A Capstone Project on

Industrial Safety Chatbot
using NLP

Submitted by :
Drishti Chopra
Srujana Rayaprolu
Aswathi Sasidharan
Akshay J S

Mentor:
Madhan Seduraman

Submission Date: January 29, 2021
Submitted in Complete Fulfilment of the requirements for PGP in AIML

Table of Contents

1. Introduction

1.1 Background

Industrial Safety refers to the management of all operations and events within an industry in order to protect its employees and assets by minimizing hazards, risks, accidents, and near misses. The Occupational Safety and Health Association (OSHA) is the primary regulatory body in the United States dedicated to ensuring industrial safety. Industrial accidents cause heavy loss to the employees, industry as well as environment. The ever-increasing mechanization, electrification, criminalization and sophistication have made industrial jobs more and more complex and intricate. This has led to increased dangers to human life in industries through accidents and injuries. In fact, the same underline the need for and importance of industrial safety.

1.2 Problem Statement:

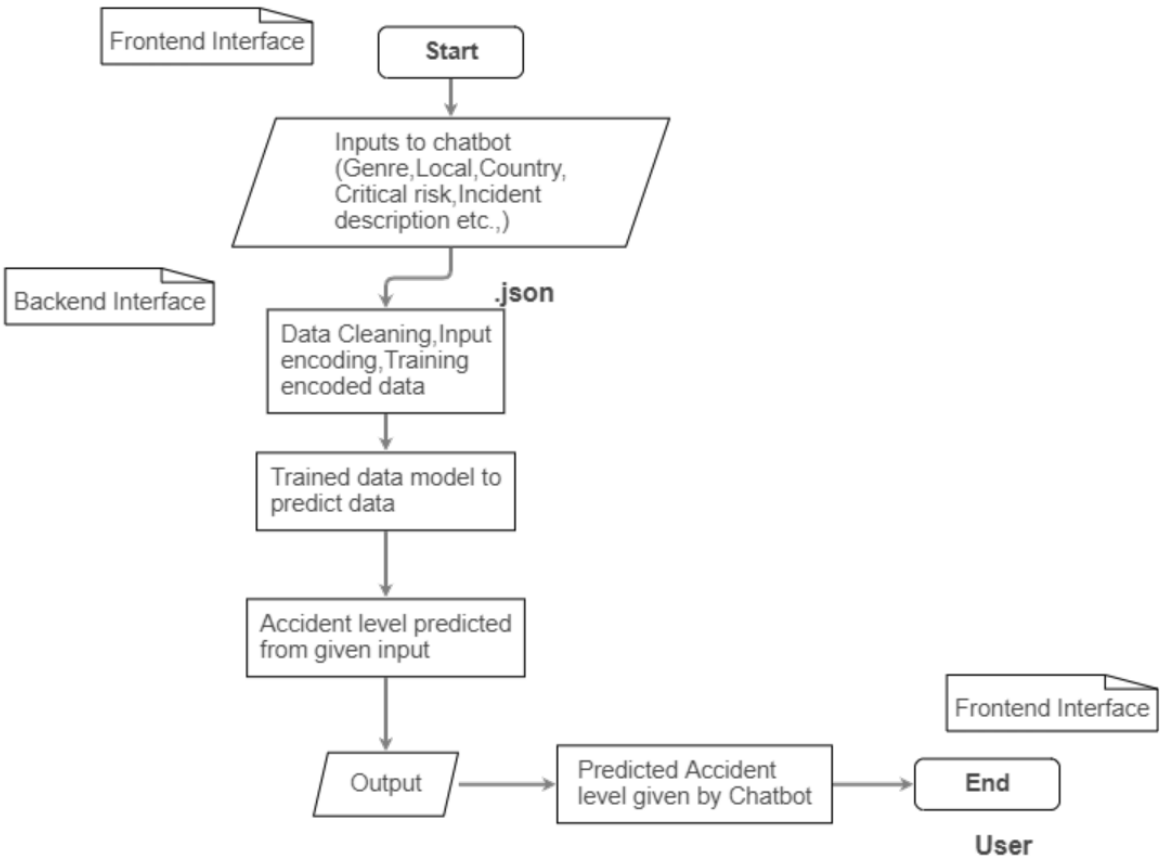
Design an ML/DL-based chatbot utility that can help the professionals to highlight the safety risk as per the incident description.

1.3 Problem Solution:

We are provided with a dataset by a Brazilian company, IHM Stefanini that includes information about accidents in 12 manufacturing plants in 3 countries. We need to use this dataset to understand why accidents occur and discover clues to reduce tragic accidents.

In this project,we analysed this accident data and developed a chatbot based on NLP aiming to help manufacturing plants to save lives as there are a lot of accidents that happen everyday.

1.4 Flow Chart:



1.5 Data Source:

The details of the data to build the model is available at the below link:
https://drive.google.com/file/d/1iSdZ4TGC7hceNbVGuIbaLc3lUtvVL3pw/view?usp=sharing

The Data source consists of a single spreadsheet with different attributes that gives information regarding the accident/injuries that happened in industries. The database comes from one of the biggest industries in Brazil and in the world. The shape of this data is **(425, 11)** i.e. 425 rows and 11 columns.

A sample data is shown below:

```
Shape of the raw data is : (425, 11)
```

	Unnamed: 0	Data	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party	Critical Risk	Description
0	0	2016-01-01 00:00:00	Country_01	Local_01	Mining	I	IV	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...
1	1	2016-01-02 00:00:00	Country_02	Local_02	Mining	I	IV	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...
2	2	2016-01-06 00:00:00	Country_01	Local_03	Mining	I	III	Male	Third Party (Remote)	Manual Tools	In the sub-station MILPO located at level +170...
3	3	2016-01-08 00:00:00	Country_01	Local_04	Mining	I	I	Male	Third Party	Others	Being 9:45 am. approximately in the Nv. 1880 C...
4	4	2016-01-10 00:00:00	Country_01	Local_04	Mining	IV	IV	Male	Third Party	Others	Approximately at 11:45 a.m. in circumstances t...
5	5	2016-01-12 00:00:00	Country_02	Local_05	Metals	I	III	Male	Third Party (Remote)	Pressurized Systems	During the unloading operation of the ustulado...
6	6	2016-01-16 00:00:00	Country_02	Local_05	Metals	I	III	Male	Employee	Fall prevention (same level)	The collaborator reports that he was on street...
7	7	2016-01-17 00:00:00	Country_01	Local_04	Mining	I	III	Male	Third Party	Pressed	At approximately 04:50 p.m., when the mechanic...
8	8	2016-01-19 00:00:00	Country_02	Local_02	Mining	I	IV	Male	Third Party (Remote)	Others	Employee was sitting in the resting area at le...
9	9	2016-01-26 00:00:00	Country_01	Local_06	Metals	I	II	Male	Third Party	Chemical substances	At the moment the forklift operator went to ma...

2. Exploratory Data Analysis

Data source Insights:

On exploring the data source, there are the insights which help to drive the solution for the problem statement. ‘Accidental Level’ attribute was considered as the decision/target column to describe the incident and the remaining attributes were taken as the data collections (independent columns).

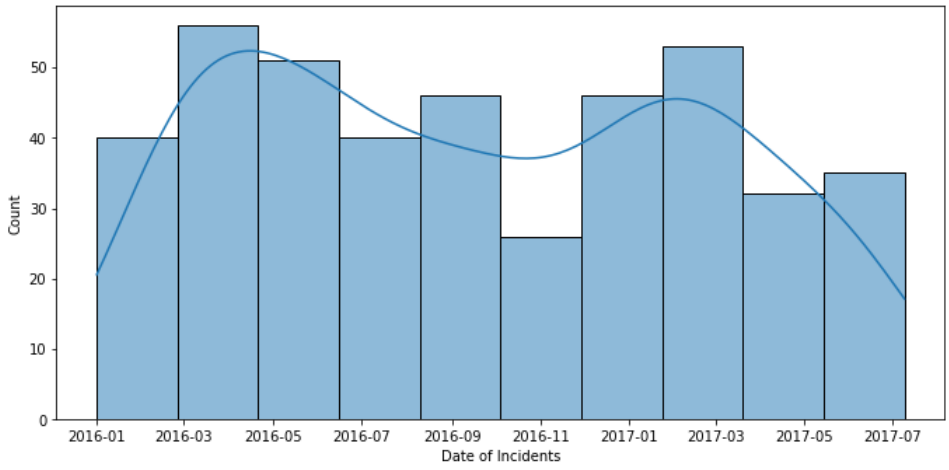
Univariate Analysis:

Unnamed: 0

This column contains index numbers from 0 to 424. It doesn't add value in the process of analyzing the data for the model development. Hence this column can be dropped.

Date:

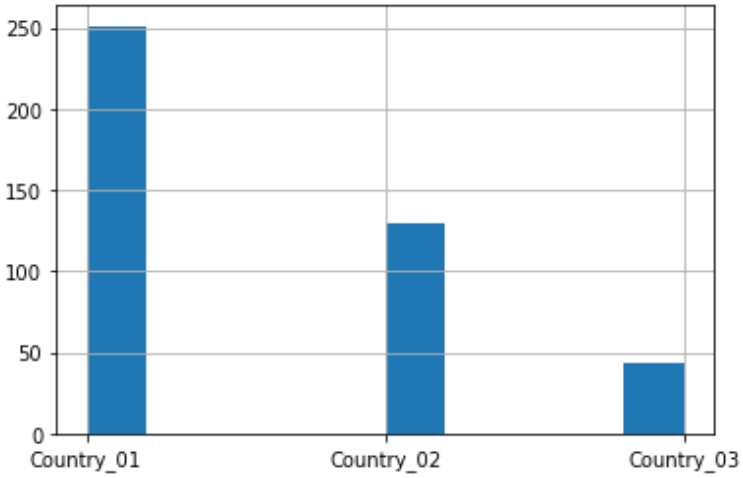
This column in the dataset contains the date and timestamp information of the accident. It is in the format of Year/Month/Date. Further, it is reframed as three different columns to examine whether any pattern can be observed in the accidents/injuries that have happened.



Univariate Analysis of **Data Column**

Countries :

It has the information in which country the accident occurred. The data source is the anonymized data from three countries in the world. They are denoted as Country_01, Country_02, Country_03 in the dataset.



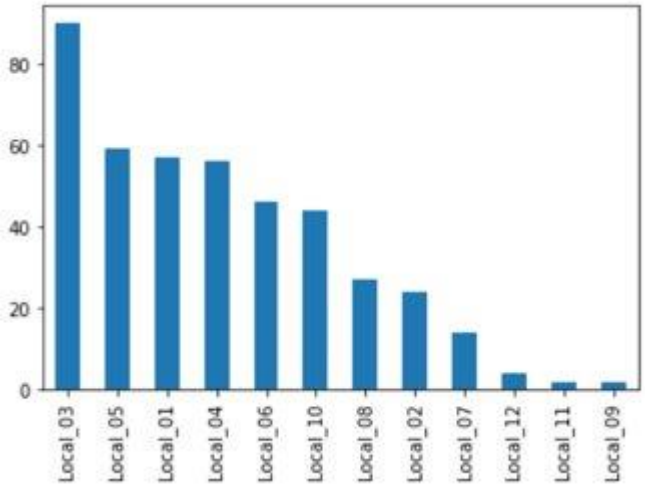
Univariate Analysis of **Countries Column**

Inference:

Country_01 is identified with the highest accidents 251 among 425 entries.

Local:

These are the local cities from the above three countries where the manufacturing plant is located.

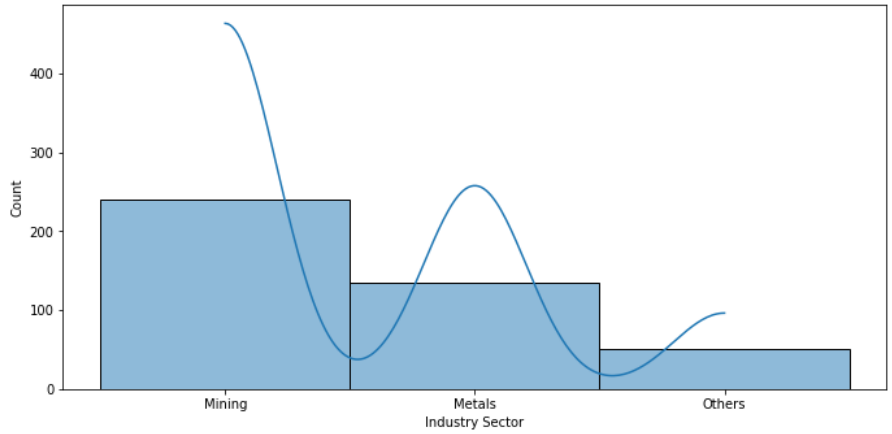


Inference:

There are 12 in total where local_03 has the highest number of accidents 90.

Industry Sector:

It is giving the information regarding the sector of the plant belongs to. With this data, the kind of accident/injury that happened can be estimated.



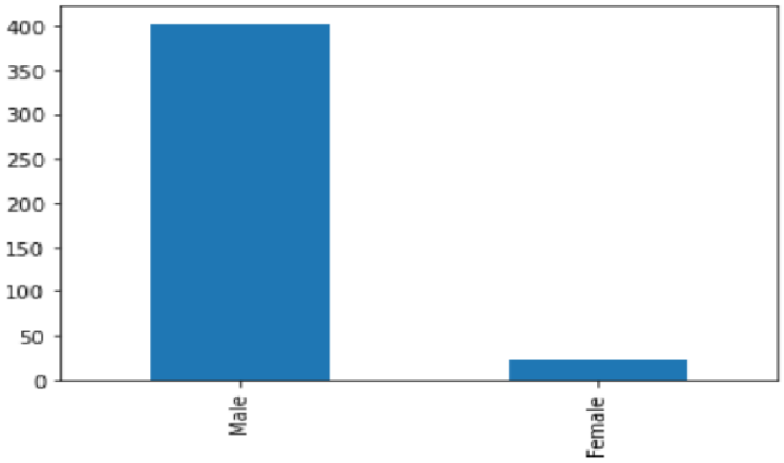
Univariate Analysis of **Industry Sector Column**

Inference:

Most of the accidents happen in the “mining” and “metals” sectors. The rest of all noted as other categories in the dataset.

Genre:

It is the Gender of the person who got affected in the accident. In this dataset, most of the people are male.



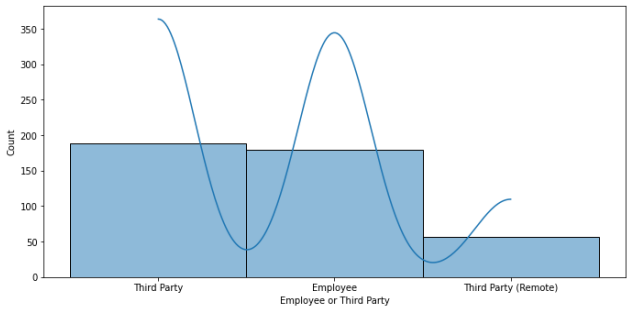
Univariate Analysis of **Genre Column**

Inference:

Only 22 female persons are injured out of 425 entries (5% of total affected persons). This shows a large class imbalance in the dataset for the Genre feature.

Employee or Third Party:

It is the column that narrates whether the injured person is an employee in that plant or an outsider..



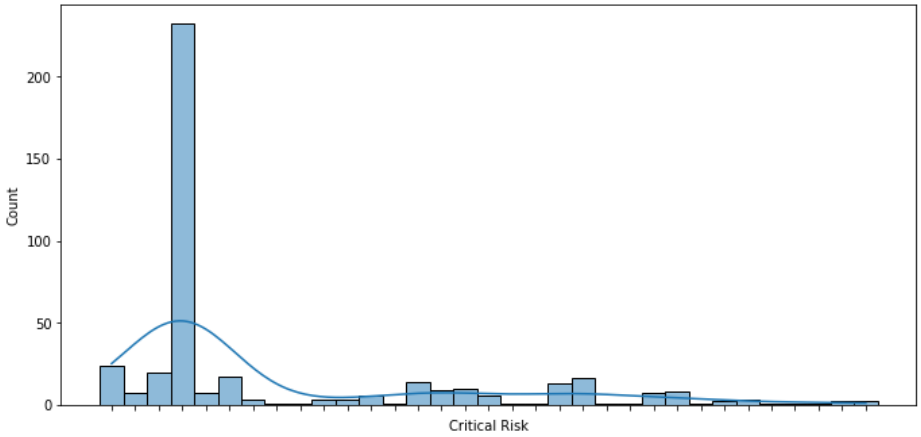
Univariate Analysis of **Employee or Third Party Column**

Inference:

In the dataset, Third-party people are affected more including the third category Third-party (Remote) rather than the employees. This also has to be converted into numerical data type using Label Encoding techniques

Critical Risk:

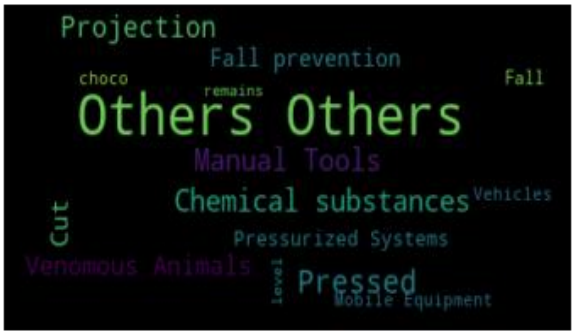
This column gives a brief idea regarding the root source involved in the accident like because of any venomous animals, manual tools, or Chemical substances, the accident has occurred.



Univariate Analysis of **Critical Risk Column**

Inference:

Maximum number of entries are noted as “Others” among 33 different reasons are listed **denoting a class imbalance**. Those other related detailed descriptions of the accidents are completely different, no pattern is observed.



Description:

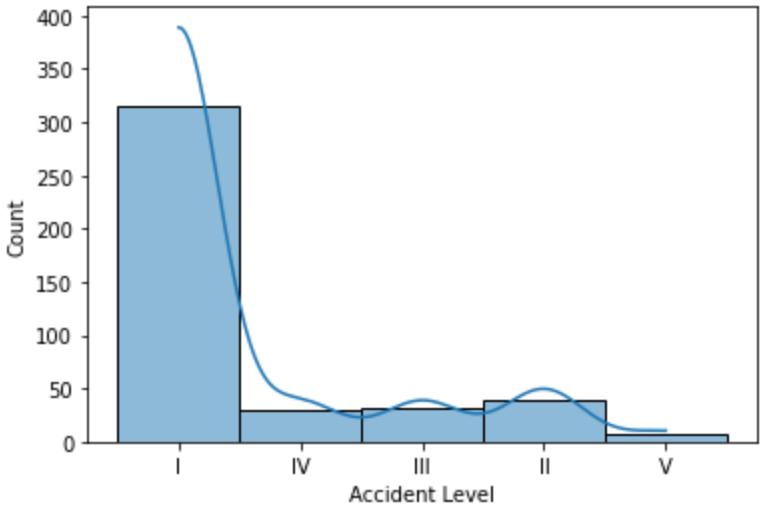
This column is a detailed description of how the accident happened. It involves complete text data and numbers with time and addresses telling that when and where the accident/injury happened. As it is the text data it needs to be handled as a part of the NLP text Pre-processing steps like tokenization, stopword, and punctuation removal, stemming, and lemmatization.

Accident Level: (TARGET COLUMN)

Here the level of accidents is registered from I to VI, how severe was the accident. I is considered as not severe while VI is a very severe accident. It will be our Target column since the chatbot must be trained in such a way that it will give a response regarding the accident and its related information when a user asks.

```
sns.histplot(dataframe['Accident Level'],kde=True)
```

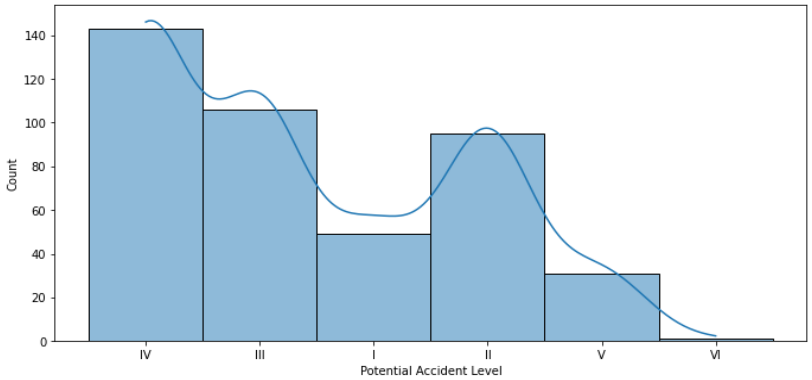
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa2636f67b8>
```



Univariate Analysis of **Accident Level Column**

Potential Accident Level:

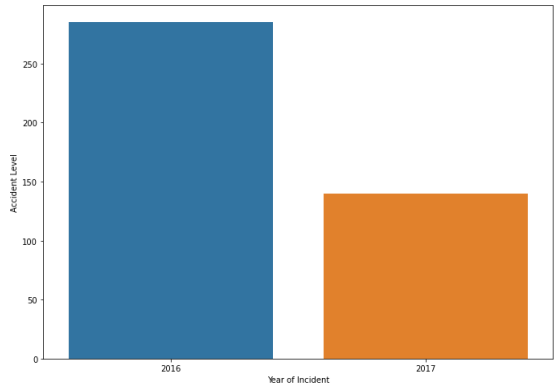
Depending on the Accident Level, the database also registers how severe the accident could have been due to other factors involved in the accident. This also followed the same notation I to VI similar to Accident Level, the target column.



Univariate Analysis of **Potential Accident Level Column**

Bivariate Analysis: Relationship between Target column and Independent columns:

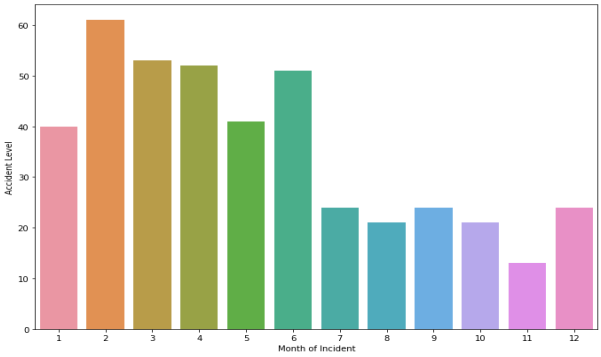
1. Bivariate Analysis of **Year of Incident vs Accident Level**



Inference:

It is shown that the number of accidents are higher in 2016 than 2017.

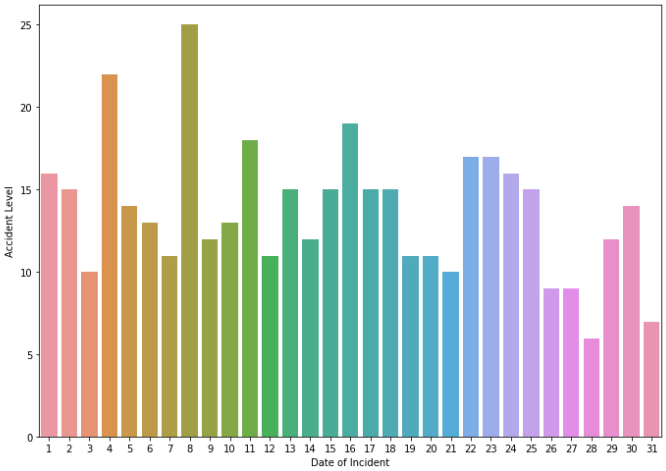
2. Bivariate Analysis of Month of Incident vs Accident Level



Inference:

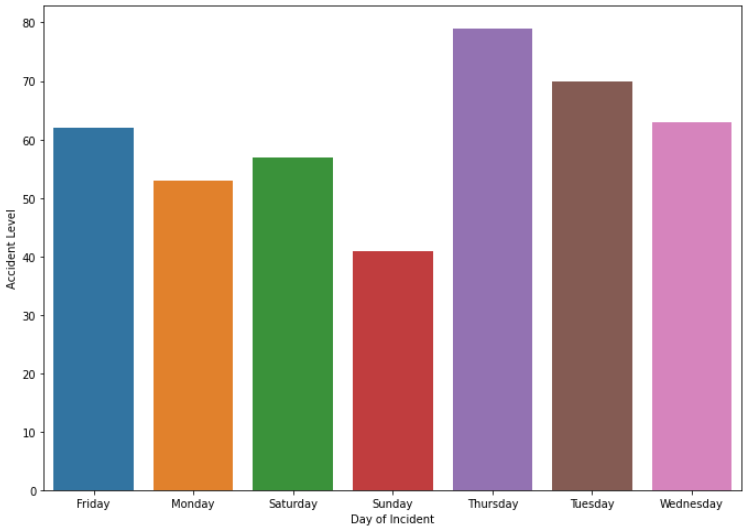
Highest number of accidents are registered in the month of February.

3. Bivariate Analysis of Date of Incident vs Accident Level



Inference:

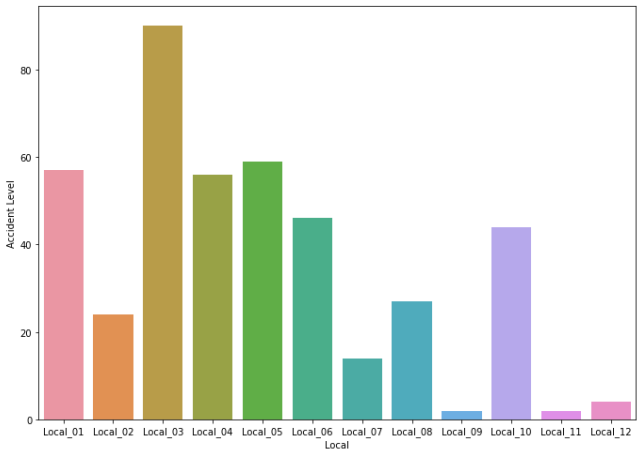
On the date 8th,more accidents are placed.



Bivariate Analysis of **Day of Incident vs Accident Level**

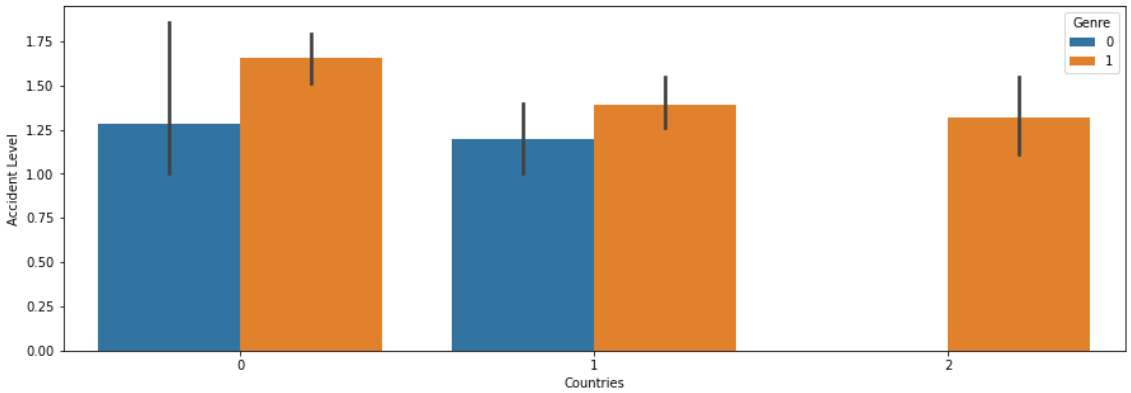
Inference:

The Highest accident level was reported on Thursday.

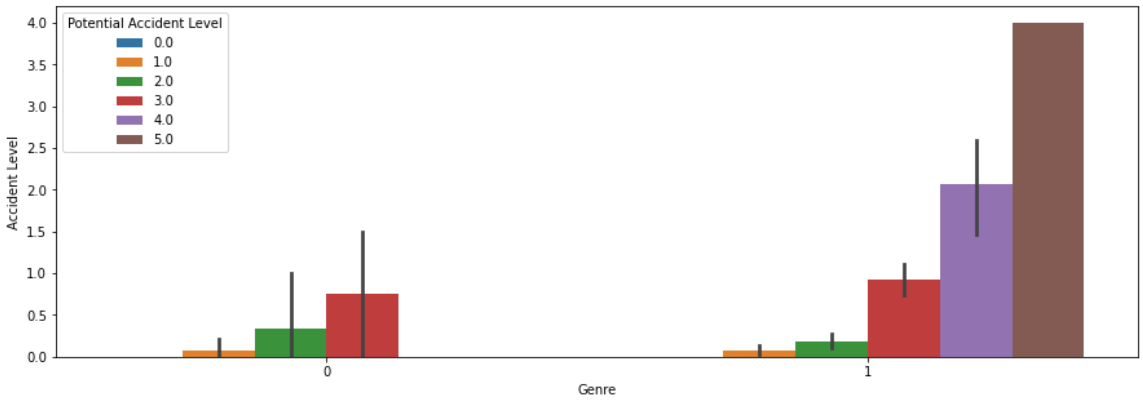


Bivariate Analysis of **Local Column**

Multi Variate Analysis:



Analysis of **Countries vs Target** column with respect to **Genre** column



Analysis of **Genre vs Target** column with respect to **Potential Accident Level** column

Summary of Data Inference for the provided data:

1. More number of accidents are registered in the year 2016 when compared to the year 2017.
2. In Country_01, location local_03 is the danger prone plant among 12 manufacturing plants as the highest are happening and registered there.
3. Mining sector is hazardous where many accidents are occurred.
4. Male affected more in the accidents than female as the manufacturing plants are in the mining and metal sector. There male workers are more, scope for female workers is less.
5. Third Party people are injured mostly than employees. Generally employees will be trained with the equipment before they start work in the plant.
6. In the “CRITICAL RISK” column, more than half the incidents are with category ‘others'. That might be the entries for this database were entered through a form or program that had a limited number of valid inputs. It could also be that these entries were changed to 'Others' as part of the anonymization process.

2.2 Data Pre-Processing

2.2.1 Checking Missing values:

The dataset is checked if missing values are any.

	Total	Percent
Description	0	0.0
Critical Risk	0	0.0
Employee or Third Party	0	0.0
Genre	0	0.0
Potential Accident Level	0	0.0
Accident Level	0	0.0
Industry Sector	0	0.0
Local	0	0.0
Countries	0	0.0
Data	0	0.0

Inference:
There are no missing values in the dataset.

2.2.2 Dropping unused columns:

In the dataset, the First column named Unnamed:0 is index values from 0 to 424. So that is removed as it will not help further. Data column is reframed as three different columns as a year, month and date. Therefore, data is also removed from the original raw dataset.

2.2.3 Label Encoding and One hot Encoding:

As the columns are object data types, they need to be converted using label encoding or one hot encoding technique.

2.2.4 Text Pre-processing using NLP:

The column description has text content. That is to be pre-processed before going through the modeling using the techniques tokenization, stemming and lemmatization, etc. Stop word removal is the most useful step in text pre-processing. It will remove all the words that are unnecessary in the modeling process.

Natural Language Toolkit (NLTK):

Natural Language Toolkit is a Python library that makes it easy to process human language data. It provides easy-to-use interfaces to many language-based resources such as the Open Multilingual Wordnet, as well as access to a variety of text-processing libraries.

2.2.5 N-Grams:

N-grams are a tool in nlp for finding sets of commonly co-occuring words, where *n* refers to the number of consecutive occurring words. The *n*-grams typically are collected from the corpus. An *n*-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram", size 3 is a "trigram". Here, bigrams and trigrams are performed to analyze the cause of the accident. The top 10 most commonly used word sets are considered for the analysis.

2.2.6 Bigrams:

```
word_fd = nltk.FreqDist(description)
bigram_fd = nltk.FreqDist(nltk.bigrams(lem))

bigram_fd.most_common(10)
```

```
[(('left', 'hand'), 70),
 (('caus', 'injuri'), 57),
 (('right', 'hand'), 57),
 (('time', 'accid'), 56),
 (('finger', 'left'), 25),
 (('employe', 'report'), 24),
 (('injuri', 'describ'), 20),
 (('da', 'silva'), 18),
 (('area', 'u200b'), 17),
 (('medic', 'center'), 17)]
```

2.2.7 Trigrams:

```
word_fd = nltk.FreqDist(description)
trigram_fd = nltk.FreqDist(nltk.trigrams(lem))

trigram_fd.most_common(10)
```

```
[(('finger', 'left', 'hand'), 24),
 (('caus', 'injuri', 'describ'), 16),
 (('finger', 'right', 'hand'), 14),
 (('injuri', 'time', 'accid'), 13),
 (('gener', 'describ', 'injuri'), 9),
 (('time', 'accid', 'employe'), 9),
 (('area', 'u200b', 'u200bth'), 8),
 (('hand', 'caus', 'injuri'), 8),
 (('manoel', 'da', 'silva'), 8),
 (('describ', 'time', 'accid'), 7)]
```

Inference:

It is observed that the frequently occurred words are “hand” either with the word left,right,injury. That means most of the accidents occured are hand injuries, which comes under the category of minor accidents. Therefore, the dataset also has more number of “ Level 1” accidents.

3. Data Preparation and Cleanup for Modelling

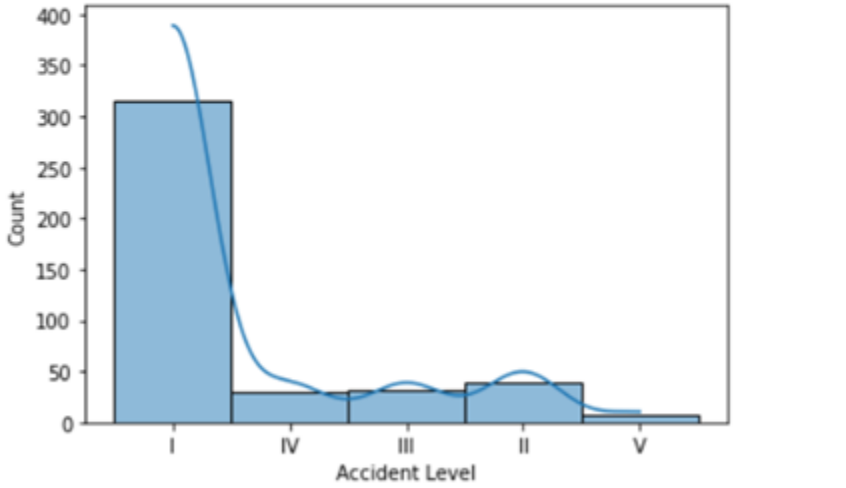
3.1 Post EDA Insights - Summary

Post getting the insights from Exploratory Data Analysis (EDA) about the dataset and the relationship amongst different features certain insights were derived to go about the pre-processing of specific features in the dataset.

- The insights can be summarized as:
  Data was found to be highly imbalanced.
- Some training features were also found to be highly imbalanced. Example: Genre column had a high imbalance with the majority of the values as male.
- The critical risk column acted as a column similar in its utility to the description column. The column appeared to be more of a textual rather than a categorical feature.

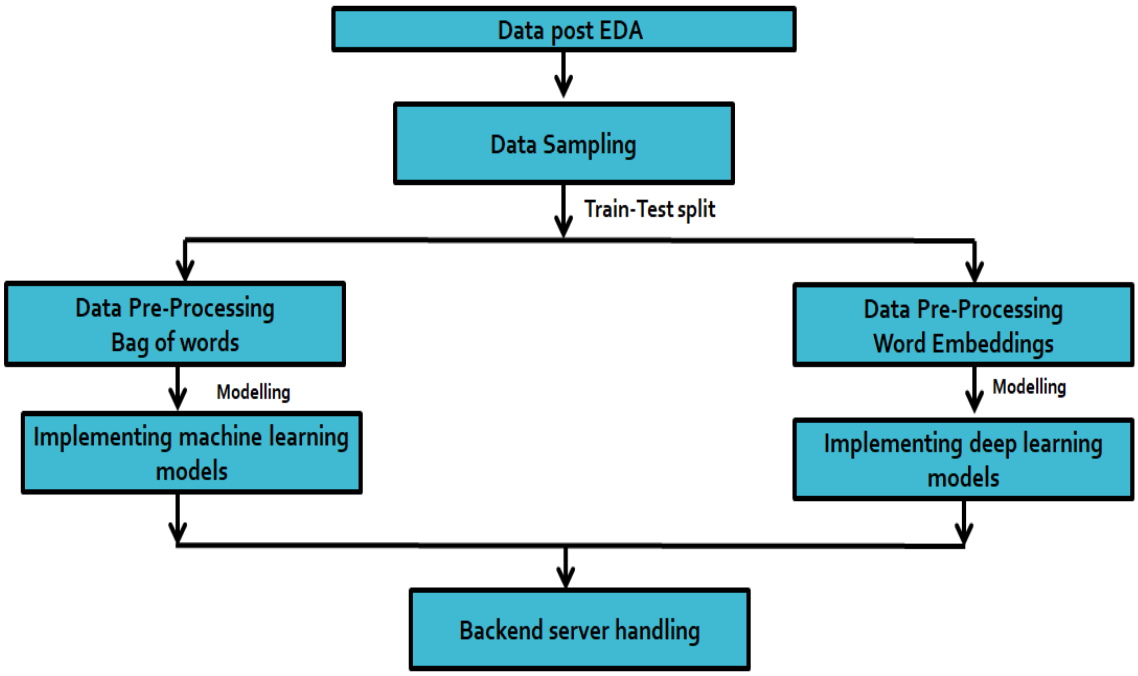
```
[ ]  sns.histplot(dataframe['Accident Level'],kde=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe954720940>
```



Representation of class imbalance of target column

3.2 Steps performed for data preparation and modeling



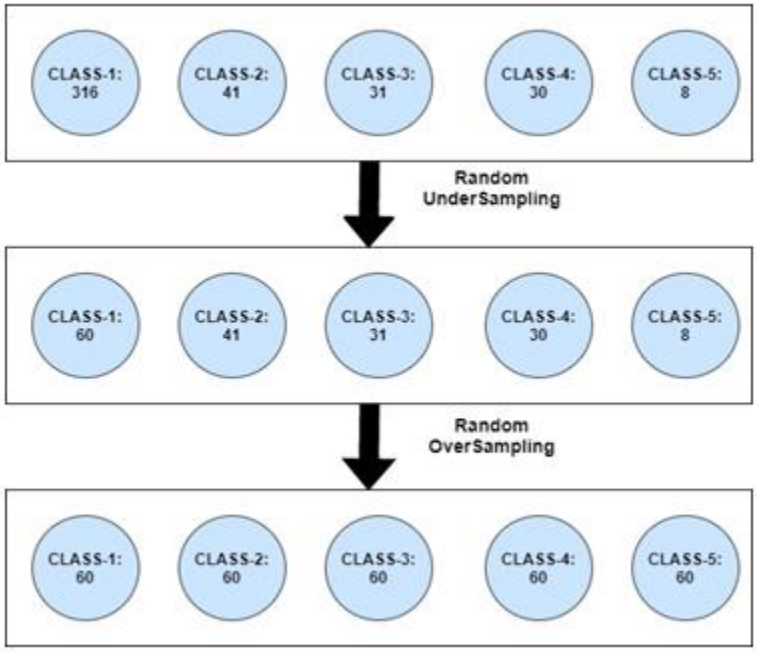
A description of the steps followed post-EDA can be summarized as follows:

- Collate data post-Exploratory Data Analysis process.
- Perform data sampling on the dataframe to handle class imbalance in the dataset.
- Split the data into train and test sets.
- Merge the Critical Risk column and Description Columns.
- Textual Data Pre-Processing (Machine Learning)
    - Implement TF-IDF vectorizer on the new Description column to create feature vector.
- Textual Data Pre-Processing (Deep Learning)
    - Create word embeddings using GLOVE
- Non-Textual Data Pre-Processing (Machine Learning and Deep Learning)
    - Label encoding for features such as Countries, Industry Sector
    - Ordinal encoding for features such as Potential Accident Level, Date of Incident
    - Binary encoding for feature Local

- Implement different machine learning and deep learning models.
- Create a dump of the trained models, encoders and a .ipynb file to execute trained models on new data provided to the file via chatbot interface. The same was shared with the interface team.

3.3 Sampling

Resampling methods are designed to change the composition of a training dataset for an imbalanced classification task.



Representation of sampling technique used

3.3.1 Need for sampling in the dataset

Due to the highly imbalanced classification of target variable with the majority of the values for Accident level feature being for class I, various resampling methods were applied to improve learning rate for minority classes and remove bias.

3.3.2 Techniques Implemented

Sampling technique used on the dataset is a conjunction of random under-sampling and random oversampling.
Other techniques tested were CNN (Condensed Nearest Neighbours), Tomek Link and SMOTE were also tried.

3.4 Data Pre-processing: Bag of Words – Machine Learning Model

A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms. The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents.

A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

- A vocabulary of known words.
- A measure of the presence of known words.

It is called a "bag" of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

There are two approaches that can be used for implementing the Bag-Of-Words model:

- **CountVectorizer**

CountVectorizer converts a collection of text documents to a matrix of token counts: the occurrences of tokens in each document. This implementation produces a sparse representation of the counts.

- **TF-IDF**

We can further transform a count matrix to a normalized tf: term-frequency or tf-idf: term-frequency times inverse document-frequency representation using TfidfTransformer. The formula that is used to compute the tf-idf for a term t of a document d in a document set is:

$$\text{tf-idf}(t, d) = \text{tf}(t, d) * log(\frac{n}{df(t) + 1})$$

when smooth_idf=True, which is also the default setting.

In this equation:

- tf(t, d) is the number of times a term occurs in the given document. This is same with what we got from the CountVectorizer
- n is the total number of documents in the document set
- df(t) is the number of documents in the document set that contain the term t

The effect of adding 1 to the denominator in the equation above is that terms with zero idf, i.e., terms that occur in all documents in a training set, will not be entirely ignored. At the end, each row is normalized to have unit Euclidean norm (by dividing l2 norm of itself).

The goal of using TF-IDF instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus.

The following parameters were set for the bag of words model:

1. Max_features (total number of feature vectors to be selected): - 2000
2. Max_df (ignore terms that have a document frequency strictly higher than the given threshold): -0.9
3. Ngram_range (Creating multiple n-grams of the given corpus) :- (1,3)

3.5 Data Pre-processing: Word Embeddings – Deep Learning Model

A word embedding is a learned representation for text where words that have the same meaning have a similar representation.
It is this approach to representing words and documents that may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems.

3.5.1 Word Embedding Algorithms

Word embedding methods learn a real-valued vector representation for a predefined fixed sized vocabulary from a corpus of text.
The learning process is either joint with the neural network model on some task, such as document classification, or is an unsupervised process, using document statistics.

- **Embedding Layer**

An embedding layer, for lack of a better name, is a word embedding that is learned jointly with a neural network model on a specific natural language processing task, such as language modeling or document classification.
It requires that document text be cleaned and prepared such that each word is one-hot encoded. The size of the vector space is specified as part of the model, such as 50, 100, or 300 dimensions. The vectors are initialized with small random numbers. The embedding layer is used on the front end of a neural network and is fit in a supervised way using the Backpropagation algorithm.
This approach of learning an embedding layer requires a lot of training data and can be slow, but will learn an embedding both targeted to the specific text data and the NLP task.

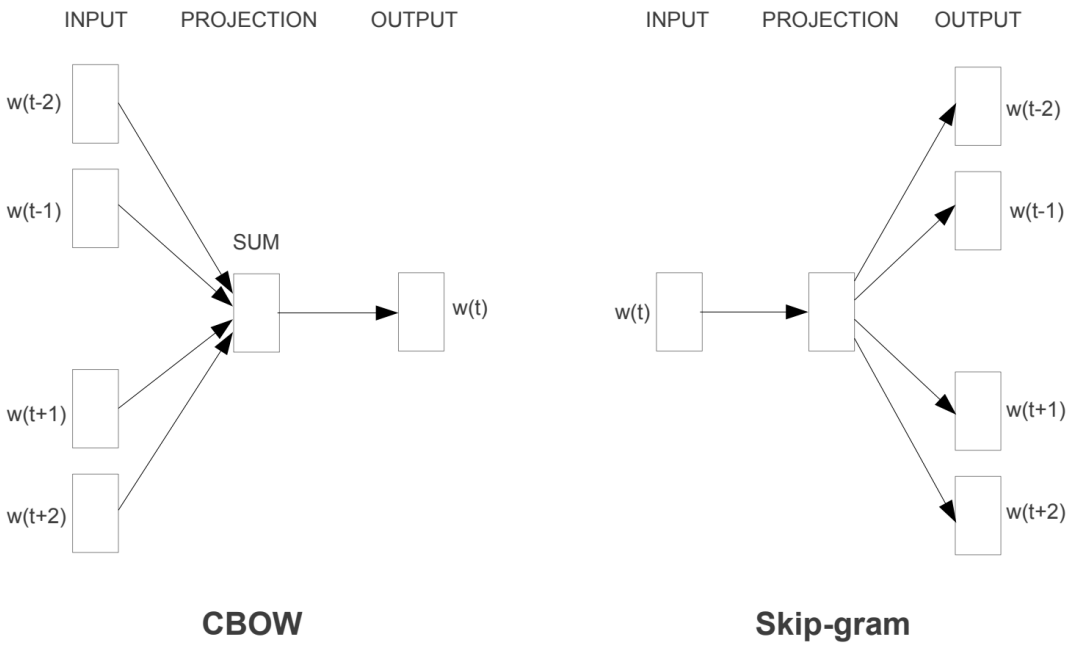
- **Word2Vec**

Word2Vec is a statistical method for efficiently learning a standalone word embedding from a text corpus.
It was developed by Tomas Mikolov, et al. at Google in 2013 as a response to make the neural-network-based training of the embedding more efficient and since then has become the de facto standard for developing pre-trained word embedding.

Two different learning models were introduced that can be used as part of the word2vec approach to learn the word embedding; they are:

- Continuous Bag-of-Words, or CBOW model.
- Continuous Skip-Gram Model.

The CBOW model learns the embedding by predicting the current word based on its context. The continuous skip-gram model learns by predicting the surrounding words given a current word.
The continuous skip-gram model learns by predicting the surrounding words given a current word.



Word2Vec Training Models

- **GloVe**

The Global Vectors for Word Representation, or GloVe, algorithm is an extension to the word2vec method for efficiently learning word vectors, developed by Pennington, et al. at Stanford.

Classical vector space model representations of words were developed using matrix factorization techniques such as Latent Semantic Analysis (LSA) that do a good job of using global text statistics but are not as good as the learned methods like word2vec at capturing meaning and demonstrating it on tasks like calculating analogies (e.g. the King and Queen example above).

GloVe is an approach to marry both the global statistics of matrix factorization techniques like LSA with the local context-based learning in word2vec.

Rather than using a window to define local context, GloVe constructs an explicit word-context or word co-occurrence matrix using statistics across the whole text corpus. The result is a learning model that may result in generally better word embeddings.

1. Feature critical risk and description were merged to create larger corpus.
2. Stop words, punctuations, emoticons etc. were removed during initial cleaning.
3. After initial cleaning all words were tokenized and padded.
4. Glove Embeddings technique was used to create Embeddings vectors for the textual data. It is done by creating an embedding layer at the start of the neural network architecture.

5. The following parameters were kept during embedding:
    1. Embedding Size: - 200
    2.Max Length: -137
    3.Max Features :- 30,000(approximately)

4. Data Modelling

The pre-processed and cleaned training data is then passed on to various machines learning models.

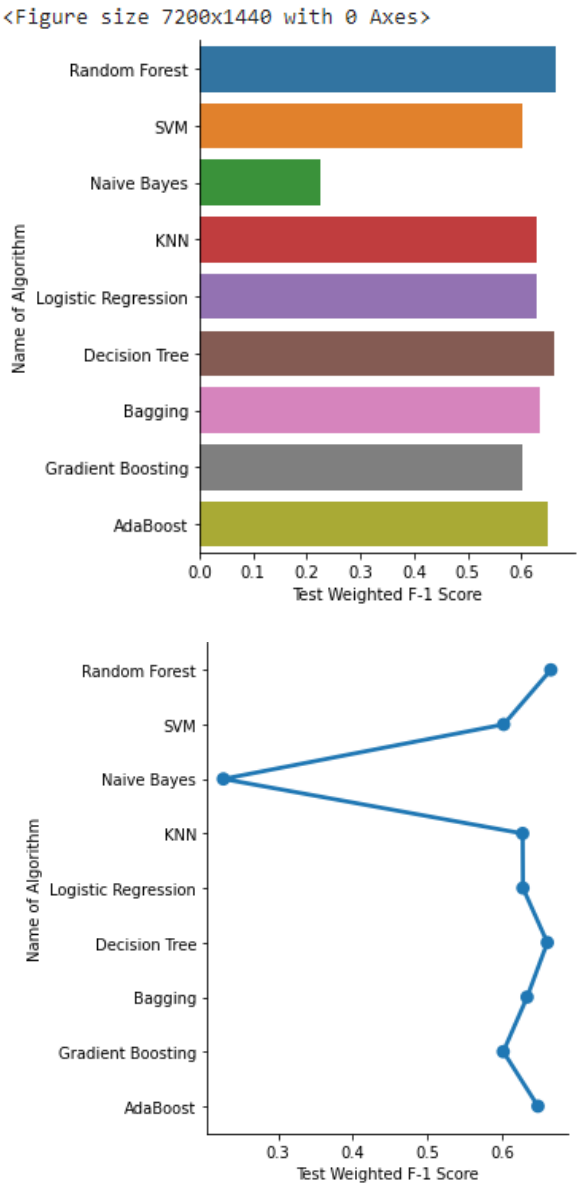
4.1. Models trained on dataset:

1. **Regression Based Algorithms**
    1. Logistic Regression
2. **Instance Based Algorithm**
    1.K-nearest Neighbour
    2.Support Vector Machine
3. **Decision Tree Algorithms**
    1.CART
4. **Bayesian Algorithms**
    1.Naïve Bayes
5. **Ensemble Algorithms**
    1.Bagging
    2.Random Forest
    3.Stacking and Voting Classifiers
6. **Boosting**
    1.Ada Boost
    2.Gradient Boost
    3.XG- Boost

4.2. Results and Analysis for different models

The results for the various models on the dataset are summarized as follows:

4.2.1 Imbalanced dataset (using the provided dataset as-is)



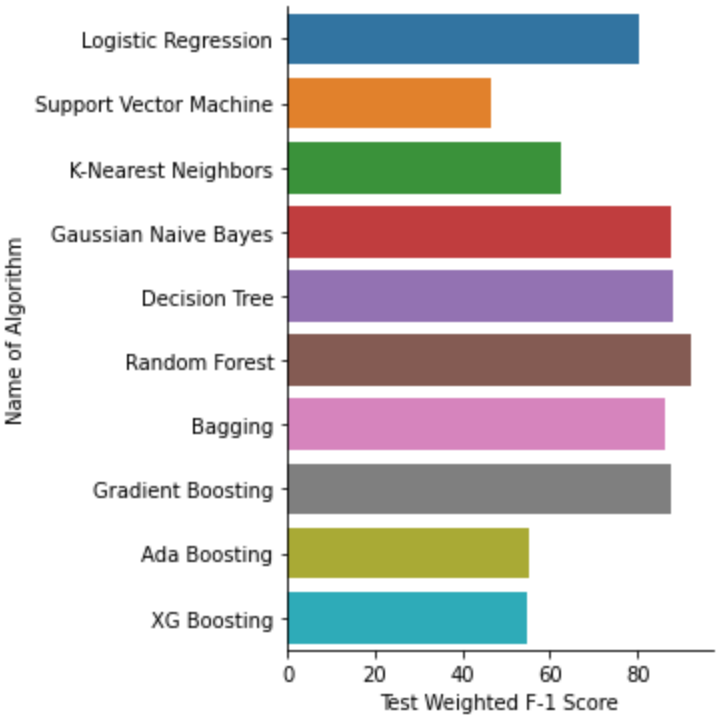
	Train Score	Test Accuracy Score	Test Weighted Precison Score	Test Weighted Recall Score	Test Weighted F-1 Score	Name of Algorithm
0	0.947788	0.738318	0.656802	0.738318	0.665493	Random Forest
1	0.644976	0.719626	0.517862	0.719626	0.602296	SVM
2	0.316556	0.177570	0.730708	0.177570	0.226310	Naive Bayes
3	0.717701	0.691589	0.594207	0.691589	0.627711	KNN
4	0.772040	0.682243	0.600569	0.682243	0.628126	Logistic Regression
5	0.993660	0.682243	0.648763	0.682243	0.660804	Decision Tree
6	0.912973	0.719626	0.566088	0.719626	0.633674	Bagging
7	0.993660	0.691589	0.532523	0.691589	0.601721	Gradient Boosting
8	0.993660	0.710280	0.623851	0.710280	0.648202	AdaBoost

4.2.2 Balanced dataset (using the dataset after data cleaning and pre-processing)

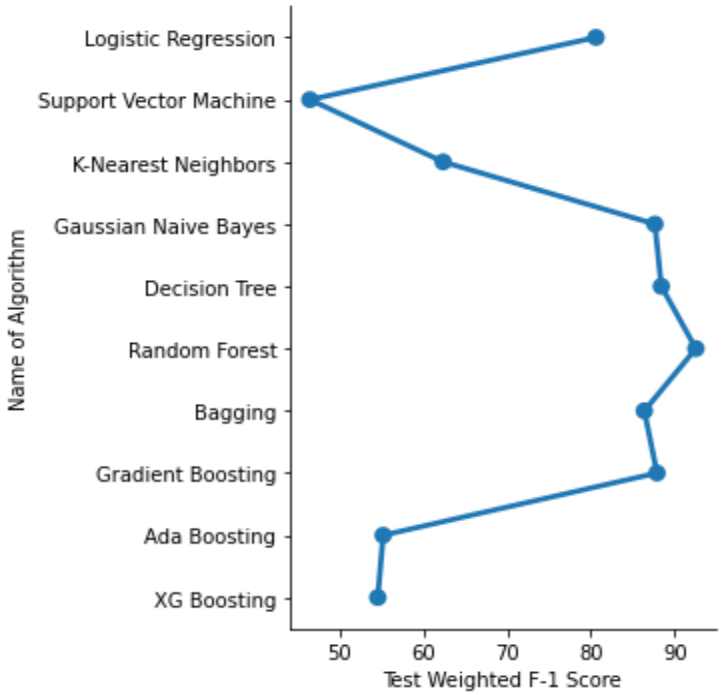
4.2.2.1 Balanced dataset without hyper-parameter optimization

	Train Score	Test Accuracy Score	Test Weighted Precison Score	Test Weighted Recall Score	Test Weighted F-1 Score	Name of Algorithm
0	0.936849	79.864502	82.580724	80.666667	80.600931	Logistic Regression
1	0.501307	50.975108	57.976523	50.666667	46.508201	Support Vector Machine
2	0.799328	61.515079	65.590933	63.333333	62.384651	K-Nearest Neighbors
3	0.982137	88.674459	89.501563	88.000000	87.622715	Gaussian Naive Bayes
4	0.986677	88.867244	89.755739	88.666667	88.392243	Decision Tree
5	0.986677	93.106061	94.113095	92.666667	92.509537	Random Forest
6	0.977738	86.954473	88.276190	86.666667	86.364874	Bagging
7	0.986677	88.287879	91.338228	87.666667	87.876013	Gradient Boosting
8	0.617484	57.032179	61.570013	57.333333	55.235352	Ada Boosting
9	0.617484	56.868543	61.476074	57.000000	54.594922	XG Boosting

<Figure size 7200x1440 with 0 Axes>

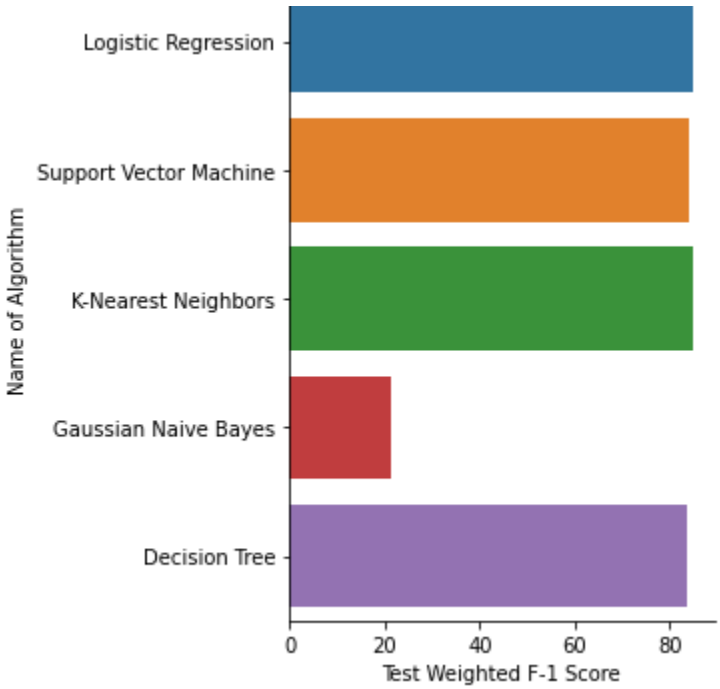


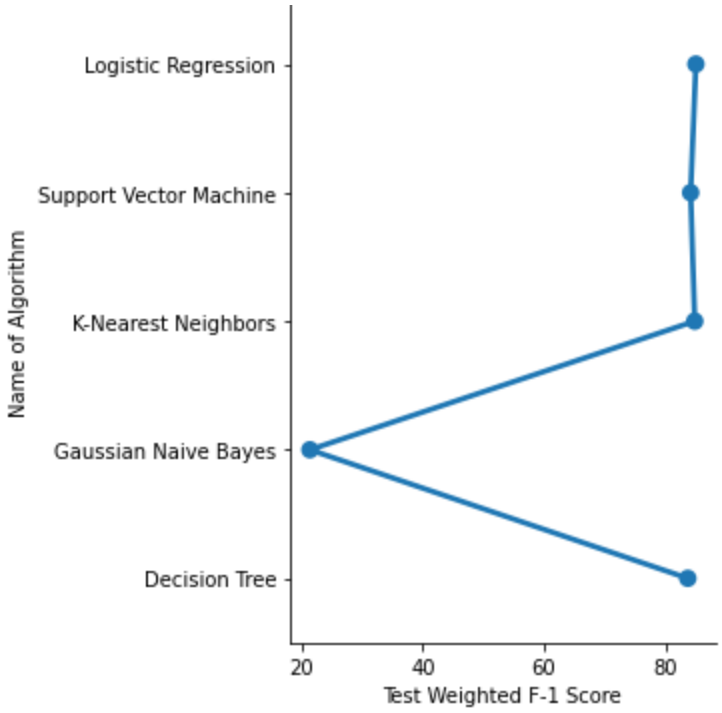
<Figure size 7200x1440 with 0 Axes>



4.2.2.2 Balanced dataset with hyper-parameter optimization

	Train Score	Test Accuracy Score	Test Weighted Precison Score	Test Weighted Recall Score	Test Weighted F-1 Score	Name of Algorithm
0	0.995558	85.408297	87.928175	85.333333	85.053126	Logistic Regression
1	0.995558	85.216234	86.675794	85.000000	84.187805	Support Vector Machine
2	0.995558	85.729798	87.097487	85.000000	84.863327	K-Nearest Neighbors
3	0.287015	34.254762	19.036267	31.000000	21.383886	Gaussian Naive Bayes
4	0.995558	83.771789	86.132744	84.000000	83.718136	Decision Tree




Logistic Regression
Support Vector Machine
K-Nearest Neighbors
Gaussian Naive Bayes
Decision Tree
Name of Algorithm
20
40
60
80
Test Weighted F-1 Score

4.2.3 Deep Learning model

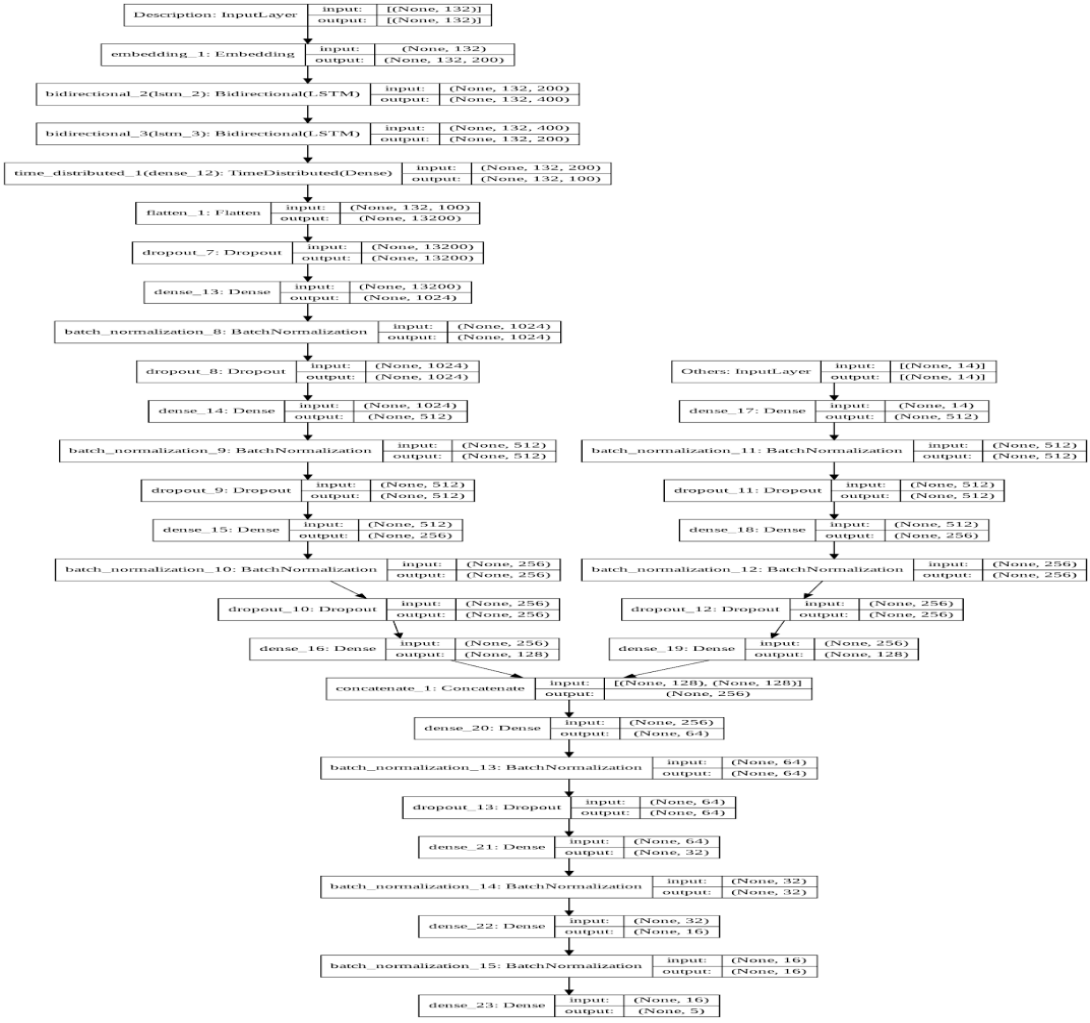
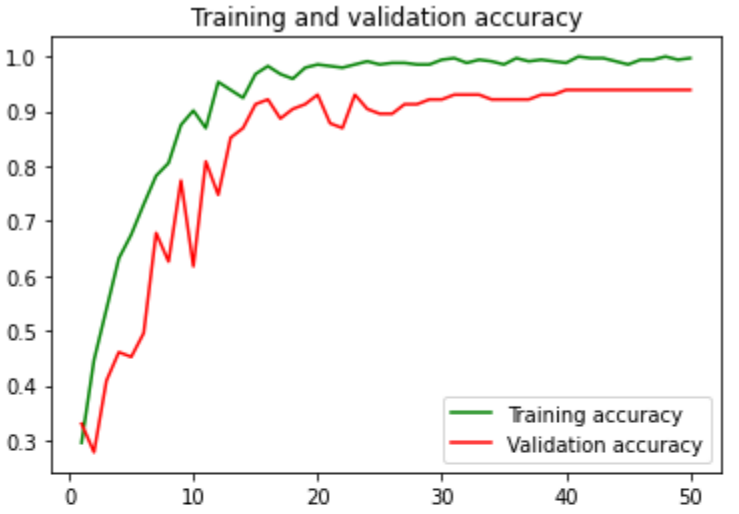
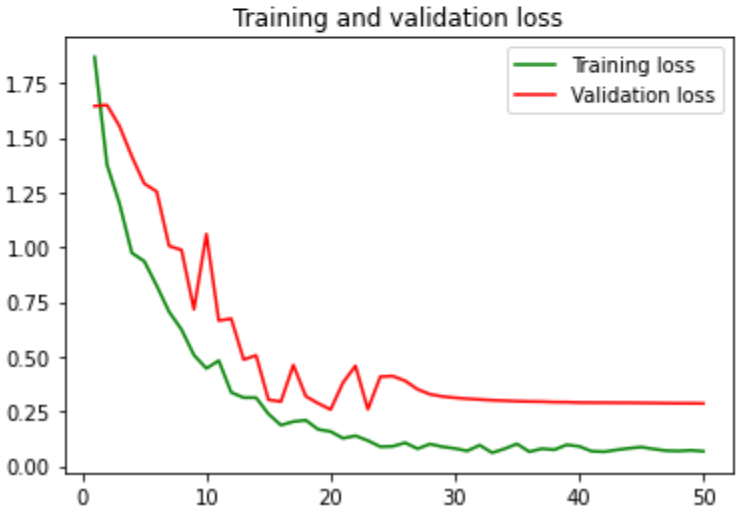


Fig: Model Architecture (File provided externally for viewing)

```
Epoch 00045: val_loss did not improve from 0.21495
Epoch 46/50
11/11 [==============================] - 27s 2s/step - loss: 0.0623 - accuracy: 0.9940 - val_loss: 0.2304 - val_accuracy: 0.9391

Epoch 00046: val_loss did not improve from 0.21495
Epoch 47/50
11/11 [==============================] - 27s 2s/step - loss: 0.0601 - accuracy: 0.9974 - val_loss: 0.2299 - val_accuracy: 0.9391

Epoch 00047: val_loss did not improve from 0.21495

Epoch 00047: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-09.
Epoch 48/50
11/11 [==============================] - 26s 2s/step - loss: 0.0638 - accuracy: 0.9962 - val_loss: 0.2291 - val_accuracy: 0.9391

Epoch 00048: val_loss did not improve from 0.21495
Epoch 49/50
11/11 [==============================] - 27s 2s/step - loss: 0.0752 - accuracy: 0.9886 - val_loss: 0.2282 - val_accuracy: 0.9391

Epoch 00049: val_loss did not improve from 0.21495
Epoch 50/50
11/11 [==============================] - 26s 2s/step - loss: 0.1048 - accuracy: 0.9722 - val_loss: 0.2280 - val_accuracy: 0.9391

Epoch 00050: val_loss did not improve from 0.21495
```

Fig: Accuracy in Final epoch





4.2.4 Best Model Selection and Model Dump Creation:

Based on the values for performance metrics obtained for different models, we observed that the random forest model gave best performance for the given dataset. For Model dump creation, encoders, vectorizers and the models were exported using pickle, and the best model was kept in the backend for deployment.

```
name=name+' Default'
path='/content/drive/MyDrive/NLP_Chatbot_Capstone_Project/Trained Default Machine Learning Classifiers/'
with open(path+'Model_'+str(name),'wb') as f:
  pickle.dump(model,f)
```

```
with open(path_cleaning_encoders+'/'+files[i] ,'rb') as f:
  encoder=pickle.load(f)
```

5. Safety Chatbots

A chatbot is a software application used to conduct an on-line chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent. Chatbots can understand human language and can process it and can interact with humans while performing specific tasks.

Chatbots have become extremely popular in recent years and their use in the industry has skyrocketed. They have found a strong foothold in almost every task that requires text-based public dealing.

5.1 Chatbots – A brief history

1966 Eliza
1972 Parry
1995 A.L.I.C.E
2001 SMART CHILD
2010 SIRI
2012 GOOGLE NOW
2015 ALEXA

5.2 Types of Chatbots

There are many types of chatbots available, a few of them can be majorly classified as follows:

- Text-based chatbot: In a text-based chatbot, a bot answers the user's questions via text interface.
- Voice-based chatbot: In a voice or speech-based chatbot, a bot answers the user's questions via a human voice interface.

The chatbot we designed for our project is of type Text-based.

5.3 Chatbot Design

Chatbots can be classified into different types, based on how they are built:

- Rule-based chatbot
- AI-based chatbot

5.3.1 Rule-based chatbot

Rule-based chatbot are pretty straight forward. They are provided with a database of responses and are given a set of rules that help them match out an appropriate response from the provided database. They cannot generate their own answers but with an extensive database of answers and smartly designed rules , they can be very productive and useful. The simplest form of Rule-based chatbots have one-to-one tables of inputs and their responses. These bots are extremely limited and can only respond to queries if they are an exact match with the inputs defined in their database.

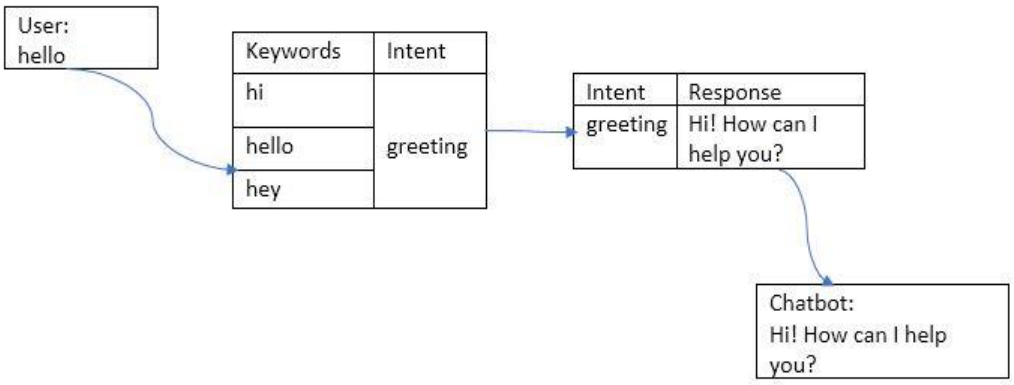
The chatbot we designed for our work is of rule-based kind.

5.3.2 AI-based chatbot

Using artificial intelligence, it has become possible to create extremely intuitive and precise chatbots tailored to specific purposes. Unlike the rule based chatbots, AI based chatbots are based on complex machine learning models that enable them to self-learn.

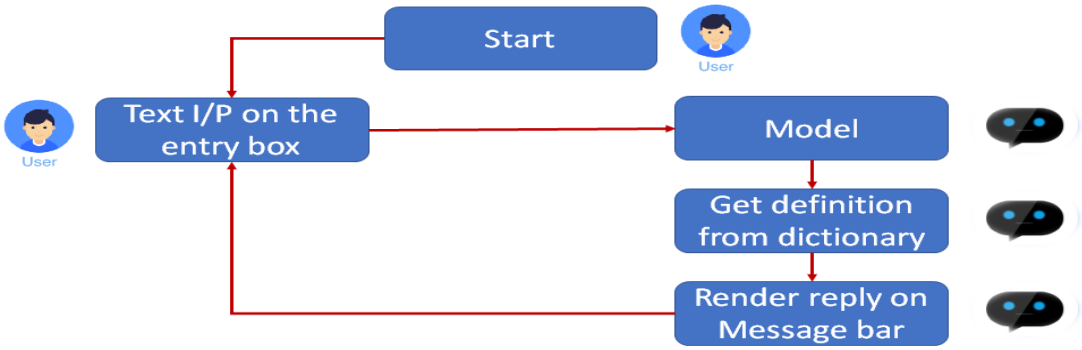
5.4 Chatbot System Overview

The rule based chatbot we designed works by searching for specific keywords in inputs given by a user. The keywords will be used to understand what action the user wants to take , ie the user's intent. Once the intent is identified, the bot will then pick out a response appropriate to the intent.



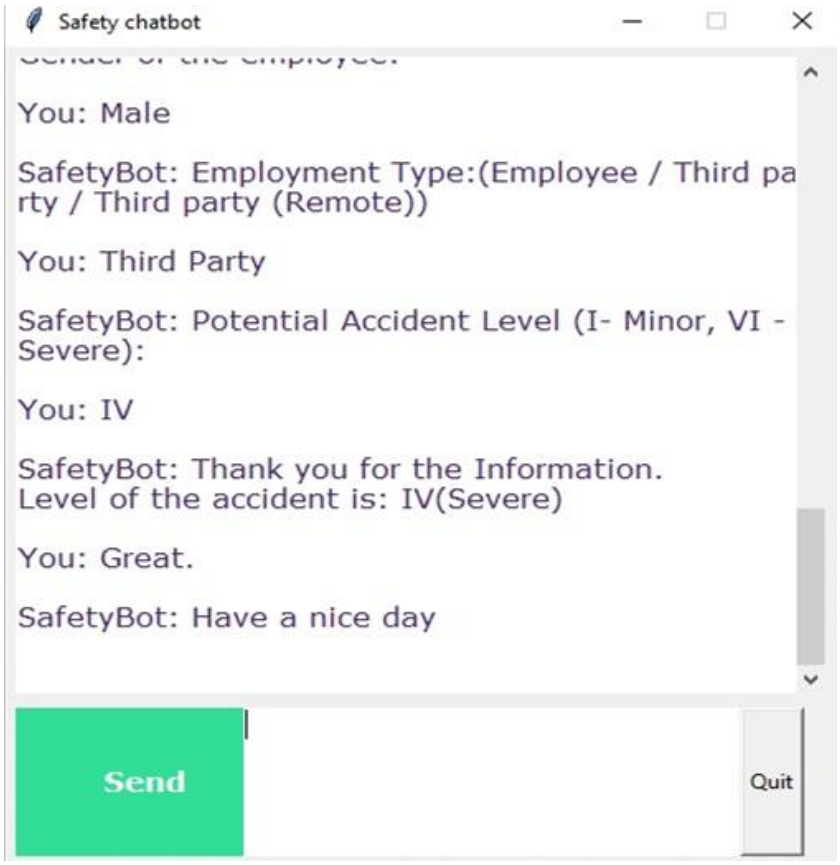
The list of keywords the bot will be searching for and the dictionary of responses will be built up manually based on the specific use case for the chatbot. A flow of how the chatbot would process the inputs is shown below:

5.5 Design flow of our safety chatbot



5.6 Tkinter GUI

Tkinter is a Standard GUI library for Python. Takes input from the user as string and processes the information and transfer the data to the backend for the prediction and displays the predicted output (Accident Level).



5.7 Future Improvements For Modelling and Chatbot

Possible Future Scope for modelling can be,

1. Using Attention Models and Transformers
2. Using BERT and XLNET
3. Bootstrap Aggregation with LSTM
4. Stacking and Voting with LSTM
5. Bayesian Hyper Parameter Optimization
6. Perform advanced Data Cleaning Techniques on Dataset
7. Perform advanced Feature Selection.
8. Perform NLG techniques to increase data.

Further improvements for chatbot could be to build a conversational chatbot which can converse with humans in interactive mode and demonstrate the severeness of the accident. Hence, highlighting the key factors which industry professionals could look upon.