Risk Prediction Sentiment Analysis 3

Authors: Sagar Mehta, Kaustubh Kale, Jaswant Kasinedi, Parvesh Samayamanthula, Arjun Vashistha

Instructor: Edward Fox

Client: Christian Johnson

CS 4624: Multimedia, Hypertext, and Information Access
Department of Computer Science
Virginia Tech, Blacksburg VA 24061
May 10, 2022

Contents

List of Figures

List of Tables

1. Executive Summary / Abstract

Workplace safety has become a pressing issue in today's society. People are concerned about what kind of environments they are working in and whether they will incur some kind of injury while working. Our client, Christian Johnson, and we have worked on a web app that intends to tackle this problem head on. Christian Johnson, founder of riskinsights.ai, proposed that by analyzing the sentiment of worker narratives in a certain workplace, we can figure out the relative risk of that workplace. In order to do this, we were tasked to continue building an intuitive web app from last year that allows users to upload worker narratives in the form of .csv files so that a machine learning NLP model predicts the sentiment of these narratives as essentially either negative, neutral, or positive.

One of the major challenges in building this web application is accurately predicting sentiment. This is a crucial problem as well since without this accurate sentiment analysis, our system will be useless to users in the workplace. Though this is one of the critical problems and research areas in the field of natural language processing and machine learning (ML), it was not our intention to fully solve it. However, we did want to use/build the best possible model for our budget. We decided to go with a cloud ML solution because it would be the easiest to scale and integrate with our application. After trying different cloud services, such as Amazon AWS and Microsoft Azure, we were able to come to the conclusion that the Google Cloud AutoML service would fit our needs the best. So after labeling 1500+ worker narratives, and building and tweaking the Google Cloud model, we were able to bring the accuracy of our sentiment analysis up to 84.6%, which is a huge improvement from last year's 65%. We were also able to meet the client's requirements of adding new frontend functionalities, including creating an instructions page (Figure 7) and implementing authentication to the web app. Therefore, we were able to significantly improve the overall functionality of the application.

2 Introduction

2.1 Problem

A major problem our client identified is the advent of hazards in the workplace. This is a crucial problem because a lot of these hazards could be life or death situations. In a national safety month survey from 2019, 39% of respondents are more concerned with on-the-job safety this year than they were in 2018. Additionally, in the same report, 25% of the workers that responded said that on a daily basis, they worry about getting some type of injury. Additionally, workers in construction, oil, and gas are getting more and more likely to fear for their safety. In fact, the top workplace safety concerns this year, in order of biggest concern to smallest, are slipping or falling, electrical hazards, ergonomic problems, vehicle and equipment strikes, and falling objects. However, the actual major concern of a worker differs per industry. The biggest concern is electric hazards for construction, ergonomic problems for manufacturing, gas or chemical exposure for oil and gas, vehicle and equipment strikes for transportation, and electrical hazards for utilities. Investment in workplace training is something these workers want in order to reduce the risks of hazards involved with working in these industries [1].

Additionally, men and women have a difference in thoughts on workplace hazards. One major difference is that women are 6 times more likely to identify workplace violence as their number one safety concern and twice as likely to list ergonomic issues as their number one safety concern when compared to men. On the other hand, men are three times more likely to identify falling objects as their number one safety concern and twice as likely to identify electrical hazards as their primary concern when compared to women. These concerns make sense because women are more likely to be a victim of homicide in the workplace while men are more likely to die based on hazards that include falls, slips, trips, or contact with dangerous objects/equipment [1].

2.2 Client and His Vision

Our client is Christian Johnson, CIH, CSP. He is an industrial and systems engineering graduate of Virginia Tech and has a variety of professional experiences. He is currently the Director of Safety, Health & Environment at AstraZeneca and the founder of RiskInsights.ai [12].

RiskInsights.ai is the company we are working for on this project. Christian had an idea when the COVID-19 quarantines started, about using artificial intelligence and machine learning in the workplace to help mitigate hazards and improve safety. This initial motivation led to the development of RiskInsights.ai and the project we are working on. This project has been iterated through by two teams previously as part of the Virginia Tech CS4624 course; we are the third team to work on this project [11].

The vision for this project comes from a very simple thesis. Basically, sentiment of narratives from the workplace, such as what workers have observed and said, can be used to determine the relative risk of a workplace. By actually analyzing the sentiment of what these workers are saying, we can determine the risk of the hazards previously mentioned actually happening. Sentiment analysis is a subsection of artificial intelligence that is gaining popularity. According to *Oxford Languages*, sentiment analysis is "the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral" [2]. Essentially, our hypothesis is that the more negative the sentiment of a worker's narrative, the higher the risk of a workplace hazard happening and the overall more dangerous that place is. On the other hand, the more positive the sentiment of a worker's response, the lower the risk of a workplace hazard happening and the overall less dangerous that place is. With this information, managers of workplaces can figure out if they are a risky place and can be proactive in reducing injuries and damage, instead of just being reactive.

2.3 Background/Work Done

As mentioned previously, this is the third iteration of this project. As a result, we had a terrific launching pad to start off from. We had a pretty good and functional website to build upon [Figure 1], and a decently large dataset of worker responses to work from in building a good sentiment analysis system.

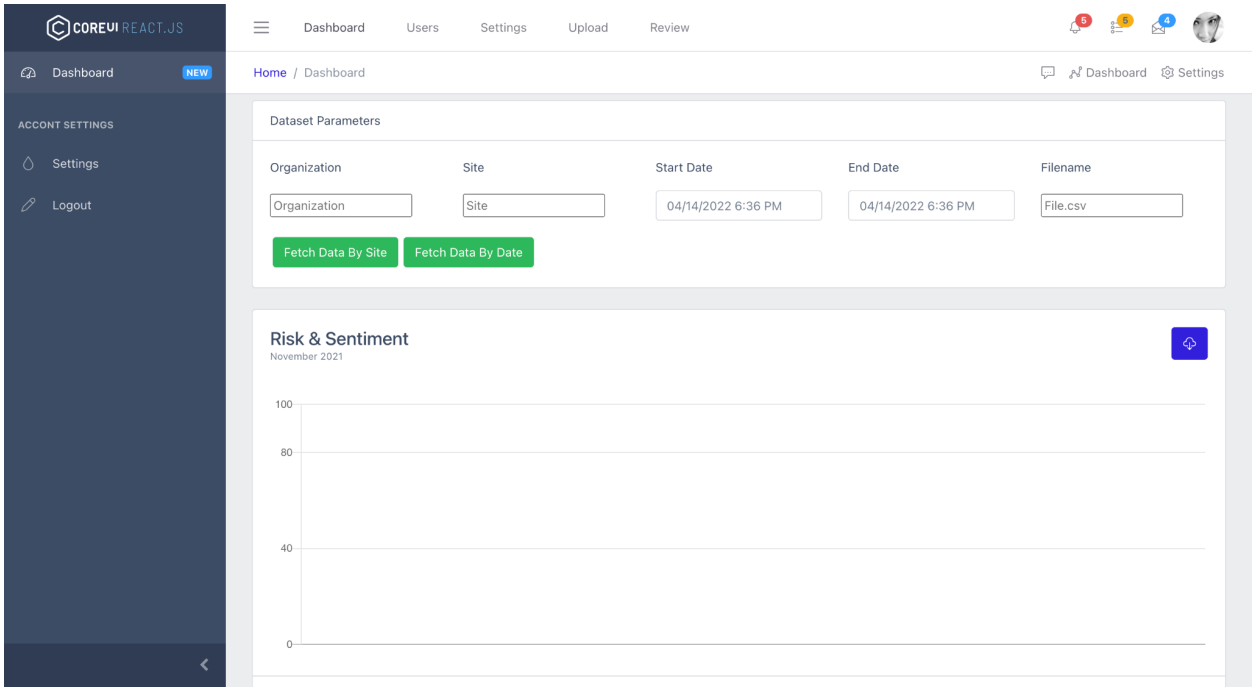

Figure 1: Home page of the project - adapted from [11]

However, not all the functionalities shown in Figure 1 were completely implemented. For example, the logout feature had a button on this site, but no form of user accounts and authentication was implemented. Essentially, a user can upload a CSV file with worker data, and the site provides the sentiment of the input data and graphs it. More details of what the system has to be able to do will be given in Section 3: Requirements. However, the actual sentiment analysis the site provides is being done by a service called AWS Comprehend [13]. After doing a human review of the sentiment analysis AWS Comprehend provided, last year's team determined that it was only accurate on 65% of test data in determining whether a certain worker's response text was positive, negative, or neutral [11]. This was the number that was a major goal of ours to improve.

2.4 Organization of Report

Next, we describe the project requirements and how we designed our system. Then we go into how we assessed our system. Afterwards, we discuss how the user can interact with our website, and how a developer can interact with it. Finally, we include lessons learned, acknowledgements, and references.

3 Requirements

3.1 Improving Sentiment Accuracy

One of the major requirements for the backend was improving the accuracy of the sentiment analysis currently being used by the system. The Custom AWS Comprehend model previously used only had an accuracy of 65%. It was a major goal of ours to improve this percentage.

3.2 Authentication

One major frontend requirement of ours was to add authentication and accounts management into the website. We wanted only certain people to have access to the website in case someone with malicious intent goes on there and keeps using the cloud sentiment analysis service, accruing great costs. We wanted some sort of security, keeping people from doing this.

3.3 CI/CD

Another major requirement for us was to integrate some sort of CI/CD pipeline [4] into our workflow. The reason for this is because the process of getting our code up to production takes too long. First, we need to run our code locally to generate the build files for the application, and then upload the build files into an Amazon S3 bucket so it can host the application. This process is quite tedious so we want to speed it up by implementing some sort of pipeline that will automatically rebuild and host the application as soon as we push code to our GitHub repo.

3.4 Minor Frontend UI Improvements

A smaller requirement for the project was to clean up the general frontend of our application. The previous team used a Core React template to build the UI. This meant a lot of the elements on the website were unused or did not necessarily contribute to functionality for the end user. As we progressed on the frontend, our client would give us feedback on what UI components to keep or remove, or restyle/reconfigure.

4 Design/Implementation

4.1 Backend

4.1.1 Infrastructure

Our client provided us with a working version of the project that a group of students from the previous semester created. We found their backend infrastructure to be quite good, as it effectively leveraged multiple technologies to satisfy the end goal of providing sentimental analysis for worker responses. They did not, however, use serverless technologies, but we deemed it not the most effective use of time. AWS S3 was used to store CSV data with worker responses that would be analyzed in the future, while AWS DynamoDB [14] was used to efficiently store and access the sentiment analysis outputs. AWS EC2 was used for a server hosting the backend endpoints. The issue with the approach of the previous group was the sentiment analysis accuracy did not meet the client's requirements. Their model predicted sentiment accurately a modest 65% of the time which was our main area of improvement to the project. The issue with their approach was they used a built-in sentiment analysis model, AWS Comprehend [1], trained on general data geared towards regular day to day conversations. It is not optimal for predicting risk in the workplace. Therefore, we decided to use a sentiment analysis model provided by and trained using the Google Cloud Platform [6]. Google Cloud's AutoML service [5] allows training sentiment analysis models based on labeled data, so a model can effectively predict the sentiment of specialized text instead of everyday conversations. The NodeJs [15] endpoint that originally relied on the AWS Comprehend service was changed to use the custom model trained using Google Cloud AutoML. After parsing through the input CSV file and submitting each worker response description to the Google Cloud AutoML endpoint, we collect all the sentiment scores returned in an array. Afterwards, we append the sentiment scores and the overall sentiment the scores represent to the original CSV. We send this CSV back to the frontend, where it'll automatically be downloaded by the user. An example input CSV is given in Table 1 and the resulting output CSV is given in Table 2.

An additional requirement provided by the client was to support outputting sentiment given an audio format of a worker response. We used Google Cloud to provide audio transcription as it was found to be accurate when tested by different speakers. However, as the semester progressed, our client discarded the audio

transcription feature from the project goals as increasing sentiment analysis accuracy was of utmost importance. Furthermore, we decided not to implement continuous learning, so the model will be periodically re-trained using new data.

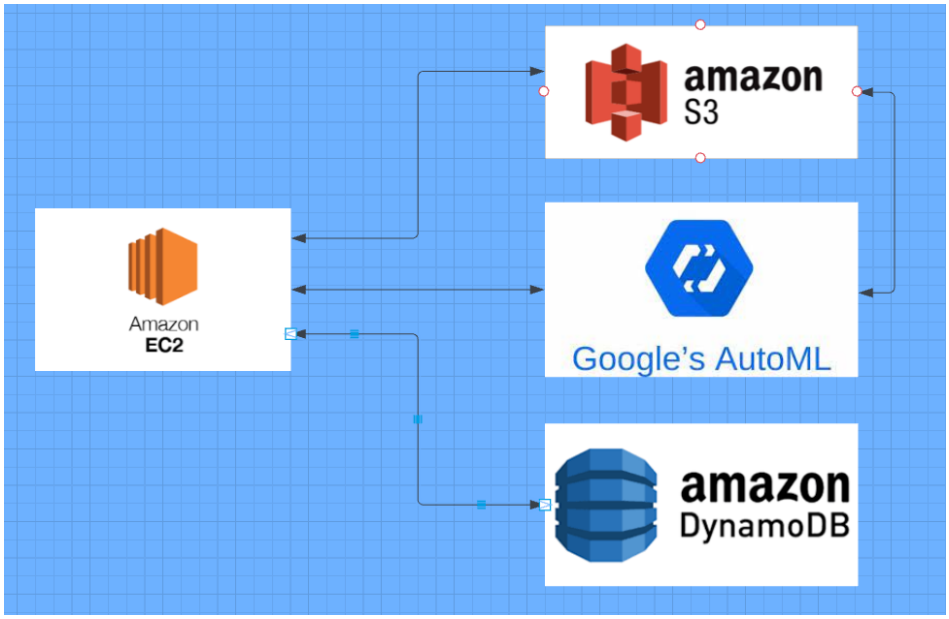

Figure 2: The infrastructure of the backend

4.1.2 Sentiment Analysis

The initial sentiment analysis model provided to us by the group from the previous semester relied on AWS Comprehend, which gave us mediocre results due to it not being geared towards worker responses [11]. Ideally, some sort of dictionary of common words (lexicon) from the workplace would be helpful to train a custom model. Many words used in the workplace have different meanings than if they were used in everyday conversations. Take for instance the phrase “oil rigging process”. This phrase should be considered neutral when taking into account that it is said by a worker talking about a process occurring in the workplace. However, for those unfamiliar with oil “rigging”, the phrase could be interpreted as talking about the process of conducting fraudulent activities with oil. Since AWS Comprehend is oriented towards general conversations, we confirmed the model treats the phrase as the latter example, hence giving it a negative sentiment. Unfortunately, our client did not have access to such a lexicon, and building one ourselves would entail a great deal of effort, expanding the expected timeline of the project quite severely.

Instead, we chose the second best option, labeling sample worker responses to conduct supervised learning. We hypothesized that a custom model based on labeled data as such would provide better accuracy, since it would be trained on data specifically consisting of worker responses. The labels would represent a relative sentiment score corresponding to a worker's response. On the other hand, the model would be trained on a limited number of worker responses (Figure 3), as the amount of data provided by our client would obviously be substantially smaller than the amount of data AWS Comprehend was trained on. Additionally, since the data didn't come with labels, our team had to manually input as many labels as possible. An issue with this approach would be the labels would be subjective towards what our team considered the appropriate sentiment, as we only had a small group of 5 people.

Description	Rating (0-4)
Asked Wetmill/Feedhouse team if machine guarding audits have been completed. Not aware of a recent audit. Removable gate in questi	2
Talked to employee about their immediate work space and the ergonomic impact of monitor position.	2
Great job and clean up	4
The driver and I discussed the use of gloves when handling the unload hose that attaches with camlock fittings. They are working in tight	3
talked about working off the scaffolding talked about the rigging process talked about the LTT	2
Observed wet mill maintenance apply fins on motor for cooling tower. They explained procedure of application of where to place fins, and	1
spoke with the people about work area condition	2
spoke with them about respirator use and my concern for their health	0
Talked about slippery road conditions and slick walking surfaces around coal rec.	1
Talked with fh operators about issues concerning the unplaned SD associated with motor. How work could have been planned , and effor	0
They were wearing the proper PPE for the actives they were performing. They needed to inspect their ladder and the issue was complete	3
Liked the fact that the press was operated safely and discussed the sampling requirements due to the site now sending press material fo	4
Had all the permits needed. They also where wearing all the correct PPE.	4
Feedback was all positive (Proper tools and body position. Proper fire watch was used during spark producing activities)	4
Larry and I discussed the hazards associated with the job. To include respiratory concern and dumping powder into a flammable atmosph	0
jdkljdl	2
good job kenny	4
very safe and thorough	4
We discussed the confined space and the hard barricading and means to get packing in	2
Darryl said that the DE sack would empty better into the hopper and with less dust if there was some different type of knife in the hopper.	2
The only concern was that while doing fire drill handling hose there may be a concern with not having fire helmet on. This will make sure	1
Contacted Corey Bearden about leak. Rotate maintenance is working on it	2
Told them i appreciated their good work.	4

Figure 3: The labeled worker responses (0-4)

Our initial thought was to stay with the AWS infrastructure and train a model using AWS Custom Comprehend, since we wouldn't need to set up administrative accounts on other cloud platforms. Unfortunately, AWS Custom Comprehend didn't provide custom sentiment analysis, but rather custom classification. We deemed this to be likely ineffective as the model would be trained using more general classification techniques rather than techniques intended for custom sentiment analysis. For example, an application of AWS Custom Comprehend could be analyzing parts of speech of input text. The service is not specific towards sentiment analysis, something that would

impede its accuracy. Next, we looked towards Microsoft Azure's Text Analytics platform [16] but it offered the same services as AWS. Still, it was worth exploring whether Microsoft's Azure would do better predicting sentiment of the workers' responses than the original AWS model. However, when testing built in models from both services, we found the AWS model to possess slightly greater accuracy. The Microsoft Azure model was accurate about 56% of the time, whereas the AWS model had a 64% accuracy. Luckily, we found that the Google Cloud Platform [8] supported custom sentiment analysis different from AWS and Microsoft Azure. It was specialized towards analyzing sentiment rather than other areas of interest in Natural Language Processing (NLP). Initially, our client was reluctant to add another service into our project lifecycle, as he believed it would add unnecessary work to correlate technologies from multiple platforms (AWS and Google Cloud). However, since the service used from Google Cloud would simply act as a cloud endpoint, we convinced our client we wouldn't have to change too much of the existing codebase - just some administrative jargon. If we chose to migrate a more fundamental service to Google Cloud such as AWS DynamoDB or EC2, we would have to consider shifting the entire technology stack to Google Cloud.

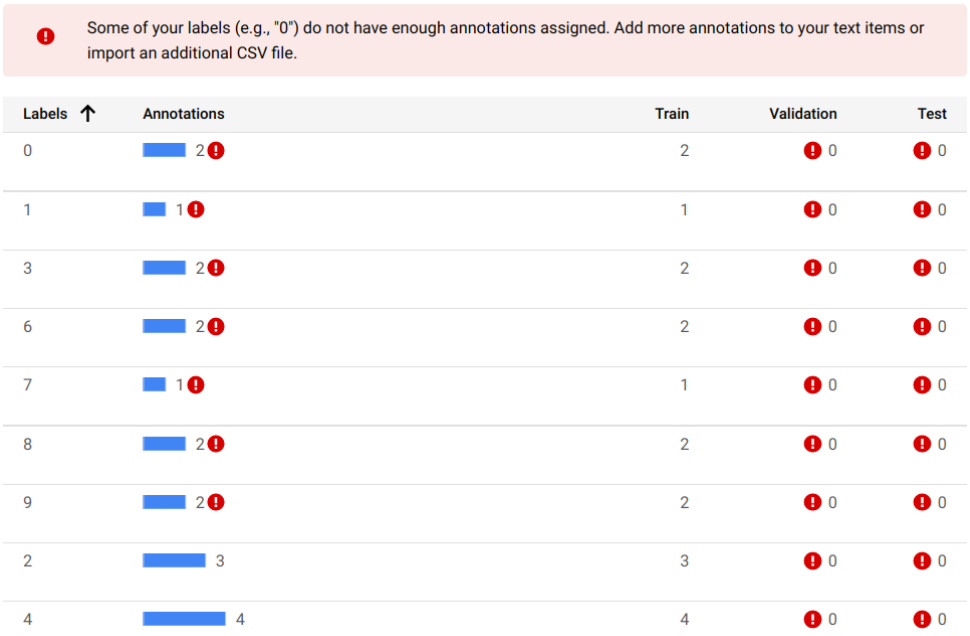

Figure 4: Insufficient numbers for training model

We initially trained a model through Google Cloud's AutoML API [9] using 350 labeled responses. The API was customizable in that we could choose a range of sentiment

scores to label, maximum from 0-10. The lower the score, the more negative the sentiment, and the higher the score, the more positive the sentiment. Our client initially proposed having 0 as extremely negative, 5 as neutral, and 10 as extremely positive, with 4 intermediate sentiment scores. The issue our team quickly found with this approach was that there simply wasn't enough data to represent all the sentiment scores in the range (Figure 4). Even if we labeled all the data, i.e., ~4000 responses, there was no guarantee that a high enough number of labels for each sentiment score would be provided. Furthermore, since labeling the data was extremely subjective, extending the range of possible sentiment scores only increased the chance of error in model prediction. Therefore, we chose a simple range (0-4) of sentiment scores to label our data with. Our final model was trained on 1544 manually labeled worker responses (Figure 3). This model produced an accuracy that was at par at best, so we decided to compute an additional metric to analyze the accuracy. The additional metric was used to broadly gauge the accuracy of the model - whether it predicts positive, negative, or neutral properly. Our client agreed to our recommendation of grouping together the "3" and "4" sentiment scores and the "0" and "1" sentiment scores. The "2" sentiment score would remain neutral, as we didn't have additional scores in our range to identify the degree of neutral sentiment. It is worth noting that this approach is not equivalent to training a basic classification model using a range of 0-2 sentiment scores since the extended range adds a buffer for labelers and the model to assess the sentiment. With a sentiment score range of 0-2, there would be a higher chance of slightly positive or slightly negative worker responses being rated as neutral, since there are simply no other values to represent the true sentiment. We concluded the new metric exceeded our expectations and our client was quite happy with the results. These results will be explored in detail in a later section.

4.1.3 CI/CD

A big improvement we made to the infrastructure of the project this semester was implementing a continuous integration continuous deployment pipelining system within AWS; the service is AWS CodePipeline [4]. The website from last semester was hosted manually on an S3 bucket. This method made updating the website a little cumbersome since it would require manually building the website locally and copying the contents of the build folder to the S3 bucket.

We felt the benefits and simplicity of CI/CD was worth the extra time it would take to implement the system. To further explain what CI/CD accomplishes, it essentially automates the above-described process. Through this service, we can connect to the cloud repo (GitHub, BitBucket, etc.) within AWS. Whenever we commit code to the master branch within the repository, the service can detect that changes have been made and subsequently pull the new source code. It then automatically builds the application files and deploys the website to our designated repository.

In our specific case, we connected our AWS account to the GitHub repository that contained our frontend code. We then created an S3 bucket and set it up to allow for public static website hosting, which included creating a static website hosting policy [7] that defines these terms. The other important file we had to add to our frontend repo is a buildspec.yml file, which can be found in the root directory of the frontend GitHub repository. This file essentially tells AWS how to build the application and which directories to pull code from. Apart from the more generic setup as you go through the CodePipeline process [4], these were the major changes needed to make our frontend CI/CD-friendly (Figure 5).

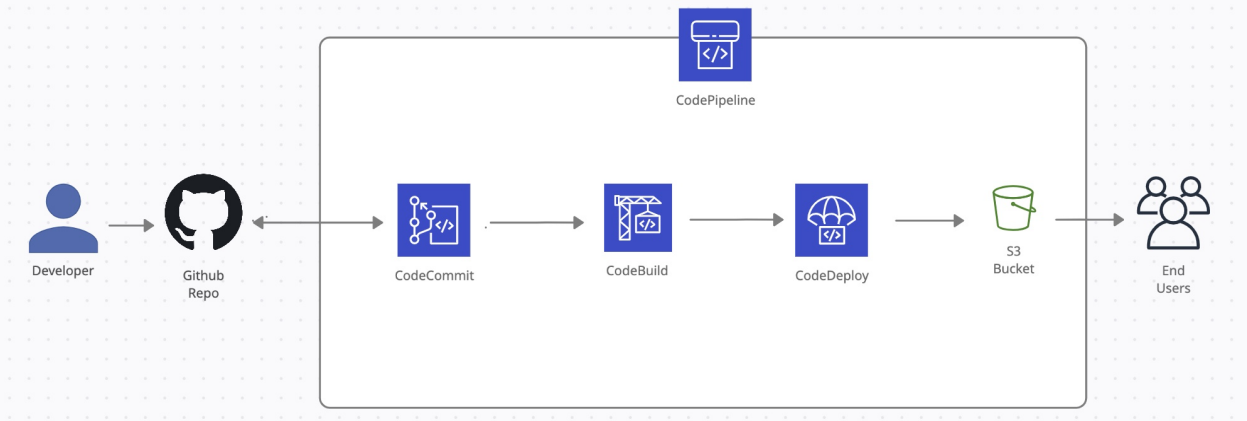

Figure 5: CI/CD Diagram

4.2 Frontend

4.2.1 Frontend Overview

Our frontend work extended the website that was created by last semester's team. The website was developed using React.js and used the Core React UI Framework [10], which provided a basic template and design. Our main additions to the website

included clean-up of various template elements that were not being used, adding in properly encrypted user login and authentication (Figure 6), a user manual page (Figure 7), and results viewing option (Figure 9) – since our sentiment analysis model design changed.

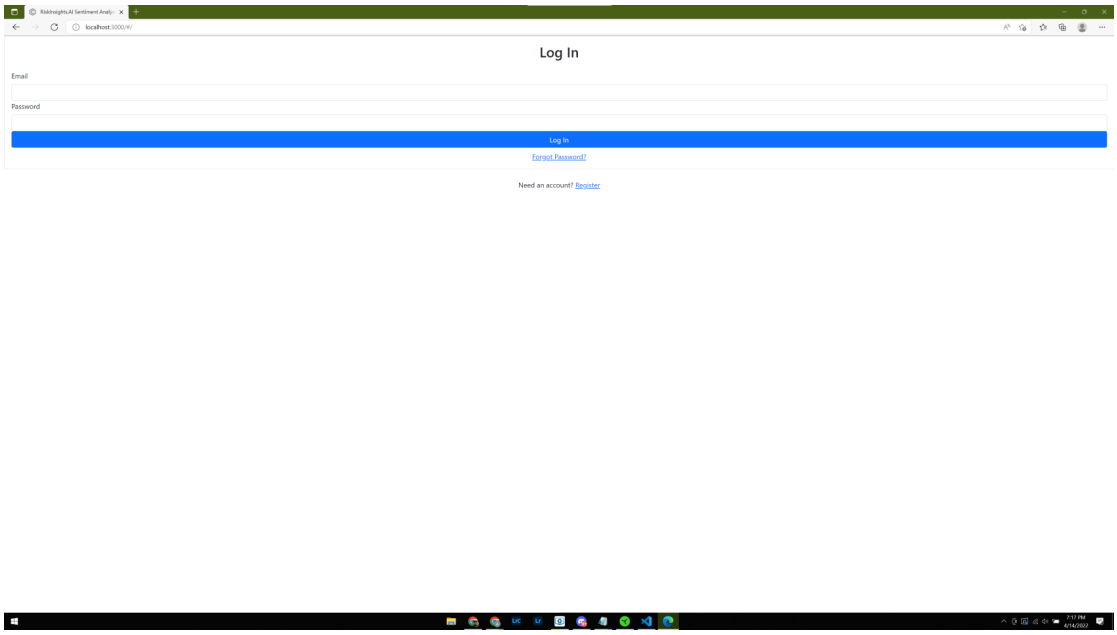

Figure 6: Login Page

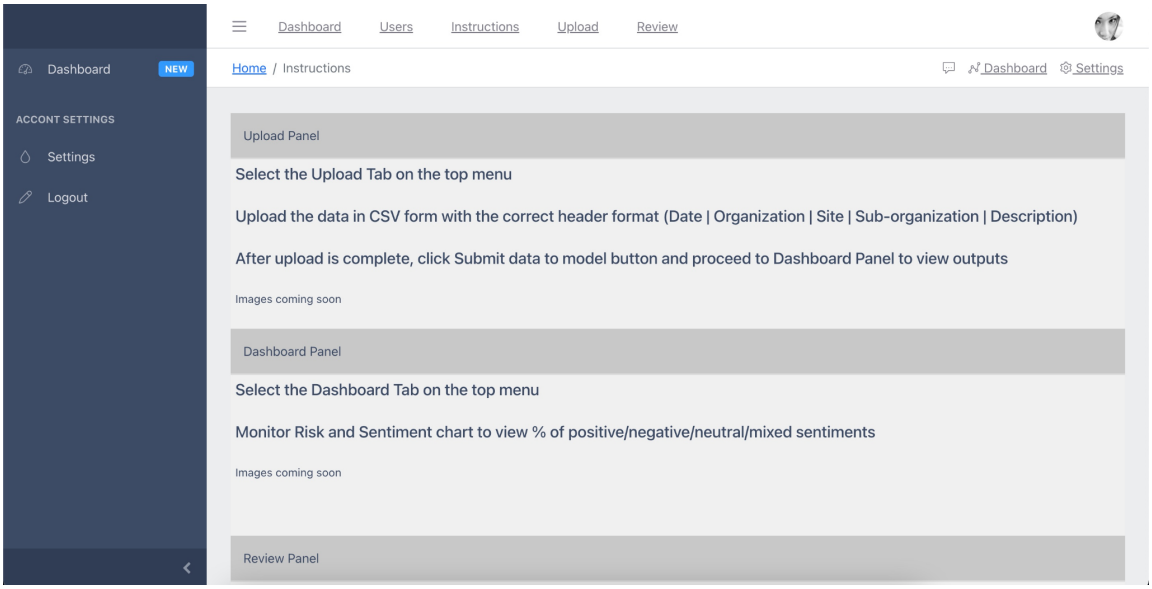

Figure 7: Instructions Page

Date	Organization	Site	Sub-organization	Description
2018-09-18	Global Operations	Site B	Wetmill	Asked Wetmill/Feedhouse team if machine guarding audits have been completed. Not aware of a recent audit. Removable gate in question should have a lock on it. Notified Area Technologist and Area Engineer and asked that the chain be replaced at the DSL dryer.
2018-10-04	Global Operations	Site B	Office area	Talked to employee about their immediate work space and the ergonomic impact of monitor position.
2018-10-20	Global Operations	Site C	Operations	Great job and clean up

Table 1: Example input CSV

Date	Organization	Site	Sub-organization	Description	Sentiment Score	Overall Sentiment
2018-09-18	Global Operations	Site B	Wetmill	Asked Wetmill/Feedhouse team if machine guarding audits have been completed. Not aware of a recent audit. Removable gate in question should have a lock on it. Notified Area Technologist and Area Engineer and asked that the chain be replaced at the DSL dryer.	2	Neutral
2018-10-04	Global Operations	Site B	Office area	Talked to employee about their immediate work space and the ergonomic impact of monitor position.	2	Neutral
2018-10-20	Global Operations	Site C	Operations	Great job and clean up	4	Positive

Table 2: Example output CSV post sentiment analysis

4.2.2 Frontend Implementation

The majority of changes to the frontend did not require any new frontend tools; they leveraged the template used last semester. The main implementation information pertains to the login and authentication.

We used Google Firebase [3] authentication because one of our team members was familiar with the framework. It was a good option because Google offers a JavaScript library [9] that integrates easily into the frontend.

In essence, the authentication service needed to be enabled and set up on our organization's Google Cloud account. Once the service is enabled, credentials are generated so the JavaScript library knows which account to link and users gain the functionality of registration, login, and resetting password. This functionality is achieved through functions that are available through the aforementioned JavaScript library.

5 Testing/Evaluation

5.1 Sentiment Analysis

The initial model created by the group from the previous semester had an accuracy of 65%. Although this wasn't extremely bad, our client felt improving the accuracy was to be the main goal of our group. Therefore, we chose to train a model using supervised learning leveraging the AutoML service from the Google Cloud Platform. Additional details of the design and implementation of the model are provided in Section 4.

However, we first attempted utilizing the built-in model from Microsoft Azure's Text Analytics Platform, in order to compare it with the existing AWS Custom Comprehend. However, we found Microsoft Azure's services to be mediocre, as the achieved accuracy was about 51%.

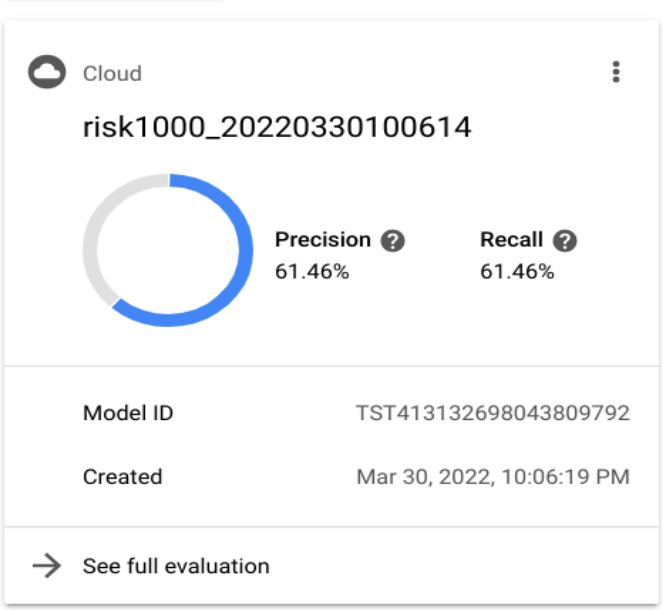

Figure 8: Overall precision and recall of the Google Cloud Model (0-4)

As described extensively in Section 4, the Google Cloud model was trained on 1544 worker responses and labeled data of sentiment scores ranging from 0-4. In our case, 0 is extremely negative, 4 is extremely positive, and 2 is neutral. At first glance, it seemed as if all the manual labeling had gone to waste, as the accuracy obtained was 61% (Figure 8), similar to the accuracy of the initial AWS Comprehend model. However, AWS Comprehend model classified "positive", "negative", and "neutral" sentiment while the supervised Google Cloud model classified sentiment across a range of 0-4,

adding two extra possible classifications to the output. Due to the subjectivity of labeling worker responses, we realized that the model had fluctuating outputs across similar sentiment scores. For example, it was treating a “3” as a “4” or a “0” as a “1” (Figure 9). In simple terms, it was effectively classifying worker responses as leaning towards positive or negative sentiment, however, it was not getting the exact sentiment correct - likely due to the low number of data points (1544). A polished built-in model of a cloud service is likely trained on millions of data points, so there was no possibility of achieving quite the same level of success.

Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in gray).

Actual score \ Predicted score	Sentiment score 0	Sentiment score 1	Sentiment score 2	Sentiment score 3	Sentiment score 4
Sentiment score 0	10%	70%	20%	-	-
Sentiment score 1	13%	78%	6%	3%	-
Sentiment score 2	4%	21%	61%	11%	4%
Sentiment score 3	-	-	-	57%	43%
Sentiment score 4	-	8%	-	25%	67%

Figure 9: Confusion matrix with spread of accuracy for sentiment scores

```
accurate = 0
for m, r in zip(model_ratings, test_ratings):
    if m == 2 and r == 2:
        accurate += 1
    elif m == 3 or m == 4:
        if r == 3 or r == 4:
            accurate += 1
    else:
        if r == 0 or r == 1:
            accurate += 1

print(accurate / len(test_ratings))
```

```
0.8469387755102041
```

Figure 10: Output of the new metric describing the accuracy of the model

To combat the aforementioned issue, our team proposed an additional metric to gauge the accuracy of the model. The metric would provide functionality similar to the original model, in that it would compare whether a response is positive, negative, or neutral, rather than checking its exact sentiment score. Since the model had issues with worker responses leaning a certain direction on the sentiment scale, we were optimistic that this metric would garner great results. We decided to group together sentiment scores “3” and “4” and sentiment scores of “0” and “1”. A more detailed explanation of our implementation choice for this metric is described in Section 4. The accuracy of the secondary metric was about 84.6% (Figure 10), so our client was quite ecstatic about our progress, as it was a 21% jump from the initial AWS Comprehend Model.

5.2 Audio Evaluation

A secondary goal of the project was to support worker responses input in an audio format, so we researched various audio transcription services to convert audio inputs into text format. In the end, we decided to use Google Cloud’s audio transcription service, as one of our team members already had experience with it. After testing the service with 5 samples each lasting over a minute, we achieved 93% accuracy, which we deemed to be quite effective. Unfortunately, our client paused audio transcription as a project goal as he believed it was more important to focus attention completely on improving the quality of sentiment analysis, but it will definitely be a point of interest for future work.

5.3 Frontend Testing

Testing of our frontend was mostly done through bug reports between the team members as well as manual testing from our client. We created a shared document/Trello board listing the current bugs and what changes needed to be made in order to fix them. During our weekly meetings with our client, we would go over any progress on bugs and accordingly update our board.

6 Developer Manual

6.1 Frontend

Setup: The frontend is based on a template from CoreUI libraries [7] and works with React.js. In order to follow the commands, it is necessary to have a Git terminal, such as Git for Windows. The commands are for the terminal command line.

Install Node.js for the npm commands and then clone the repository to a folder ./Risk-Sentiment-Analysis-FrontEnd with the following command:

git clone https://github.com/Parvesh-S/Risk-Sentiment-Analysis-FrontEnd.git

This will create the folder in the current directory. To enter the newly created folder, use the following command:

cd ./Risk-Sentiment-Analysis-FrontEnd

The command will load you into the main branch. To install the libraries required to run the server on the frontend, use the following commands:

npm install
npm install bootstrap

Now to run the frontend, use the following command:

npm start

This should get the server for the frontend running on localhost with port 3000. This can be accessed by going to the following URL:
http://localhost:3000/#/

Methodology: The purpose of the frontend is to facilitate the connection between devices and backend server, which occurs while sending the CSV file from the device to the backend and then sending a few data points back to the device from backend to review. All the components of the frontend server are separated into templates. The frontend server communicates with the backend and client browsers. The sentiment analysis of the data and data storing are handled by the backend.

6.2 Google Cloud & Backend

Our backend code is also hosted in a GitHub repository:

https://github.com/Parvesh-S/Risk-Sentiment-Analysis-Backend

Cloning and deploying the backend can be found in last semester's report [11]. We did not make any modifications to this process.

To train a new custom AutoML sentiment analysis model through the Google Cloud Platform, all that is required is the Google Cloud Console credentials and a labeled dataset.

1. Navigate to the project "riskAnalysis" in the Google Cloud Platform.
2. Select AutoML Sentiment Analysis and input a range of sentiment scores the worker responses are labeled with. For the existing model, the range should be 0-4.
3. Upload the labeled dataset to a Google Cloud bucket. This should take around 30 minutes to an hour for completion. It took us 45 minutes to complete.
4. Once the dataset is uploaded, simply follow the prompts to train the model. It is advised to dedicate enough time for this step, as the model can take multiple hours to train. We left the model training overnight, so the exact time is unclear.
5. Evaluate the accuracy in the "Evaluate" tab and export the data to a bucket using the Export Data button in the "Test" tab.

6.3 Deployment of Application

6.3.1 Frontend

In order to update the frontend, implement the following commands:

1. Update the GitHub files with the local files that were edited; the commands below should run when in the root directory of the frontend folder.

   ```
   git add <file names of modified files>
   git commit -m "<message describing what changes were made>"
   git push
   ```

2. With the CI/CD, the code from GitHub is automatically pulled by the AWS service and compiled to produce the build code. Then it updates the frontend with the refreshed code.

7 User Manual

Our website includes an added Instructions page [Figure 7], which explains to the users the order of the steps to utilize the sentiment analysis process. The Instructions page describes the user interface of the frontend. The application has four subsections for the user, which are: data upload, data human review, display chart of output data, instructions for client to use application, and security of the user.

For the client instructions to use the application, there is a separate page, as seen in Figure 7. By clicking the “Instructions” on the header bar, the user can access the instructions that will be useful while working on their data and for the upload of the data. This page also provides the user with the flow of work of the application to get a good user experience.

In order to provide more security for the application, there is a login requirement for the users, which is administered by our client through Google Firebase. The users that are able to access the application will be required to login or sign up for the application. Figure 11 is the sign up / registration page, while Figure 6 is the login page.

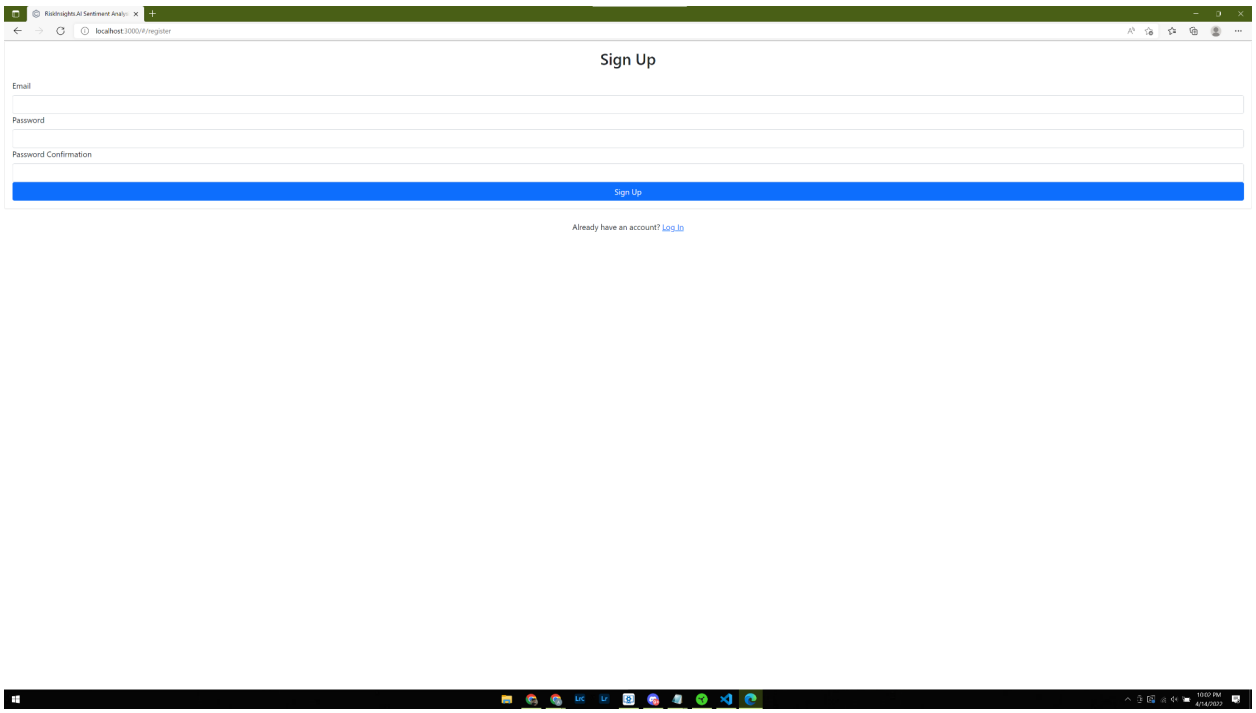

Figure 11: Registration Page

8 Lessons Learned

8.1 Timeline/Schedule

- January 27 - February 15:
    - Finish setting AWS permissions/roles for all team members
    - Review/test previous codebase
- February 15 - February 28:
    - Test/Compare AWS and other Cloud ML models (Azure, Google Cloud)
    - Research transcription and login/authentication APIs
- February 28 - March 15:
    - Begin to implement authentication (prioritized over audio transcription functionality)
    - Finalize and begin to implement chosen ML model
    - Begin to manually label ~1500 worker narrative responses
- March 15 - April 1:
    - Finalize UI improvements (minor styling, cleanup of unused elements)
    - Deploy Google Cloud backend functionality
    - Start testing and recording bugs
- April 1 - April 15:
    - Connect backend to frontend
    - Implement CI/CD
    - Fix any remaining bugs (mainly routing)

8.2 Problems

We ran into a number of problems throughout the course of this project, but one of the first problems was our schedule, which was not able to properly gauge which features to prioritize and also how long it would take to complete. Primarily, there were some features that were put on the backburner during the course of the semester, specifically the audio transcription and continuous learning for the ML model, due to feasibility and time constraints. Additionally, more resources were needed than we originally thought for testing and creating a new, more accurate ML model.

The second problem had to do with the quality of data. A lot of the data was very objective, as in the worker responses were simple recounts of which activities were

observed rather than actual feedback regarding the safety of the workplace. Because of this, a large part of our training data was neutral. Our other data skewed to the positive side, but we only had approximately 400 data points for those. Due to this skew in the data, our initial model skewed heavily to the positive side and our test results showed the model was often incapable of correctly classifying negative worker responses. The model was accurate around 79% of the time when supplied with positive responses while the negative accuracy was a mediocre 23%.

The third problem we had was that we were building off of a previous project rather than starting from scratch. Because of this, initially, there was a lot of information that was lost in translation on how to properly work with the previous semester's work, including setting up permissions, getting access to repositories and resources, and properly understanding the existing codebase. The latter, in particular, affected a lot of our team member's ability to properly work. Since this learning segment took longer than expected, delays occurred regarding other feature development.

8.3 Solutions

We solved the first problem by working with our client to discuss our blockers and understanding what features are important and should be prioritized. By doing this, we were able to decide that the main goal should be to have an improved ML model ready by the end of the semester and to have login and authentication functionality working for the frontend. By knowing this, we were able to properly allocate time and resources to certain features and put a pause on features that weren't important. Solid communication between team members and our client helped solve this issue.

The second problem was solved by combining our existing training data set with new data that our client had available. This table fell under the "Near Misses" and "Concern Report" titles, which meant they skewed heavily to the negative side. This balanced out with our previous positively skewed data to enable a more accurate model. This additional data also brought our manually labeled number to ~1500, which was a much larger quantity than last year's model and our initial Google Cloud model. Simply having more data, and data that was diverse in classification, definitely helped improve our model to a much higher accuracy.

The third problem was more minor compared to the other issues, but it was solved through two means. First, we were able to get in contact with the previous semester's

team members and set up a small meeting so they could provide our team with a more comprehensive explanation of their codebase and resources. Additionally, having their contact information also helped us quickly resolve situations that had blockers related to the previous semester's work. The second way was that we made sure to have more frequent meetings amongst our team (outside of our weekly meetings) so we could readily share updates and knowledge with each other, so everyone would overall be more knowledgeable. CI/CD was also implemented to help future teams with deploying new versions of the frontend site, since previously it was a hassle to deploy changes as the process was all manual.

8.4 Future Work

Future work for the project will most likely occur next semester, given that this project has already spanned multiple semesters.

8.4.1 Next Semester

A lot of possibilities exist in terms of improvement. Two of the features we had to pause for this semester, but were features our client was interested in, were audio transcription and continuous learning/improvement of the ML model. Working on these two features would be interesting and useful work during the next semester. Additionally, as mentioned previously, setting up a testing and feedback tool could greatly improve the quality of our UI and give our client more data on what aspects to improve and perhaps which features to implement as the project progresses. The CI/CD tool could also be improved as it currently does not include the testing feature, which ensures new changes properly function before deploying. The backend deployment method could also be improved since it is still manual. Lastly, the graphs which displayed a visualization for the analysis results need to be modified since our ML model and format changed.

9 Acknowledgements

We would like to acknowledge Dr. Edward Fox, our professor, for guiding us and supporting us throughout the project, especially with the machine learning model.

Dr. Edward Fox

We would also like to acknowledge Mr. Christian Johnson for being accommodating with the challenges faced and being professional. He was very supportive of the approaches our team took and the tools our team decided to work with.

Christian Johnson

10 References

[1] 360training. New Survey Reveals Rise In Workplace Safety Concerns. https://www.360training.com/blog/new-survey-reveals-rise-workplace-safety-concerns#:~:text=Top%20Workplace%20Safety%20Concerns%20in%202019&text=Electrical%20Hazards%20(13%25),Falling%20Objects%20(6%25), 2022. Accessed: 30-Jan-2022.

[2] Oxford Language. Oxford Languages and Google. https://languages.oup.com/google-dictionary-en/, 2022.

[3] Google Firebase. Firebase Documentation. https://firebase.google.com/docs, 2022. Accessed: 10-Mar-2022.

[4] Amazon Web Services. Set Up a CI/CD Pipeline on AWS. https://aws.amazon.com/getting-started/hands-on/set-up-ci-cd-pipeline/, 2022. Accessed: 17-Apr-2022.

[5] Google Cloud. AutoML beginner's guide. https://cloud.google.com/vertex-ai/docs/beginner/beginners-guide, 2022. Accessed: 20-Feb-2022.

[6] Google Cloud. Sentiment Analysis Tutorial https://cloud.google.com/natural-language/docs/sentiment-tutorial, 2022. Accessed: 20-Feb-2022.

[7] Amazon Web Services. Bucket policy examples. https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html, 2022, Accessed: 21-Feb-2022.

[8] Google Cloud. APIs and references. https://cloud.google.com/speech-to-text/docs/apis, 2022. Accessed: 21-Feb-2022.

[9] Google Firebase. Authenticate Using Google with JavaScript. https://firebase.google.com/docs/auth/web/google-signin, 2022. Accessed: 10-Mar-2022.

[10] CoreUI. Free bootstrap admin templates - CoreUI. https://coreui.io/ , 2021. Accessed: 15-Feb-2022.

[11] Himler, Tom; Sharma, Pranav; Taing, Eriq; Akula, Akshay; and Joaquin, Evan. Risk Prediction Sentiment Analysis. 2021. CS4624 submission to VTechWorks, Virginia Tech, Blacksburg, VA 24061 USA. http://hdl.handle.net/10919/107004, accessed May 3, 2022.

[12] Christian Johnson. Risk Insights AI. https://riskinsights.ai/, 2022. Accessed: 15-Feb-2022.

[13] Amazon Web Services. AWS Comprehend. https://aws.amazon.com/comprehend/, 2022. Accessed: 28-Feb-2022.

[14] Amazon Web Services. AWS DynamoDb. https://aws.amazon.com/dynamodb/, 2022. Accessed: 15-Mar-2022.

[15] NodeJs. NodeJs. https://nodejs.org/en/, 2022. Accessed: 20-Feb-2022.

[16] Microsoft. Microsoft Azure Text Analytics. https://azure.microsoft.com/en-us/free/cognitive-services/, 2022. Accessed: 21-Feb-2022.