Harvard University Extension School
"Principles of Big Data Processing"
CSCI E-88, Fall 2022
Final Project

by Ruchi Asthana, Kemi Olateju

Project Goal and Problem Statement

Explain the problem you are going to solve or the use case you will be implementing in your project

Elon Musk is a South African-born American entrepreneur who co-founded large companies like PayPal and SpaceX, and is currently the CEO of Tesla and Twitter. His recent purchase of Twitter has been especially controversial, affecting not just Musk's reputation but also the global economy.

Among the many Elon Musk controversies associated with the Twitter acquisition is the massive layoff that started in mid-November, resulting in half of its employees losing their jobs.
Based on media outlets, it seems like working conditions for the remaining employees are also bleak as many have to work 80 hours a week and come in and have to work from the office. Our team is interested to see if the sentiment on Twitter matches the negative sentiment towards him in the media and press. We will specifically dive deep into the positive and negative sentiments associated with #elonmusk from the past week.

We hypothesize that the overall sentiment of tweets will be generally negative, as there is still a lot of frustration associated with the recent Twitter layoffs.

We will leverage different Big Data Processing frameworks to extract Tweets that reference Elon Musk and analyze the sentiment associated with them. We will then create visualizations in an attempt to understand trends over the last week to answer our proposed hypothesis.

YouTube Video URL

https://youtu.be/CdzAkxTHQlo

GitHub Link

https://github.com/ruchiasthana/CSCI-E-88-Final-Project

Big Data Source

Specify the data source for your project - it should be either a big public data source set or self-generated datasets if needed. Describe the data, size, schema and any other details that are needed to understand the data.

Our project leveraged the Twitter API v2, to fetch recent Tweets (reference). To access the Twitter v2 APIs we had to create a Twitter account and sign up for Standard credentials through OAuth. Once we had OAuth credentials, we could call a select number of Twitter v2 APIs using a REST interface. Initially, we sought to search for **#elonmusk** across all Tweets in November. However, we quickly learned that one limitation of the standard credentials was that we could not hit the **`search_all_tweets`** API. We then pivoted to search for **#elonmusk** across all **recent** Tweets using the **`search_recent_tweets`** API. Below is the schemas for the input and output of this API for a more recent date range:

Input Schema

```
{
  query: "#elonmusk",
  start_time: "2022-12-06T00:00:00.000Z",
  end_time: "2022-12-10T00:00:00.000Z",
  max_resuts: 2

}
```

Output Schema


```
{
  "data": [
  {
    "edit_history_tweet_ids": [
      "1601366063827935232"
    ],
    "id": "1601366063827935232",
    "text": XXX
  {
    "edit_history_tweet_ids": [
      "1601366005074505728"
    ],
    "id": "1601366005074505728",
    "text": XXX
  },
],
    "meta": {
  "newest_id": "1601366063827935232",
      "oldest_id": "1601365730116915201",
      "result_count": 10,
      "next_token": "b26v89c19zqg8o3fqk0zzqsq5ufbd2p74vxhz94io8iv1"
}
}
```

As you can see above, input and output data are both in JSON format. We made this call for 2 types of batches: batch_1: 2,000 recent tweets and batch_2: 20,000 tweets. While working with these batch sizes, we encountered a second limitation with the Twitter v2 APIs where we hit a limit on the number of calls we can make with the **`get_recent_tweets`** API. While we were able to make one API call to fetch 20,000 Tweets, for the purpose of this demo, we will go with a smaller Tweet size of 2,000 recent Tweets.

Expected results

Describe the expected results/end state of the project

Expectations for the Pipeline
With regards to the pipeline flow for this project, we expect to:

1. Consume Twitter data from the Twitter API
2. Extract relevant Information about the Tweet
3. Pass the Tweet through a sentiment analysis software
4. Visualize Tweet sentiments over a specific timeframe (eg., the month of December)

Expectations for the End State
We expect the end state of our project to capture a meaningful distribution of POSITIVE, NEGATIVE, and MIXED sentiment for recent re-tweets with #elonmusk in the past week. We will heavily focus on visualizations with Kibana and use these visualizations to answer our hypothesis: ***the overall sentiment of tweets will be generally negative, as there is still a lot of frustration associated with the recent Twitter layoffs.***

Expectations for the Results
Considering the media coverage of Elon Musk is still so negative, we expect the majority of Tweets to have negative Sentiment about #elonmusk.

Processing Pipeline

Explain the design of your pipeline and the reasons for your choices of technologies for each tier. Describe all tiers that will be used and their configuration if applicable. Provide more details about the new technology[s] that you used.

Data Source
We used the GET /2/tweets/search/recent API to pull up to 20,000 Tweets about #elonmusk. We leveraged this API because it was one of a limited number of APIs available for standard OAuth credentials. We also chose this dataset because the data from this API will tell us if the majority of recent Tweets have a POSITIVE or NEGATIVE sentiment.

Distributed Deployment
Docker containers were leveraged for seamless and consistent deployment of **Apache Kafka clusters** between different users and devices.

Collection & Message Tier
We chose to use **Apache Kafka** as our stream processing platform because of its low latency and high throughput capabilities. A **Kafka producer** was used to generate messages from our data stream source (Twitter APIs), and we published them using a **Kafka consumer** to **Amazon s3.**

Master storage tier
We chose **Amazon s3** for our storage layer primarily for its high data availability, scalability, and performance capabilities. It was also easy to use and accessible through the various other technologies deployed.

Sentiment Analysis
We leveraged **AWS Comprehend** for sentiment analysis of individual Tweets. We chose AWS Comprehend because it is a state-of-the-art natural language processing tool for finding insights in text. It especially excels in extracting key phrases, people, and brands, as well as understanding how positive or negative a text is. For each tweet passed into AWS Comprehend, we collected the overall sentiment (POSITIVE, NEGATIVE, NEUTRAL, or MIXED) along with the numerical positive sentiment score and negative sentiment score. Our analysis was primarily conducted on the categorical sentiment field; however, there are natural extensions of this work that could involve the numerical sentiment scores as well.

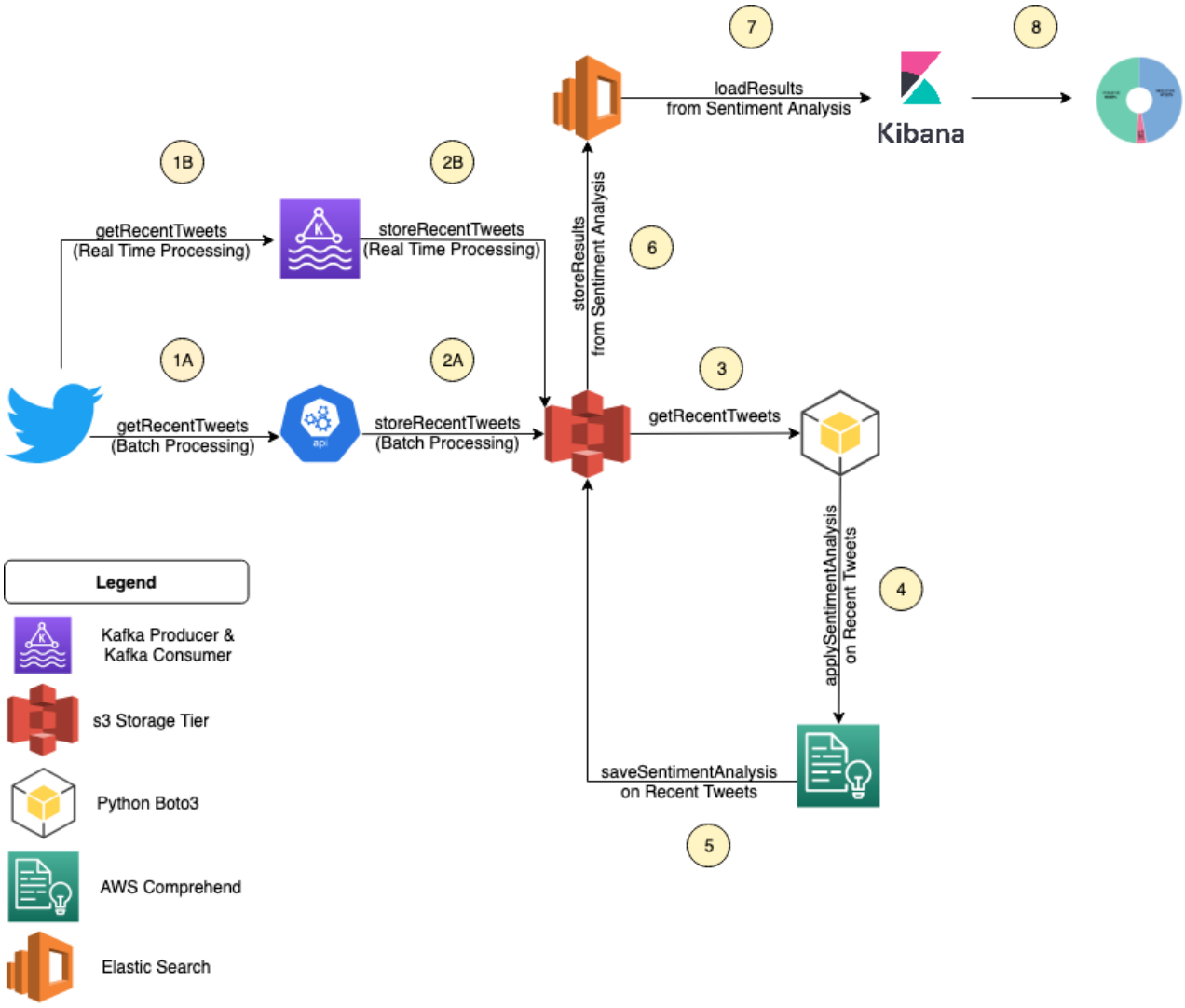
Distributed Indexing/Search
We leveraged the **Elasticsearch cloud** for a portion of our data analysis to process the results of the sentiment analytics and filter out the NEUTRAL sentiments. We chose Elasticsearch for its ability to handle JSON format files natively and full-text search engine capabilities at speed.

Visualization
We chose **Kibana** for our data visualization capabilities of its seamless integration with Elasticsearch. Within Kibana, we focused primarily on visualizations for categorical variables, including pie charts and bar graphs over a period of time.

We used **Github** for code sharing, version control, and collaboration.

Implementation

This is the main section. It should contain all implementation details for each tier. Copy all relevant configuration and code. The code should be well documented.



Collection and Messaging Tier (1A, 2A, 1B, 2B)

We were interested in ingesting data both through batches and in real-time. We connected both batch ingestion and real-time streaming to the Twitter v2 API for recent tweets and dumped data collected from both paths into an s3 container. Below we will outline each data collection tier in more detail.

Setup for Batch-Processing (1A, 2A)
Python file: get_twitter_data_dump.py

Python command: **python3 get_twitter_data_dump.py**
Python Code: We first check to see if recent_tweet_data already exists locally; if it does not, we pull the data from the Twitter v2 API. Once we have recent_tweet_data, we load it into s3.

```
"""
The objective of this file is to call the Twitter API and get a dump of Tweets over a given time period
"""
import os
import boto3
import tweepy
import pandas as pd


"""
Below is basic credentials for accessing the Twitter API:

API Key: F58ZzwAG8D2y582kHGwiTEXqi
API Key Secret: o5n0ASypriKMmWSdOrvkeU25dfKLTrBIw789jK7LQTEdIMkCkf
Bearer Token: AAAAAAAAAAAAAAAAAAAAAPh8jwEAAAAA5FLMueSNftI3RHF0rZPXSKdo0dc%3DEyX313BYU98GdqPWNG3grnxhhX2AkjGE2N8efR1RXZYT1H4WZq
"""

bearer_token = "AAAAAAAAAAAAAAAAAAAAAPh8jwEAAAAA5FLMueSNftI3RHF0rZPXSKdo0dc%3DEyX313BYU98GdqPWNG3grnxhhX2AkjGE2N8efR1RXZYT1H4WZq"
client = tweepy.Client(bearer_token)
s3 = boto3.resource('s3')

elon_musk_recent_tweets_bucket = 'elon-musk-recent-tweets'
path_to_elon_musk_recent_tweets = '/Users/rasthana/Desktop/CSCI-E-88-Final-Project/elon_musk_recent_tweets'
file_for_elon_musk_recent_tweets = path_to_elon_musk_recent_tweets + '/' + 'elon_musk_recent_2000_tweet_data.csv'
if os.path.exists(path_to_elon_musk_recent_tweets):
    print('A path to Elon Musk Recent Tweets already exists!')
    print('Here is its path: ', path_to_elon_musk_recent_tweets)
    if s3.Bucket(elon_musk_recent_tweets_bucket) in s3.buckets.all():
        print('Recent Tweet data already exists in s3')
    else:
        print('Recent Tweet data does NOT exist in s3')
        print('Uploading Recent Tweet data to an s3 bucket')
        s3.create_bucket(Bucket=elon_musk_recent_tweets_bucket)
        boto3.client('s3').upload_file(file_for_elon_musk_recent_tweets, elon_musk_recent_tweets_bucket, 'elon_musk_recent_2000_tweet_data.csv')
        print('Completed uploading data to the elon_musk_recent_tweets bucket')
```

```
else:
    print('A path to Elon Musk Recent Tweets does NOT exist!')
    print('We will pull tweet data and save it to s3')
    elon_musk_november_tweets_query = '#elonmusk -is:retweet lang:en'
    elon_musk_november_tweets = tweepy.Paginator(
        client.search_recent_tweets,
        query=elon_musk_november_tweets_query,
        tweet_fields=['context_annotations', 'created_at'],
        max_results=100).flatten(limit=2000)

    count = 0
    elon_musk_november_tweet_data = []
    for tweet in elon_musk_november_tweets:
        tweet_data = [tweet.created_at, tweet.text]
        elon_musk_november_tweet_data.append(tweet_data)
        count += 1

    print("We collected data from {} tweets".format(count))
    elon_musk_november_tweet_data_frame = pd.DataFrame(elon_musk_november_tweet_data, columns=['created_at', 'text'])
    print("Data Frame for Elon Musk's Recent Tweets", elon_musk_november_tweet_data_frame.head())
    elon_musk_november_tweet_data_frame.to_csv("elon_musk_recent_2000_tweet_data.csv")
    s3.create_bucket(Bucket=elon_musk_recent_tweets_bucket)
    boto3.resource('s3').upload_file(file_for_elon_musk_recent_tweets, elon_musk_recent_tweets_bucket, 'elon_musk_recent_2000_tweet_data.csv')
```

Environment Setup for Real-Time Processing with Kafka (1B, 2B)

We first created a **YML** file to store the configuration details of our Kafka environment

Docker Commands

To run the kafka yml in docker container

docker-compose -f docker-compose-finalkafka.yml up

To create our new topic. We chose to only use one broker because of the limited dataset used.

docker exec -it broker1 /bin/kafka-topics --bootstrap-server broker1:29092 --create --topic finalproject --replication-factor 1 --partitions 1

Commands to run Kafta on CLI (Showcased in our demo)

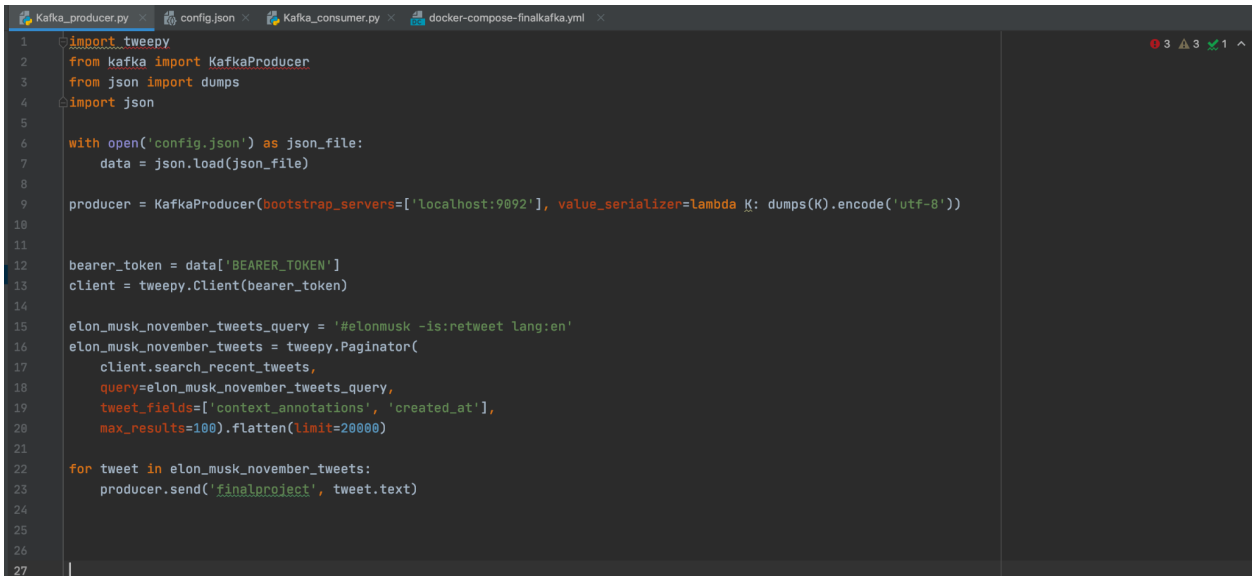
To start the Kafka Producer
Python3 kafka_producer.py

To use the Kafka console consumer
docker exec -it broker1 /bin/kafka-console-consumer --bootstrap-server broker1:29092

--topic finalproject

Kafka API - Python Code

Kafka producer



Kafka consumer

```python
from kafka import KafkaConsumer
import boto3
import json

with open('config.json') as json_file:
    data = json.load(json_file)

client= boto3.resource("s3", aws_access_key_id=data['AWS_ACCESS_KEY_ID'], aws_secret_access_key= data['AWS_SECRET_ACCESS_KEY'])
elon_musk_recent_tweets_with_sentiment_bucket = 'elon-musk-recent-tweets-with-sentiment-bucket-1'
client.create_bucket(Bucket=elon_musk_recent_tweets_with_sentiment_bucket)
consumer = KafkaConsumer('finalproject', bootstrap_servers=['localhost:9092'])

s3_file_name = "elon_musk_recent_20000_tweet_data_with_sentiment_analysis.csv"

for message in consumer:
    values = message.value.decode('utf-8')
    with open(s3_file_name, 'w') as f:
        print(message.value)
        f.write(f"{values}\n")

boto3.client('s3').upload_file(elon_musk_recent_with_sentiment_analysis, elon_musk_recent_tweets_with_sentiment_bucket,
                               'elon_musk_recent_20000_tweet_data_with_sentiment_analysis.csv')
```

Sentiment Analysis (3,4,5)

We leverage the AWS Comprehend for sentiment analysis from recent_tweets acquired through both batch processing and real-time processing. Our recent_tweet data is pulled from a **elon-musk-recent-tweets** s3 bucket using the Python Boto3 library. The data is then passed to the AWS Comprehend API to retrieve a sentiment, positive_sentiment_score, and negative_sentiment_score. These values are stored in a pandas Data Frame and then saved locally and in a **elon-musk-recent-tweets-with-sentiment-bucket** s3 bucket.

Python file: apply_sentiment_analysis_with_aws_comprehend.py
Python command: **python3** apply_sentiment_analysis_with_aws_comprehend.py
Python code:

```
"""
The goal of this document is to apply sentiment onlysis on tweet data
We will be leveraging AWS Comprehend for sentiment analysis
"""

import io
import os
import boto3
import pandas as pd


print("Searching for the Recent Tweet Data")
recent_tweets_file_path = './elon_musk_recent_tweets/elon_musk_recent_2000_tweet_data.csv'
if os.path.exists(recent_tweets_file_path):
    print("Recent Tweet Data is found in local computer at the following file path: {}".format(recent_tweets_file_path))
    elon_musk_recent_tweets = pd.read_csv(recent_tweets_file_path)
    print("Head of elon_musk_recent_tweets: ", elon_musk_recent_tweets.head())
else:
    print("Recent Tweet Data is NOT found on the local computer, fetching it from s3")
    #Fetch Data from the s3 bucekt and load it into a Pandas Data Frame Object
    s3_client = boto3.client('s3')
    bucket_name = 'elon-musk-recent-tweets-2'
    file_name = 'elon_musk_recent_2000_tweet_data.csv'
    obj = s3_client.get_object(Bucket=bucket_name, Key=file_name)
    elon_musk_recent_tweets = pd.read_csv(io.BytesIO(obj['Body'].read()), encoding='utf8')
```

```
#Initialize the AWS Comprehend Client for analysis
client = boto3.client('comprehend')

#We start by reading in a CSV file of elon musks most recent tweets
# elon_musk_recent_tweets = pd.read_csv('./elon_musk_recent_tweets/elon_musk_recent_20000_tweet_data.csv')

#Initialize fields that we will want to populate with our analysis
rows, cols = elon_musk_recent_tweets.shape
elon_musk_recent_tweets['sentiment'] = 'initial sentiment'
elon_musk_recent_tweets['positive_sentiment_score'] = 0.0
elon_musk_recent_tweets['negative_sentiment_score'] = 0.0
```

```
#Pass each tweet to the AWS Comprehend API for sentiment analysis
count = 0
for row_idx in range(rows):
    tweet_text = elon_musk_recent_tweets['text'][row_idx]
    sentiment_analysis = client.detect_sentiment(Text=tweet_text, LanguageCode='en')
    #Extract Relevant fields from AWS Comprehend Sentiment Analysis
    sentiment = sentiment_analysis['Sentiment']
    positive_sentiment_score = sentiment_analysis['SentimentScore']['Positive']
    negative_sentiment_score = sentiment_analysis['SentimentScore']['Negative']
    elon_musk_recent_tweets['sentiment'][row_idx] = sentiment
    elon_musk_recent_tweets['positive_sentiment_score'][row_idx] = positive_sentiment_score
    elon_musk_recent_tweets['negative_sentiment_score'][row_idx] = negative_sentiment_score
    print("Tweet Text Number {} has been passed through Sentiment Analysis".format(count))
    count += 1

print(elon_musk_recent_tweets.head())
elon_musk_recent_with_sentiment_analysis = 'elon_musk_recent_20000_tweet_data_with_sentiment_analysis.csv'
elon_musk_recent_tweets.to_csv(elon_musk_recent_with_sentiment_analysis)

#Upload Sentiment Analysis Data to s3
s3_client = boto3.client('s3')
elon_musk_recent_tweets_with_sentiment_bucket = 'elon-musk-recent-tweets-with-sentiment-bucket-1'
s3_client.create_bucket(Bucket=elon_musk_recent_tweets_with_sentiment_bucket)
boto3.client('s3').upload_file(elon_musk_recent_with_sentiment_analysis,
                               elon_musk_recent_tweets_with_sentiment_bucket,
                               'elon_musk_recent_2000_tweet_data_with_sentiment_analysis.csv')
```

ElasticSearch-Kibana Implementation (6,7,8)

We loaded the sentiments stored in Amazon s3 to the Elasticsearch cloud using the current free version of the tool. We had to do this manually by first downloading the sentiments from s3 to our local device and then manually uploading the sentiments to Elasticsearch. Of course, this is not scalable, so for a future iteration, we will look into utilizing a Lambda function to automate the data connection.

Command to copy sentiment file from s3 to local

aws s3 cp --recursive

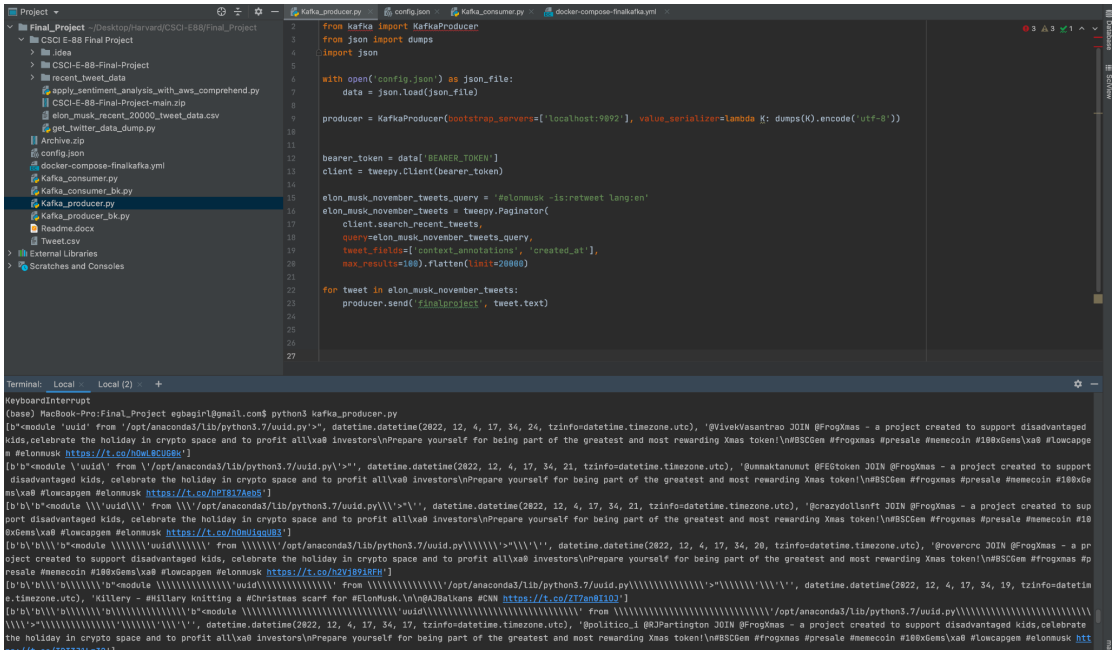
s3://elon-must-recent-tweets/elon_musk_recent_tweets_with_sentiment_analysis.csv/ .

ElasticSearch GUI navigation flow
Stack Management => index Management => Data views => Discover =>Dashboard =>Lens

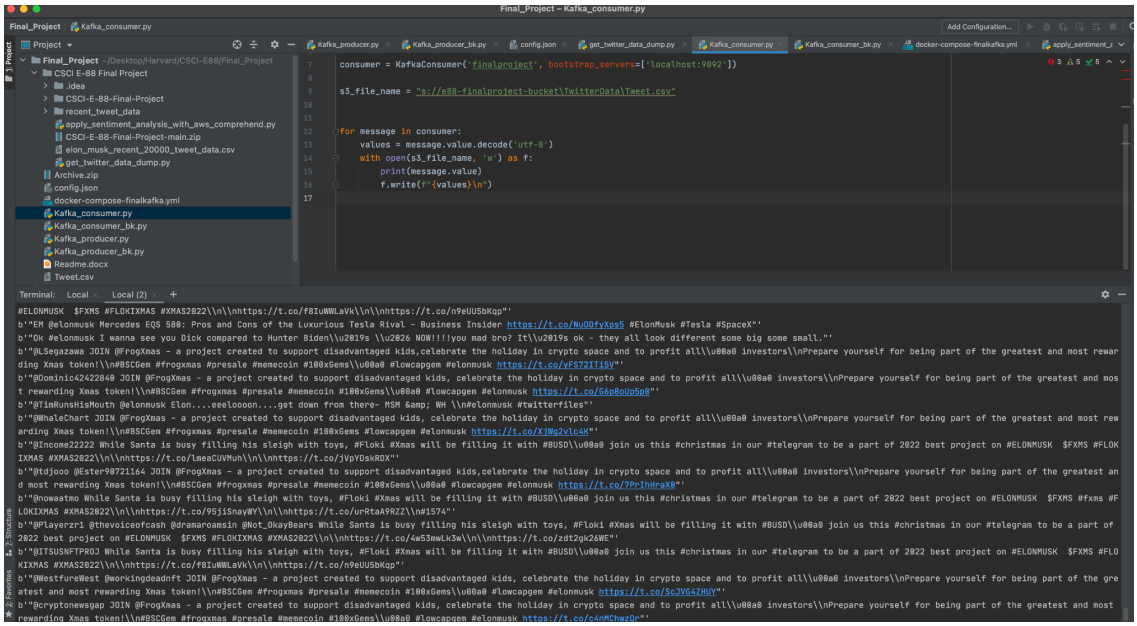
Results

Demonstrate your results. Provide all relevant screenshots of the log files and data stored in DB and GUI tier if applicable.

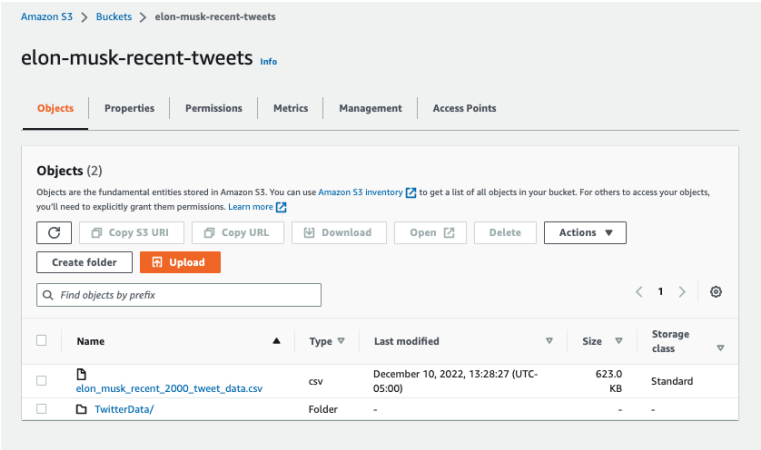
Screenshot of the Kafka Producer

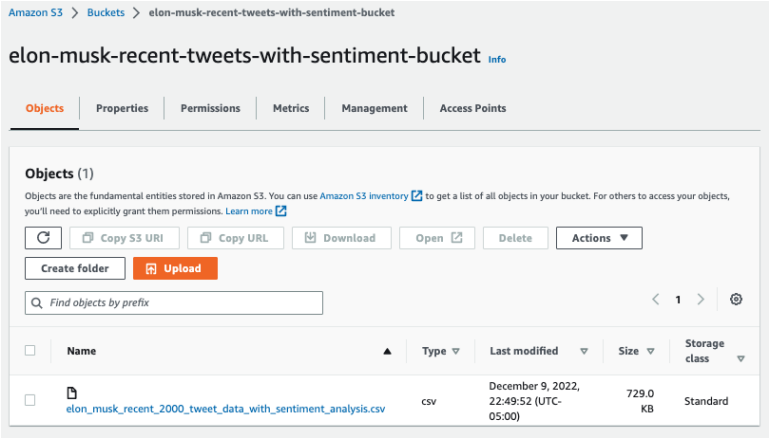


Screenshot of the Kafka Consumer



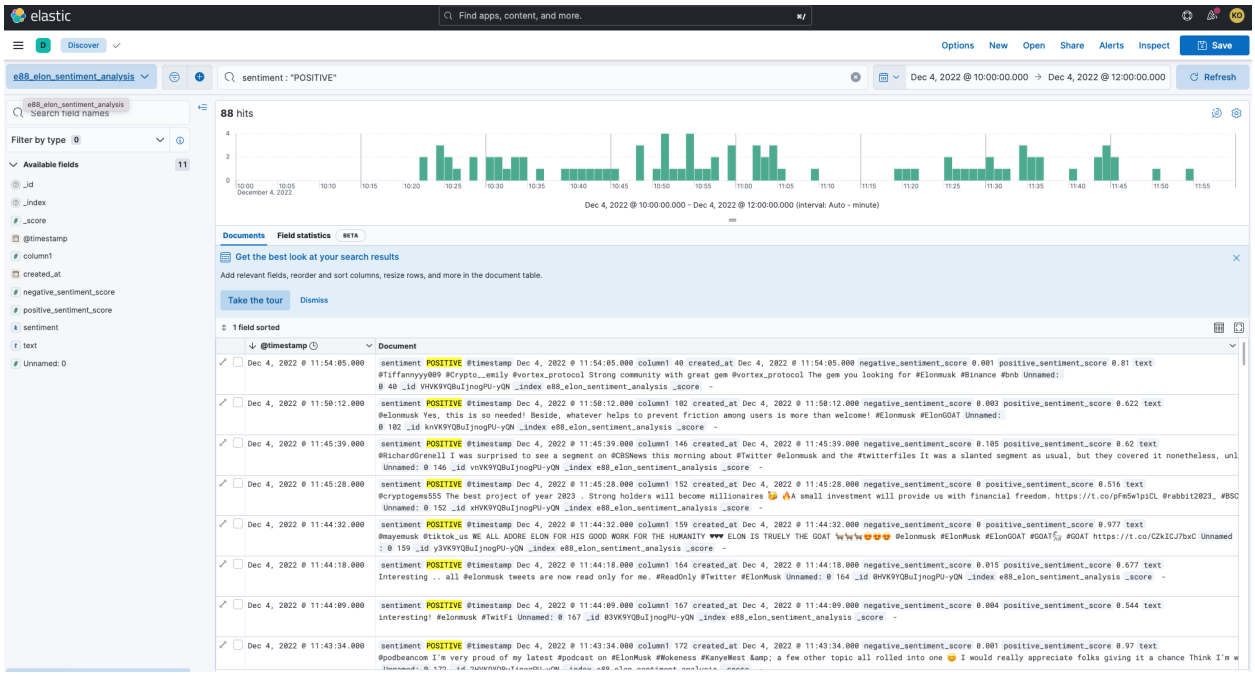
Screenshot of Amazon s3 Bucket for Real-Time and Batch Processing



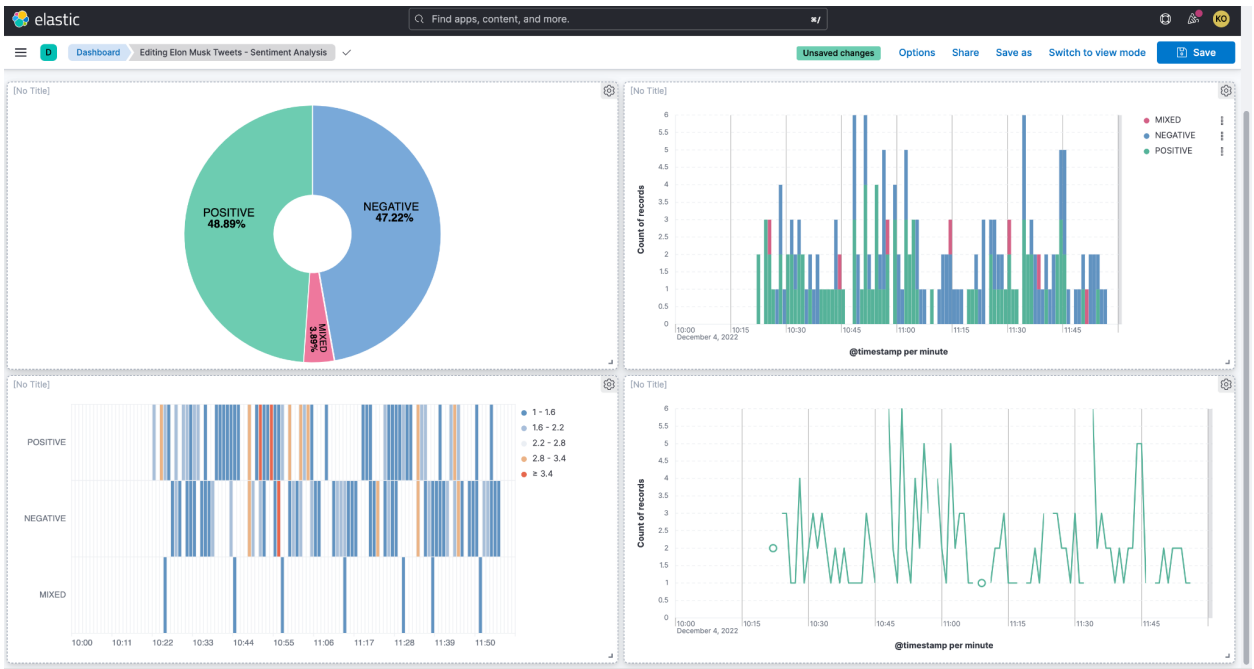


(more content on the next page)

Screenshot of the Tweets in ElasticSearch



Screenshot of the Kibana Visualization



Observations
We were able to isolate 2000 tweets related to #elonmusk between the hours of 10.22 am and 11.55 am eastern timezone. Of these tweets, only about 10% were labeled by our sentiment analyzer as either NEGATIVE, POSITIVE or MIXED - with the remaining tweets having a NEUTRAL sentiment. Overall for this snapshot in time, we observed that the sentiment towards Elon Musk was quite balanced. **These results contradict our hypothesis that *the overall sentiment of tweets will be generally negative, as there is still a lot of frustration associated with the recent Twitter layoffs.***

Conclusions and Lessons Learned

Describe what you have learned during this project:

- ***What issues did you have?***
- ***What limitations, if any, did you run into with the technologies used?***
- ***What would you do differently next time?***
- ***What alternatives to technologies you use might you consider?***
- ***Where would you take your project next?***

What issues did you have?
We faced a lot of issues getting the right data from the Twitter API. It seems like access tiered credentials must have recently changed, and the documentation is not updated to reflect that. APIs that used to have basic credential requirements but now require additional access credentials. We also experienced issues with the search_all_tweets API and, as a result, had to end up using the search_recent_tweets API.

What limitations, if any, did you run into with the technologies used?
It was interesting to see how cost spiked for AWS Comprehend when we scaled the sentiment analysis script from analyzing 2,000 tweets to 20,000 tweets. We stopped this scaled analysis after one round when we got a cost notification from AWS saying that we exceeded the threshold for a Basic tier and would need to pay more for a preferred tier to process batches of 20,000 tweets at a time.

What would you do differently next time?
We will be more frugal with resources next time. Instead of running the same script on the same date range and overwriting results, we would add checks earlier to see if the data had already been processed and stored. This would help conserve our budget, especially in the testing and debugging phases of this project.

What alternatives to technologies you used might you consider?
One gap in our project is the verification that tweets retired from the Twitter v2 API actually contain meaningful information about Elon Musk. To do this, we can leverage NLU technologies to extract keywords and phrases to ensure they actually have strong semantic references to Elon Musk. We could also work with more language preprocessing techniques to decrease the text so that it contains less noise and only relevant information. If we wanted to continue

expanding the size of our analysis to 200 thousand or 2 million tweets, we might also want to consider using Elastic Container Service (ECS).

Where would you take your project next?

There are multiple avenues where we could take this project:

- Generalizing the code so that we can apply this time series sentiment analysis for any hashtag, not just #elonmusk
- Adding post-processing to the actual tweets before passing them through sentiment analysis so we can get rid of the noise that might steer a tweet toward being labeled as neutral sentiment and isolate tweets directly related to the named entity
- Since we can only use the get_recent_tweets API, we would set up an automatic system to run every week so we could collect data over a larger time range and include more time series analysis in our portfolio