline is coded as a vector, a long list of integers of the same size (the total number of unique words). Each position contains the number of occurrences of the corresponding word in the headline. The resulting two-dimensional array of headline vectors ( X axis value) and corresponding labels ( Y axis value) need to be split into training (80%) and test (20%) sets. The X-train and Y-train are then used as a training set for the algorithm. The trained model (a grid of learnt probabilities) is then tested against X-test and Y-test to compare predicted labels with real labels and calculate the accuracy of prediction.

Overall Advantages:

1. It performs well with multiclass predictions
2. Can perform better than logistic regression
3. Needs less test data than most logistic algorithms.
4. Easy to understand and learn

Overall Disadvantages:

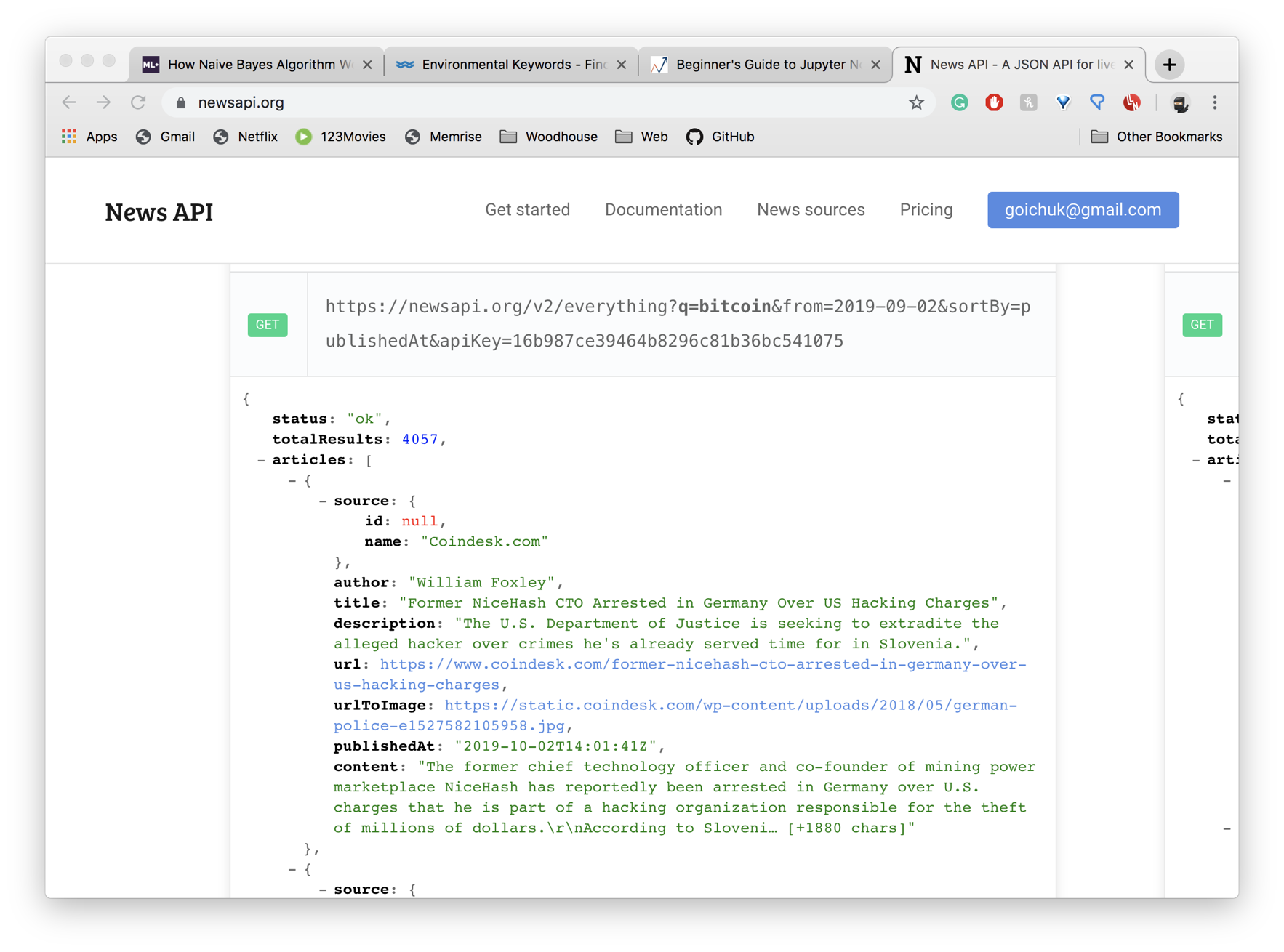
1. Is assumes all features are independent when in real life nothing is truly independent
2. All data must be vectorised and converted into numbers
3. If new categorical variable/feature is seen in test data without existing in the training data, the model will be unable to make a prediction and would assign zero probability.

**Training Data and API:**

To train my model, I will need an excessive amount of training data- a large labelled dataset. The dataset must contain headlines, a category for each headline. The source, URL and the PublishedAt time will be needed for my real data for the site. The source will be needed when scraping the news body from the site to form a summary about the event. The URL is to display it for the user to be able to visit the original news page (and for news scraping). The PublishedAt time is to check now new tbe news is and to keep track of what headlines have gone to the database already, to avoid having duplicated data in the database. However, for training my model I will only need the headline and the category.

The service I will be using is newsapi.org. It will provide me a list of 400,000 headlines and an assigned category (health, business, entertainment and tech). As I am planning to have 9 categories mot 4, I will have to have extra query classification to overlay the category predicted by my model, to further depict the more specific category of the news. I will be doing this to give the user a better opportunity to find the news they want, and as I could not find data with my exact categories, and it would take way too long to categorise a large number of headlines myself, I decided to further query my headlines to more categorisation.

The newsapi service provided me a json file with this structure about each headline

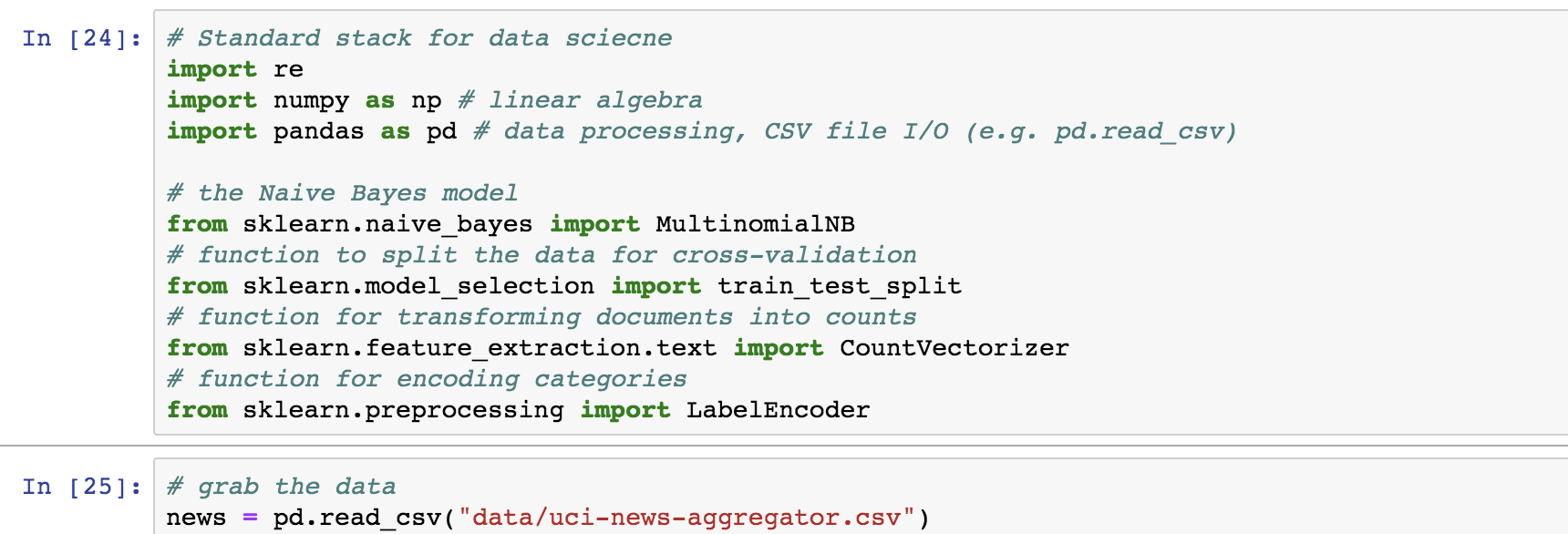
****

The total results showed how many headlines the api has provided me with, (4057 is example data). The articles had a name, id, author, title, url, urlToImage, PublishedAt and the content (a summary). For data training this API is perfect as it provides a large amount of data, however, I there are issues with is when using it for my actual site, as it includes too many sources (some of which might be unreliable), and only feeds 15 headlines at a given time, with large time intervals before it updates any headline or adds any new ones (around 30 mins). This therefore means I will have to use another API that gives me the same content I need but in larger amounts.

Now having the the test data and knowing the algorithm, I was able to train my model.

**Training Model:**

I used the standard stack for machine learning (helping functions from sci-kit learn) to train and test my model. I used this stack as I read about its effectiveness and was then happy with the end result:



I imported numpy, a library that does mathematical calculations including linear algebra, matrices and vectors multiplication, that is used by the Multinomial Naive Bayes algorithm. It is an optimised array that is used specifically for number calculations, unlike standard python lists.

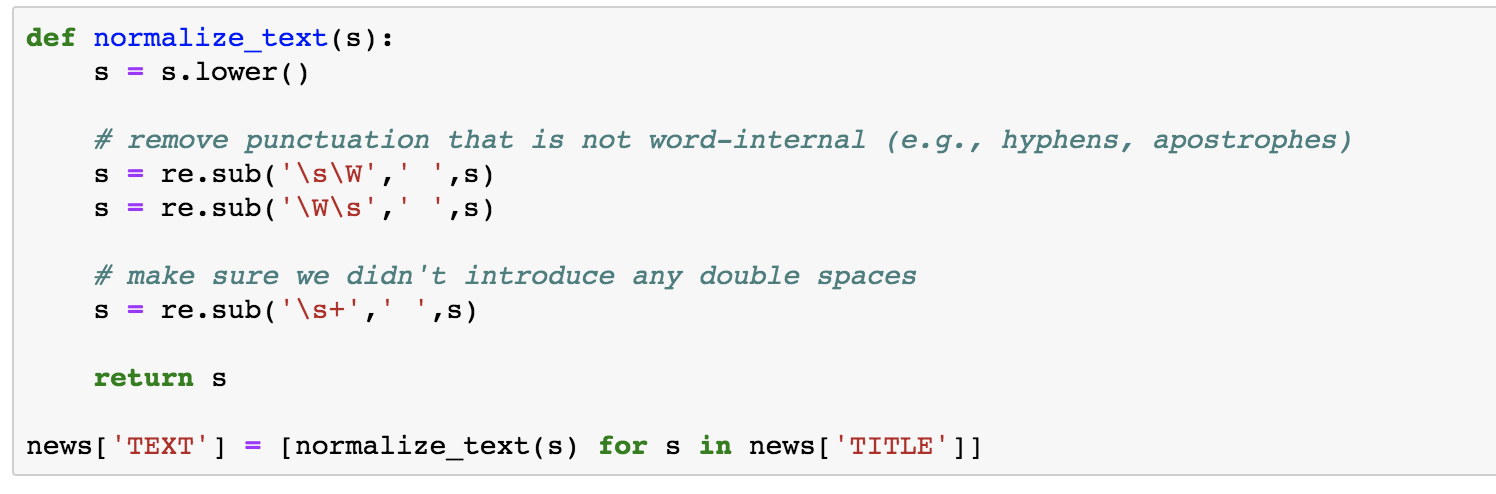
Pandas is a library that allows you to manipulate with tabular memory data/ lists of records, each column being your numpy array.

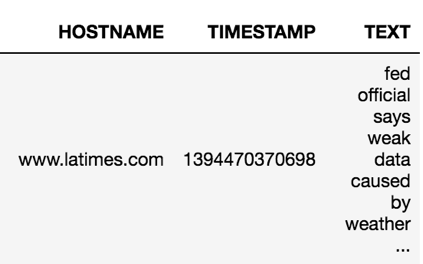
The train\_test\_split helps to split your data set into a training set and a data set (a helper function). You provide the ratio, by which the data is split.

The CountVectoriser function is a common helper method for text processing: turning text into vectors. When given a list of headlines, it finds all unique words across all headlines, which will form a vector space of high dimensionality. After that, the CountVectoriser, turns each headline into a vector in that space, where in each position (word), you have the total of that word repetition in that headline.

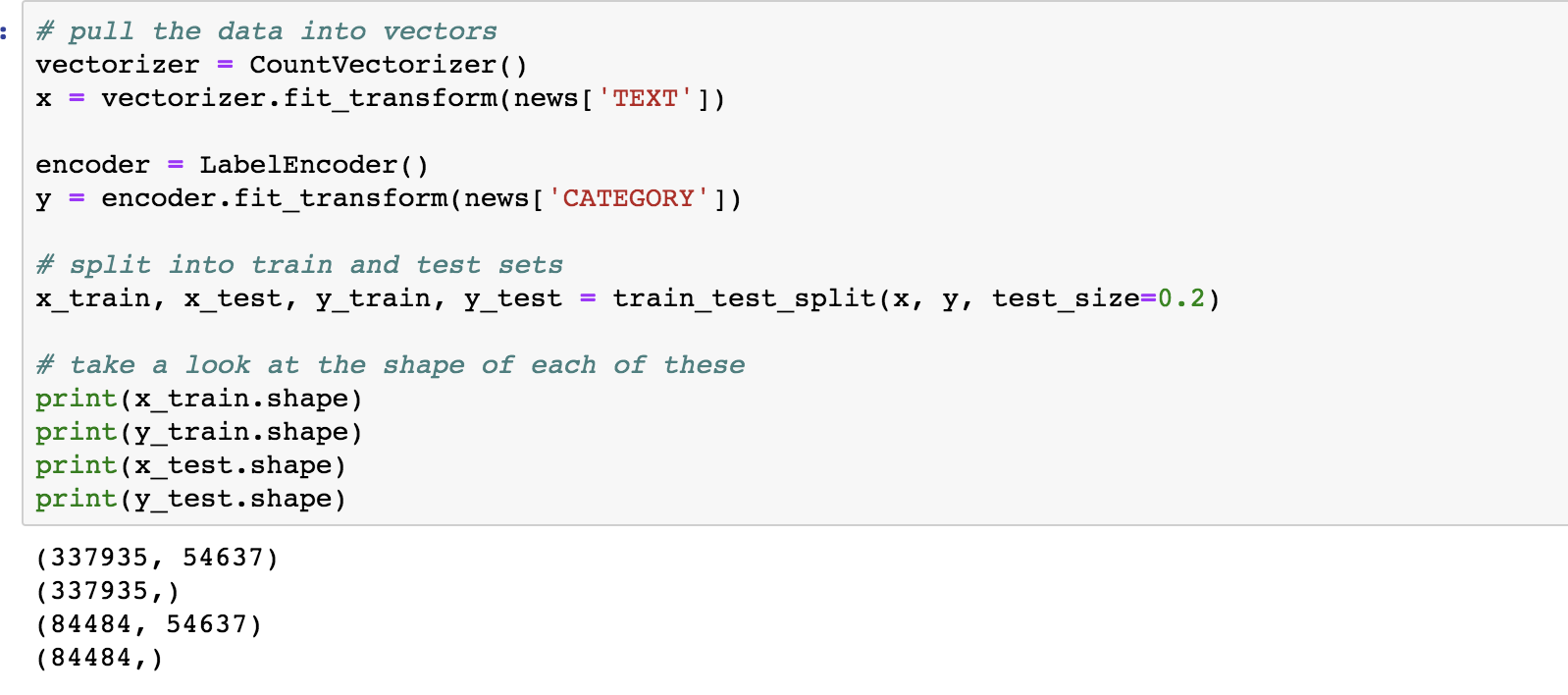
The LabelEncoder assigns a number to each unique label to encode each unique category with a number.

**Normalising Text:**

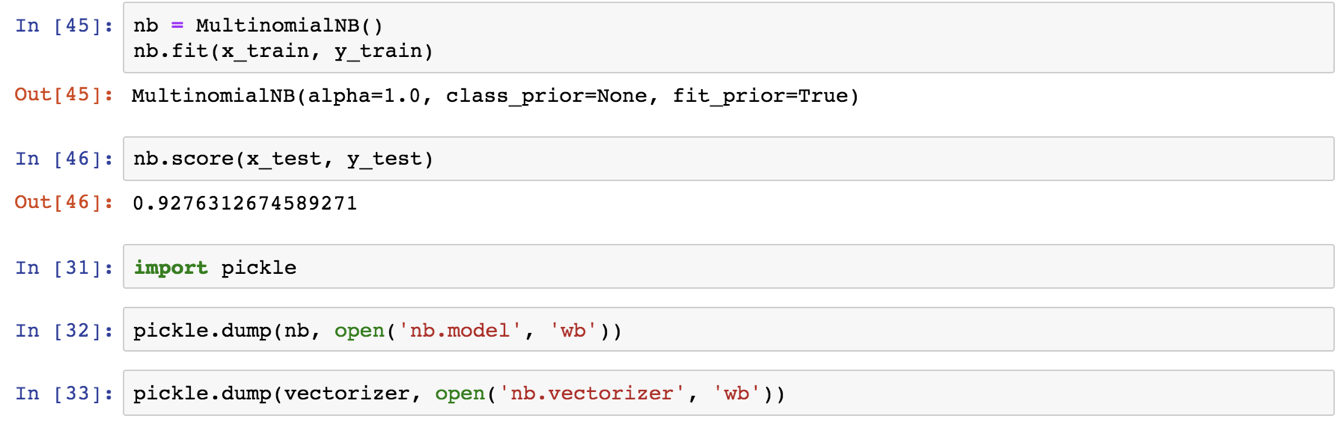
Firstly, to use the data in text processing, we need to clean the data. The data was taken from the News API service. Cleaning the data meant stripping all punctuation, replacing multiple spacing with one space and converting to lowercase. This must be done to more accurately vectorise the data.

****Next were showing the pandas dataframe built from csv file:

Where the first column is the original headline and the last in the normalised headline ready to be vectorised.

Next I vectorised the TEXT column and encoded the category column as a list of whole integers and split the data into a training set and a test set.

When printed, the results outputted is the length of all vectors (number of total unique elements is the length). After our headlines were vectorised, I used the Multinomial Naïve Bayes algorithm to train and test my data, getting a whole 92% accuracy. After this I downloaded my model and vectorizer as a binary file so I then can use my trained model in visual studio.

**Limitation for the data and algorithm:**

The data I used from NewsAPI could only be split into 4 categories (business, entertainment, health and tech) meaning that I will need to categorise other categories through queries or use a different set of data. NewsAPI also only refreshes its news stories evey 15-30 mins, not very efficient for my news site.

The Algorithm can also classify the news into categories, but it cannot detect whether 2 news stories are written about the same event, therefore I will need to design my own algorithm and use natural language processing to define whether 2 headlines by different agencies are written about the same event.

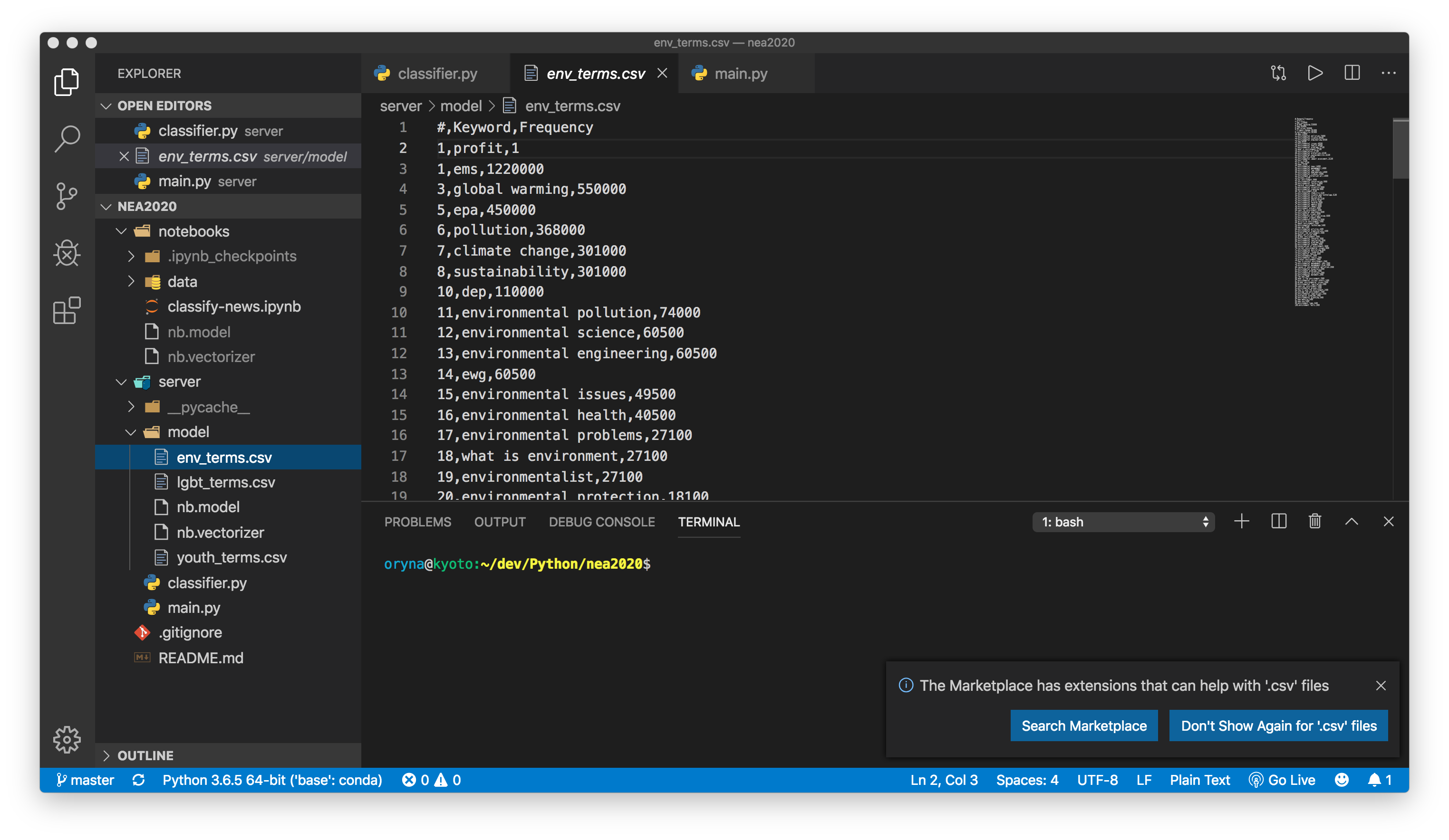
The accuracy also can be higher than 92% is more test data is used but as it is extremely difficult t find a large amount of testing data, I must settle for 92%.

Another limitation is that all data must be normalised and vectorised to be efficient, as without normalising the headlines, the same word written in lower and upper case would be considered a different word.

**Extra Queries:**

To improve my classification methods, I am also planning to use topic keyword queries to classify news into categories to add an extra level of accuracy and effectiveness. I will use keyword files associated with a topic to classify headlines by checking what key terms the headline consists of. This is done also to make up for the fact that I was unable to create my own algorithm for classification, due to its complexities.

This is the environment keywords csv file that will be used for extra query headline analysis. The keywords list for the environment was downloaded and edited and the lgbt category keywords were created fully from scratch containing about 100 words. To make my categorising more efficient, will create an algorithm that also accepts two-grams and three-grams not just one word, so the algorithm will not only track the presence of one word but up to 3 words. This means I can categorise news by the more specific queries eg “climate change” so that it doesn’t make my categorisation less effective. Apart from the keywords, all the other information on the csv file is not needed but I did not remove it for simplicity purposes.



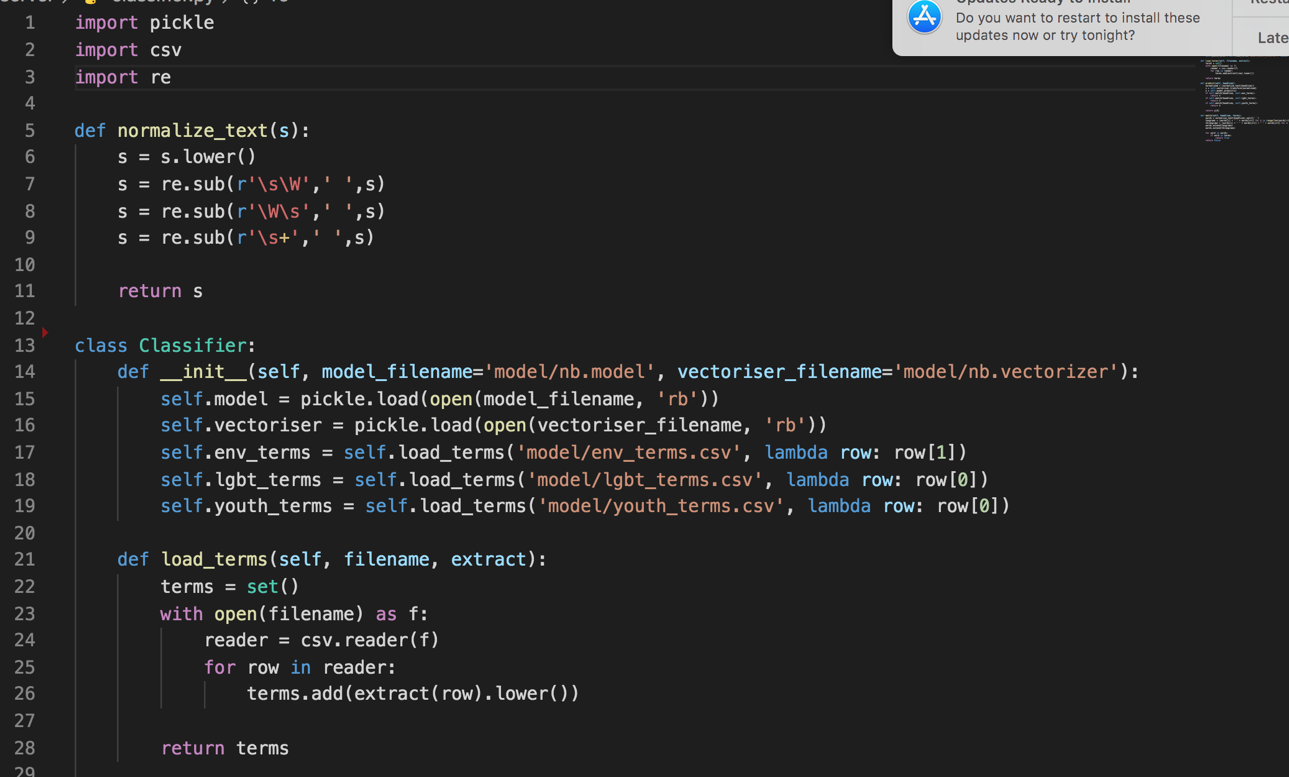
**The Classifier:**

After having a trained model and extra keywords, I was able to create a full news classifier that was able to predict all 9 categories.

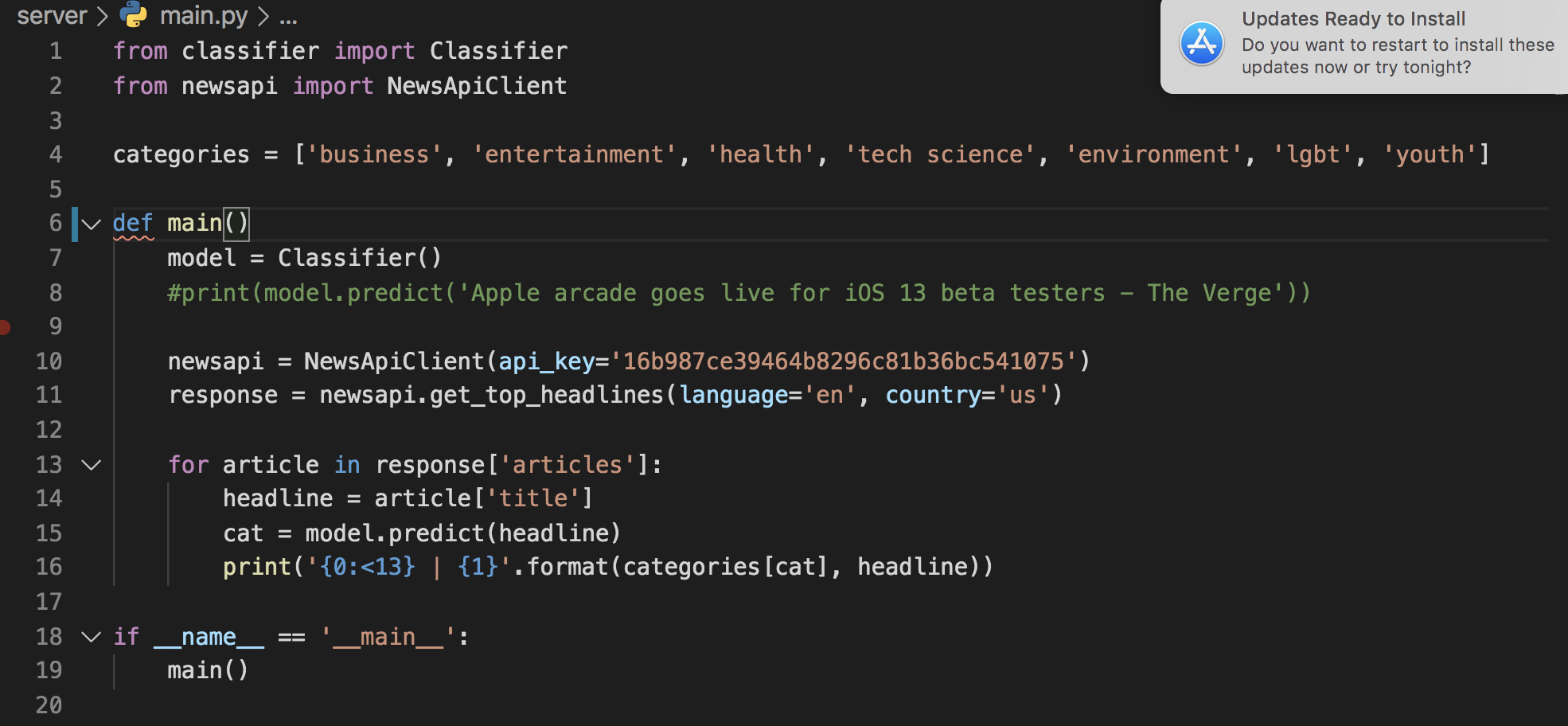
It consists of multiple parts. The first part is the main classifier. Here I will download my model and my vectorizer, ready to work on real data. I will also load my keyterms and create a function that will load the terms. The class will have a separate function for loading terms, predicting categories and matching the keywords (two-grams and three-grams also) from the keyword files and the actual headlines. The class should them return an encoded category.

This is the Class Diagram for my classifier:

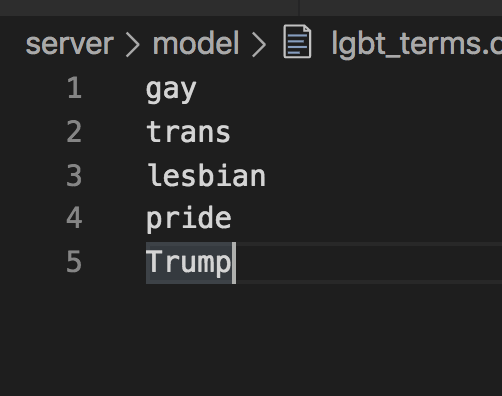
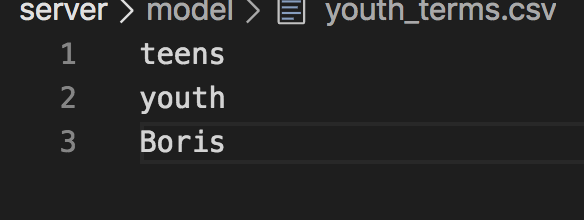
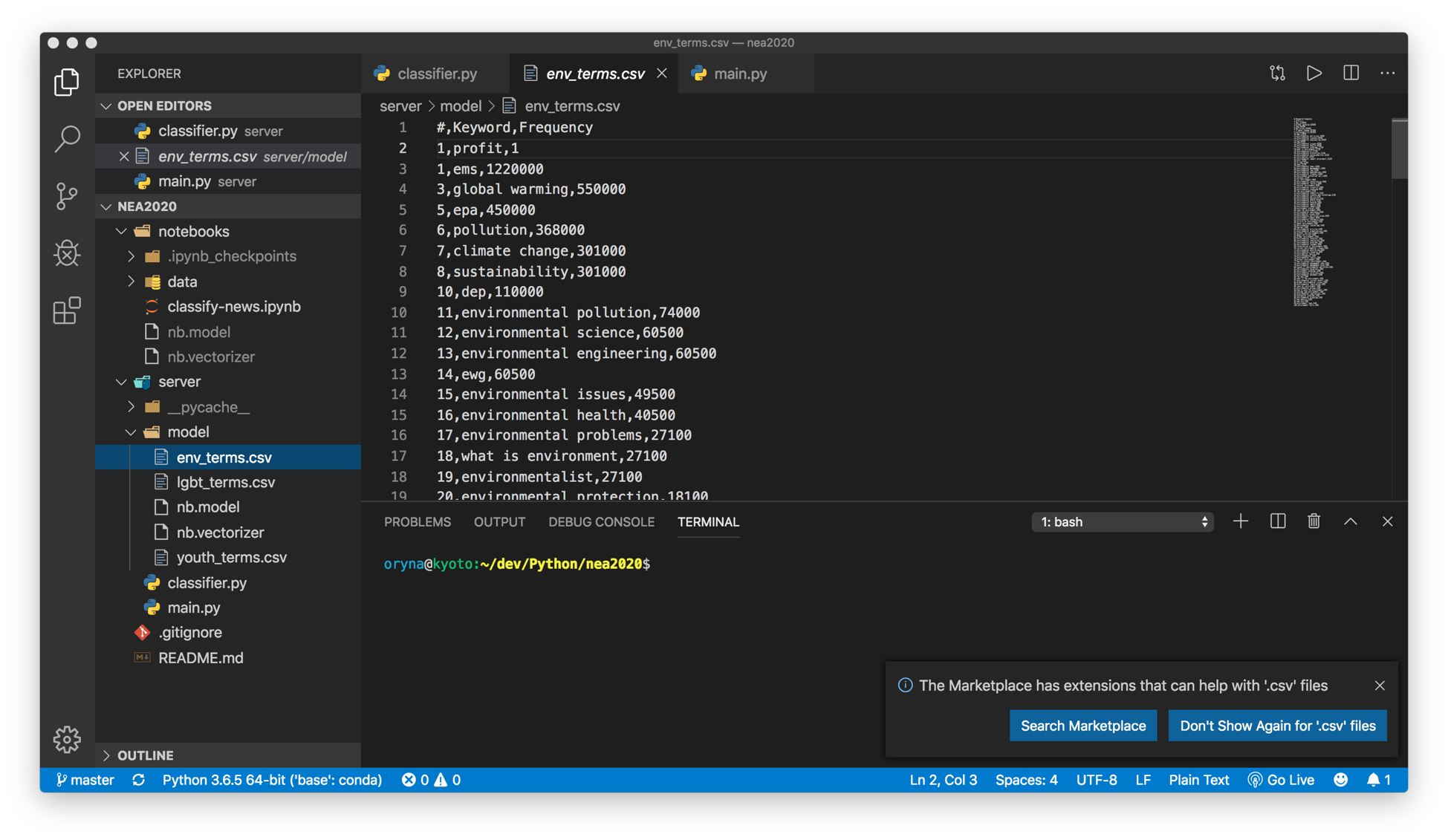
After creating a class diagram, I created the actual class and combined it with other elements to complete my news classifying task. I also created a normalising function to get my data prepared for vectorisation.

Next, I created the predict function, that normalises the text, vectorises the headlines and then predicts it using the .predict() function that is included with the ML model. It will then return 0,1,2,3 if the headlines do not contain any queries from keyword files. The numbers 1,2,3 and 4 correspond to the 4 categories that the model can categorise. If there are other keywords from the keyword file, it will overwrite the category to the category of the keywords. I decided to overwrite the category instead of having 2 to prevent too many duplicated events/headlines in the wordcloud. It also doesn’t matter which category to put the news in as it is in many categories at the same time. The more specific categories of the keywords give a more specific category of the news. If it has keywords from multiple files, it will assign the most latter category to the headline for simplicity purposes.

The .match function looks for 1,2 or 3grams from all given keyword files. It will return true or false depending on whether the word is in the file. This will later be used by the predict function that will return 4,5,6 for the last 3 categories. The current categories in this prototype are: environment, youth and lgbt, but this may be changed later on, as I have already decided to not include ‘youth’ as one of my categories.

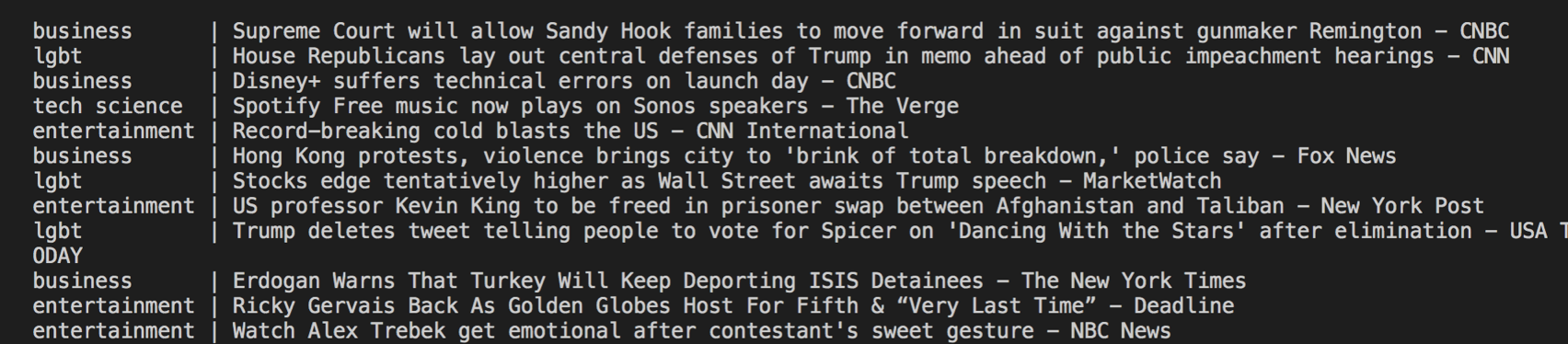
The second part of classifier is the main file that uses an apikey to load the headlines from newsapi.org and filters them by country and language. In the prototype, it only filers for English and USA. It then applies the prediction of the classifier. It also has to load the classifier from another file and assigns the number returned by the classifier to the category.

These are the keyword files I used for my testing:



Lgbt youth environment

To make sure my application works, I used sample words for lgbt and youth terms.

After running the program, I got the desired outcome. My headlines were categorized.

The classifier categorised headlines from both the ML model and the keyword files, as both types of categories appeared correctly.

## Detecting headline duplicates:

After putting news into topic categories, we will find the most important words in each headline by eliminating stopwords (filler words, pronouns, adverbs, article, etc), and counting the number of occurrences of the rest of the words, After I will also apply stemming and lemmatisation to convert each word into its base from (present tense, singular) and by applying names entity recognition, we will be able to detect names, geographical locations, company names etc.

After detecting the most relevant words, I will find headlines with similar relevant word occurrences to group these headlines to be describing the same event. This means that all news articles and headlines from different sources and news agencies about the same event will be collected and as one unit and a tag will be created to mark that event that will go in the wordcloud. This means I will not have any headline duplicates in my wordcloud.

A LOT MORE TO ADD AFTER SECTION CODED

## Generating Short Headlines for wordcloud:

## Twitter Loader:

## News Loader:

## Scraping News Body:

## Summaries of News:

## UI Design & HTML:

More SQL queries

Class Diagram

Talking about dataframes