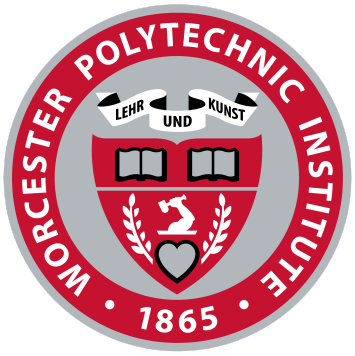


USDA Foodborne Illness Outbreak Detection and Visualization

A Major Qualifying Project
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements
for the Degree of Bachelor of Science

Submitted By:
Katy Hartmann - Data Science
Timothy Kwan - Computer Science/Data Science
Jasmine Laber - Data Science
Anne Lapsley - Data Science/Mathematical Sciences
Liam Rathke - Computer Science

Date:
2023-04-27

Report Submitted to:

Professor Elke Rundensteiner
Professor Oren Mangoubi

Worcester Polytechnic Institute

Abstract

Our project aimed to create the groundwork for a tool that identifies foodborne illness cases from Twitter as the first steps in creating an unofficial warning system to slow the spread of foodborne illness. We collected and stored historical tweets, created visualizations that can display the trends of foodborne illness over time, compared Twitter data to official foodborne illness data, and evaluated the machine learning model in order to set up the framework for this early warning system.

Executive Summary

The Centers for Disease Control explains that the foodborne illness outbreak detection timeline is approximately a 3 to 4 week process dependent on the bacteria associated with the illness. This process includes when an individual comes in contact with the bacteria, to the onset of symptoms (anywhere between a matter of hours to weeks later), to running lab tests, to then being able to attempt to associate the disease with other cases (CDC, 2022). The aim of this project is to attempt to explore the viability of getting early warnings of foodborne illness from Twitter. We attempted to start this process by creating an outline for a future tool for foodborne illness detection and evaluating it against archive tweet data from Twitter and past foodborne illness outbreaks to see if they lined up. With the outline, we can more easily update the tool as needed or help us to understand if we should pursue another method.

Our first step was determining how to collect and store useful data by developing a pipeline. Previous iterations of this project and similar projects used Twitter data, so we decided to do the same. We chose to collect historical Twitter data so we could compare it to ground truth data and past foodborne illness outbreaks, to be able to assess the viability of the tool. Using the Tweetkit Python package we created a specific query to run on the Twitter archive in order to obtain tweets that were most likely related to foodborne illness. The second step in creating our pipeline was developing a system for storing our data to be easily queried later. We created a PostgreSQL database with a simple schema to store our tweets and the attributes we deemed necessary. We made sure to choose the appropriate data types for each attribute. Then we created a pipeline that would perform the following tasks in sequence. First, the pipeline efficiently retrieves tweets from the Twitter archive that may indicate a case of foodborne illness. These tweets are passed through a refined BERTweet machine learning model, which will provide a

prediction on whether or not the tweet indicates a foodborne illness case. The tweets and their predictions are then stored in our database, including symptom and food entities identified by the machine learning model. We set up a system that would collect tweets cyclically through a specified time interval. We also performed the appropriate data cleaning before data was inserted into our database.

We then sought to create effective visualizations that displayed the trends in foodborne illness. We created dynamic histograms that displayed results for various symptoms and food items over time, allowing us to track the difference in frequencies over time. We also plotted geo-coordinates for each tweet with a discernible location on a heatmap, so that foodborne illness discussion hotspots stood out immediately. Where possible, these visualizations were served on a website using data directly from our database.

Our final step was to evaluate the usefulness of Twitter data in indicating foodborne illness cases and outbreaks. In order to determine the validity of our Twitter data, we compared it to ground truth data from the National Outbreak Reporting System [NORS], the Center for Disease Control [CDC], and the Food and Drug Administration [FDA]. In comparing our collected Twitter data to real outbreak data we were able to begin to assess whether this system could work. We also evaluated the performance of the BERTweet machine learning model by comparing the predictions to hand-labeled data, to determine if there could be improvements for the machine learning model.

We created the foundations of a system that could be used in the future to analyze discussions of foodborne illnesses on Twitter and Twitter-like websites. With access to more data in the future, this system could help policymakers and public health analysts discover new

foodborne illness events to investigate and warn the public of potential outbreaks before they are officially declared.

Acknowledgments

We would like to thank the CDC's National Outbreak Reporting System (NORS) for providing us with ground truth foodborne illness case data. We would also like to thank our advisor Professor Rundensteiner and Ph.D. students Ruofan Hu and Dongyu Zhang for all of their help.

We acknowledge USDA grant #2020-67021-39133 for supporting the collaborative project entitled FACT: Innovative Big Data Analytics Technology for Microbiological Risk Mitigation Assuring Fresh Produce Safety, between Professor Rundensteiner of WPI and Professor Hao Feng of North Carolina Agricultural and Technical State University, as this grant supports the Ph.D. students in their work, and gives them the time to help mentor our team.

Table of Contents

List of Figures and Resources

Authorship

Section	Author	Editor
Abstract	Jasmine	Anne, Katy
Executive Summary	Jasmine	Anne, Katy
Acknowledgments	Jasmine/Timothy	Anne
1. Introduction	**See Below:**	
1.1 Motivation	Jasmine	Anne, Katy, Liam
1.2 Objectives	Jasmine	Katy, Liam
2. Background	**See Below:**	
2.1 Food Science and Foodborne Illness	Anne	Katy
2.2 Tweet FID	Anne	Jasmine, Katy
2.3 Machine Learning Model	Jasmine/Anne	Katy
2.4 Prior MQP Work	Katy	Jasmine
2.5 iWasPoisoned	Anne	Katy
3. Methodology	**See Below:**	
3.1 Data Collection	Anne	Katy
3.2 Database	Jasmine	Katy
3.3 Data Processing	Timothy	Jasmine, Katy
3.4 Pipeline Setup	Jasmine	Anne, Katy
3.5 Front End	Liam	Katy
3.6 Forecasting Tweets	Anne	
3.7 Predicting NORS Cases	Anne	
4. Results	**See Below:**	
4.1 Twitter Data Collections	Timothy/Anne	Jasmine, Katy

4.2 Food Token Analysis and Comparison	Jasmine/Timothy	Katy
4.3 Frontend Visualizations	Liam/Timothy	Katy
4.4 Machine Learning Model Performance	Katy	Jasmine
4.5 Forecasting Tweets Results	Anne	
4.6 Predicting NORS Cases	Anne	
5. Future Works	**See Below:**	
5.1 Word Importance Analysis	Timothy	Katy
5.2 Collaboration with iWasPoisioned	Timothy	Anne, Katy
5.3 Streaming Tweet Collection	Timothy	Anne, Katy
5.4 Frontend User Interface	Liam	Katy
5.5 Forecasting and Predicting	Anne	

1. Introduction

1.1 Motivation

Foodborne illness is a widespread issue: approximately 1 in 6 Americans get sick every year. The CDC outbreak detection process tends to take weeks and lab testing can be slow, costly, and potentially inaccessible, so often the word does not get out until after many have already gotten sick (CDC, n.d). An early warning system could slow the spread of foodborne illness. The goal of this project is to continue developing an efficient framework to create and evaluate a tool to warn of the early signs of foodborne illness outbreaks.

1.2 Objectives

Our first objective was to create an efficient pipeline that will collect the appropriate foodborne illness related data. To do this, we created a suitable query made up of keywords that will retrieve tweets from Twitter that are most likely related to foodborne illness. After retrieving these tweets, we passed them through our machine learning to get predictions on whether or not each tweet indicates a case of foodborne illness. This data was then cleaned and stored appropriately so it could be easily retrieved for analysis.

Our second objective was to create a website that contains user-friendly visualizations. We want to convey our results to both laypeople and those involved in public policy or public health, and by displaying interactive results on an easily accessible webpage, we hope to make it easy for users to understand what foods and symptoms commonly related to foodborne illness, and for them to see which areas of the US are most impacted.

Our third objective was to explore the validity of our collected Twitter data in indicating foodborne illness outbreaks. We compared the data we collected for our pipeline to ground truth

data obtained directly through the CDC/NORS. By first preparing the data to be in comparable formats, we performed analysis to see how closely our data collected relates to real foodborne illness cases.

Our fourth objective was to evaluate the performance of our machine learning model on the data that we collected. To do this, we hand-labeled data as ground truth data and compared it to what the model predicted to see how accurate the predictions were.

2. Background

2.1 Food Science and Foodborne Illness

Food science is a discipline that covers a wide variety of topics related to food, from nutrition to food safety to development. This project focuses on the food safety side of this discipline as that is where foodborne illness falls. Foodborne illness, generally, is from bacteria in food that causes different illnesses and infections. There are many causes of foodborne illness such as lack of cleanliness, improper preparation, improper storage, etc. Foodborne illnesses can be caused by many different bacterias, some of which are more harmful than others. (USDA, n.d). Agencies like the CDC, FDA, and USDA focus on different aspects of food safety relating to foodborne illness, from precautions put in place to prevent foodborne illnesses, to detecting and reporting foodborne illnesses and processes that help stop them from spreading.

To identify an outbreak the Center for Disease Control (CDC) collects data such as locations food was acquired (restaurants, grocery stores, farms, etc.), type of food, and specific bacteria. (CDC, 2022) The CDC explains that the foodborne illness outbreak detection timeline is approximately a 3 to 4 week process dependent on the bacteria associated with the illness. This process includes when an individual comes in contact with the bacteria, to the onset of symptoms

(anywhere between a matter of hours to weeks later), to running lab tests, to then being able to attempt to associate the disease with other cases. The process is dependent on the individual actually going to a doctor to get tested for an illness. If that is not the case, and the patient doesn't get tested, then there is less data to correlate to an outbreak. The end goal of identifying an outbreak is to be able to respond accordingly, either by issuing recalls, warning the public, or closing contaminated food facilities, which in turn will minimize the spread of the outbreak.

In 1996, the CDC coordinated PulseNet, a network of laboratory agencies with a common set of practices with the goal of identifying clusters of illnesses. (Boxrud, et al., 2010) They use pulsed-field gel electrophoresis to attempt to identify illnesses with similar origins. Due to the standardized protocols across all laboratories, PulseNet can share results across state lines. This creates quicker communication and a larger dataset to look at to potentially identify multi-state illness outbreaks. PulseNet has many different programs to monitor different aspects of foodborne illnesses. These programs include *VetNet*, testing for Salmonella and other diseases animals can carry, *FoodNet*, performing active surveillance for foodborne illness, and *OutbreakNet*, created to facilitate rapid communication during an outbreak. Due to a lack of funding and resources, PulseNet cannot monitor and test every sample that comes to them, but that means that there are possible missing data for potential outbreaks.

The improvements in technologies have created potential new avenues to examine foodborne illnesses and their spread. Machine learning algorithms exist that can identify common foodborne illnesses with pretty high accuracy. When creating a self-reporting platform for foodborne illness Du and Guo described "Hence, a foodborne disease pathogen prediction model using an extreme gradient boosting (XGBoost) model is designed to aid in the diagnosis of foodborne pathogens [...] the model achieved a 69% accuracy on Salmonella, Norovirus,

Escherichia coli, and Vibrio parahaemolyticus infection diagnosis". (Du, Guo, 2022) These technologies create an interesting possibility for an outbreak detection system that is based on self-reporting illnesses. Models like these aren't as accurate as lab testing would be, but they do reduce the time and cost of reporting foodborne illness and the step of having to visit a doctor could potentially be removed from the process. This project, similarly, sets out to attempt to identify potential foodborne illness outbreaks through social media posts. Social media posts are a form of self-reporting illness, in an indirect way.

2.2 Tweet FID

The Tweet-FID dataset is a dataset of tweets collected that are potentially related to foodborne illness. (Hu et al. 2022) These tweets were gathered by searching for specific keywords and hashtags that could be indicative of foodborne illness. This dataset was then taken and crowd-source labeled, using MTurk, to identify 5 things: what [foods], symptoms, where [location], and other related keywords, as well as whether or not the tweet indicated foodborne illness. The labels were checked for quality of labeling and the procedure was adjusted to attempt to minimize the amount of labels deemed low quality. Experts within the food science discipline were then asked to label the same dataset of tweets to create a ground truth labels dataset to compare the crowd-sourced dataset to. This data set was then used to train a machine-learning model with different tasks, illustrated in Figure 2.1. The model is able to identify different entities within a tweet allows us to use that information for analysis later in the process.

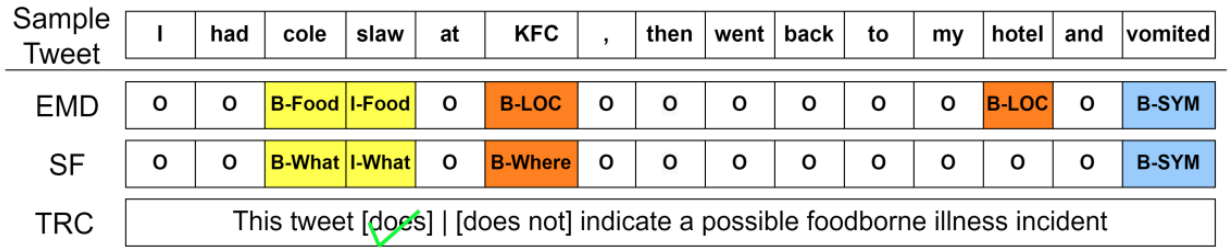


Figure 2.1: *Sample tweet with sample outputs from three different tasks Text Relevance Classification (TFC), Entity Mention Detection (EMD), and Slot Filling (SF) (explained in more detail in the next section),* (Hu et al. 2022)

2.3 Machine Learning Model

Machine learning, which is a subfield of artificial intelligence, generally aims to replicate intelligent human behavior (Brown 2021). Machine learning models can be used to generate predictions based on given data. In order to train machine learning models, they must first be trained on designated training data, and then tested against validation data. Only after this can the models' accuracy be evaluated on the remaining data, known as the test data. This process is used in order to avoid data snooping, which can lead to a model that performs only well on our selected data, and not necessarily in general (Halloway, 2019). In our case, we are interested in using machine learning to classify social media posts to identify foodborne illness incidents and different factors involving the incident. Training a machine learning model on labeled foodborne illness tweets can result in a strong tool to identify cases of foodborne illness.

RoBERTa, a pre-training method for language models, was created in order to improve upon the pre-existing BERT training model. The name RoBERTa was chosen as the approach can be described as Robustly optimized BERT (Liu et al., 2019). The four main objectives of the RoBERTa model were to have the model trained longer with more batches, to remove the next

sentence predictor, to train on longer text sequences, and to change the masking process. In order to improve upon BERT more, further research was conducted on what data to use for pretraining and the number of training batches. The evaluation of the RoBERTa model showed improvements compared to BERT in terms of performance on GLUE, RACE, and SQuAD.

BERTweet is a pre-trained RoBERTa model using Twitter data that can be used for downstream tasks: text classification and name-entity extraction. We used a fine-tuned BERTweet model attempted to achieve the following tasks: first, classify if a tweet indicates a foodborne illness incident. Second, determine critical entities related to the incident (food, location, symptoms, etc.). This model, therefore, helps improve on past studies as, in addition to binary classification (a tweet indicating foodborne illness or not), the model also provides insight into the entities of tweets. The performance of BERTweet was then measured on the testing data (Tao et al., 2021). BERTweet was able to achieve an accuracy of .87, which was much higher than previous models. However, in the process of identifying critical entities, it was difficult to collect location information due to the scarcity of geolocation data. Looking more into obtaining accurate location data can help improve this model's effectiveness in helping to provide warnings of foodborne outbreaks.

As for the training and testing with the Tweet-FID dataset, once the labeled dataset Tweet-FID was created, it was able to be used to train and test deep learning models utilizing different methods of performing the same tasks as the human labelers. (Hu et al. 2022). The tasks performed by the model were Text Relevance Classification (TFC), identifying whether or not the tweet was related to foodborne illness, Entity Mention Detection (EMD), identifying all mentions of target entities (any mention of foods, locations, symptoms, and keywords, not just relevant mentions), and Slot Filling (SF), extracting text identified to fill a slot (what, symptoms,

where, keyword). These tasks were then tested in different combinations using different deep learning models to see what model and combination of tasks perform the best at labeling and identifying the tweets. In order to train this model, 3000 tweets were broken up into a testing, training and validation set. 2400 tweets were used in the training set, 300 in the validation set, and 300 in the testing set.

2.4 Prior MQP Work

This project is a continuation of a previous year's MQP project. The past group wanted to identify, collect, and analyze data to create a display with relevant information about foodborne illnesses. They also wanted to create a framework that allowed the collected data to predict a foodborne illness outbreak. In order to do this they first decided on and created a database to store this data. They chose to use PostgreSQL because it would be easy to maintain the data's integrity and consistency. They were very focused on determining geolocation for tweets using various inference methods. The columns include location data along with the individual foods and symptom tokens.

They used various sources for data collection. They wanted to expand the focus beyond just tweets, but also data from other websites. The main websites they searched were iWasPoisoned.com and the CDC. The first website allows users to report their own experiences with food poisoning with the newest reports showing at the top of the site. Users are able to report symptoms, locations, ingredients/foods, and a description. iWasPoisoned would publish how long ago the report was created and then the team used a web crawler and used it to convert the time since into a timestamp. From the CDC website, they had access to all official foodborne illness outbreak reports and the web crawler was able to store relevant links the team could use

and reference later. They also utilized the Twitter API. They searched Twitter for relevant keywords and time periods. From this, they collected account IDs, time of posting, geo-tagging, tweet ID, language, metrics, and text.

For the front end of the project, they decided to build the application using ReactJS along with Bootstrap 4. On their website, they created four main pages which include the home page, explore page, about page, and tracker page. On the homepage, their main goal was to inform the user about the project. This included data visualizations and various infographics. The explore page allows users to explore more of the data that was collected. A word cloud was created to allow an outbreak to be selected so the common words used can be explored. The about page includes information regarding the team, a timeline of the work progression, and the sponsor/partner of the project. The last page, the tracker page, includes additional maps, charts, and outbreak descriptions. This page is primarily used for more of the data to be observed from the sources.

2.5 iWasPoisoned

iWasPoisoned.com is a website that was created in 2009 to crowdsource individual foodborne illness cases from the public. Its tagline is “Report a food safety issue. Protect others.” iWasPoisoned has the user input their location and symptoms as well as a rating of the restaurant and a description/comments (*iWasPoisoned.com*, n.d). All of this is with the intention of informally warning the public of foodborne illness without it having to travel through official channels, like the CDC, or even hospitals. In 2017 iWasPoisoned created a secondary Twitter account (secondary to their branded Twitter account, created in 2009) with the sole purpose of posting every entry on their website onto Twitter. From our collections, it appears that

iWasPoisoned did not start tweeting from this account until the start of 2020, but starting in 2020 a large portion of the positively identified tweets the machine learning model picked up on were from iWasPoisoned.

3. Methodology

3.1 Data Collection

3.1.1 Data Source

Depending on the definition many websites can fall under the term “social media”, which left us with many data sources to choose from. We considered using a platform such as Yelp or Google Reviews, but Google Reviews has rate limits that would be hard to work around and Yelp’s keyword search returns restaurants, not specific reviews so scraping for relevant reviews would be difficult. There was also the potential issue of having to retrain the model on different data since it was trained on tweets, so we decided to continue to use Twitter. (When we made this decision there was free access to the Twitter API from any account, which was changed on February 9th, 2023).

The tweet collection method we used was using the python package Tweetkit (julianfssen, n.d). We chose to use Tweetkit because it was an already built system that accomplished what we needed it to. We built upon the Tweetkit package to fit it into our pipeline and customized it for what we did and did not want to collect. Using Tweetkit we could collect all of the data associated with a tweet and the user associated with it. This includes attributes such as tweet id, user id, user name, geo-location for the profile and the tweet if available, and entities (hashtags, URLs, etc.) as well as much more information (“Tweet objects”, n.d). For each tweet predicted by the machine learning model to be foodborne illness (positively predicted

tweets) we chose to keep the tweet id, author id, tweet text, place name (either a city and state or just a state, split into city and state columns in the database), and coordinates within our database. On top of that, we also have a raw data storage system that keeps all of the predicted tweets positive or negative. In this system, we save all of the same information, as well as the full user (can include information like follower count, profile information, whether the account is verified, as well as other things) and geo (can include information like place type, coordinates, name, and full name, as well as other things) information ("User objects", n.d) ("Geo objects", n.d). Using these data storage methods we can access and use our data in different ways.

The Tweetkit package is able to collect from the tweet archive, as well as the tweet stream. For our project we decided to focus on the tweet archive because it takes time for the CDC to declare an outbreak, so ground truth data would not exist for the data we collected and therefore it could not be verified. For future implementations of this project, the goal would be to use the streaming tweets and be able to real-time predict outbreaks as they happen. This would require collecting and predicting tweets as they come in to attempt to identify an outbreak as it is happening.

3.1.2 Specifics about our collection method

We decided to only collect tweets with location data because if an outbreak were to happen the location of the outbreak would be important information to have to take preventative measures. Collecting only tweets with geo also gives us the ability to filter out tweets from outside of the United States, since our project is focused within the United States and our ground truth data, which will be discussed later, is from only the United States. Hypothetically tweets without geolocation could help identify an outbreak if the volume of tweets shows an outbreak, but that may not be helpful in acting on an outbreak if its location is not known. We also only

collected tweets that are written in English, since the machine learning model is trained on English sentences.

The data collection works by setting a start date and an end date. The script then collects tweets in 24-hour intervals, querying tweets that match the keywords: ‘#foodpoisoning', '#stomachache', '"food poison"', '"food poisoning"', 'stomachache', 'vomit', 'puke', 'diarrhea', '"the runs"', 'nausea', '"stomach cramp"', 'nauseous'. These keywords attempt to narrow down our tweet collection to more relevant tweets. We chose these words as they are common symptoms of foodborne illness. Foods and locations are not going to be as universally related (i.e adding “chipotle” as a keyword would probably collect more tweets that are not relevant to foodborne illness because not every tweet about chipotle is going to be about illness).

The tweet collection could be set to run for a certain amount of time, and that time frame could be broken into smaller intervals. For example, it could run for a week-long time but collect in hour-long intervals. We ended up collecting in week-long intervals, broken into seven-day-long collection periods. This was done in case Twitter rate limits or system failures caused issues for the collection, so at least parts of it would be saved. We did some trials to make sure the collection remained consistent in the amount of tweets over longer periods of time, and then the same interval broken up. We found that collecting in intervals produced the same amount of tweets as the longer time frame, so collecting like that would not impact how much data we were receiving, but it may mean more is saved in the case of something going wrong. Each interval batch is saved as a data frame, then put in a queue for later analysis. The queues are stored locally on the server that it is run on, to more easily retrieve it later.

3.1.3 Choosing Ground Truth Data

Within the scope of this project there are two types of ground truth data, there is data that is hand labeled by humans which we spoke about in our background section of the project about the Tweet-FID data to train/test the machine learning model, and then there is also the data of actual outbreaks. The ground truth data relating to real-world outbreaks can be used to see if the trends in the Twitter data correlate to real-world data. We identified two potential places to acquire that data from, both coming from the CDC. One of them is the CDC List of Multistate Foodborne Outbreak Notices. This is a list of outbreaks/recalls identified by the CDC from 2006. For each outbreak, it lists what food it is associated with, what illness/bacteria, how many cases/hospitalizations/deaths, recall status, general information, as well as a map and a timeline ("List of Multistate Foodborne Outbreak Notices", n.d). Exploring this data, it has a few drawbacks. First, it only lists multistate outbreaks, so smaller outbreaks won't be reported on this dashboard. The other potential issue is that the data for when cases are from is not readily available. The data from 2021-2022 is available to download for individual outbreaks, but prior to 2021 that data is only available on an image of a bar graph, so it cannot be scraped, nor can it even be collected by hand, especially not for larger outbreaks. Another issue with this data is that due to how recent the data is combined with how long outbreaks last sometimes, the data seems incomplete. In Figure 3.1 it can be seen that there is a spike in the July 2021 - October 2021 which seems to be due to the fact that the data for January 2021 - June 2021 may be associated with earlier outbreaks, and the data from November 2021 - December 2022 will end up being associated with later outbreaks. That is all data we do not have access to, and therefore we decided not to use the CDC for the number of cases, but rather just the timing of outbreaks.

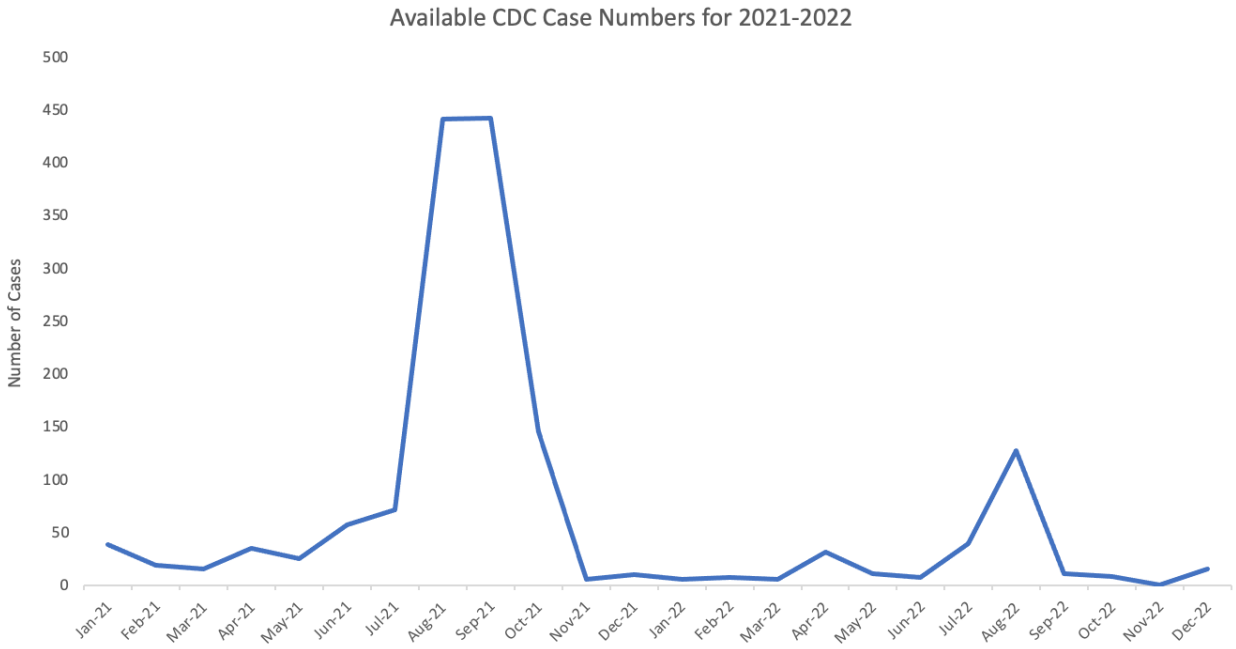


Figure 3.1: *Graph of CDC data available for 2021-2022*

The other option we have for ground truth data is the National Outbreak Reporting Service [NORS]. NORS is a reporting system for state/city/local officials to report outbreaks. This is for any outbreaks of foodborne, waterborne, animal, environmental, and person-to-person illnesses. The data that is available for each outbreak is Year, Month, State, Primary Mode, Etiology, Serotype or Genotype, Etiology Status, Setting, Illnesses, Hospitalizations, and Deaths. We can then filter by foodborne illness and get all reported outbreaks, confirmed and suspected for a certain timeframe (NORS, n.d). One of the downsides to the NORS data is that it is only available for years before 2021. We had originally collected tweets from 2018-2022, but we soon realized that it did not line up with the ground truth data that we had access to. So we went back and collected the tweets from 2017 and requested data from NORS for 2021, to maintain our 5-year timeframe. One issue that we ran into with the NORS data was that there was only a start date for the illnesses and not an end date so that means our data is skewed to the left of what the actual time frame of the illnesses is. We were later able to acquire end dates for the illnesses as

well, but we still do not know the spread of the number of cases across the time frame to get an accurate day-to-day case count. In the following graphs the cases are binned by month based on the date first ill. Approximately 20% of outbreaks start and end in different months, but due to time constraints and lack of information on day-to-day case numbers, we chose to keep the dates based on the date first ill for consistency. An alternative for further use of this data could be averaging case numbers over the illness time frame to estimate day-to-day totals for a more accurate idea of case numbers over time.

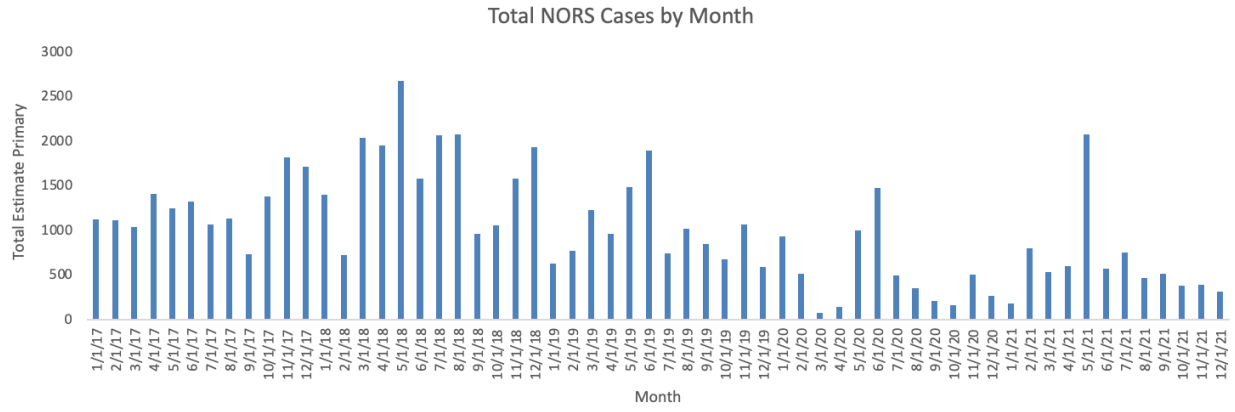


Figure 3.2: *Graph of NORS data 2017-2021*

3.2 Database

In order to store a large amount of data that could be easily and quickly accessed, we decided to create a database. By doing this, we are able to store large amounts of Twitter data that can be easily queried for our front-end visualizations. The previous iteration of this project used a different database schema design, with a different table for location-specific data. In order to ensure our queries would run easily and simply, we opted for a more simple approach. We used the pre-existing MQP server to create a new PostgreSQL database called “MQP22”. Within

this database, we created one table which we named “final_tweets". The following diagram summarizes this setup:

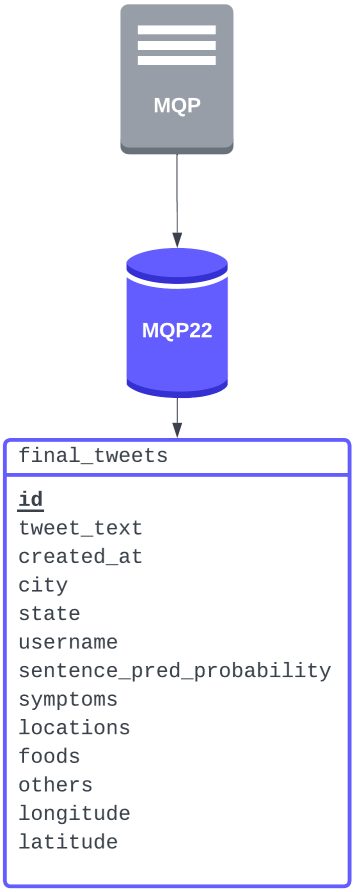


Figure 3.3: *Diagram that displays database setup and table attributes*

We chose to use a PostgreSQL database as opposed to MySQL or others, as PostgreSQL has more complex data types. Specifically, PostgreSQL has the array data type, which is useful for storing multiple values (of any specified type) in one singular column. We wanted to use this data type to store our ‘symptoms’, ‘locations’, ‘foods’, and ‘others’ values, as these all contain multiple token words per tweet. Storing these values in an array allows for easy querying later, as we can unpack these arrays and get counts of tweets that contain specific food or symptom entities. For example, if we wanted to look at the number of tweets that contained the symptom

“nausea”, we could unpack our ‘symptoms’ arrays and then get the count of tweets that contained this symptom. This proves particularly useful for our planned visualizations on our website, as well as a general analysis of popular keywords contained in our collected tweets.

As mentioned above, our schema consists of one table named ‘final_tweets’. This table contains 13 fields, each of which we consider useful for querying on our front-end. For each of these fields, we carefully chose the appropriate data type that would be most efficient for front-end use, as well as database storage. The ‘id’ is the primary key of our final_tweets table, as it uniquely identifies each individual tweet. Querying the count of ids can be used to get tweet counts based on different factors. We stored the ‘id’ as the bigint type as the integer type is ideal for storing ids in a database, and we found that the regular integer type was not large enough to store some of our tweet ids. The ‘tweet_text’ column is of text type, as we found issues trying to limit storage to a certain number of characters. The ‘created_at’ column stores tweet timestamps as PostgreSQL timestamps so that we can easily query data over certain time intervals. The ‘city’ ‘state’, and ‘username’ columns are both of type varchar(100), as to ensure that these values will be stored without taking up too much space in our database. The ‘sentence_pred_prob’ is stored as a numeric type, as it has a large number of digits past the decimal point that we want to preserve for precision. As mentioned above, we stored our ‘foods’, ‘symptoms’, ‘locations’, and ‘others’ entities as text arrays. Lastly, ‘longitude’ and ‘latitude’ were both stored as numeric types in order to account for precision.

We decided to only retain these 13 fields in our database in order to optimize queries for our front-end. The columns here contain data that we need to display in front-end visualizations or simple statistics. The data collected in this database is also only tweets that were predicted by our ML model to be foodborne illness related. In order to perform analysis later of all of the

tweets that ran through our collection process, we implemented raw data storage that contains all of the fields we initially collected from Twitter.

3.3 Data Processing

We saved two sets of the same predicted output data, one set is strictly for the frontend user interface, and the other set is raw unprocessed data used for in-depth analysis and backup in case something goes awry in the pipeline. Before inserting the collected data into the database for front-end use, there are a couple of data processing steps we performed.

Firstly, since the frontend interface will only examine the positively predicted presence of foodborne illness in a tweet, we filtered the data so that we only keep tweets that have a sentence prediction equal to 1.

Next, we created new columns based on the existing columns for more efficient database querying. Longitude, latitude, city, state, and country code columns were created by extracting information from the large “geo” column which was a dictionary data type. Some of the tweets did not have specific location data, so sometimes this left some of the new columns blank. For example, if the location was only listed as the state of Ohio, then there would be no city, state, longitude, or latitude columns. We chose to not include longitude or latitude information of states because certain states like Michigan, Florida, and California might produce a longitude and latitude that is located in the ocean or a lake. This is due to the fact that the Twitter geo data stores the coordinates in terms of a bounding box, so in order to get a single coordinate, we calculate the center of this box. The other new column produced was the “username” column which was extracted from the “user” column. The “user” column was of type dictionary, and we

chose the much more simple username of type string because it will be much more efficient to query and because we only need the username information in the frontend interface.

In the frontend interface, we will only be examining documented cases of foodborne illness from the US, so we will use the newly created “country code” column to filter out any countries that are not The United States.

The last step is to drop all the unneeded columns before inserting them into the database to conserve memory. We dropped the following columns: author_id, geo, country_code, user, tweet_token, token_prediction, and sentence_prediction.

3.4 Pipeline Setup

In order to effectively collect and store Twitter data from a specified timeframe, we created a pipeline that can perform both of these tasks in sequence as well as in parallel. We chose to use Python to build our pipeline, as this was the language we were all most familiar with. Python also has quite a few useful packages for handling data such as pandas and NumPy.

We split up our pipeline into two main components: producer and consumer. The “producer” focuses on handling the collection of Twitter data from the Twitter archive. The “consumer” portion of our pipeline focuses on taking the collected tweets and appropriately storing the ML model prediction data in our database and raw data storage. The following diagram summarizes our pipeline setup and process:

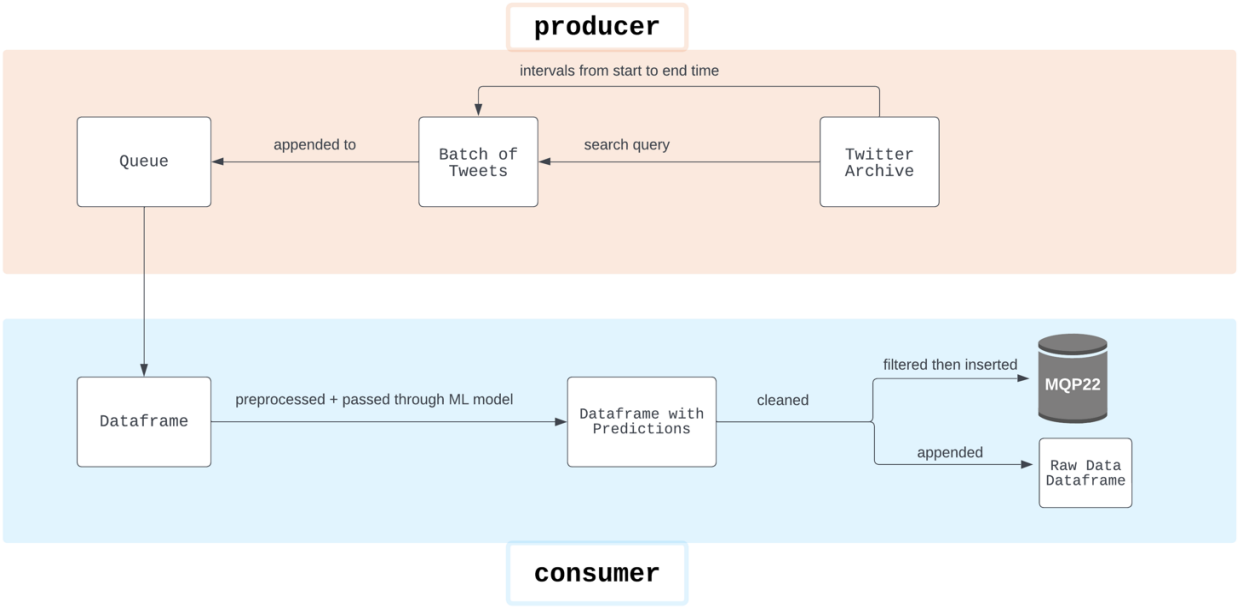


Figure 3.4: *Diagram of our pipeline setup*

We stored the code for our pipeline under the Data-collection folder on our server. We chose to run this code on the server, due to the fact that the requirements for the machine learning model were already established there. Also, this would allow the code to run for an extended period of time without a local computer having to sustain it.

Under the Data-collection folder, we have multiple files that serve separate purposes in our pipeline. Our ‘main_collection’ file carries out the consumer part of our pipeline. This file runs a search query (based on foodborne illness identified keywords) and retrieves a batch of tweets within a specified time interval. These tweets are then each appended into a queue. For our project, we chose to export each of these queues into their own .pt files that we could access later. Our ‘main_predict’ finishes our process by carrying out the producer portion. The tweet queue is made into a data frame, which then gets appropriately preprocessed before being sent

into the machine learning model. The machine learning model then runs on this batch of tweets, and a new data frame is created, containing the raw tweet data as well as the prediction results. This data frame goes through a data cleaning process and then is inserted into our database and raw data storage respectively. We only inserted data we felt would be useful for the front end into the database (explained in section 3.1.2) and stored all of the data in our raw data storage. We have organized our pipeline files this way so that the components of our pipeline can be run and tested separately if needed. Finally, our 'main_multiprocess' file is intended to run the entire pipeline from start to finish. Because of time constraints, we do not plan to use this to process the data for our project. However, we have created this file for an easier collection process to be used in future projects.

3.5 Frontend

Data visualization is a key part of any research project, and thus, given both the internet theme of our study and our plans to store data in a format that would be easy for applications to interact with, we decided to create a website capable of displaying our results to a wide variety of users. Regardless of what results we discovered, we wanted to ensure that our conclusion would be clearly communicated to users with even minimal experience interacting with online visualizations.

To begin the process, we audited and assessed the work completed by a previous group of students working on the same project. They had also created a website, with three key features: a timeline, a map, and a word cloud. With these ideas as a basis, we decided to find ways to improve upon the existing design, and came up with several actionable insights.

First, we needed to connect our future website with the actual data from the project where possible. In order to make sure that our visualizations would scale with whatever data we ended up collecting, we needed to create a pipeline so that a Tweet could be collected, included in our analysis, and most critically, sent to our frontend website, without any manual human interaction. This would hopefully save us time in the long run, while also reducing opportunities for human error. As far as we could tell, data for the existing website was mostly hardcoded in CSV files, which added additional steps and complexity which we sought to reduce.

Second, we needed to add interactivity to the data we were displaying. Many of the previous visualizations, such as the word cloud, did not allow for meaningful user interactions, meaning that users were effectively just viewing images on the website, meaning that many of the benefits of web apps for communicating trends were left untouched. We wanted to create experiences where users could look at our study, come to their own conclusions, and then see how those stacked up against the data trends we observed. Or, users could use our website as a springboard to raise new questions about foodborne illnesses by playing around with the different interactive options we presented them.

Finally, we needed to design our code in a way that is easy for future projects or researchers to begin developing quickly. The previous project contained a large number of unused files, obscuring the development process and adding unnecessary bloat. By streamlining the final product, hopefully, the results of this study will be more useful for everyone.

For the final product, we chose a variety of modern development frameworks and tools to help ensure that we could efficiently develop a web interface usable by our audience. Next.js, a frontend web development framework (Next.js, 2023) based on the React (React, 2023) library, was chosen as our frontend web framework, as it includes many utilities, such as file

optimization and static asset bundling, which ease the process of deploying our app to a server. An additional benefit of Next.js is that it leverages the React frontend library, which is widely used across modern web applications, meaning that there are well-supported React resources online, should a developer need help. D3.js (D3.js, 2023) was chosen as our visualization library for our frequency graph, because, much like React, it is widely used and considered by many to be an industry standard. Mapbox, a hosted GIS mapping service (Mapbox, 2023), was used to power our live map - we were able to connect the tweets in our database to an interactive Mapbox map, allowing users to interact with the tweets on a US map. Finally, Tailwind CSS (Tailwind CSS, 2023) was chosen as our UI library, because it is small and efficient while also providing significant flexibility to developers.

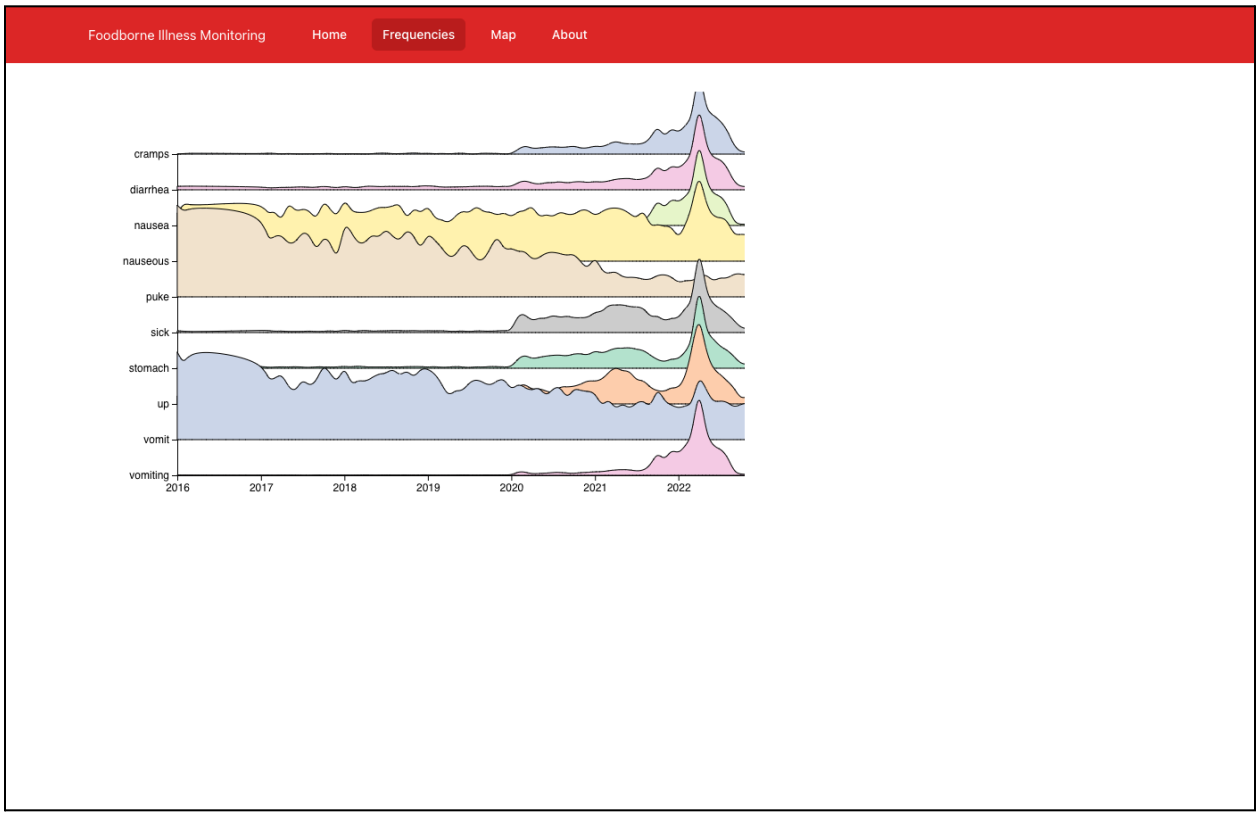


Figure 3.5: *Screenshot of the frequencies diagram on the visualization webpage, which was created with D3.js*

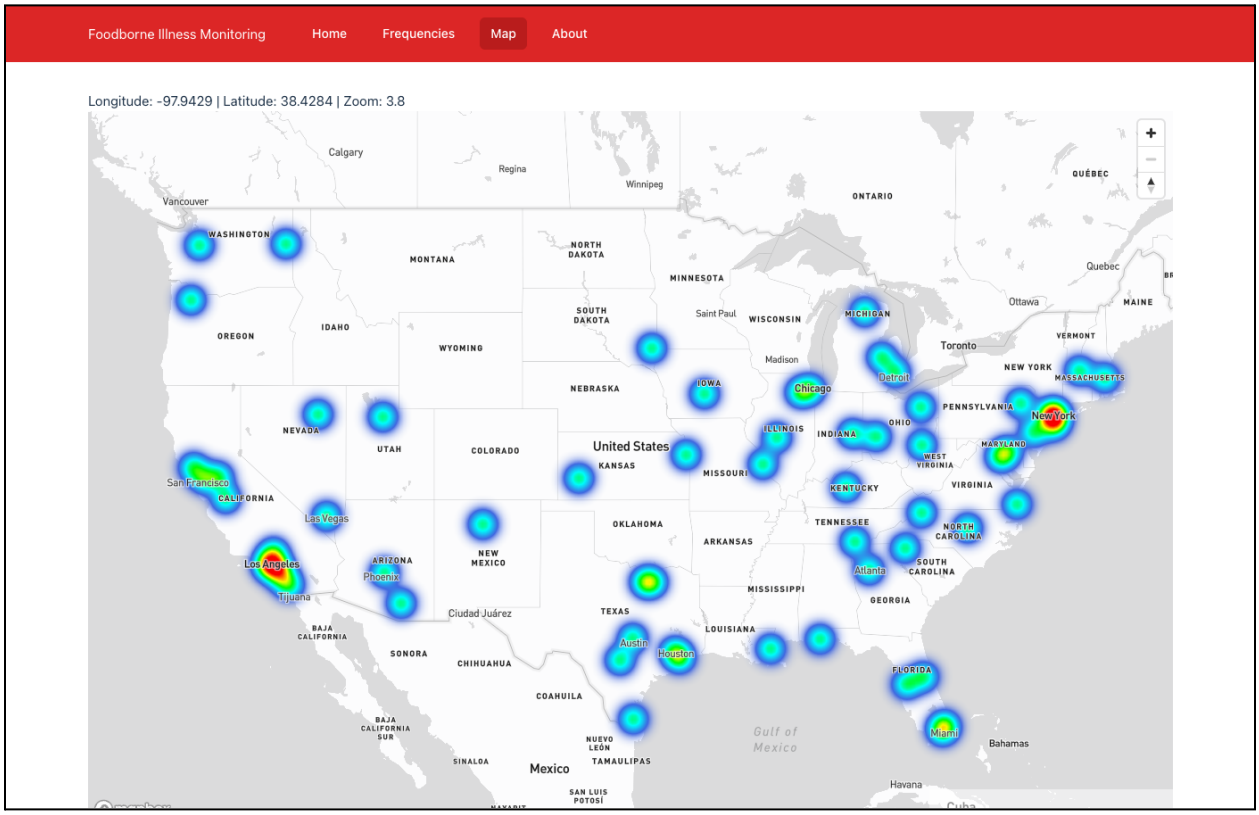


Figure 3.6: *Screenshot of the map on the visualization webpage, which was created with Mapbox*

3.6 Forecasting Tweets

Forecasting takes previous information and uses it to predict future information. In this scenario, forecasting allows us to infer the future amount of tweets based on the previous amount of tweets. This would allow us to predict Twitter data sooner than it would actually occur. If one was able to use daily Twitter data to predict foodborne illness, being able to forecast it would allow an even earlier warning that the CDC is able to provide.

iWasPoisoned makes up a large portion of our collected Twitter data set. Since iWasPoisoned tweets only exist from 2020-2022 in our Twitter dataset, it will impact the data trends because there are more tweets in those years. Normalizing the data, such as looking at the

percent of positively predicted tweets, would also have impacted the trends because there are more positively predicted tweets from those years. There are more positively predicted tweets because almost all of the iWasPoisoned tweets were predicted to be foodborne illness. For these reasons, in this section, we decided to focus on the dataset without iWasPoisoned so that it would not impact the scaling of our data, or the data trends.

3.7 Predicting NORS Cases

To test the viability of predicting NORS cases from tweets, we decided to test linear regression and logistic regression on different combinations of past tweets and NORS cases. This was done to determine what the ideal amount of tweets and NORS cases given to the model to train for each prediction. Then use the model to predict with and determine what combination of tweets and NORS cases would be necessary to get the best results when predicting and also to see overall how well this model does at predicting.

The first thing that needs to be done is to create a training set X to train the model on. We decided to use the first 200 weeks of NORS cases, and the first 1400 days of tweets to allow for about one year's worth of NORS cases to be in the validation set. The set X would be created with different amounts of tweets and previous NORS cases for each label. For example, X with n tweets and m NORS cases prediction would look like this:

$$X_{NORS^{T}} = \alpha_1 * X_{tweet^{7T}} + \ldots + \alpha_n * X_{tweet^{7T+n}} + \alpha_{n+1} X_{NORS^{T-1}} + \ldots + \alpha_{n+m} X_{NORS^{T-m}} \quad (1)$$

The first few entries of a sample X-training, for 4 tweets and 2 NORS cases, would look like:

Label	X_1	X_2	X_3	X_4	X_5	X_6
$NORS^3$	$tweet^{14}$	$tweet^{15}$	$tweet^{16}$	$tweet^{17}$	$NORS^1$	$NORS^2$
$NORS^4$	$tweet^{21}$	$tweet^{22}$	$tweet^{23}$	$tweet^{24}$	$NORS^2$	$NORS^3$
$NORS^5$	$tweet^{28}$	$tweet^{29}$	$tweet^{30}$	$tweet^{31}$	$NORS^3$	$NORS^4$

Figure 3.7: *Sample entries of training set X, where tweet^n refers to the number of tweets on day n, and NORS^m refers to the number of NORS cases in week m. Since week 5 contains days 28-35, we use tweets within those days to train the model on for week 5*

The next thing to do was to loop through different combinations of n tweets and m weeks of NORS cases to see what yielded the best R-Squared and the best error. Tweets is in days because we have access to that information, whereas NORS is in weeks because that is the smallest bin we have access to. The regression models that we used is from the Python package sklearn, as well as statsmodel to calculate the significance of the coefficients (*Scikit-Learn*, n.d.). The pattern would loop through NORS cases and then increase by one tweet. As an example, it would be (n tweets, m NORS), (n tweets, m+1 NORS), (n tweets, m+2 NORS), (n+1 tweet, m NORS), (n+1 tweet, m+1 NORS), (the formula is 4*tweets+NORS to convert the numbers on the X axis of the graph to the combination of model parameters used, since the method loops through 4 different options for number of NORS cases before changing amount of tweets), and

so on. From this, the R-Squared, RMSE, Adjusted R-Squared, and validation set RMSE would be reported for further analysis.

4. Results

4.1 Twitter Data Collection

4.1.1 General Outcome

The tweet collection from the Twitter API collected ~600,000 tweets from the beginning of 2016, to the end of 2022. The process of passing in the tweets through the model, cleaning, then inserting them into the database took the longest amount of time, taking ~10.32 hours. The database had ~110,000 tweets after processing. For the raw data saving process, the script didn't append all the separated queues into a single data frame during predicting runtime because this would require a constant reading and writing of a large file, so instead it initially only saves the individual data frames. The concatenation of these files took about a minute.

4.1.2 Tweet Locations

The following map in Figure 4.1 displays the number of positively predicted tweets by state. From this, we see that most tweets are coming from California and Florida. This makes sense because these states are the largest population-wise. More people tweeting in general would likely lead to more foodborne illness tweets identified in these areas.

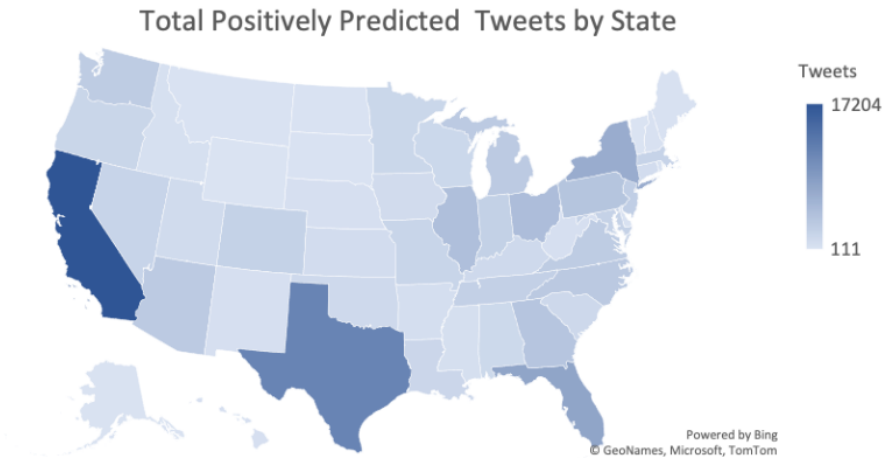


Figure 4.1: *Map of positively predicted tweets by state 2017-2022*

This next graph, Figure 4.2 is the total number of tweets collected by state divided by the population of the state (divided by 100,000 to create a per capita amount) ("State Population Totals", n.d). This shows different information than the previous map because it normalizes the amount of tweets by the amount of people in the state. This fixes the issue of larger states having a large number of foodborne illness related tweets because of their larger population. If a smaller state had an outbreak, with a larger amount of tweets than it normally would have, it may be overshadowed by larger states. Normalizing the data like this allows trends in smaller states to appear more significant than they would without adjustment for population size. There are two versions of the map, with the graph on the left including Washington DC, and the right without. Since DC has the highest per capita amount of tweets (displayed better on the next graph), but doesn't show up on the map due to its size it changes the appearance of the map.

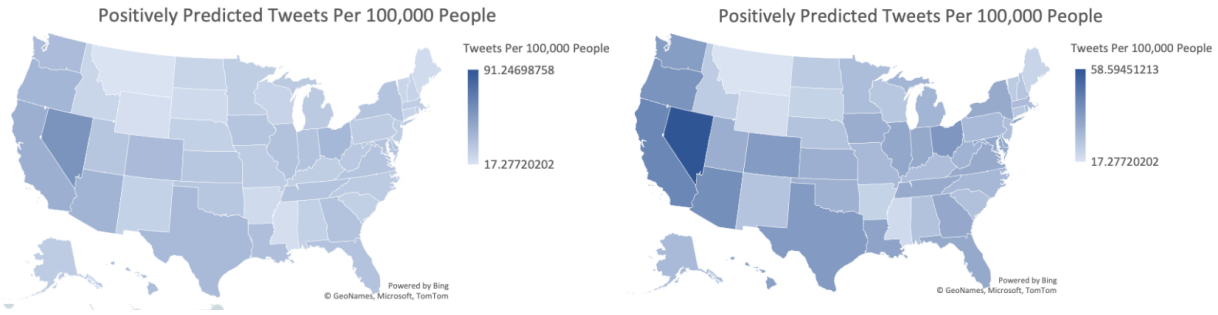


Figure 4.2: *Map of positively predicted tweets by state 2017-2022 corrected for population based on the 2022 census estimates (left includes DC, right excludes DC)*

The final tweet data by state graph, Figure 4.3, is one that compares the amount of tweets by the state to the amount of tweets per capita. This displays the same information in a way that combines the two maps. This visualization allows one to visually compare the difference between the amount of positively predicted tweets from each state, compared to the per capita amount of tweets by state. This also illustrates the importance of having the previous two maps because both are important for different reasons.

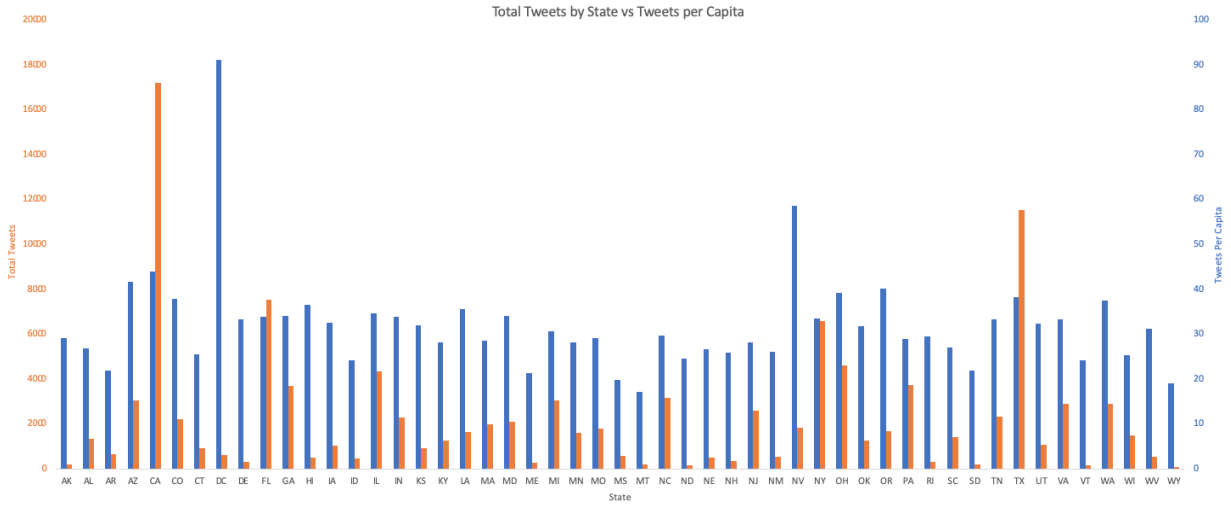


Figure 4.3: *Graph of collected tweets compared to per capita tweets 2017-2022 corrected for population based on the 2022 census estimates*

4.1.3 Author ID Analysis

One potential issue we considered that we may come across was repeated author id, which could possibly mean a Twitter account dedicated to foodborne illnesses. This would not be an issue if the tweets were relevant to what we were looking for (i.e the iWasPoisioned Twitter account tweets about instances of foodborne illness, specifically what we are looking for), but if there was a source of not relevant tweets that kept coming up we wanted to be able to filter that out (i.e the CDC, FDA, or news sources tweeting about foodborne illnesses after the fact, or a Twitter account dedicated to tweeting foodborne illness information, rather than instances of foodborne illness). Luckily, in searching through samples of collected tweets searching for repeated author IDs, the only one that did appear to come up often was the iWasPoisioned Twitter account, so we deemed that not an issue since the iWasPoisoned Twitter account is most likely cases of foodborne illness.

The following charts in Figure 4.4 show the amount of tweets collected that came from iWasPoisoned vs the amount of only positively predicted tweets that came from iWasPoisoned. Of over 56,000 tweets collected from iWasPoisoned, less than 0.5% (less than 300 tweets) were predicted to not be foodborne illness. (Less than 56,000 tweets from iWasPoisoned ended up in our database because we collected tweets from everywhere, but only included tweets from the US in our database). The other Twitter accounts that came up multiple times, had significantly fewer tweets that were predicted to be foodborne illness related so they did not impact our dataset as much.

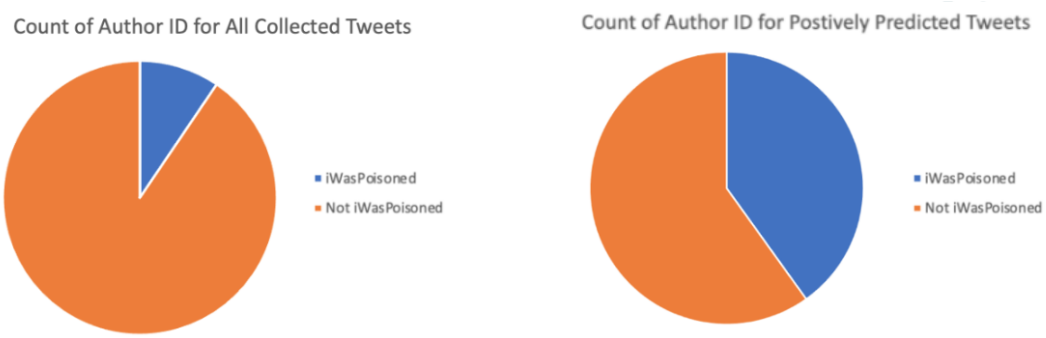


Figure 4.4: *Chart of all collected tweets by username, versus positively predicted tweets by username*

Looking at these charts, it is apparent that iWasPoisoned makes up a larger portion of our collected data, and an even larger portion of our predicted data. This led us to wonder if collecting iWasPoisoned tweets was impacting our collection. Figure 4.5 shows that this potentially could be the case. The fact that our total number of tweets increases with the addition of iWasPoisoned tweets, but the number of non-iWasPoisoned tweets begins to decrease seems to suggest that iWasPoisoned may have an impact. In later iterations of this project, it may be interesting to collect tweets while excluding iWasPoisoned to see if there are more of them. iWasPoisoned is potentially interfering with how the spikes in tweets show up because people are posting there first rather than on Twitter.

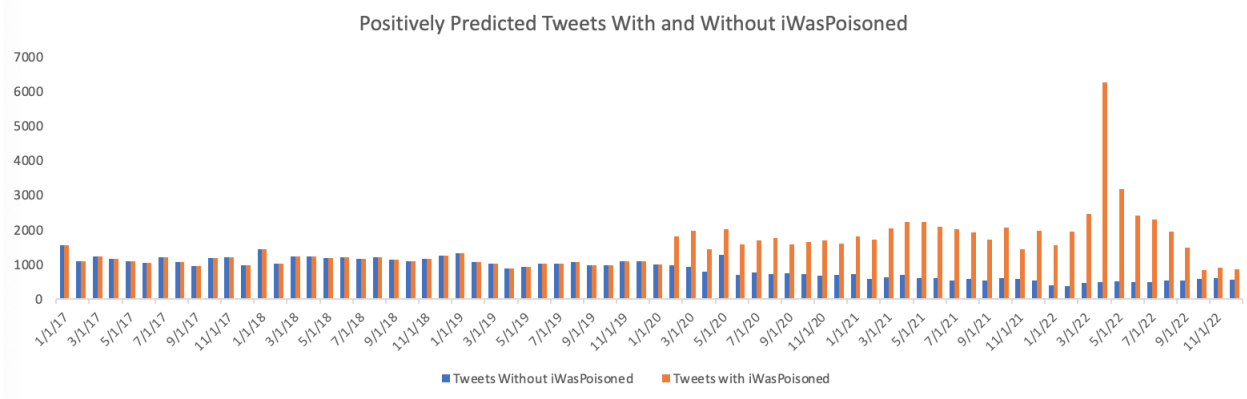


Figure 4.5: *Graph of positively predicted tweets with and without iWasPoisoned*

4.2 Food Token Analysis and Comparison

4.2.1 Most Frequent Food Tokens

As mentioned before, our machine learning model identifies ‘food’ entities from a tweet. These entities are identified words that most likely indicate a key food or ingredient. To evaluate the validity of our collected tweets in indicating real foodborne illness cases, we chose to compare the most frequent food entities with the real food names mentioned in NORS data. We first explored the top 15 most frequent food tokens that appeared in tweets in our database from 2017 -2021. Below is a graph displaying the most frequent food tokens and their counts.

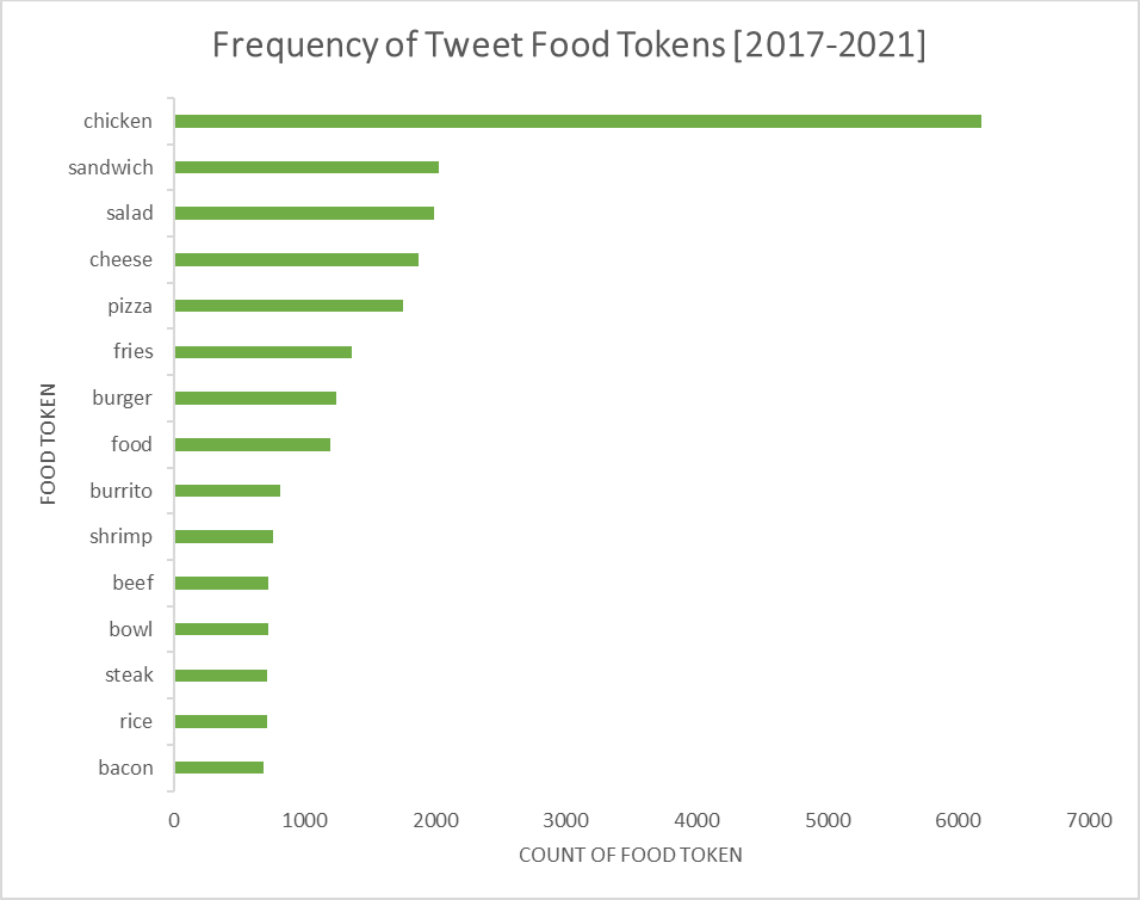


Figure 4.6: *Graph displaying the 15 most frequent food tokens found in collected tweets from 2017-2021*

Chicken was identified as the most frequent food token, with a much greater count compared to the rest of the food tokens. We also observed that 'bowl' appeared as a food token, most likely because it was associated with other food terms such as 'burrito bowl'. Also, the word 'food' was identified as a food token, which is not very helpful in identifying specific foods involved in outbreaks. Besides these terms, the other 13 tokens appear to be relevant.

As noted in earlier sections, we noticed that iWasPoisoned tweets made up a significant portion of our tweet collection. In order to explore 'genuine' foodborne illness tweets that were not outsourced ahead of time, we decided to run the same query, this time excluding the iWasPoisoned username. By doing this, we received a different set of most frequent food tokens.

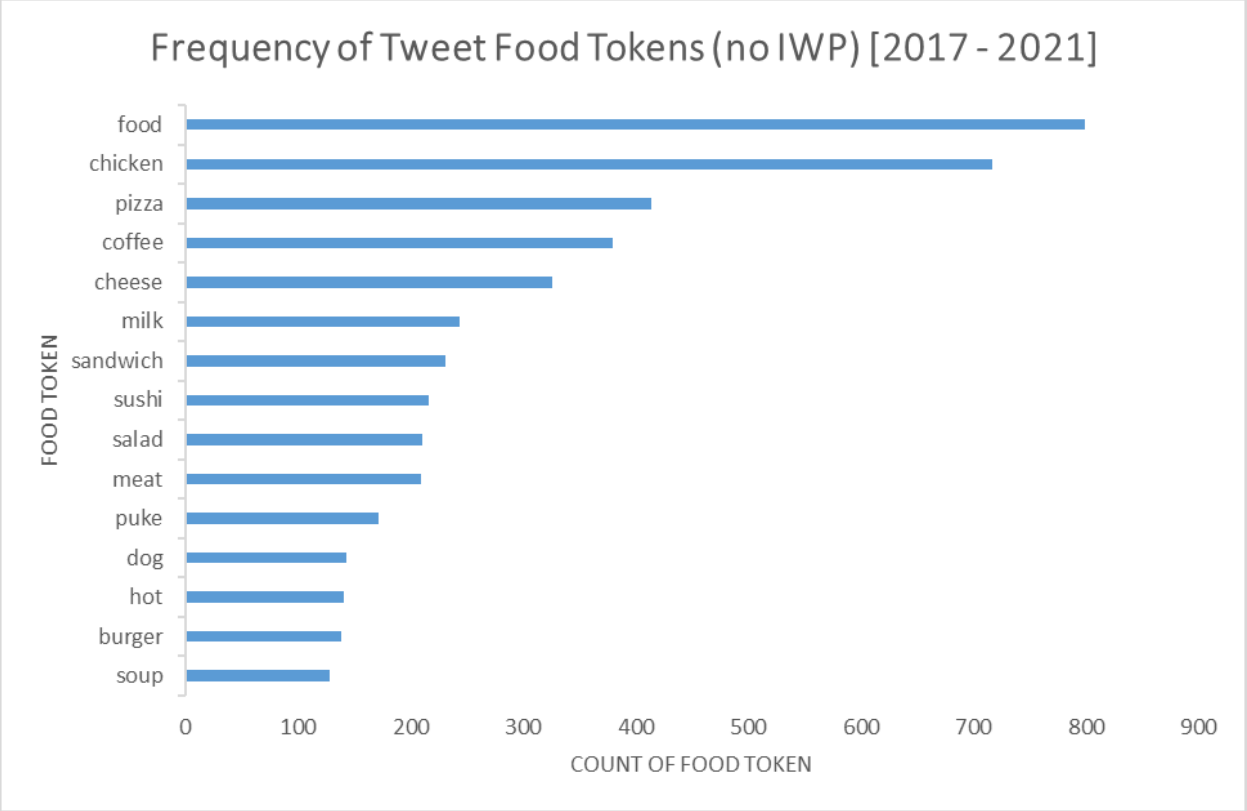


Figure 4.7: *Graph displaying the most frequent food tokens from 2017-2021 excluding iWasPoisoned tweets*

This time, the most frequent food token was ‘food’, as opposed to ‘chicken’ in the previous graph. Is it also important to note that ‘puke’ was recognized as a food token by the machine learning model, indicating that perhaps the model does not perform as well on tweets that are not retrieved from the iWasPoisoned Twitter. There do appear to be some consistent tokens, however, as 7/15 of the tokens were exact matches. Another important thing to note is, out of the tokens that differ, the first graph displays more ‘specific’ described foods (such as ‘shrimp’, ‘beef’) while non-iWasPoisoned tweets contain more generically described foods (‘meat, ‘burger’, ‘hot dog’). This could be due to iWasPoisoned requesting more information about ingredients that may have made users ill.

We then chose to explore the top 15 most frequent food tokens mentioned in NORS reports. To do this, we separated each word included in the ‘Food Name’ column of the NORS report data and counted the frequency of each word. We then graphed the top 15 most frequent food tokens from 2017-2021.

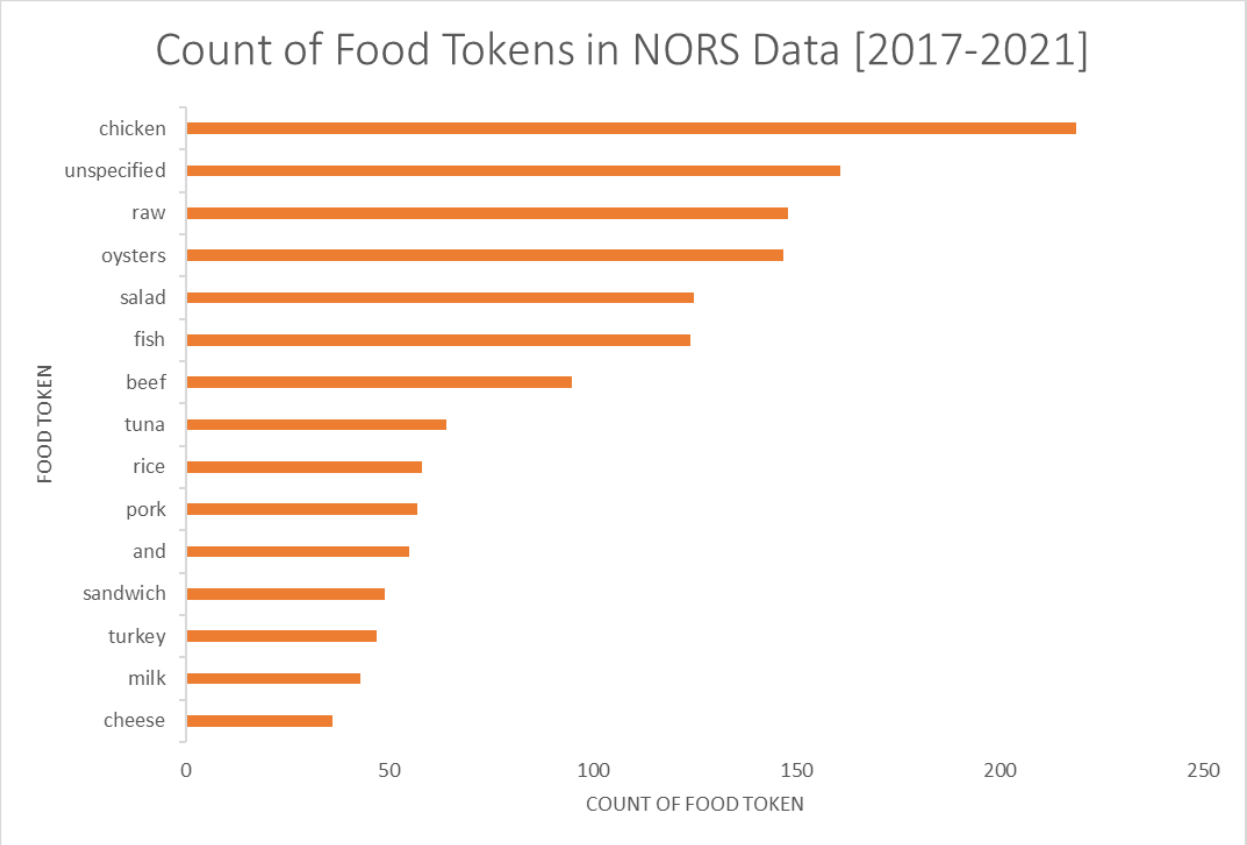


Figure 4.8: *Graph displaying the most frequent food tokens in NORS report data*

We first noticed that some of the food tokens retrieved from NORS were not necessarily food related. For example, 'unspecified' and 'and' appeared in the most frequent food names, as they were used as descriptive words within the 'Food Name' column. To perform a more accurate comparison in the future, it would be useful to identify these non-food related words beforehand or perform word importance analysis. Otherwise, we see that 'chicken' also appears as the most frequent food token in the NORS reports. In order to compare the counts of the food tokens to our Twitter data, we performed further analysis.

To increase the chances of finding matching food tokens, we chose to identify the top 30 most frequent food tokens and compare these. We first began by comparing the top 30 most frequent food tokens with tweets including iWasPoisoned data to the top 30 most common food

tokens in NORS reports. To perform concrete comparison, we chose to only compare food tokens that exactly matched. There were other words that could be associated (such as ‘beef’ and ‘hamburger’), but further analysis would need to be done to fairly compare these. Out of the top 30 food terms, 13/30 were exact matches. In order to compare the counts of these terms, we normalized the counts by the maximum token count for both the Twitter and NORS data. After doing so, we calculated the absolute value of the difference between these normalized counts. Below is a graph displaying the differences in counts for the matching food terms.

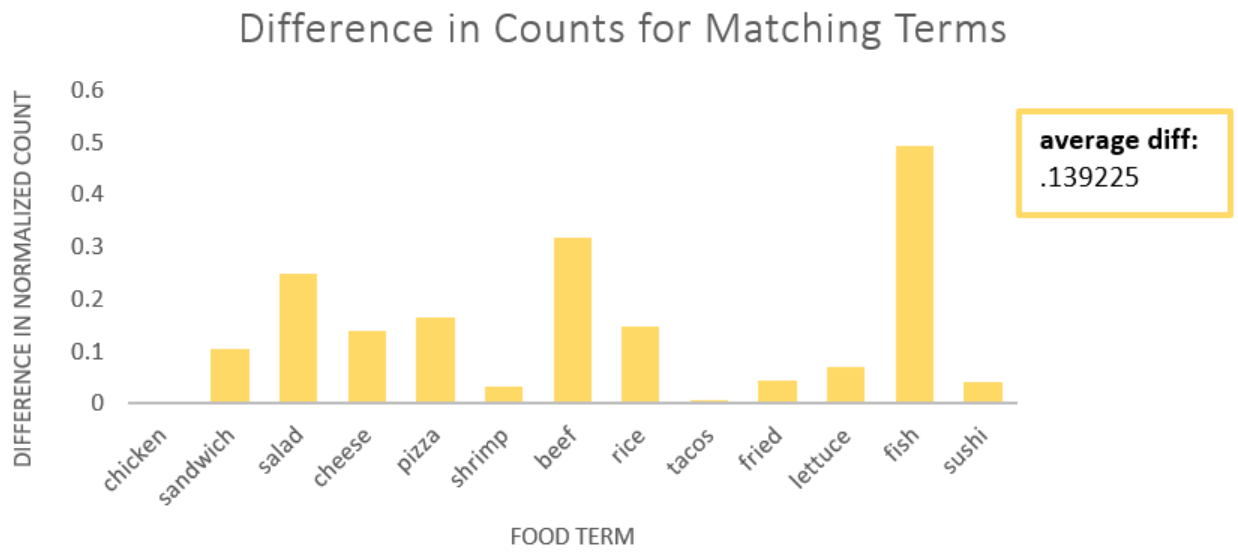


Figure 4.9: *Graph displaying the difference of normalized counts of matching food terms from Twitter and NORS data*

Because ‘chicken’ was the most frequent term for both datasets, there is a difference of 0. However, we can see that there is a greater difference in the count of ‘fish’ and ‘beef’ terms. Overall, the average normalized difference across all of the matching terms was .139225.

We then chose to perform the same comparison, this time without iWasPoisoned tweets included. In this case, 12/30 terms matched. In order to see if these terms were more or less

consistent, we graphed the difference between each matching term and calculated the average difference overall.

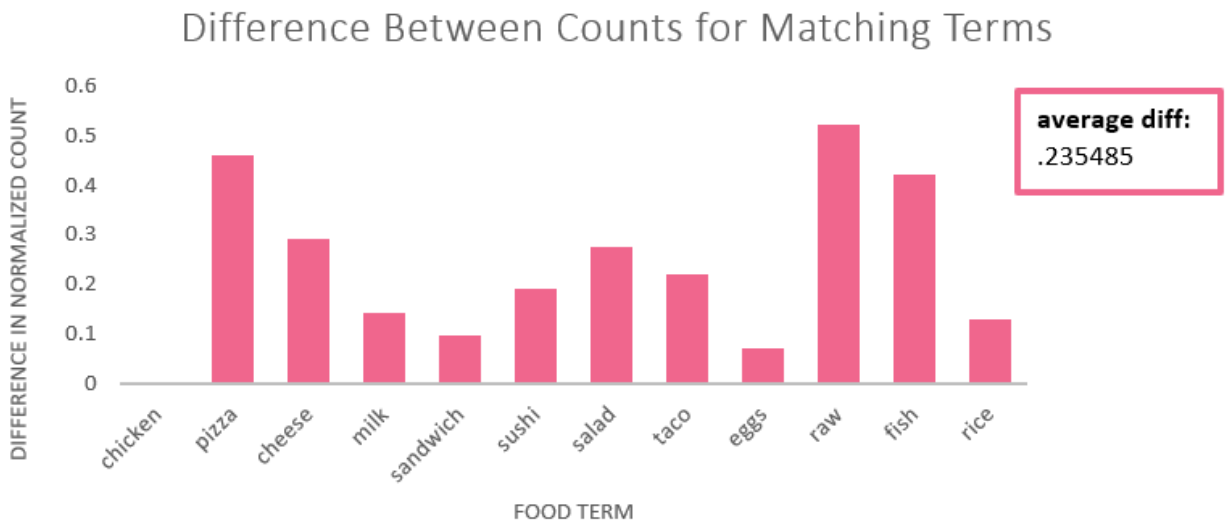


Figure 4.10: *Graph displaying the difference between normalized counts of matching food terms from Tweet data (excluding iWasPoisoned) and NORS data*

We observed a larger average difference in terms of the relative count of matching food terms. This could suggest that the iWasPoisoned data had matching terms that differed less, on average, to ground truth foodborne illness reports. Therefore, it appears that iWasPoisoned tweets contribute to a more accurate depiction of real foodborne illness cases. When it comes to predicting tweets that are not from iWasPoisoned, there appears to be a greater distinction. This could be due to the machine learning model not performing as well on these types of tweets, or that the tweets collected may not have as relevant food information.

4.2.2 Comparison to Ground Truth Outbreaks

During our exploratory analysis of our Twitter data, we explored the most popular food keywords from 2017-2022. To our surprise, we found that ‘lucky’ and ‘charms’ were the second

and third most frequent food entities that appeared within this timeframe. We found this interesting, as these were the only food tokens within the top 15 recognized that contained a specific brand name. It is also important to note that these were identified as two separate food tokens when they should be identified as a single food entity. We chose to research this and see if there were any official outbreaks related to Lucky Charms cereal.

Through our research, we discovered that there was a suspected foodborne illness outbreak involving Lucky Charms starting at the end of 2021 and lasting until April 2022. As of May 6, 2022, it was reported by iWasPoisoned that there were more than 7,300 reports of people becoming sick from the cereal (Lee, 2022). Many complained about experiencing extreme nausea and stomach aches after consuming Lucky Charms. The immense amount of reports and complaints prompted the FDA to launch a 4-month long investigation. However, the FDA reported that there was 'no pathogen or cause' found in causing these illness cases (Casey, 2022).

We were curious to see if the large number of tweets including the 'lucky' and 'charms' entities lined up with the timeline of this event. Therefore, we chose to graph the frequency of these entities from the end of 2021 through 2022. We also included the frequency of the 'chicken' entity as a baseline, since this was the most frequent food entity in our Tweet collection. Below is the graph showing the frequency of the entities over time:

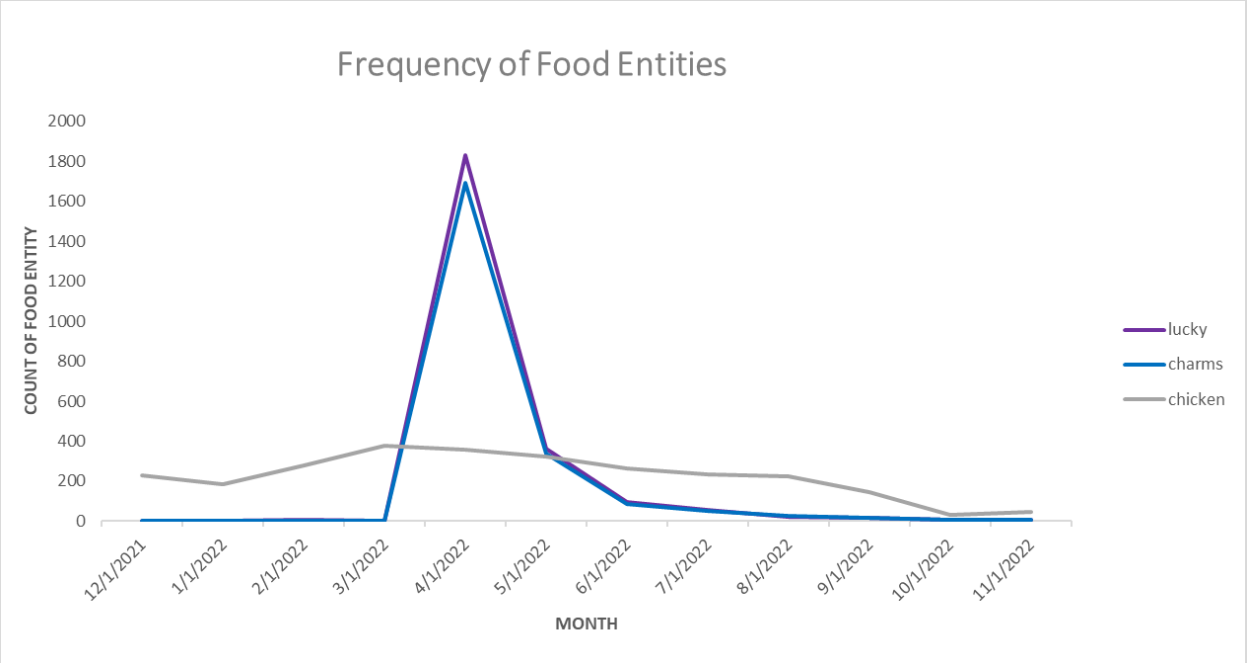


Figure 4.11: *Figure displaying the counts of 'lucky', 'charms', and 'chicken' entities from the end of 2021 through 2022*

We observe an extremely large peak in the appearance of 'lucky' and 'charms' food entities in April. Comparing these entities to 'chicken' proves that there is a significant indistinguishable spike involving these specific entities. In order to look into the timing of these tweets further, we chose to plot the frequency of the entities by day for March - May of 2022.

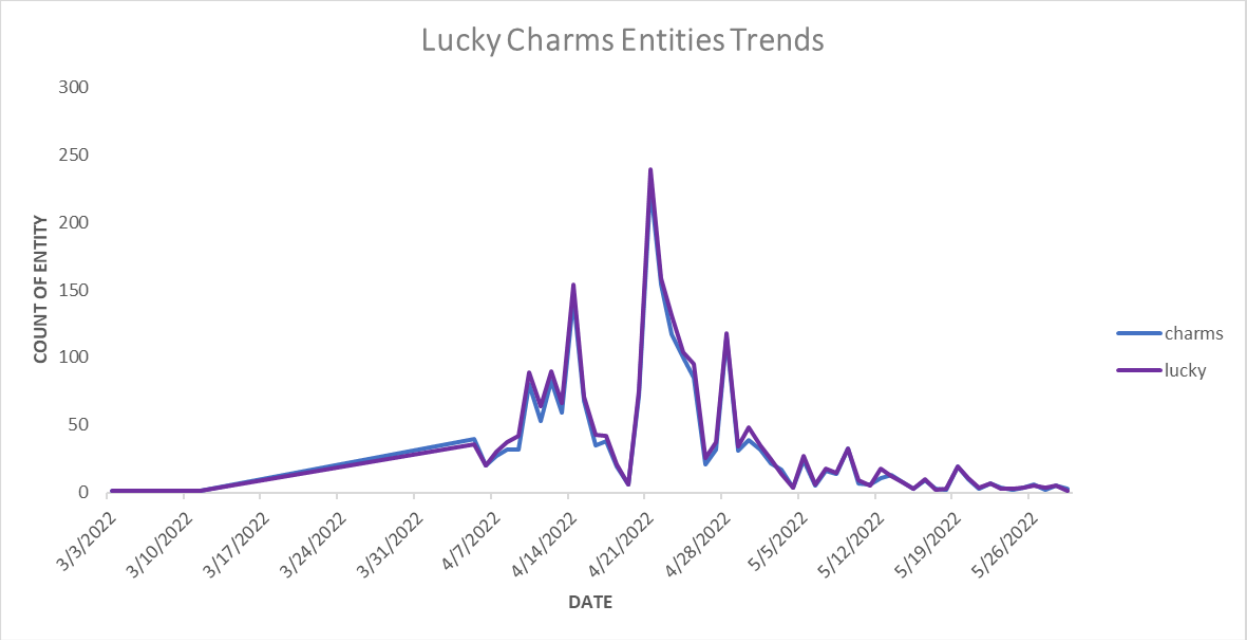


Figure 4.12: *Graph displaying the trend of tokens 'lucky' and 'charms' by day (March - May 2022)*

From this graph, we can observe that there are strong peaks in the frequency of these entities in the middle and end of April. This lines up with the ground truth, as it was reported that the FDA began investigating the outbreak in April, and it was reported that by May there were over 500 reported cases. Therefore, we are seeing proof that our machine learning model was able to detect Tweets about these illnesses at about the same time that people were becoming sick. It is however important to note that all of these tweets were obtained from iWasPoisoned data. When attempting to create the same graphs without iWasPoisoned tweets, no usable data was obtained. Therefore, this provides more proof that iWasPoisoned greatly contributed to useful foodborne illness data.

We also explored if the trends of specific food entities aligned with reported outbreaks in NORS data. In order to identify specific outbreaks to investigate, we first sorted the NORS reports by the total number of estimated cases. The reasoning behind this was to compare

foodborne illnesses outbreaks with the most cases, as this would most likely yield the greatest amount of related tweets. We chose to identify the 30 NORS reports with the greatest number of estimated cases and then explore the food names mentioned in each report. The report with the greatest amount of estimated cases mentioned ‘red onion’ as the key food. As ‘onion’ is a rare food entity in our Twitter database, we chose to not investigate this specific outbreak. The third report with the most cases mentioned ‘salad’, and as this was quite a common food entity found by our model, we chose to explore this specific outbreak. This outbreak took place from May to June of 2020, and the food identified was salad mixes. As the NORS data includes both the first and last day of illness, we explored the trend of the ‘salad’ entity around these dates. Here is a graph of the ‘salad’ entity trend in 2020.

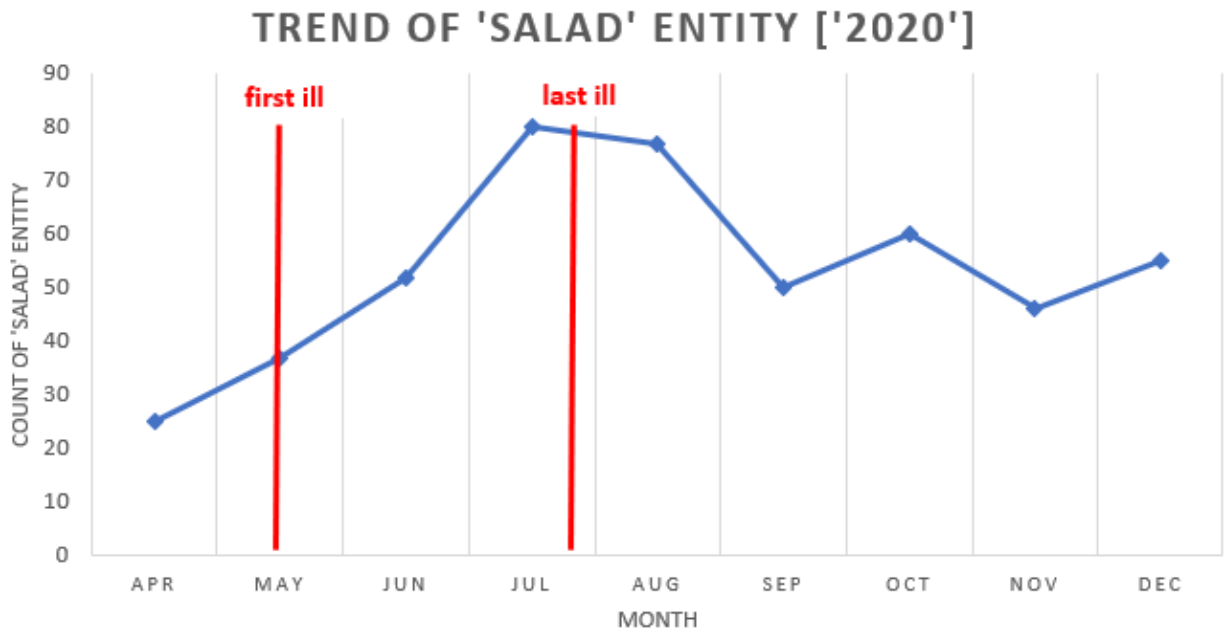


Figure 4.13: *Graph displaying the trend of the ‘salad’ entity in 2020, with the first and last illness of outbreak identified*

There appears to be an upward trend in the appearance of salad entities between the first and last reported illness in this outbreak. The trend also reaches its peak right before the last illness was reported, which could confirm that the Twitter data collected is somewhat consistent with the ground truth case data.

We also explored another outbreak involving salad mixes that took place in 2018. This time, we did not observe a clear correlation. Within the time frame, there actually appears to be a dip in the amount of 'salad' food entities observed.

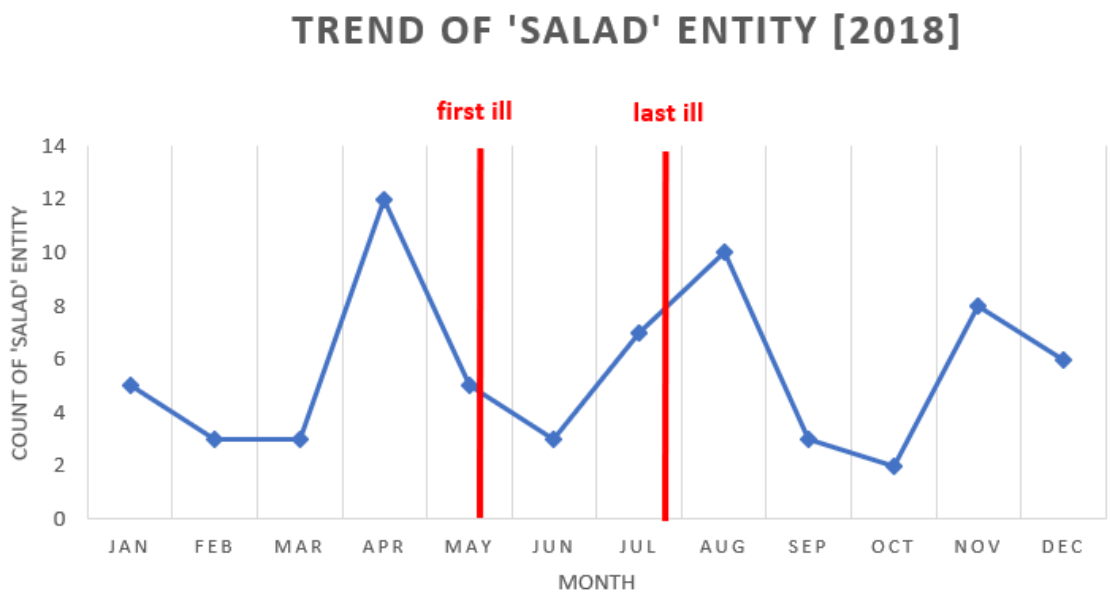


Figure 4.14: *Graph displaying the trend of the 'salad' entity in 2018, with the first and last illness of outbreak identified.*

We also explored significant outbreaks that involved chicken, as 'chicken' was a very frequent food entity found in our database. We identified an outbreak in 2021 that involved chicken as a key food, however, the reported duration appeared to be very short (2/16/2021 - 2/17/2021). Therefore, once we graphed the trend of the entity, it was difficult to truly notice if

the Twitter data had any indication of an outbreak between these days when plotting monthly. Therefore, we chose to plot entity frequency by day for February 2021 to see if there was any apparent spike.

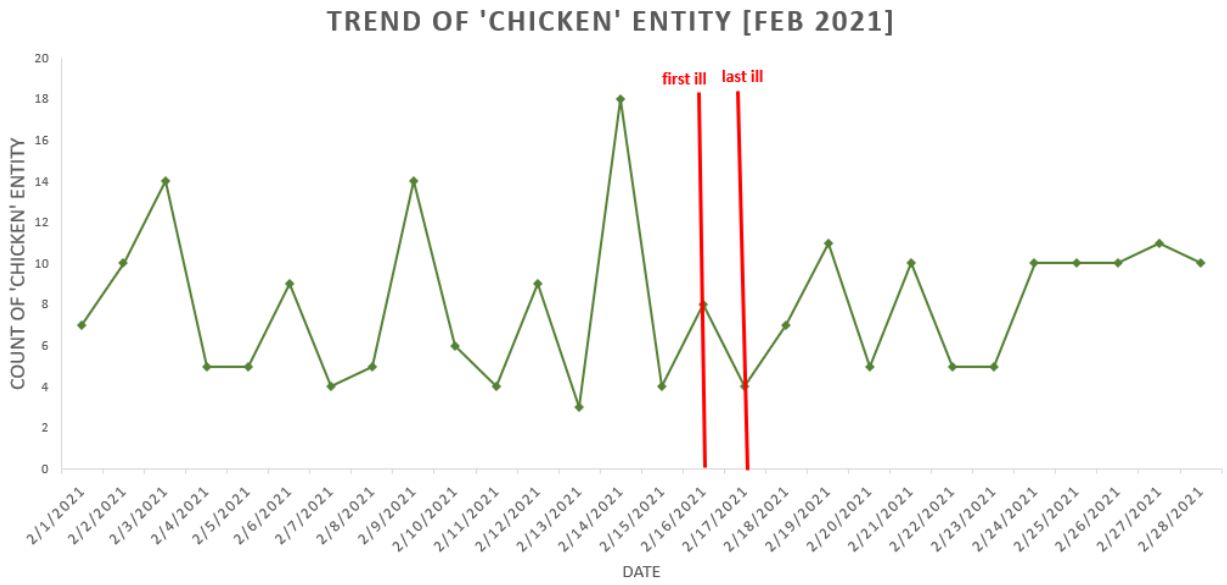


Figure 4.15: *Graph displaying the trend of the 'chicken' entity in Feb 2021, with the first and last illness of outbreak identified by red bars.*

There appears to be a spike in cases two days before the first reported illness, which could indicate a possible correlation. However, the number of tweets greatly drops within the specified interval of reported illnesses. This could either indicate that tweets involving the event were created before patients reported being ill, or that our data appear inconsistent.

We then investigated another outbreak involving chicken, this time with a longer duration for the chance of obtaining a greater volume of Twitter data. There was an outbreak involving chicken wraps that took place from 4/2/2018 to 5/4/2018.

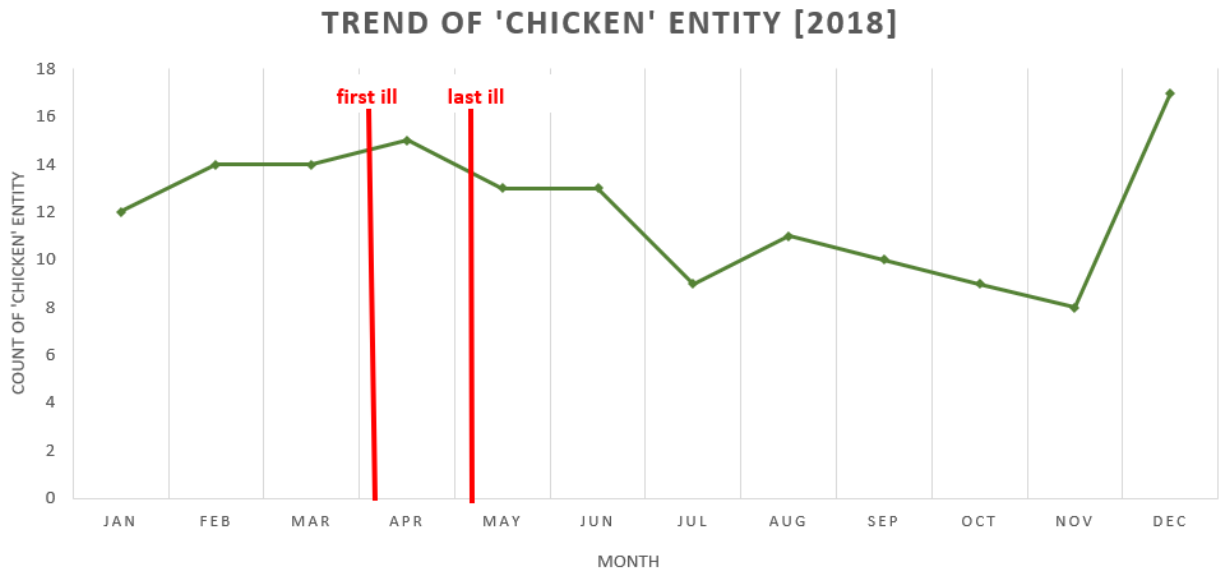


Figure 4.16: *Graph displaying the trend of 'chicken' token in 2018, with the first and last illness of outbreak identified.*

There does appear to be a peak in this time range, however, it does not appear as significant as it was in previous months. However, this could be due to the fact that there was a chicken salad foodborne illness outbreak that lasted from January to around March of the same year.

There were many limitations in comparing our Twitter data to NORS report data. One limitation was that most outbreaks in the report had a limited duration, which made it difficult to compare to a significant number of tweets within the same time frame. Another limitation was that key foods mentioned in certain reports were very rare to find in our tweet collection. This could be both due to the fact that our machine learning model did not recognize the food word, or simply because users chose not to tweet these specific foods/ingredients. Another issue was that more significant outbreaks occurred before 2020, and the Twitter data obtained from this time period did not generally line up well with the ground truth data. This could be due to the fact that iWasPoisoned data was not prevalent during this time period.

4.2.3 Comparison Using The CDC's Food Categories

The CDC has a food labeling system call the IFSAC food categorization scheme which generalizes foods into similar characteristics (Center for Disease Control and Prevention, 2023). Because of the sheer number unique food tokens, compressing the foods into general categories for analysis could provide valuable insight. Because there are no available resources to automatically convert common foods into their corresponding categories, we hand labeled the top ~250 common food tokens, from both the Twitter archive and the NORS dataset, into their respective categories. These labels were put into a JSON dictionary which could be easily read to convert tokens to categories.

The following figure shows the total percentage breakdown of each category once the conversion took place.

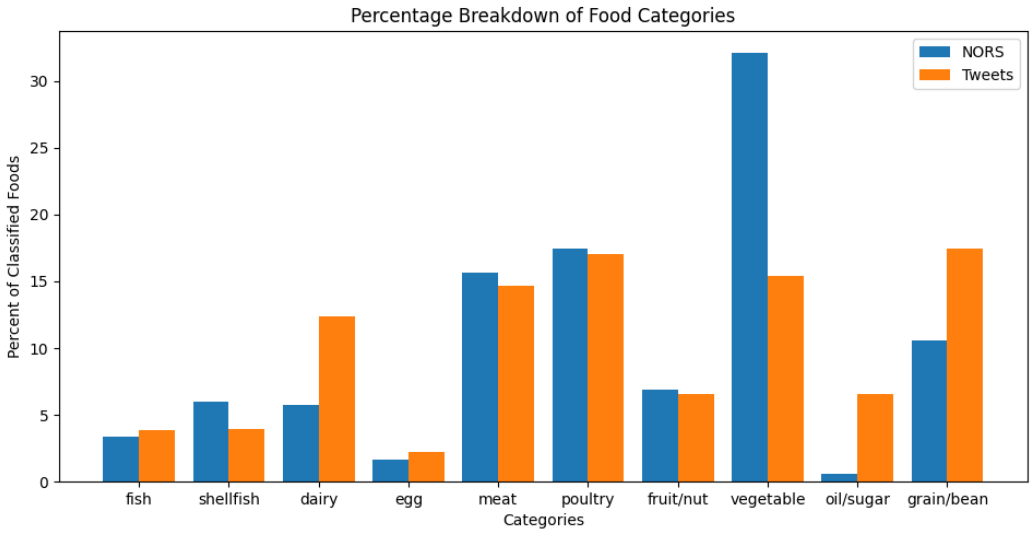


Figure 4.17: *Graph showing normalized volume of data between IFSAC categories by dataset.*

The volume of food categories between the NORS dataset and the Twitter dataset were relatively similar, which is promising for accuracy comparison. To further analyze this data, we chose to view the volume of categories overtime in each food category.

Before analysing and to make sure the two dataset are comparable, we first filtered the datasets to be between the dates of 2016 and 2022, as there were missing datapoints outside this domain. Then, we filtered out the iwaspoisoned twitter account as it was creating an artificial spike of cases near the end of the twitter datatset. This issue is discussed more in depth in a previous section. Once the data was cleaned, it was then normalized by the total number of cases in the corresponding datasets to make comparable on the same graph.

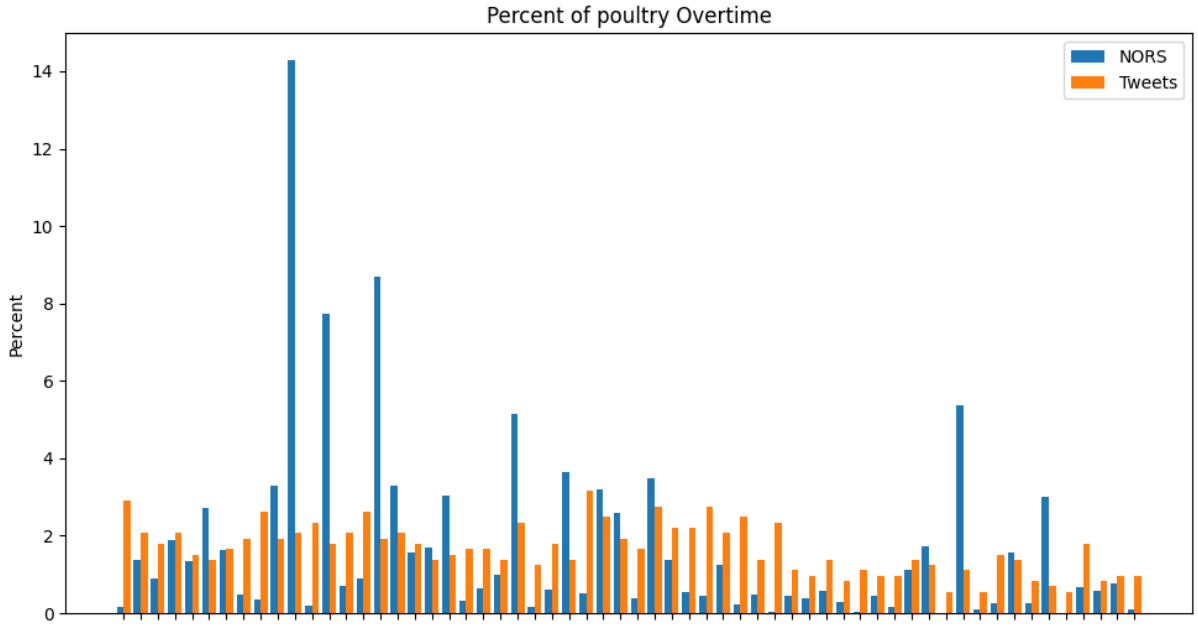


Figure 4.18: *Graph showing normalized volume of poultry overtime by dataset.*

The last procedure we performed to analyze the food categories, was a cross correlation between the two datasets with time lags. We are particularly interested in Tweets preceding NORS cases because the Tweets then, could be used as an early indicator for foodborne illness cases.

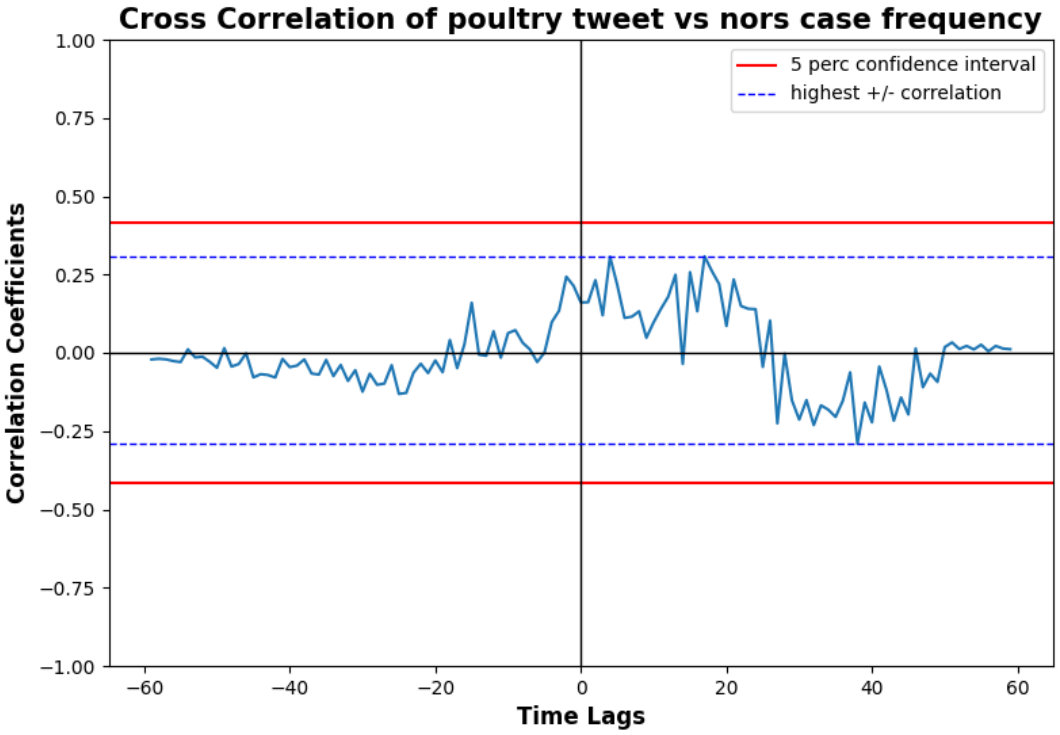


Figure 4.19: *Figure showing cross-correlation of poultry volume overtime with respect to time lags between datasets.*

Because we are interested in a positive correlation where the Tweets precede the NORS cases, in the graph about we will focus on the first quadrant (top-right quadrant). In the first quadrant, the max correlation was 0.307 with a lag of 4 months. This shows a weak positive correlation with a some lag, so this indicates that our positively predicted poultry Tweets could possibly warn us of an outbreak, assuming NORS cases is an accurate ground truth measure. Other food categories showed similar metrics, however, the smaller categories like the fish and egg categories did not show great results as there was simply not enough data.

4.3 Frontend Visualizations

We created a website, freely available to the public at *https://usda-foodpoisoning.wpi.edu/* to convey the results of our work to both policymakers and ordinary people. There are three core data sections, each showing a different aspect of our data collection.

First, on the homepage, the user is presented with a number of meaningful statistics about the tweets we have collected. We display the number of tweets collected, the number of tweets we are more than 90% confident were correctly identified by the algorithm, the number of tweets with symptom tokens, the number of tweets with city information, and then rank the top five states by tweet volume (in order, those are California, Texas, Florida, New York, and Ohio). The purpose here is to provide quick insights into the sheer volume of tweets analyzed (Figure 7.8).

Second, we display a graph showing the top symptoms over time for key foodborne illness-related keywords (Figure 3.5). This data is generated in real time from the backend because users will be able to filter results from within our time range. A keyword spiking in frequency could be an indication that a foodborne illness event is occurring.

Finally, we created an interactive map that displays the geocoordinates of every tweet containing location data in a heatmap (Figure 3.6). Areas with the greatest number of tweets about foodborne illness are dark shades of red, while areas with lower foodborne illness tweet density are teal. The heatmap aspect of the map is crucial for accurate data display, as a result of a limitation on the geodata we received - in almost all cases, the geolocation is simply a city or neighborhood, as opposed to exact coordinates that match the location of the original user

tweeted from. Thus, in many cases, several tweets have the exact same geocoordinates, which we must then plot on the map - for example, there are over 700 tweets with the same coordinates in different parts of New York City. Fortunately, with the heatmap's properties, more tweets with the same coordinates mean a darker shade of red, which should allow users to note that there is significant tweet volume associated with a specific point.

Beyond zooming in and out, the map also supports click actions. If a user taps on a location with tweet density, a popup will appear with the tweet's text. This behavior is key in validating our model's results - users can see the raw text of a tweet we've predicted to relate to foodborne illness and come to their own conclusion about whether our prediction was accurate. In the case where there are multiple tweets stacked on top of each other, users are presented with "Previous" and "Next" buttons, allowing them to cycle through the potentially hundreds of tweets associated with a specific location.

To create an interactive dashboard, we used the python module Dash. This tool help us develop a web based dashboard thats built on the React framework. This dashboard at the moment is disconnected from the rest of the website, however, because it is built on the React framework, we can merge the dashboard and the website relatively easily in future iterations of this project.

This prototype emphasized interactivity, giving users the freedom to change date ranges, to examine specific food groups, and filter out iwaspoisoned data points. This interactivity could benefit both government officials and the general public to have a broader view of potential foodborne illness outbreaks.

4.4 Machine Learning Model Performance Evaluation

As tweets were run through our machine learning model, a prediction value was given to show how confident the algorithm thought it was on each prediction. Overall the model had an average probability of .963 over approximately 598,000 predicted tweets. We split the data into predicted yes (positively predicted) having around 157,000 tweets and predicted no (negatively predicted) having around 441,000 tweets. For predicted yes the average probability was .940 and for predicted no the average was .972. This means that the model was more confident in predicting not related than predicting relating to foodborne illness.

We wanted to further analyze the performance of the machine learning model. To do this we hand-labeled around 1,000 tweets, this is a small portion of the tweets but will help give us an idea of the performance. Half of these tweets were from predicted yes and the other to predicted no. Tweets were selected at random but made sure to cover a distribution of varying probabilities given by the machine learning model. When hand labeling the tweets we used our best judgment to decide whether or not it was related to the foodborne illness. When we came across a tweet that we could not discernibly label we would skip that tweet. There were thousands of tweets so skipping tweets would not skew our data. We consider this new label for the data to be the ground truth data because it is what we created as the correct label. In the figure below there are 4 tweets we labeled. The ones with prediction 0 are predicted no from the machine learning model and then 1 is predicted yes. There is an example of when the model was correct and when it was incorrect for both positively and negatively predicting tweets.

Tweet Text	Prediction	Probability	Ground Truth
I been feeling nauseous all day ðŸ¥´	0	0.500685	no
No more tuna sub way that specific subway ever again. Feel terrible and cant kick nausea	0	0.9980692	yes
my back hurts so badly i feel nauseous	1	0.6701319	no
sorry I came back late from my lunch break. I was in the bathroom with food poisoning from TacoBell.	1	0.999419	yes

Figure 4.20: *Labeled tweets*

To compare the machine learning labeled data to our ground truth data we made a confusion matrix shown in Figure 4.18. From this we see that the number of correct values predicted is 740 or 74%. From the matrix we see that there were 14 falsely identified as negative when they should have been positive and there were 254 incorrectly identified as positive. The overall accuracy for correctly predicting tweets is .732 with the accuracy of the negatively predicted tweets to be .972 and the accuracy of the positively predicted tweets to be .492. The precision for this model is that 49% of the predicted yes is actually yes while for predicted no is no the precision is 97%. When looking at recall we find that all the instances that are actually yes the model predicted correctly 95% of the time. However for all the actual instances of no the model only correctly predicted 66% of them. The F1 score for predicted no is .78 and for yes is .65. The overall F1 score is .73. Since the F1 score is on the higher end it shows that it is a good model. There is still plenty of room for improvement but it is somewhat accurate.

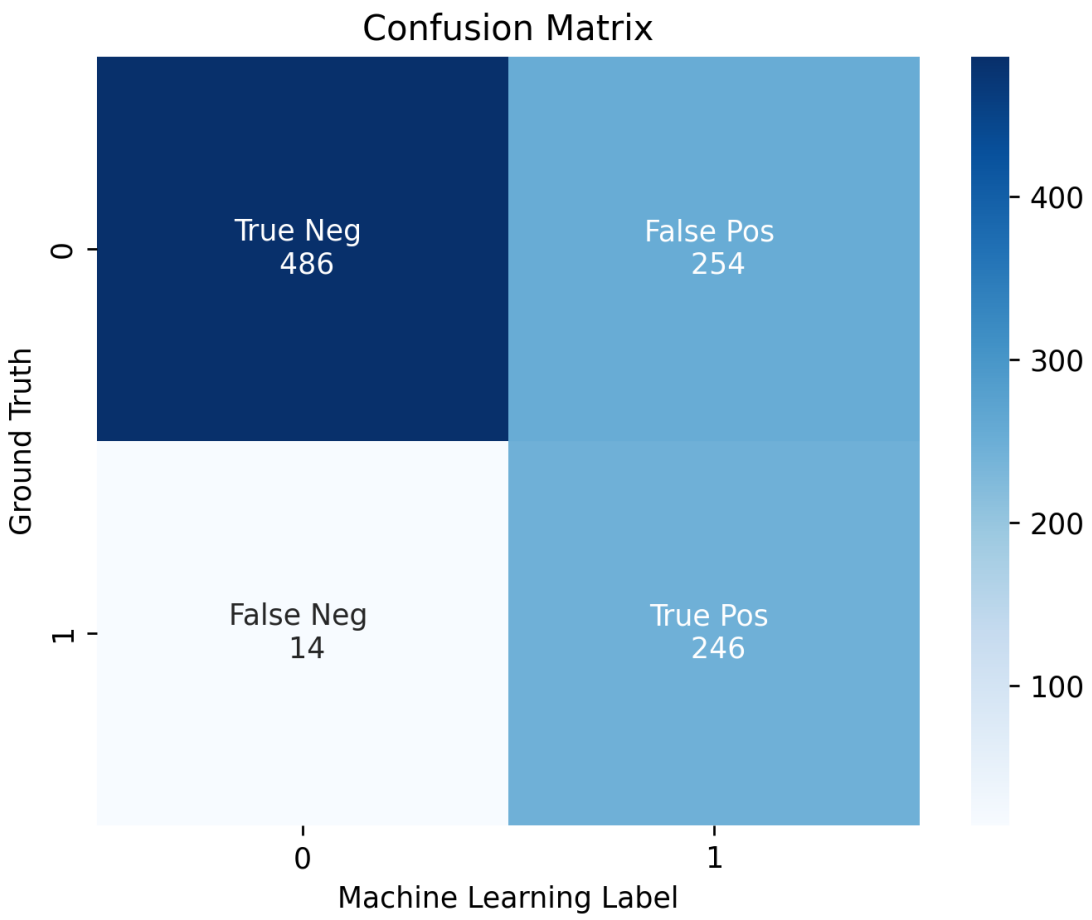


Figure 4.21: *Graph displaying the confusion matrix*

We wanted to compare the probabilities from the machine learning model to the ground truth data. From Figure 4.19 we see that for the predicted no when the probability was closer to .5 then there were more incorrectly predicted tweets then when the probability gets closer to 1. This shows that the model is pretty confident in its predicting ability for the negative tweets. However when we look at the predicted yes it is more split in half if it is correct or not. When comparing it to the probabilities there are more correctly predicted tweets towards 1. Looking closer to .5 probability there are more incorrectly predicted tweets. From this we can see that the

model is good at predicting negative tweets but has trouble distinguishing between positive and negative when it is closer to being related to foodborne illness.

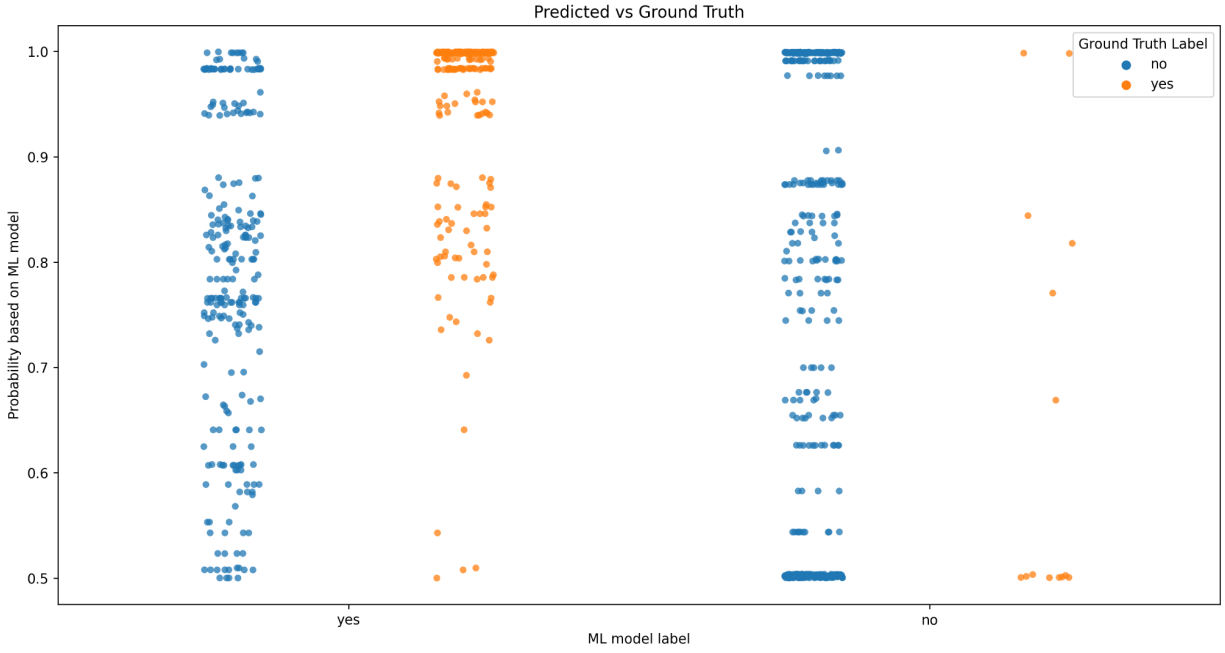


Figure 4.22: *Graph displaying the probability of the machine learning model vs ground truth*

4.5 Forecasting Tweets Results

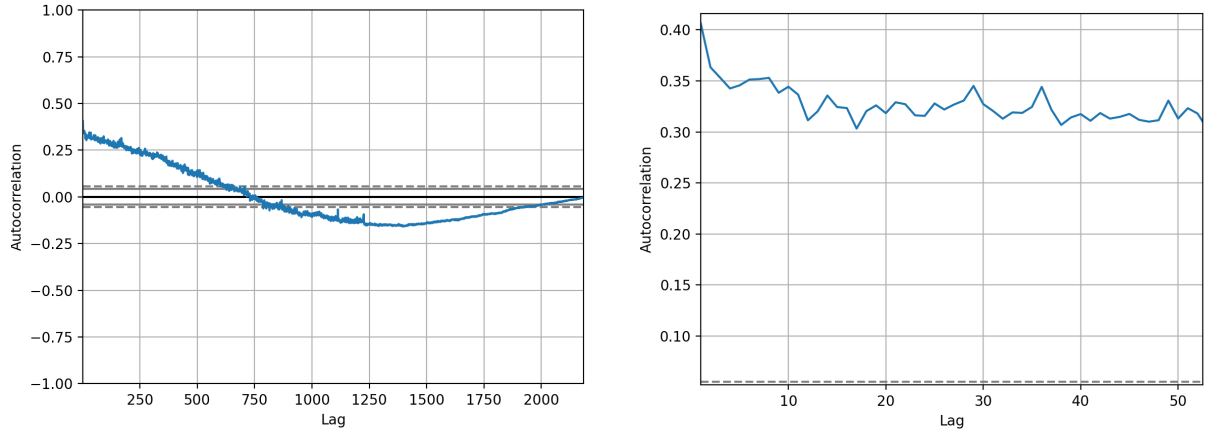


Figure 4.23: *The autocorrelation graph of the collected Twitter dataset, the left is the full graph and the right is zoomed in to the first 50 days of lag*

The above graphs in Figure 4.23 show the same thing, the autocorrelation of the Twitter data versus the lag. These graphs were created using the Python package Pandas, with the function autocorrelation_plot. They are the same graph, the one on the right is just zoomed in to be easier to see. The autocorrelation is highest around the one-to-two-day mark, at approximately 0.37, and then hits its next peak at the seven-to-eight-day mark at approximately 0.35. *MachineLearningMastery.com* suggested trying a lag with a higher autocorrelation as the lag for the time series (Brownlee, 2017). The autocorrelation is not very high, but there is some amount of correlation with the tweets around the same time frame evidenced by the fact that the autocorrelation decreases as the lag increases.

The forecasting method we used is autoregressive integrated moving average, ARIMA, model which uses previous data to predict future data (*What Is ARIMA Modeling?*, n.d.). The ARIMA model is within the Python package statsmodels (*Statsmodels*, n.d.). ARIMA forecasting is formatting ARIMA(p, d, q), where p is the number of autocorrelation terms, d is the differences needed for stationarities, and q is number of lagged forecast errors (*Introduction to ARIMA Models*, n.d.). For the preliminary testing we did we only varied the p value to change the number of autocorrelation terms, changing the number of days used to predict. Te equation with only varying p terms would look like:

$$\hat{Y}_t = Y_{t-1} + \ldots + Y_{t-p} + \phi_1 (Y_{t-1} - Y_{t-2}) - \theta_1 e_{t-1}$$

4.5.1 One-Day Prediction

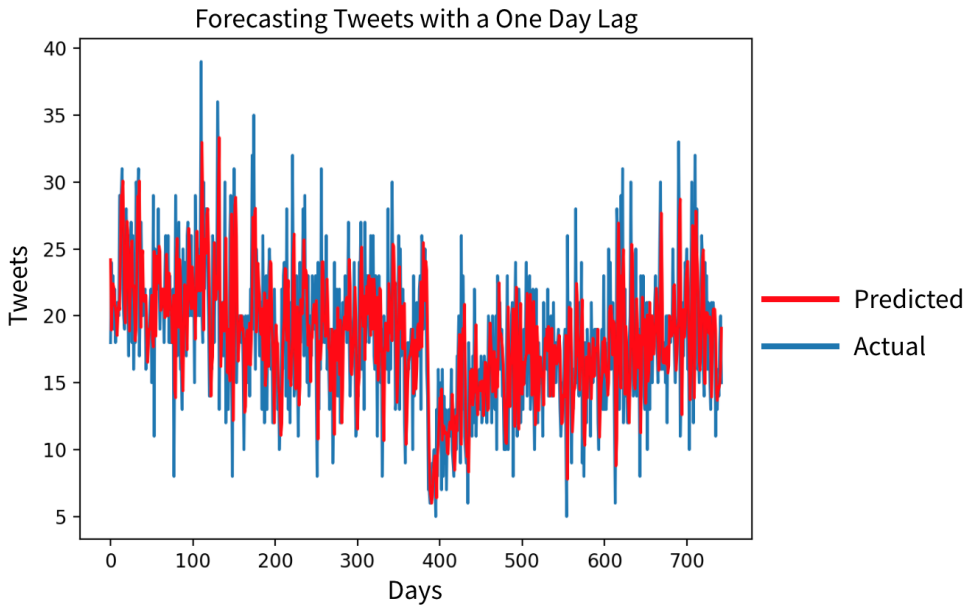


Figure 4.24: *Forecasting with one-day lag, ARIMA(1,1,1)*

The graph in Figure 4.24 shows the forecasted and actual tweets for a one-day lag. The one-day lag means that it is predicting today's tweets based on yesterday's tweets. The root mean squared error, RMSE, for this prediction is 5.591. This means that each day was off by approximately 5 tweets. Since some of the days only have 10 or so tweets, an error of 5 tweets could make a significant difference.

4.5.2 Seven-Day Prediction

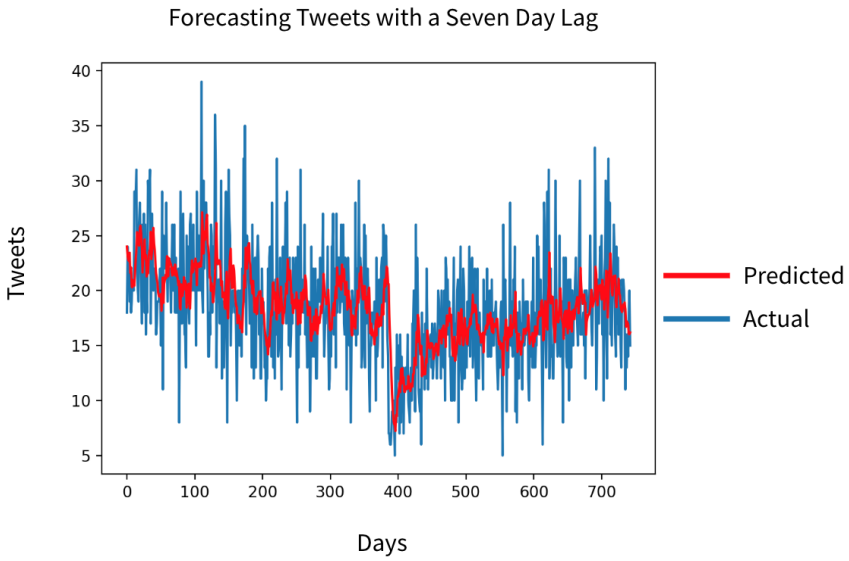


Figure 4.25: *Forecasting with a seven-day lag, ARIMA(7,1,1)*

The next lag we decided to test from the autocorrelation graph was a seven-day lag. This means that the ARIMA model is using a weeks-worth of tweets to predict the eighth day's tweets. The RMSE for a seven-day lag is 4.925. This is still a pretty large error compared to some of the data points' size, but it is an improvement. The difference between the one-day lag and the seven-day lag is that the one-day lag could potentially be fitting the spikes better, but that also leads to larger error if they predict a spike wrong, whereas the seven-day lag fits the trend better so it has a larger error on the spikes but fits the general data well.

4.5.3 General Prediction

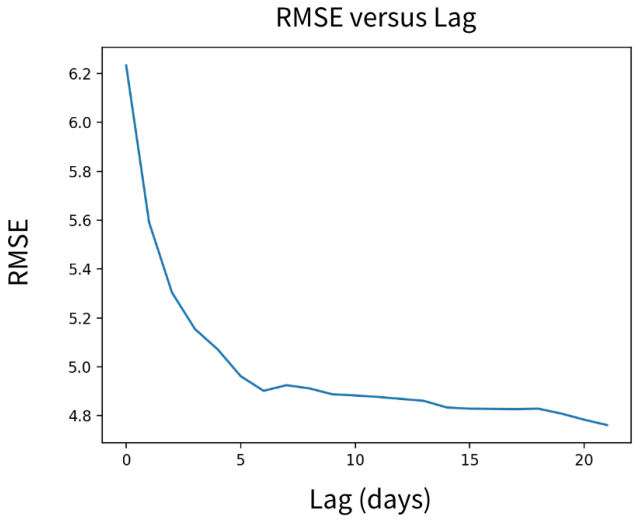


Figure 4.26: *RMSE versus lag in days*

The graph in Figure 4.26 shows the RMSE of the predictions with different lags. It can be seen that as the lag increases the RMSE decreases, but it stops decreasing as rapidly around the six-day mark. As seen by comparing Figures 4.24 and 4.25 as the lag increases the predictions seem to fit the trends of the data rather than trying to guess it exactly. This is because they have more information to base the predictions on and therefore match the overall trends rather than matching the previous day. So if there was a high amount the day prior, a one or two day lag would predict a high amount, but if the overall week's amounts were lower than that one day a longer lag would predict with the trend of however long the lag is. Both have merit in certain situations, but the longer lags do produce lower error.

Being able to predict tweets in advance would be helpful if there is a method to use tweets to predict whether or not there is a foodborne illness outbreak, attempts of a prediction methhod are detailed in the next section. Forecasting takes longer as the lag increases, so for the

sake of time if this method is used it seems like a six-to-seven-day lag is good enough to decrease the RMSE by a lot while also keeping the runtime shorter. Potential further improvements for this project include trying to forecast tweets further than one day out. The method that we are currently using only allows for tweets to be forecasted the day before, but if a different model were to be trained to allow for further out predictions that would allow for further out warning systems for foodborne illness.

4.6 Predicting NORS Cases

4.6.1 Linear Regression

To test the viability of the ability to predict NORS cases, the first method we tried was Linear regression. Linear regression will train on a certain number of tweets and previous NORS cases, with the process detailed in the previous paragraph. Thereafter, it then will predict a number of NORS cases for the next week (one number for the week, as that is the bins we have access to for the NORS data). The model will not have access to the current week's worth of NORS cases, only the previous week. It will have access to the tweets within the week it is predicting for, because they could be measured daily. We then looked at different metrics of each model, R-Squared, adjusted R-Squared, and RMSE, as well as the significance of the coefficents. These metrics allow us to assess how well each model is performing to determine if linear regression should be used.

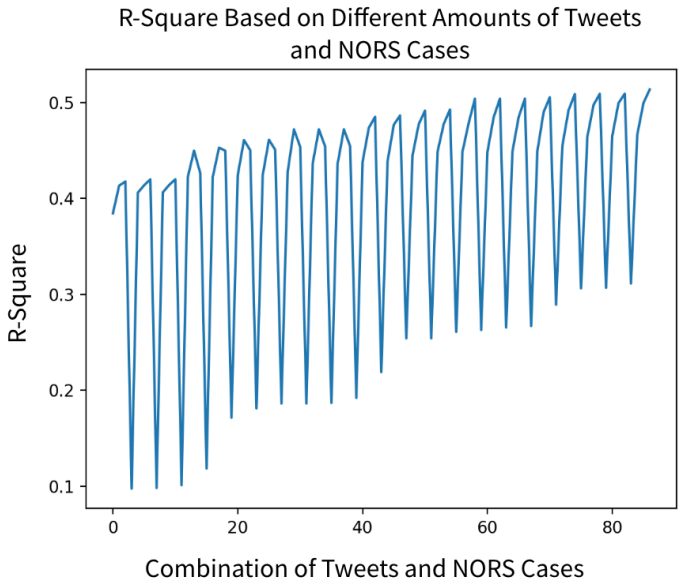


Figure 4.27: *R-Squared versus the number of tweets and NORS data used to predict*

Figure 4.27 is a graph of the R-Squared of the linear regression models with different numbers of Tweets and NORS cases, to see what has the best results. Based on this graph it seems like using 21 tweets and 3 weeks of NORS cases, yields an R-Squared of approximately 0.5. This means that the model can explain about half of the variation in the graph. For this model, of the 24 coefficients that are used, 6 have a p-value of less than 0.05, making them statistically significant. The rest of the coefficients have pretty high p-values, 16 of which have a p-value over 0.5. This means that a lot of the coefficients are not statistically significant meaning they support the null hypothesis and therefore they are not helping the prediction. The training RMSE for the model with the highest R-Squared, using 21 tweets and 3 NORS, was 119.5. When that model was applied to the validation set it had an RMSE of 93.1, meaning the model is not overfitting the data.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.513
Model:	OLS	Adj. R-squared:	0.447
Method:	Least Squares	F-statistic:	7.695
Date:	Wed, 26 Apr 2023	Prob (F-statistic):	3.69e-17
Time:	17:02:15	Log-Likelihood:	-1210.2
No. Observations:	200	AIC:	2470.
Df Residuals:	175	BIC:	2553.
Df Model:	24		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-61.1132	50.217	-1.217	0.225	-160.222	37.996
x1	-3.1094	1.164	-2.672	0.008	-5.406	-0.813
x2	0.1158	0.232	0.498	0.619	-0.343	0.574
x3	-0.4050	1.114	-0.364	0.717	-2.603	1.793
x4	2.2283	1.127	1.978	0.050	0.005	4.452
x5	-0.8153	1.215	-0.671	0.503	-3.214	1.583
x6	-0.5725	1.267	-0.452	0.652	-3.074	1.929
x7	0.2145	1.366	0.157	0.875	-2.482	2.911
x8	-1.1661	1.323	-0.882	0.379	-3.777	1.445
x9	-0.0357	0.231	-0.155	0.877	-0.491	0.420
x10	3.0017	1.123	2.674	0.008	0.786	5.217
x11	0.3168	1.123	0.282	0.778	-1.899	2.532
x12	1.1748	1.220	0.963	0.337	-1.233	3.582
x13	0.6918	1.254	0.552	0.582	-1.783	3.167
x14	2.3666	1.342	1.763	0.080	-0.283	5.016
x15	-0.1806	1.369	-0.132	0.895	-2.883	2.521
x16	-0.0028	0.231	-0.012	0.990	-0.460	0.454
x17	0.9390	1.116	0.841	0.401	-1.264	3.142
x18	-1.2548	1.112	-1.129	0.261	-3.449	0.940
x19	-0.5482	1.210	-0.453	0.651	-2.936	1.840
x20	0.0510	1.264	0.040	0.968	-2.443	2.545
x21	1.6065	1.262	1.273	0.205	-0.885	4.098
x22	0.0229	0.075	0.306	0.760	-0.125	0.171
x23	0.1972	0.079	2.496	0.013	0.041	0.353
x24	0.4094	0.075	5.492	0.000	0.262	0.556

Omnibus:	49.444	Durbin-Watson:	1.997
Prob(Omnibus):	0.000	Jarque-Bera (JB):	105.507
Skew:	1.149	Prob(JB):	1.23e-23
Kurtosis:	5.716	Cond. No.	3.38e+03

Figure 4.28: *Regression report for highest R-Squared model*

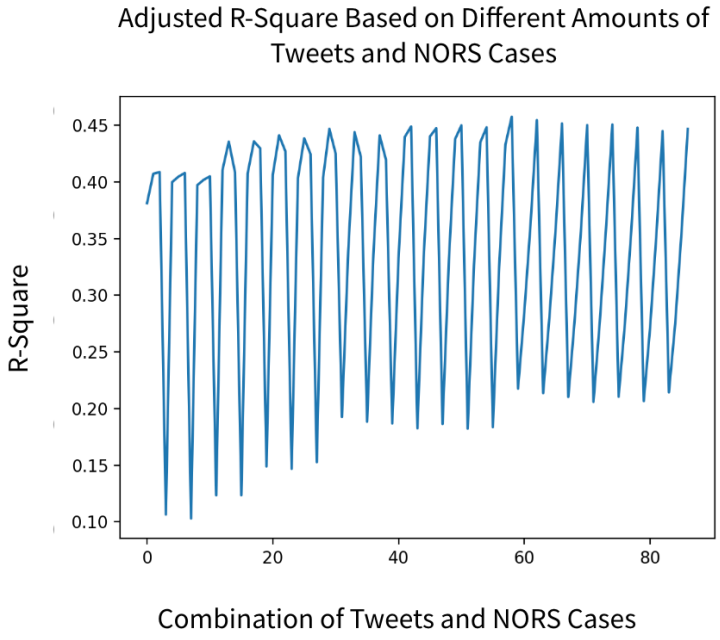


Figure 4.29: *Adjusted R-Squared versus the number of tweets and NORS data used to predict*

Figure 4.29 is a graph of the adjusted R-Squared for the same method that loops through different combinations of tweets and NORS cases. This gives us a different result than the regular R-Squared graph. This R-Squared is adjusted to reflect whether the extra coefficient actually makes a difference, or if it is just allowing the model to overfit. This shows that the regular R-Squared is probably overfitting with extra data rather than each factor being significant. For the adjusted R-Squared the best model uses 14 tweets and 3 weeks of NORS cases, this yields an R-Squared of 0.45. Of the 17 coefficients, 5 of them have a p-value less than 0.05, making them statistically significant. 9 have a p-value over 0.5. The training RMSE for the model with the highest adjusted R-Squared, with 14 tweets and 3 NORS, was 126.2. When that model was applied to the validation set it had an RMSE of 74.1, meaning the model is not overfitting the data, and performing with less error than expected.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.504
Model:	OLS	Adj. R-squared:	0.457
Method:	Least Squares	F-statistic:	10.87
Date:	Wed, 26 Apr 2023	Prob (F-statistic):	6.28e-20
Time:	17:02:15	Log-Likelihood:	-1212.2
No. Observations:	200	AIC:	2460.
Df Residuals:	182	BIC:	2520.
Df Model:	17		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-56.0271	46.718	-1.199	0.232	-148.206	36.152
x1	-3.0820	1.129	-2.731	0.007	-5.309	-0.855
x2	0.0882	0.227	0.389	0.698	-0.359	0.535
x3	-0.2861	1.085	-0.264	0.792	-2.426	1.854
x4	2.0457	1.075	1.903	0.059	-0.076	4.167
x5	-0.5602	1.180	-0.475	0.636	-2.889	1.769
x6	-0.4154	1.219	-0.341	0.734	-2.820	1.990
x7	0.2207	1.319	0.167	0.867	-2.382	2.823
x8	-1.1094	1.305	-0.850	0.396	-3.684	1.466
x9	-0.0797	0.224	-0.356	0.722	-0.521	0.362
x10	3.0227	1.087	2.782	0.006	0.879	5.167
x11	0.3097	1.095	0.283	0.778	-1.851	2.470
x12	1.1947	1.176	1.016	0.311	-1.125	3.514
x13	0.5298	1.200	0.441	0.659	-1.838	2.898
x14	2.4897	1.220	2.040	0.043	0.082	4.897
x15	0.0196	0.073	0.268	0.789	-0.125	0.164
x16	0.1930	0.077	2.508	0.013	0.041	0.345
x17	0.4082	0.072	5.644	0.000	0.265	0.551

Omnibus:	48.715	Durbin-Watson:	1.993
Prob(Omnibus):	0.000	Jarque-Bera (JB):	105.115
Skew:	1.127	Prob(JB):	1.49e-23
Kurtosis:	5.745	Cond. No.	3.13e+03

Figure 4.30: *Regression report for highest adjusted R-Squared model*

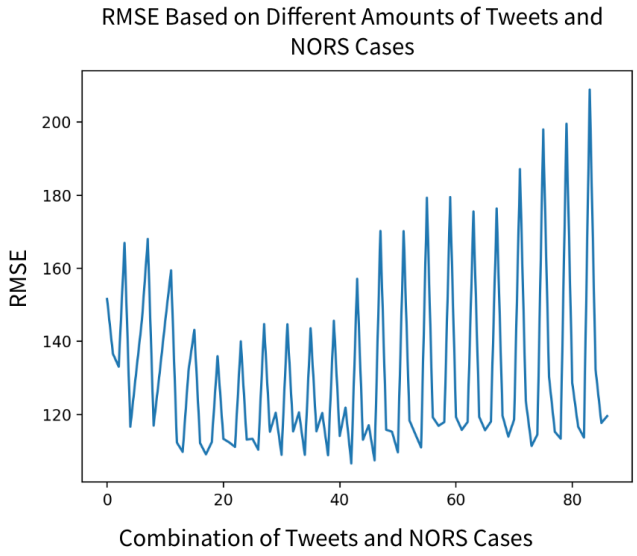


Figure 4.31: *RMSE verse number of tweets and NORS cases used to predict*

The root mean squared error is a metric that conveys the difference between the model-predicted value and the actual value. This can be used to look at the average amount the model will be off by. This graph shows the different amounts for all the different models tested. This shows that the lowest error comes from the model that uses 10 tweets and 3 NORS cases. The training RMSE for the model with the lowest RMSE, using 10 tweets and 3 NORS, was 106.5. When that model was applied to the validation set it had an RMSE of 77.6, meaning the model is not overfitting the data.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.485
Model:	OLS	Adj. R-squared:	0.449
Method:	Least Squares	F-statistic:	13.47
Date:	Wed, 26 Apr 2023	Prob (F-statistic):	8.97e-21
Time:	17:07:38	Log-Likelihood:	-1215.9
No. Observations:	200	AIC:	2460.
Df Residuals:	186	BIC:	2506.
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-13.4994	43.711	-0.309	0.758	-99.733	72.734
x1	-2.8748	1.128	-2.549	0.012	-5.100	-0.650
x2	-0.0039	0.223	-0.018	0.986	-0.444	0.436
x3	-0.0286	1.087	-0.026	0.979	-2.173	2.116
x4	2.5265	1.060	2.383	0.018	0.435	4.618
x5	-0.1599	1.174	-0.136	0.892	-2.477	2.157
x6	-0.2968	1.211	-0.245	0.807	-2.686	2.092
x7	0.6822	1.297	0.526	0.600	-1.877	3.242
x8	-0.5900	1.298	-0.455	0.650	-3.150	1.970
x9	-0.1195	0.225	-0.532	0.595	-0.563	0.324
x10	3.5172	1.061	3.314	0.001	1.424	5.611
x11	0.0490	0.072	0.683	0.495	-0.093	0.191
x12	0.2083	0.077	2.711	0.007	0.057	0.360
x13	0.4065	0.073	5.604	0.000	0.263	0.550

Omnibus:	56.716	Durbin-Watson:	1.988
Prob(Omnibus):	0.000	Jarque-Bera (JB):	131.747
Skew:	1.278	Prob(JB):	2.46e-29
Kurtosis:	6.046	Cond. No.	2.88e+03

Figure 4.32: *Regression report for lowest RMSE model*

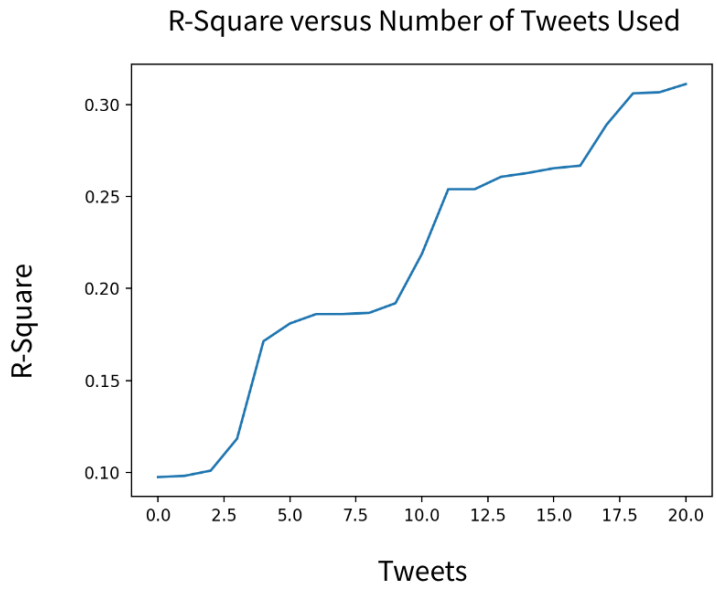
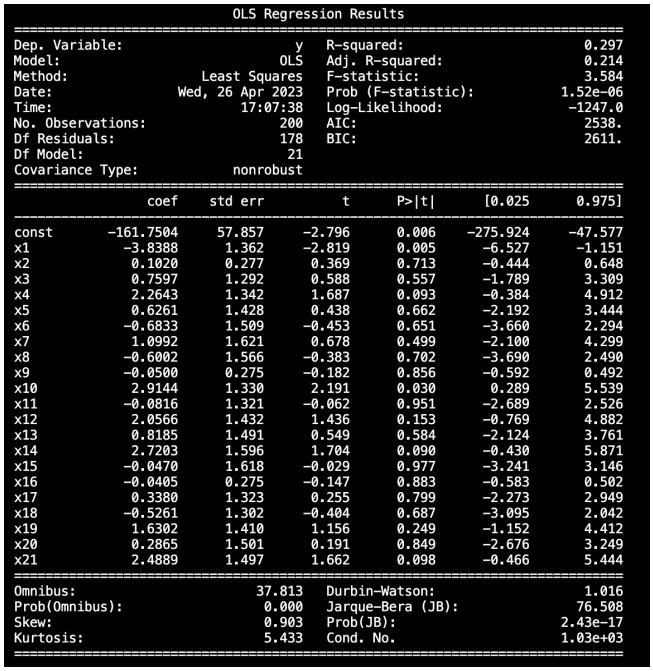


Figure 4.33: *R-Squared verse number of tweets used to predict, no NORS data used*

This is the increase in R-Squared without any NORS data graph, this graph shows how the R-Squared increases with the addition of tweets, with 21 tweets having the highest

R-Squared of 0.28. This model, however, is overfitting, in the next graph, the adjusted R-Squared does not show the same levels of R-Square. This is increasing just because there is more information given to the model, not that it is learning to fit better. It is learning to overfit. Of the 21 coefficients, three have a p value less than 0.05, but 13 have a p value greater than 0.5.



OLS Regression Results

Dep. Variable:	y	R-squared:	0.297
Model:	OLS	Adj. R-squared:	0.214
Method:	Least Squares	F-statistic:	3.584
Date:	Wed, 26 Apr 2023	Prob (F-statistic):	1.52e-06
Time:	17:07:38	Log-Likelihood:	-1247.0
No. Observations:	200	AIC:	2538.
Df Residuals:	178	BIC:	2611.
Df Model:	21		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-161.7504	57.857	-2.796	0.006	-275.924	-47.577
x1	-3.8388	1.362	-2.819	0.005	-6.527	-1.151
x2	0.1020	0.277	0.369	0.713	-0.444	0.648
x3	0.7597	1.292	0.588	0.557	-1.789	3.309
x4	2.2643	1.342	1.687	0.093	-0.384	4.912
x5	0.6261	1.428	0.438	0.662	-2.192	3.444
x6	-0.6833	1.509	-0.453	0.651	-3.660	2.294
x7	1.0992	1.621	0.678	0.499	-2.100	4.299
x8	-0.6002	1.566	-0.383	0.702	-3.690	2.490
x9	-0.0500	0.275	-0.182	0.856	-0.592	0.492
x10	2.9144	1.330	2.191	0.030	0.289	5.539
x11	-0.0816	1.321	-0.062	0.951	-2.689	2.526
x12	2.0566	1.432	1.436	0.153	-0.769	4.882
x13	0.8185	1.491	0.549	0.584	-2.124	3.761
x14	2.7203	1.596	1.704	0.090	-0.430	5.871
x15	-0.0470	1.618	-0.029	0.977	-3.241	3.146
x16	-0.0405	0.275	-0.147	0.883	-0.583	0.502
x17	0.3380	1.323	0.255	0.799	-2.273	2.949
x18	-0.5261	1.302	-0.404	0.687	-3.095	2.042
x19	1.6302	1.410	1.156	0.249	-1.152	4.412
x20	0.2865	1.501	0.191	0.849	-2.676	3.249
x21	2.4889	1.497	1.662	0.098	-0.466	5.444

Omnibus:	37.813	Durbin-Watson:	1.016
Prob(Omnibus):	0.000	Jarque-Bera (JB):	76.508
Skew:	0.903	Prob(JB):	2.43e-17
Kurtosis:	5.433	Cond. No.	1.03e+03

Figure 4.34: *Regression report for highest R-Squared, just tweets model*

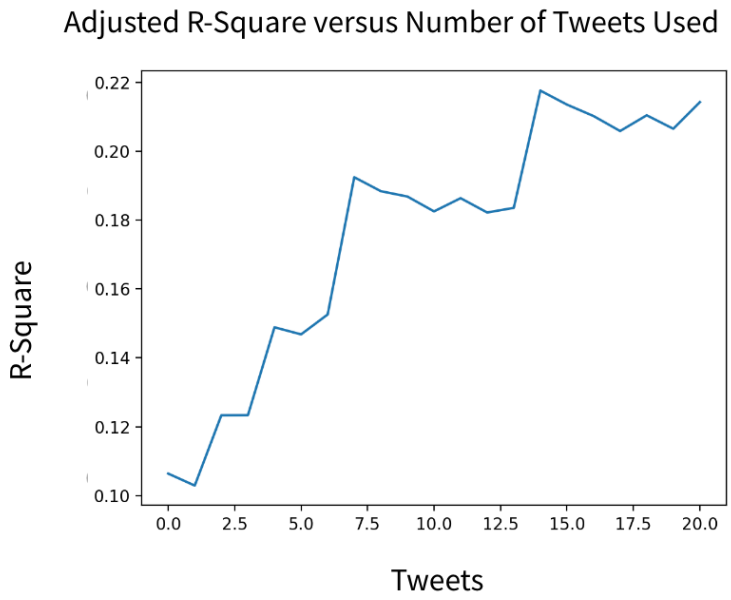
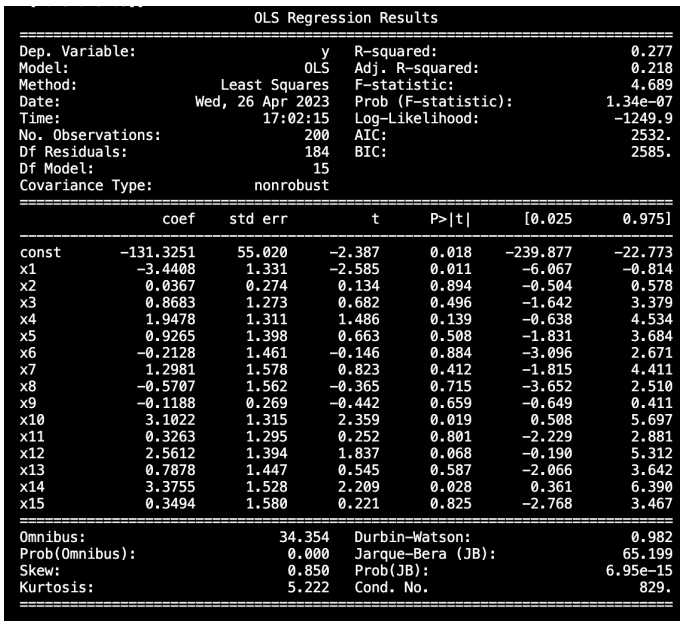


Figure 4.35: *Adjusted R-Squared graph, only tweets*

Although the adjusted R-Squared for only tweets is not that high, the increase in adjusted R-Squared as more tweets are added does show that more tweets does have an impact on the prediction. The increase in tweets does allow the model to do slightly better at predicting the NORS cases. Fifteen tweets has the best adjusted R-Squared, with an adjusted R-Squared of 0.21. Of 15 coefficients, three are statistically significant with a p-value less than 0.05. Eight of these coefficients have a p-value greater than 0.5. The linear regression that trains using 15 tweets has an RMSE of 149.7, when tested on the validation set it had an RMSE of 98.6. This means that the model fit well, and did not overfit for the error.



OLS Regression Results

Dep. Variable:	y	R-squared:	0.277
Model:	OLS	Adj. R-squared:	0.218
Method:	Least Squares	F-statistic:	4.689
Date:	Wed, 26 Apr 2023	Prob (F-statistic):	1.34e-07
Time:	17:02:15	Log-Likelihood:	-1249.9
No. Observations:	200	AIC:	2532.
Df Residuals:	184	BIC:	2585.
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-131.3251	55.020	-2.387	0.018	-239.877	-22.773
x1	-3.4408	1.331	-2.585	0.011	-6.067	-0.814
x2	0.0367	0.274	0.134	0.894	-0.504	0.578
x3	0.8683	1.273	0.682	0.496	-1.642	3.379
x4	1.9478	1.311	1.486	0.139	-0.638	4.534
x5	0.9265	1.398	0.663	0.508	-1.831	3.684
x6	-0.2128	1.461	-0.146	0.884	-3.096	2.671
x7	1.2981	1.578	0.823	0.412	-1.815	4.411
x8	-0.5707	1.562	-0.365	0.715	-3.652	2.510
x9	-0.1188	0.269	-0.442	0.659	-0.649	0.411
x10	3.1022	1.315	2.359	0.019	0.508	5.697
x11	0.3263	1.295	0.252	0.801	-2.229	2.881
x12	2.5612	1.394	1.837	0.068	-0.190	5.312
x13	0.7878	1.447	0.545	0.587	-2.066	3.642
x14	3.3755	1.528	2.209	0.028	0.361	6.390
x15	0.3494	1.580	0.221	0.825	-2.768	3.467

Omnibus:	34.354	Durbin-Watson:	0.982
Prob(Omnibus):	0.000	Jarque-Bera (JB):	65.199
Skew:	0.850	Prob(JB):	6.95e-15
Kurtosis:	5.222	Cond. No.	829.

Figure 4.36: *Regression report for highest adjusted R-Squared, just tweets model*

There are limitations to the data that we can assess in a certain timeframe that makes this regression problem more difficult. The NORS data that we are using as the ground truth realistically won't be available to the public for about a year after it occurs. Even if we could work closely with the CDC and NORS to get previous outbreak data, outbreaks may not be reported as soon as they happen. To try to make this prediction more realistic, we decided to use a three-week lag. This could potentially allow enough time for cases to be reported and processed, so they can then be used to predict future cases.

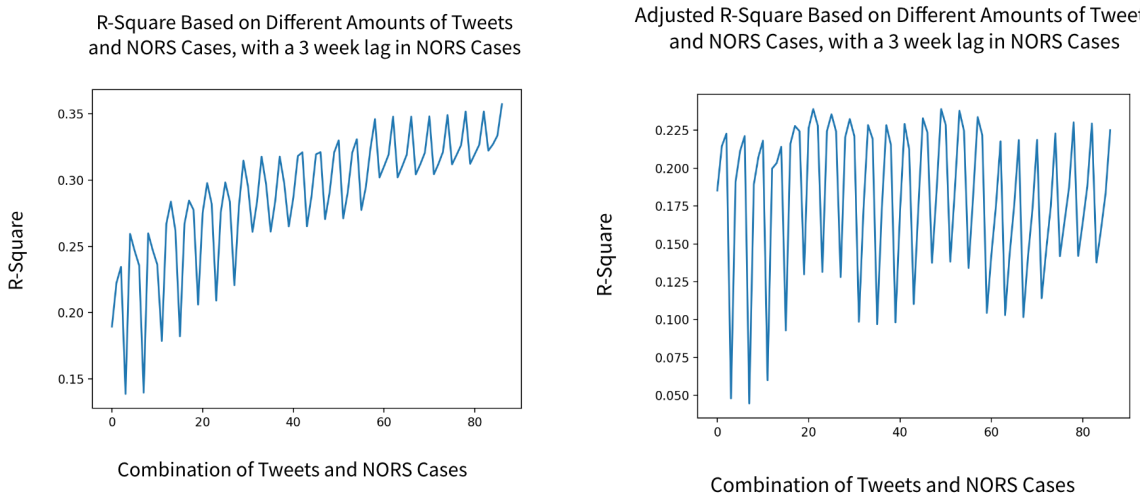


Figure 4.37: *R-Squared and Adjusted R-Squared with a 3-week lag on the NORS Cases*

These graphs show the R-Squared and adjusted R-Squared, respectively. Overall with a three-week lag there is a decrease in the R-Squared levels we were seeing before, because now on top of trying to predict something in advance we are trying to predict something with not the most up to date data. The regular R-Squared maxes out at around 0.35, and the adjusted around 0.22. These are weak R-Squared, but we do still see an overall improvement in both with the use of previous NORS cases, despite them being out of date.

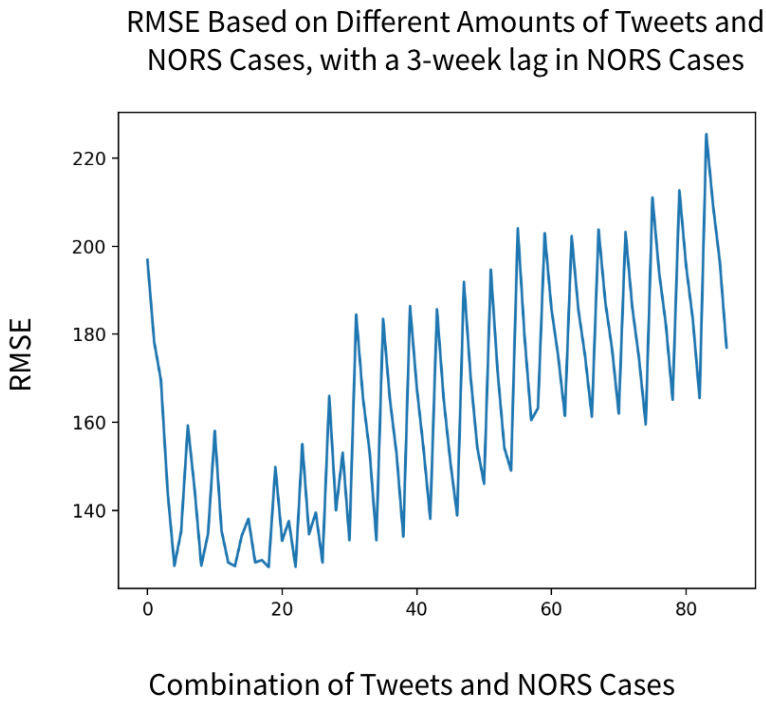


Figure 4.38: *RMSE with a 3-week lag on the NORS Cases*

This graph shows the training RMSE of the model with the three-week lag. The training error did not increase by that much compared to the non-lagged graph. Both the model with the highest R-Squared and the model with the highest adjusted R-Squared actually performed better on the validation set than they did on the training set. The RMSE for the highest R-Squared was 133.1 and the validation RMSE was 86.2. The RMSE for the highest adjusted R-Squared was 129.2 and the validation RMSE was 80.1. These show that the model is fitting the data well and not overfitting.

Overall, from the different test surrounding linear analysis, it does not seem to be the best model to use to predict the number of NORS cases from tweets and previous NORS cases. The models do perform well on on testing data, but even the best performing models are only moderately correlated.

4.6.2 Logistic Regression

In order to explore different methods to see what yields the best results for this prediction method, we decided to look into logistic regression after linear regression. Doing logistic regression on this dataset requires a shift in our goal because it is a classification method. Instead of predicting the number of cases, we will instead be looking at binary classification. This binary is created by establishing a certain threshold for "outbreak" - 1 or "not outbreak" - 0. For the sake of our testing, we used 200 cases to be that threshold, but this could be changed and the model could be retrained on that new binary threshold. This is useful to show days that have more than a certain number of cases, to show when the higher risk times are.

The equation for this logistic regression would look similar to the equation for linear regression, but it would produce a number between 0 and 1, and then whichever it is closer to would be the predicted label.

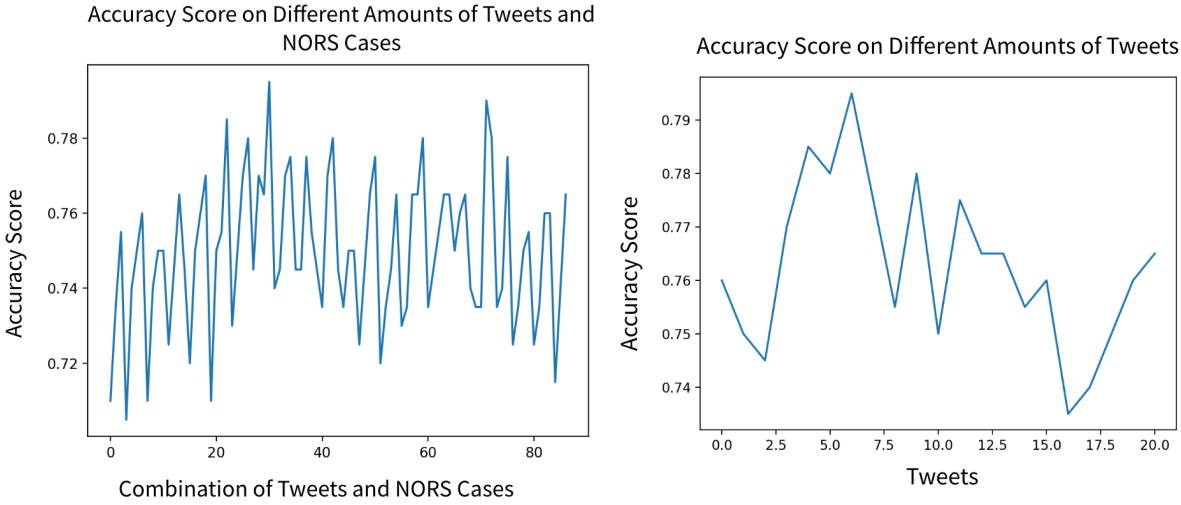


Figure 4.39: *Accuracy score of logistic regression with tweets and NORS cases, and with tweets alone*

The accuracy score of a logistic model conveys how many predictions the model predicted correctly. The highest accuracy score for a model with tweets and NORS came from the model with 7 tweets, 2 NORS and produced an accuracy score of 0.77. This same model when used on the validation set produced an accuracy score of 0.76. This shows that the model is fitting the data well. Similarly, the model with just tweets, the best accuracy score came from the model with 7 tweets, with an accuracy score of 0.75. When run through the validation set this came back with an accuracy score of 0.78. The logistic regression model seems to be pretty accuracy in determining whether or not the NORS cases will be above 200.

From this preliminary analysis, it seems like logistic regression may be a better model to use to predict tweets. Although it would require more testing to establish how accurate it may be.

5. Future Work

5.1 Word Importance Analysis

In the context of natural language processing, the ability to attach explainable terms to tokens is extremely important for both improving model performance and for explainability to non-data scientists for decision making. One of the techniques that could be utilized to explain the influence of the input tokens is a salience technique for examining the gradient changes based on an input token (Bastings, 2020). From a model engineering perspective, this technique could help create a subset of important keywords which could be utilized to update the query search keywords for the twitter data collection step. This technique could also help understand new keywords to be added to the keyword list. From an analysis/decision making perspective, this technique could help us understand which foods, locations, or spoken words indicate a

higher risk of foodborne illness, strictly in the twitter domain as the model is trained on only twitter data.

5.2 Collaboration with iWasPoisoned

As noted in the earlier sections, the iWasPoisoned twitter account made up a large portion of the collected tweets that were predicted positive. Due to this fact and the fact that iWasPoisoned has a similar goal to our tool's, we suggest a collaboration with representatives at the iwaspoisoned.com. With more experience working in the information technology field and with more niche knowledge about food poisoning, we believe that this project would benefit from a collaboration. We would be collecting the iWasPoisoned data from the source, rather than through Twitter. They currently do not have a map or timeline set up, so we could help them create those. Their established reporting system could be used with our visualizations to show trends in their data to create a warning system dashboard.

5.3 Streaming Tweet Collection

Currently, the working pipeline is set up so that the collection and processing steps need to run individually and manually by a user. In the future, this process could be set up in a way where the pipeline constantly collects new tweets on an interval, to be passed into the predictive model, and updated in the database and server. Another alternative is to collect and update only when there is a call from the front end user interface. This would save resources as we would only make calls on a user need basis. Another update to the collection method would be to start collecting tweets from streaming, or streaming from the archive with more real time tweets. This will potentially allow more real time detection.

5.4 Frontend User Interface

User interface development is an iterative process, and there are many improvements that could be made to both improve the information conveyed to the user, and the accessibility of the site. Future work could see an expansion of data visualization formats, but there are several ways to improve the visualizations we are currently displaying:

First, the data from the homepage could be rearranged to include more statistics, and research could be done to identify the most relevant statistics to display. The homepage data could also include various interactivity options. For example, when a user clicks on a specific data card, they could be presented with an explanation of why the piece of data is especially relevant and significant.

Second, further filtering options could be added to the frequency graph. A robust filtering system could be shared by both the frequency graph and the map, and in addition to just filtering by time, users could also potentially filter by prediction confidence, as well as several other factors we also kept track of. For the frequency graph specifically, future improvements could also include the ability to filter out tweets by geolocation, so that if the user wanted to see the symptoms over time for a specific city or state, they could do so using the graph.

Third, the map could be improved to convey more information about the specific tweets themselves, such as the tweet poster's username, specific keywords identified as relating to foodborne illness, prediction confidence score, as well as a link to the tweet itself.

Finally, accessibility of the website could be improved by running the site through various accessibility analytics tools, to make sure that it is easily usable by users leveraging tools such as screen readers. User testing could identify problem areas in real-world usage of the site, and plans could be made to mitigate or fix those specific problems.

5.5 Forecasting and Predicting

There are a few improvements that need to be looked into for the future of the prediction method. First, since the dataset is pretty small for NORS data, to get more accurate testing done they should probably use bootstrapping, cross-validation, or a larger dataset. This would help mitigate the potential error that comes from having a small dataset.

The next step in this process would be to connect the tweet forecasting method to the NORS prediction method and see how that performs. This can then potentially be improved by using forecasted NORS cases to forecast more NORS cases.

6. References

[1] (n.d.). Report Food Poisoning Now. Protect Others. Retrieved February 28, 2023, from https://iwaspoisoned.com/

[2] Bastings, J., & Filippova, K. (2020). *The elephant in the interpretability room: Why use attention as explanation when we have saliency methods?* https://arxiv.org/pdf/2010.05607.pdf

[3] Boxrud, D., Monson, T., Stiles, T., & Besser, J. (2010). *The Role, Challenges, and Support of PulseNet Laboratories in Detecting Foodborne Disease Outbreaks*. Sagepub. Retrieved February 27, 2023, from https://journals.sagepub.com/doi/pdf/10.1177/00333549101250S207

[4] Brown, S. (2021, April 21). *Machine learning, explained*. MIT Sloan. Retrieved February 27, 2023, from https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

[5] Brownlee, J. (2017, January 9). *How to Create an ARIMA Model for Time Series Forecasting in Python - MachineLearningMastery.com*. Machine Learning Mastery. Retrieved April 27, 2023, from https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

[6] *Burden of Foodborne Illness: Findings | Estimates of Foodborne Illness | CDC*. (n.d.). Centers for Disease Control and Prevention. Retrieved March 1, 2023, from https://www.cdc.gov/foodborneburden/2011-foodborne-estimates.html

[7] Casey, C. (2022, September 12). *FDA closes investigation into reported illnesses from Lucky Charms*. Food Dive. https://www.fooddive.com/news/fda-closes-investigation-into-reported-illnesses-from-lucky-charms/631572/

[8] Center for Disease Control and Prevention. (2023, 3 10). *IFSAC Food Categorization Scheme*. CDC. Retrieved 4 24, 2023, from https://www.cdc.gov/foodsafety/ifsac/projects/food-categorization-scheme.html

[9] *D3.js*. (n.d.). D3.js Documentation. Retrieved March 19, 2023, from https://www.d3js.org

[10] Du, Y. (2022, January). *Machine learning techniques and research framework in foodborne disease surveillance system*. ScienceDirect. Retrieved February 27, 2023, from https://doi.org/10.1016/j.foodcont.2021.108448

[11] *Food Safety | Food Safety and Inspection Service*. (n.d.). USDA Food Safety and Inspection Service. Retrieved February 27, 2023, from https://www.fsis.usda.gov/food-safety

[12] *Geo objects | Docs | Twitter Developer Platform*. (n.d.). Twitter Developers. Retrieved February 28, 2023, from https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/geo

[13] Holloway, E. (2019, November 7). *Machine Learning, Part 3: Don't Snoop on Your Data*. Mind Matters. Retrieved February 27, 2023, from https://mindmatters.ai/2019/11/machine-learning-part-3-dont-snoop-on-your-data/

[14] Hu, R., Zhang, D., Tao, D., Hartvigsen, T., Feng, H., & Rundensteiner, E. (2022). *TWEET-FID: An Annotated Dataset for Multiple Foodborne Illness Detection Tasks.* ArXiv [Cs.CL]. Retrieved from http://arxiv.org/abs/2205.10726

[15] *Introduction to ARIMA models*. (n.d.). Duke People. Retrieved April 27, 2023, from https://people.duke.edu/~rnau/411arim.htm#pdq

[16] *julianfssen/tweetkit: ⚠️ WIP: Twitter v2 API client for Ruby*. (n.d.). GitHub. Retrieved March 3, 2023, from https://github.com/julianfssen/tweetkit

[17] Lee, B. Y., & Forbes. (2022, May 7). *Over 7,300 Lucky Charms Cereal Illness Complaints On Iwaspoisoned.com, FDA Investigating*. Forbes. https://www.forbes.com/sites/brucelee/2022/05/07/over-7300-lucky-charms-cereal-illness-complaints-on-iwaspoisonedcom-fda-investigating/?sh=52992554230a

[18] *List of Multistate Foodborne Outbreak Notices | CDC*. (n.d.). Centers for Disease Control and Prevention. Retrieved February 28, 2023, from https://www.cdc.gov/foodsafety/outbreaks/lists/outbreaks-list.html

[19] *Mapbox*. (n.d.). Mapbox Documentation. Retrieved March 19, 2023, from https://www.mapbox.com

[20] *National Outbreak Reporting System (NORS) Dashboard | CDC*. (n.d.). gov.cdc.wwwn. Retrieved February 28, 2023, from https://wwwn.cdc.gov/norsdashboard/

[21] *Next.js*. (n.d.). Next.js Documentation. Retrieved March 19, 2023, from https://www.nextjs.org

[22] *React*. (n.d.). React Documentation. Retrieved March 19, 2023, from https://www.react.dev

[23] *Scikit-Learn*. (n.d.). scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation. Retrieved April 27, 2023, from https://scikit-learn.org/

[24] *State Population Totals: 2020-2022*. (n.d.). U.S. Census Bureau. Retrieved February 28, 2023, from https://www.census.gov/data/tables/time-series/demo/popest/2020s-state-total.html

[25] *Statsmodels*. (n.d.). Statsmodels. Retrieved April 27, 2023, from https://www.statsmodels.org

[26] *Tailwind CSS*. (n.d.). Tailwind CSS Documentation. Retrieved March 19, 2023, from https://www.tailwindcss.com

[27] *Timeline for Identifying and Reporting Cases in Foodborne Outbreaks | CDC*. (2022, September 22). Centers for Disease Control and Prevention. Retrieved February 27, 2023, from https://www.cdc.gov/foodsafety/outbreaks/basics/reporting-timeline.html

[28] *Tweet object | Docs | Twitter Developer Platform*. (n.d.). Twitter Developers. Retrieved February 28, 2023, from https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet

[29] *User object | Docs | Twitter Developer Platform*. (n.d.). Twitter Developers. Retrieved February 28, 2023, from https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/user

[30] *What Is ARIMA Modeling?* (n.d.). Master's in Data Science. Retrieved April 27, 2023, from https://www.mastersindatascience.org/learning/statistics-data-science/what-is-arima-modeling/

7. Appendix

	cdcid	DateFirstIll	EstimatedPrimary	DeathsInfo	...	FoodNames	IFSACLevel1	TaxonomyConfirmation	DateLastIll
3700	296551	2020-06-19	1132	708.0	...	[red, onion]	[plant, produce, vegetables, root, underground...	confirmed	9/14/2020 0:00
4308	299302	2021-05-31	1040	0.0	...	[onions]	[plant, produce, vegetables, root, underground...	confirmed	12/24/2021 0:00
3788	296798	2020-05-11	771	0.0	...	[salad, mix, bagged]	[plant, produce, vegetables]	confirmed	7/24/2020 0:00
2294	282606	2018-11-26	690	290.0	...	[]	[]	confirmed	12/6/2018 0:00
2053	281107	2018-07-26	647	647.0	...	[]	[]	confirmed	8/1/2018 0:00
2427	283434	2018-05-20	559	517.0	...	[salad, mix, containing, lettuce, and, carrots]	[plant, produce, vegetables]	confirmed	7/23/2018 0:00
1438	277325	2017-11-14	557	557.0	...	[turkey, gravy]	[land, animals, meat, poultry, poultry, turkey]	confirmed	11/21/2017 0:00
2859	287637	2018-03-09	500	500.0	...	[]	[]	confirmed	3/23/2018 0:00
2810	287467	2018-08-05	436	374.0	...	[ground, beef]	[land, animals, meat, poultry, meat, beef, non...	confirmed	2/8/2019 0:00
1545	278110	2017-12-30	420	420.0	...	[]	[]	confirmed	1/11/2018 0:00
2222	282294	2018-08-27	389	0.0	...	[]	[]	confirmed	9/6/2018 0:00
1173	275924	2017-08-06	381	376.0	...	[doughnuts]	[multiple]	confirmed	8/16/2017 0:00
1850	279828	2018-04-02	377	377.0	...	[burrito, unspecified, chicken, wrap, turkey, ...	[multiple]	confirmed	5/4/2018 0:00
37	267485	2016-02-14	375	75.0	...	[]	[]	confirmed	2/18/2016 0:00
2806	287448	2017-11-20	358	302.0	...	[turkey]	[land, animals, meat, poultry, poultry, turkey]	confirmed	3/31/2019 0:00
3879	297111	2021-02-16	324	324.0	...	[chicken, beans, chili]	[multiple]	confirmed	2/17/2021 0:00
3663	296464	2019-06-10	303	287.0	...	[fresh, basil]	[plant, produce, vegetables, leafy, vine, stal...	confirmed	7/26/2019 0:00
2248	282392	2018-10-01	293	293.0	...	[]	[]	confirmed	11/10/2018 0:00
293	269345	2016-06-16	281	280.0	...	[scallops, raw]	[aquatic, animals, shell, fish, mollusks, biva...	confirmed	10/9/2016 0:00
2339	282824	2018-12-05	280	280.0	...	[turkey]	[land, animals, meat, poultry, poultry, turkey]	confirmed	12/7/2018 0:00
3741	296676	2020-01-11	280	0.0	...	[coleslaw]	[multiple]	confirmed	NaN
2722	286925	2018-05-14	267	267.0	...	[vegetable, tray]	[multiple]	confirmed	6/20/2018 0:00
1862	279867	2018-01-08	265	262.0	...	[chicken, salad]	[multiple]	confirmed	3/16/2018 0:00
2282	282535	2018-04-06	258	175.0	...	[]	[]	confirmed	8/19/2018 0:00
2183	282186	2018-09-06	251	245.0	...	[]	[]	confirmed	10/1/2018 0:00
298	269372	2016-07-11	251	0.0	...	[]	[]	confirmed	7/13/2016 0:00
1533	278073	2017-10-17	244	244.0	...	[]	[]	confirmed	11/19/2017 0:00
2150	282045	2018-03-13	239	226.0	...	[romaine, lettuce, unspecified]	[plant, produce, vegetables, leafy, vine, stal...	confirmed	8/22/2018 0:00

Figure 7.1: Top 28 reports from NORS data by count of estimated primary cases.

Twitter Tokens (with IWP)

Food	Count
chicken	6177
sandwich	2032
salad	1996
cheese	1873
pizza	1757
fries	1366
burger	1245
food	1195
burrito	816
shrimp	757
beef	725
bowl	723
steak	716
rice	715
bacon	688
meat	671
cream	612
spicy	611
wings	610
tacos	599
coffee	599
nuggets	598
soup	532
fried	525
meal	509
lettuce	490
fish	448
sushi	431
sauce	428
hot	422

NORS Food Names

Food	Count
chicken	219
unspecified	161
raw	148
oysters	147
salad	125
fish	124
beef	95
tuna	64
rice	58
pork	57
and	55
sandwich	49
turkey	47
milk	43
cheese	36
ground	34
lettuce	33
beans	33
fried	28
pizza	26
steak	25
sushi	24
pulled	24
cilantro	24
taco	23
liver	22
ice	21
mahi	20
shrimp	20
eggs	20

Figure 7.2: Top 30 most frequent food terms for Twitter and NORS data (2017-2021).

Twitter Tokens (no IWP)

Food	Count
food	799
chicken	717
pizza	414
coffee	380
cheese	326
milk	244
sandwich	231
sushi	216
salad	211
meat	209
puke	172
dog	143
hot	141
burger	138
soup	128
cream	125
tacos	121
chocolate	118
burrito	117
eggs	116
taco	113
chinese	112
raw	111
fries	107
fish	103
lunch	100
sauce	98
bacon	97
rice	96
seafood	90

NORS Food Names

food	count
chicken	219
unspecifiec	161
raw	148
oysters	147
salad	125
fish	124
beef	95
tuna	64
rice	58
pork	57
and	55
sandwich	49
turkey	47
milk	43
cheese	36
ground	34
lettuce	33
beans	33
fried	28
pizza	26
steak	25
sushi	24
pulled	24
cilantro	24
taco	23
liver	22
ice	21
mahi	20
shrimp	20
eggs	20

Figure 7.3: Top 30 most frequent food terms for Twitter (excluding iWP) and NORS data (2017-2021).

Resource 7.4: Copy of data_collection read_me

Description

This directory includes all scripts for data collection and prediction. You need to put the model "pytorch_model.bin" into this directory. The path on the server is /home/cgnoreika/pytorch_model.bin.

Data collection

main_collection.py is for data collection only . It will save the collected tweets into *.pt files in the geo_tweets folder.

Run code below to start collecting. You need to change the start_time_iso and end_time_iso.

python main_collection.py

It will save the log info into logs/datacollection.log

Tweet Prediction

main_predict.py is for prediction only. It will read all the *.pt files in the geo_tweets folder and do the prediction.

❗**To do: Connect to your database and save the prediction into tables**

python main_predict.py

It will save the log info into logs/Predcition.log

Whole pipeline (Don't touch this until the database part is ready)

Run the code below for data collection and prediction parallelly.

python main_multiprocess.py

It will save the log info into logs/whole_pipeline.log

Send email

It will send us emails when the data collection/prediction is done or it occurs some errors. You can change the receivers by editing the variable to_address in the main_.py files.

Tasks

I will start the data collection for tweets with geo on the server. All the collected tweets will be saved into folder home/rhu/geo_tweets

Complete the database part in 'main_prediction.py'

Run 'python main_prediction.py' on the server. (For now, it will take the files in geo_tweets_test as input)

Resource 7.5: GitHub organization link

https://github.com/wpi-foodborne-illness-monitoring

Resource 7.6: GitHub repository for backend code

https://github.com/wpi-foodborne-illness-monitoring/backend/blob/main/README.md

Resource 7.7: GitHub repository for frontend code

https://github.com/wpi-foodborne-illness-monitoring/frontend/blob/main/README.md

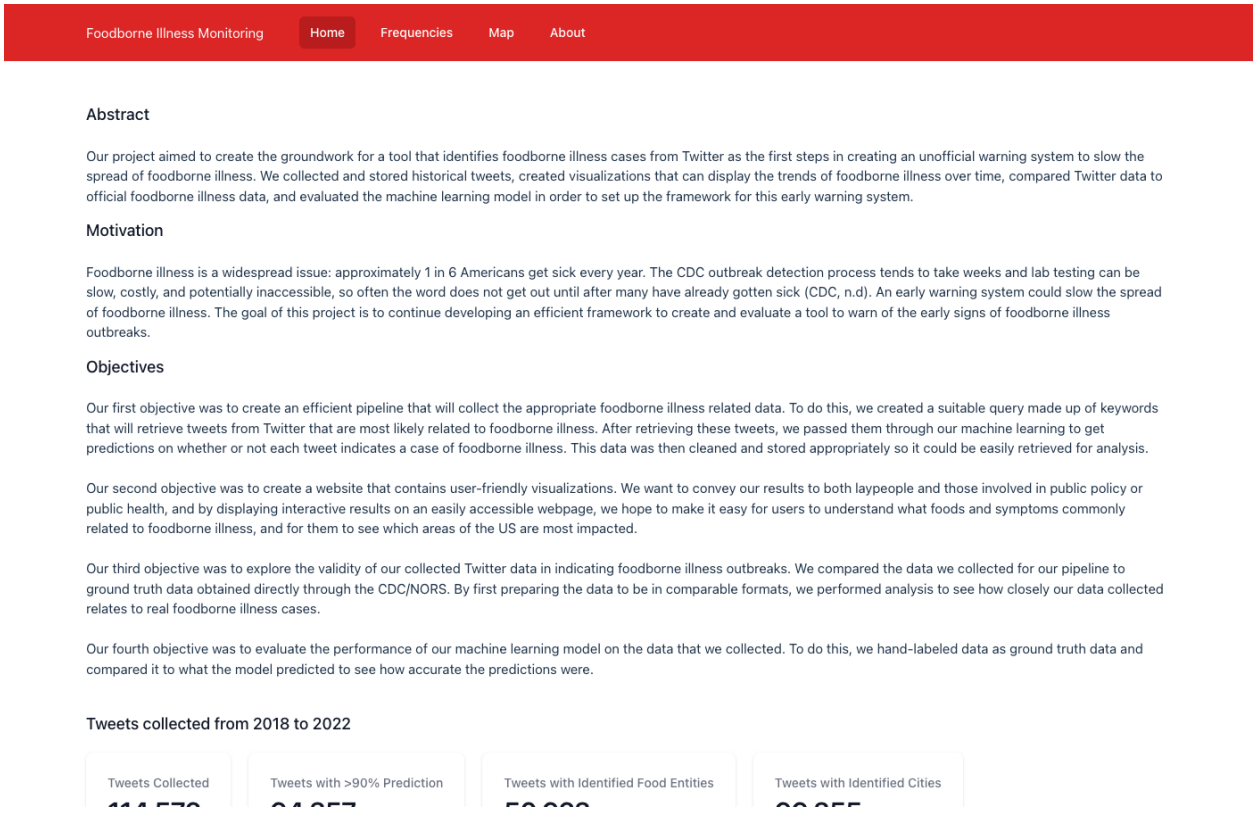


Figure 7.8: Screenshot of the homepage of the website

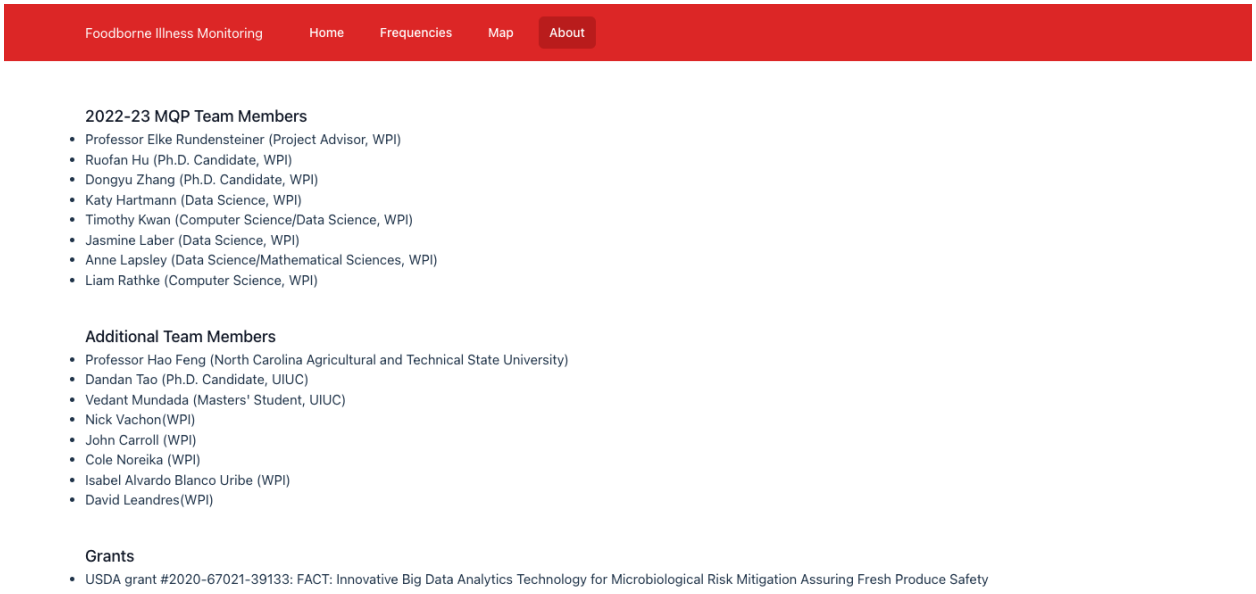


Figure 7.9: Screenshot of the about section of the webpage

Resource 7.10: GitHub repository for frontend code

https://usda-foodpoisoning.wpi.edu/

Resource 7.11: Forecasting model (i corresponds to the lag terms)

```
for i in range(22):
    series = read_csv('/Users/annelapsley/Desktop/tweets_by_day.csv', header=0, parse_dates=[0], index_col=0, squeeze=True, date_parser=parser)

    print(series.head())
    series.plot()
    plt.show()
    autocorrelation_plot(series)
    series.index = series.index.to_period('M')
    # fit model
    model = ARIMA(series, order=(i,1,0))
    model_fit = model.fit()
    residuals = df(model_fit.resid)
    residuals.plot()
    plt.show()
    residuals.plot(kind='kde')
    X = series.values
    size = int(len(X) * 0.66)
    train, test = X[0:size], X[size:len(X)]
    history = [x for x in train]
    predictions = list()
    # walk-forward validation
    for t in range(len(test)):
        model = ARIMA(history, order=(i,1,0))
        model_fit = model.fit()
        output = model_fit.forecast()
        yhat = output[0]
        predictions.append(yhat)
        obs = test[t]
        history.append(obs)
        '''
        print('predicted=%f, expected=%f' % (yhat, obs))
        '''
    # evaluate forecasts
    rmse = sqrt(mean_squared_error(test, predictions))
    rmse_list.append(rmse)
    print('Test RMSE', i ,'day lag: %.3f' % rmse)
```

(*What Is ARIMA Modeling?*, n.d.)

Resource 7.12: Creating the training and validation sets of X for regression

```
def make_X(num_of_tweets, num_of_nors):
    X = np.empty(0)
    for i in range(0, 200):
        X_temp = tweets_array[7*i:7*i+num_of_tweets]
        temp = nors_array[i:i+num_of_nors]
        X_temp = np.append(X_temp, temp)
        if i==0:
            X = np.array(X_temp)
        else:
            X = np.append(X, X_temp, axis = 0)

    X_validate = np.empty(0)
    for i in range(200, 250):
        X_temp = tweets_array[7*i:7*i+num_of_tweets]
        temp = nors_array[i:i+num_of_nors]
        X_temp = np.append(X_temp, temp)
        if i==0:
            X_validate = np.array(X_temp)
        else:
            X_validate = np.append(X_validate, X_temp, axis = 0)

    X = np.reshape(X, [200, num_of_tweets+num_of_nors])
    X_validate = np.reshape(X_validate, [50, num_of_tweets+num_of_nors])
    return X, X_validate
```

Resource 7.13: Evaluate Linear Regression model

```
def evaluate_linear(num_of_tweets, num_of_nors):
    weeks = max(math.ceil(num_of_tweets/7), num_of_nors)
    X_train, X_validate = make_X(num_of_tweets, num_of_nors)
    y = df['NORS Cases'][(0+weeks):(200+weeks)]
    y = np.array(y)
    y = np.reshape(y, [len(y), 1])
    y_validate = df['NORS Cases'][(200+weeks):(250+weeks)]
    y_validate = np.array(y_validate)
    y_validate = np.reshape(y_validate, [len(y_validate), 1])
    reg = LinearRegression().fit(X_train, y)
    y_hat = np.dot(X_train, reg.coef_.T)
    print("xtest", np.shape(X_train))
    print("coef", np.shape(reg.coef_.T))
    rmse = math.sqrt(np.sum((y_hat-y)**2)/200)
    print(reg.coef_.T)
    y_hat_validate = np.dot(X_validate, reg.coef_.T)
    rmse_validate = math.sqrt(np.sum((y_hat_validate-y_validate)**2)/50)
    adjusted_r = 1 - ( 1-reg.score(X_train, y) ) * ( len(y) - 1 ) / ( len(y) - X_train.shape[1] - 1 )
    X2 = sm.add_constant(X_train)
    est = sm.OLS(y, X2)
    est2 = est.fit()
    print(est2.summary())
    return reg.score(X_train, y), rmse, adjusted_r, rmse_validate
```

Resource 7.14: Evaluate Logistic Regression model

```
def evaluate_logistic(num_of_tweets, num_of_nors):
    weeks = max(math.ceil(num_of_tweets/7), num_of_nors)
    X_train, X_validate = make_X(num_of_tweets, num_of_nors)
    y = df['NORS Cases'][(0+weeks):(200+weeks)]
    y = np.array(y)
    y = np.reshape(y, [len(y), 1])
    y_validate = df['NORS Cases'][(200+weeks):(250+weeks)]
    y_validate = np.array(y_validate)
    y_validate = np.reshape(y_validate, [len(y_validate), 1])
    y_bin = np.empty([200,1])
    for i in range(0, len(y)):
        if y[i]>200:
            y_bin[i] = 1
        else:
            y_bin[i] = 0
    y_bin_validate = np.empty([50,1])
    for i in range(0, len(y_validate)):
        if y_validate[i]>200:
            y_bin_validate[i] = 1
        else:
            y_bin_validate[i] = 0
    reg = LogisticRegression(penalty = "l2")
    reg.fit(X_train, y_bin)
    #print("coefficents", reg.coef_)
    #print("R-squared", reg.score(X_test, y))
    prediction = reg.predict(X_train)
    pc = sklearn.metrics.accuracy_score(y_bin, prediction)
    prediction_validate = reg.predict(X_validate)
    pc_validate = sklearn.metrics.accuracy_score(y_bin_validate, prediction_validate)
    #rmse = math.sqrt(np.sum((y_hat-y)**2)/200)
    return sklearn.metrics.r2_score(y, prediction), pc, pc_validate
```

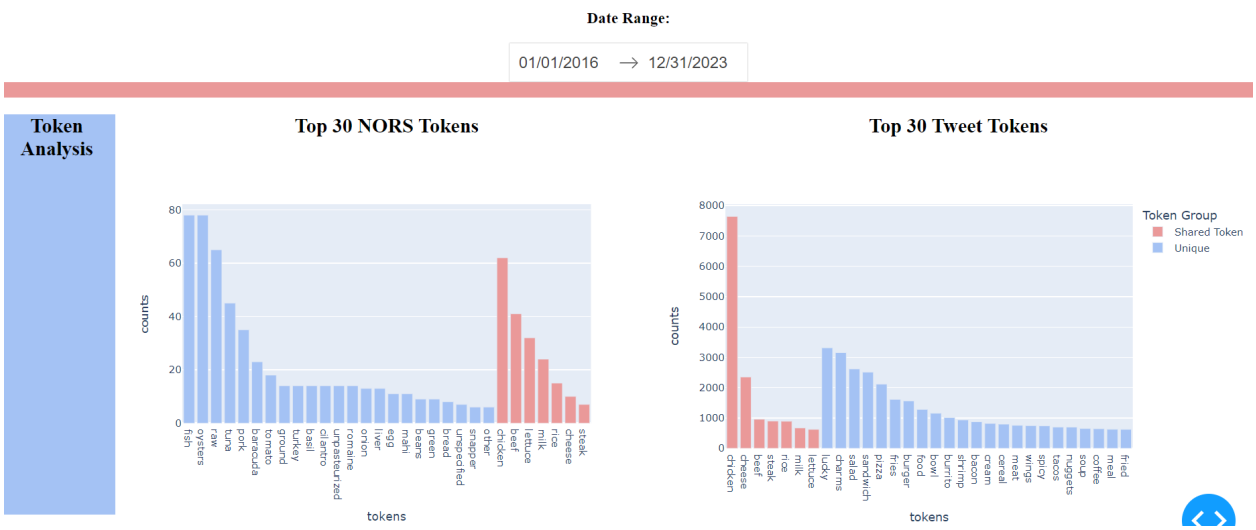


Figure 7.15: Screenshot of dashboard showcasing date range selector and interactive graph showing top 30 food tokens from both datasets

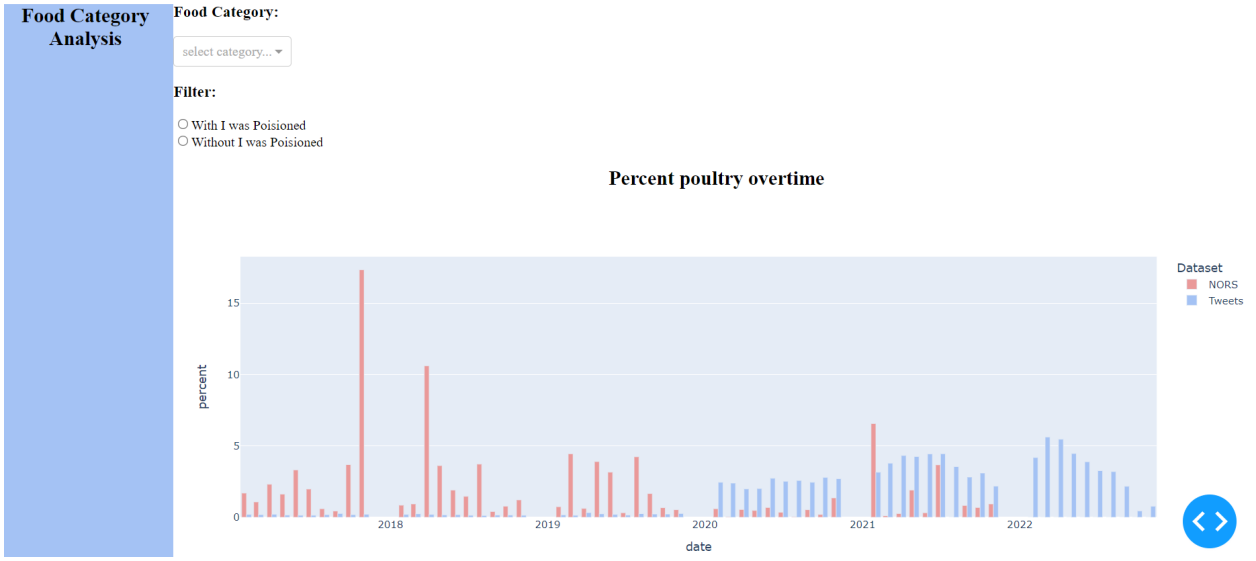


Figure 7.16: Screenshot of dashboard showcasing interactive filtering and food selection to view volume of cases overtime between the two datasets