TEXAS
The University of Texas at Austin

greatlearning

GREAT LAKES

Spock V1.0

The Chatbot – Powered By AI

AIML Batch Apr 20B – NLP Capstone Project

Program Manager: *Anurag Sah*

Project Mentor: *Aditya Bandaru*

Data Scientists Team

Satyanarayan Sai Veluganti

Amit Kumar Jain

Nitin Kumar

Pramod Kumar

Ajay Shenoy

Table of Contents

1 Acknowledgement

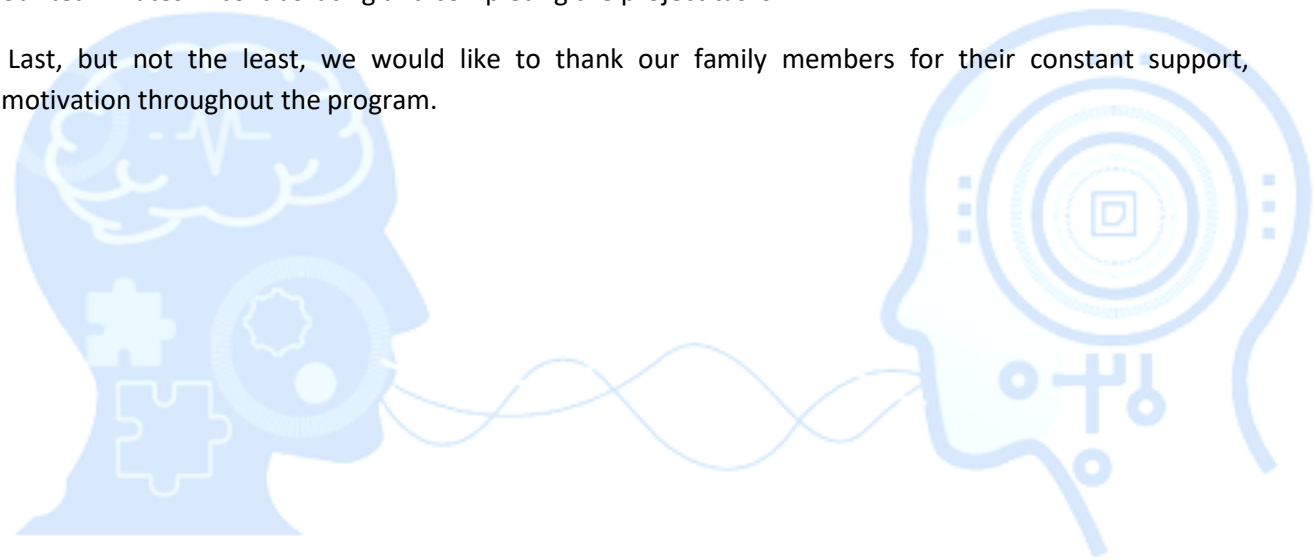
The satisfaction and excitement that accompany successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We express our most sincere and grateful acknowledgement to faculty members of all the courses in Great Lakes and TEXAS McCombs the university of Texas at Austin.

We would like to sincerely thank our project mentor, **Mr. Aditya Bandaru** for his expert guidance, sharing valuable insights and persistent encouragement throughout the project. We would like to thank our Program Manager, **Mr. Anurag Sah** for her encouragement and valuable support in our endeavor.

We would also thankful to all the team members of **Great Learning** for providing us with the required facilities and support towards the completion of the project. We are also extremely thankful to each of our teammates in collaborating and completing the project tasks.

Last, but not the least, we would like to thank our family members for their constant support, motivation throughout the program.



2 Abstract

It is an urgent need for industries/companies around the globe to understand why employees still suffer some injuries/accidents in plants. Sometimes they also die in such environment. Better tools need to be deployed to provide such analysis at run time to the industries which are involved in high risk like mining and metal.

Leveraging smart automations and Artificial Intelligence helps to take some guesswork out of such analysis and also able to reduce the time drastically. For business owners who want better support at faster pace are best suited to deploy such tools in their environment. The use of Chatbot evolved rapidly in numerous fields in recent years, including Marketing, Supporting Systems, Education, Health Care, Cultural Heritage, and Entertainment. Though the usage is still low in industrial safety, but the need is huge and if good solutions are provided, business is ready to adopt them provided the huge benefits that business gets out of this.

Conversational Chatbot are one of such artificial intelligence tools which can fill in the gap of providing such support services where human intervention is both risky and costly.

The intent of this project is to build a conversational Chatbot which can handle user queries based on machine learning models. User shall be able to access the Chatbot 24x7 through various channels. However, for the scope of this project, we have limited it text based interaction using Chatbot interface.

Apart from general interactions, Chatbot will be able to provide the critical risk based on the user inputs like information related to the country, local/city, industry, accident level, gender and most importantly the free text description. Using Natural language processing techniques this data will be processed and feed into the machine learning model. This model will provide the prediction based on the inputs.

The dataset will keep growing with every passing accident and prediction will improve over a period of time.

The database comes from one of the biggest industries in Brazil and in the world. This document covers in detail exploratory analysis on the dataset. This document also captures in detail approach and steps of data pre-processing, feature engineering, model building and predictions, Chatbot user interface and deployment.

3 Introduction

One of the key objectives of Industrial Safety 'IT Division' is to provide 24 x7 supports to users working in hazardous industries like mining and metal industries. This includes identification, categorization and providing the 'critical risk' to the users based on the description user provides of the accidents. This information should be provided to users run time round the clock.

Accidents can happen any time any place, hence the support should be available whenever needed. The solution provided in this document is scalable, flexible and highly available.

During such accidents the people involved are already in distress situation and hence human like interaction without losing patience and showing frustration is needed.

Chatbot are excellent solution to handle all the above problems and can provide human like support round the clock with all the patience at low cost.

Chatbot is a tool to retrieve information and generate humanlike conversation. It is mainly a dialog system aimed to solve/serve a specific purpose. The basic problem that these bots try to solve is to become an intermediary and help users become more productive. They do this by allowing the user to worry less about how information will be retrieved and about the input format that may be needed to attain specific data. Bots tend to become more and more intelligent as they handle user data input and gain more insights from it. Chatbot are successful because they give you exactly what you want. The Chatbot has the capability to know this information already and is intelligent enough to retrieve what is needed when you ask it in your own language or in what is known in computer science as Natural Language.

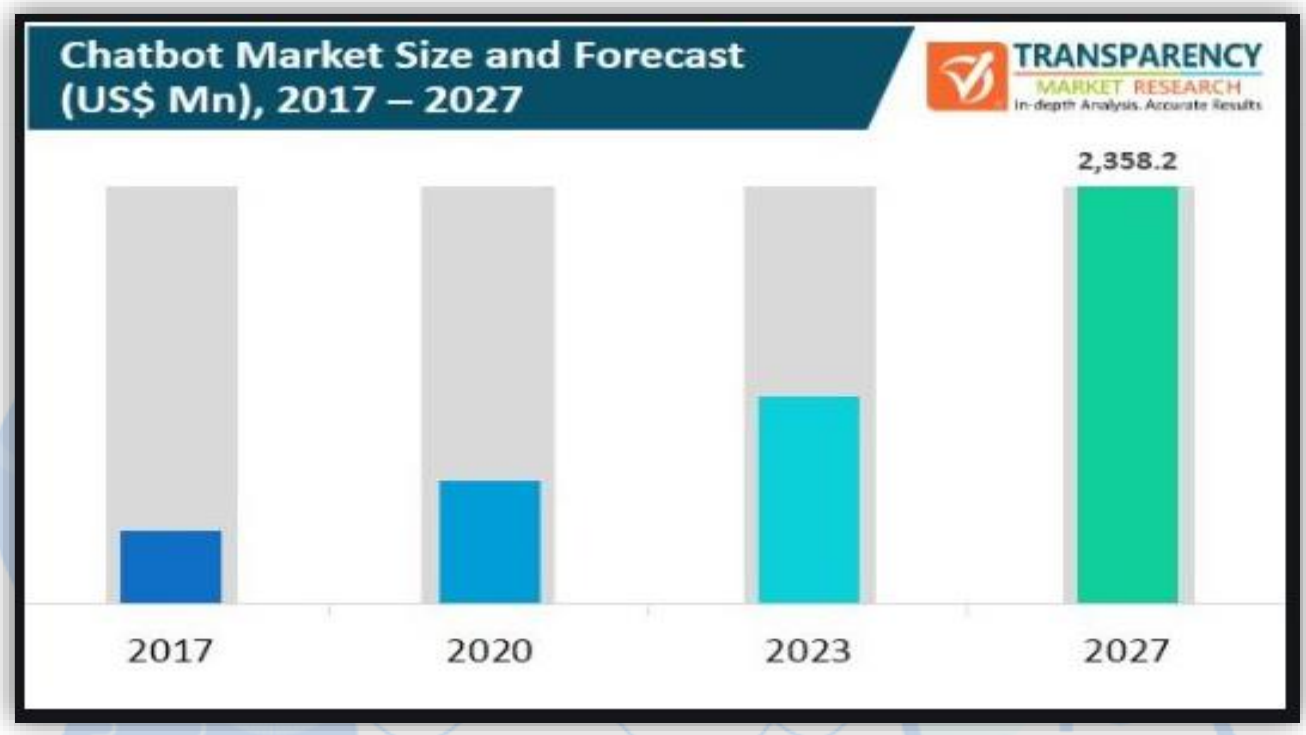
The reason Chatbot get an edge over traditional methods of getting things done online is you can do multiple things with the help of a Chatbot. It's not just a Chatbot; it's like your virtual personal assistant. You can think of being able to book a hotel room on booking.com as well as booking a table in a nearby restaurant of the hotel, but you can do that using your Chatbot. Chatbot fulfill the need of being multipurpose and hence save a lot of time and money. In this document we are going to learn how to build natural conversational Chatbot using Natural Language Processing (NLP) and how to teach a Chatbot to understand our natural language and make it do tasks for us from a single interface.

History of Chatbot

Reference link: *https://www.mygreatlearning.com/blog/basics-of-building-an-artificial-intelligence-chatbot/*

Popularity of Chatbot and growing market opportunities

Chatbot have become popular just as anything from the recent past. Let's try looking at figure below, which depicts the rise of Chatbot, and also try to understand why there is a huge demand for building Chatbot. As per Transparency market Research Chatbot market is expected to be 2Billion USD by 2027.



Reference Link: https://www.transparencymarketresearch.com/chatbot-market.html

4 Business Value Preposition

Chatbot are generally an organization's first foray into using artificial intelligence. While most organizations may not have large budgets to implement AI in edge cases like self-driving cars and robotics, Chatbot are a great alternative to explore an innovative medium to help drive business outcomes.

Mining and Metal are the most hazardous industries to work and safety and security of the workers deployed in these industries has been ongoing concern of the business owners in these industries. Now that the industry has had three years to explore what the technology has to offer, Chatbot are now providing more clarity than ever on the value that they can offer. Industrial Safety is also approaching to implement the Chatbot into their environment. Diagram below shows the benefits that business achieves by implementing Chatbot in industrial safety and security.

Reduction in Cost of Support

Companies' need for growing the safety and security department can be managed by rolling out increasingly capable bots handling more and more complex queries. The implementation of Chatbot will create a certain amount of investment costs. However, this cost can be lower compared to consumer service salary, infrastructure, and education. Except for the implementation of investment costs, the extra costs of Chatbot are quite low. These items can be topics such as ensuring Chatbot security and improving it. But it is not optimistic to think that the costs will decrease considering the long term.

Improved User Experience

Users will have the ability to interact frictionless with the company, making find answers to their concerns easier and more immediate. While support provider teams and users at the mining and metal

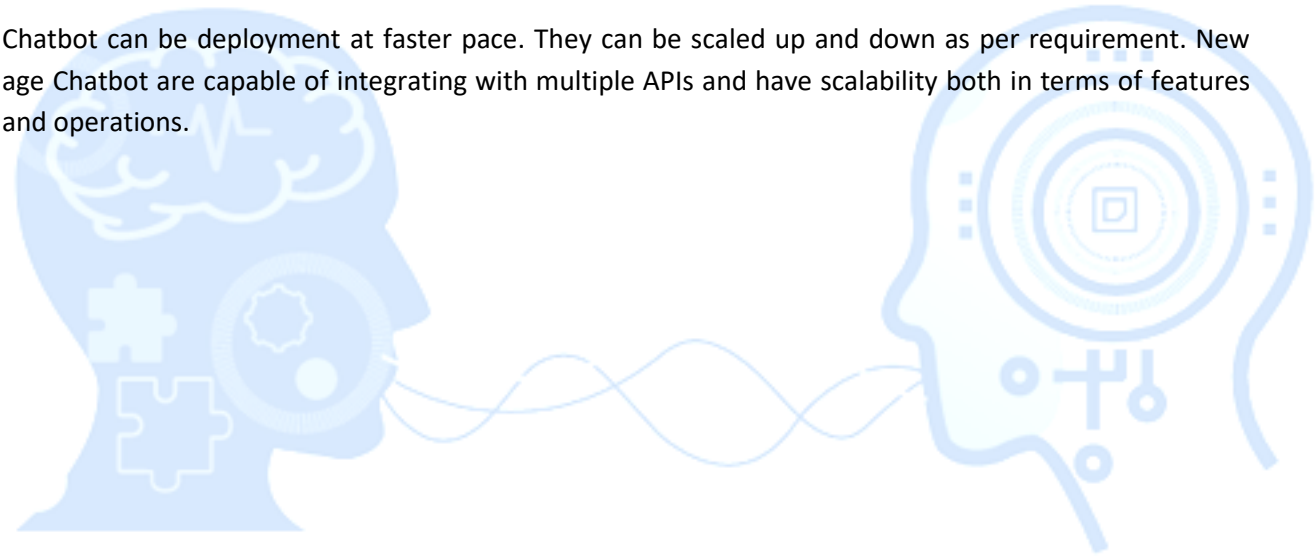
industry production platform sometimes lose their patience, that's something bots are yet incapable of. The impatience of the representative and the consumer during the solution of a problem is one of the human-related failures. The representative is expected to be more patient as much as possible so that the company can keep consumer satisfaction high. Chatbot can show the patience that no human can provide. At this point, a human-sourced consumer service problem can be resolved directly. New age Chatbot with conversational ability using natural language understanding provide human like interactions and improves the user experience

24x7Availability

According to studies, all of users expect support to be available 24/7. Waiting for the next available operator for minutes is not a solved problem yet, but Chatbot are the closest candidates to ending this problem. Maintaining a 24/7 response system brings continuous communication between the user and the safety and security support provider teams.

Faster Deployment

Chatbot can be deployment at faster pace. They can be scaled up and down as per requirement. New age Chatbot are capable of integrating with multiple APIs and have scalability both in terms of features and operations.



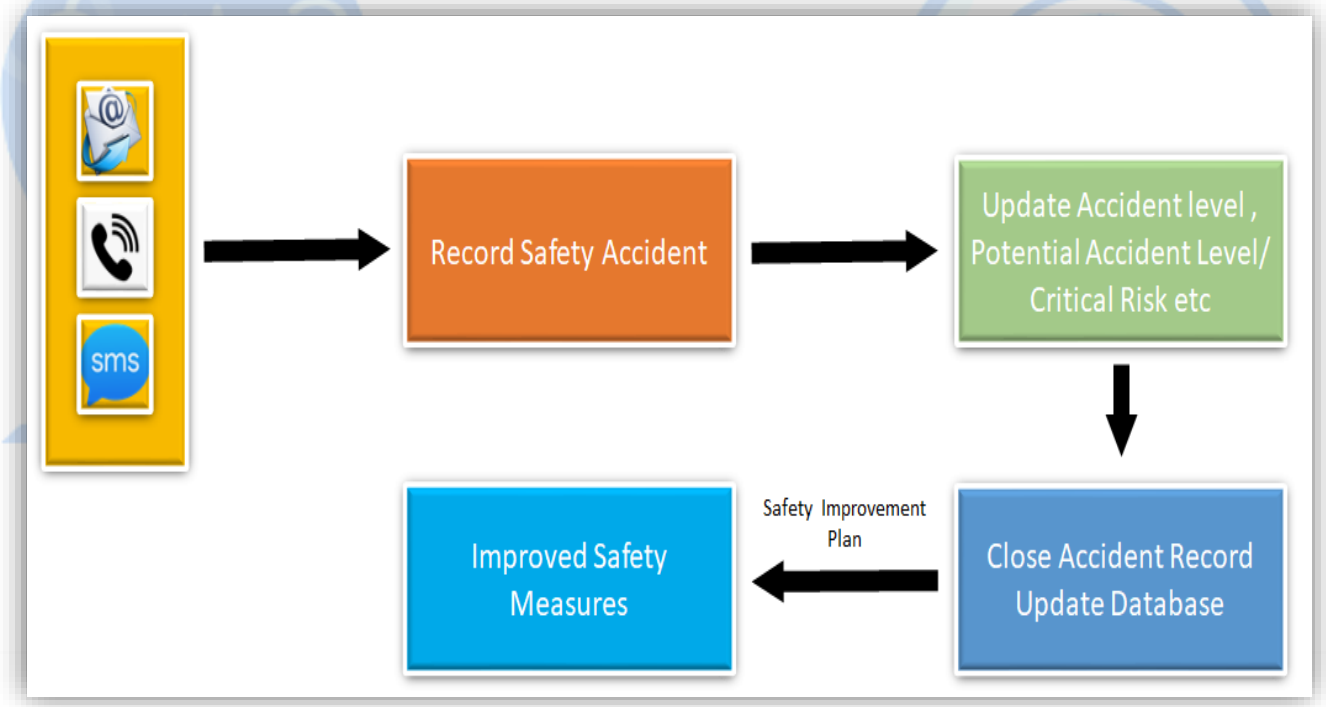
5 Business Process Overview:

5.1 Existing Business Process

At a high level, in the mining and metal industry, user logs the ticket vide different channels like email, phone or SMS to the safety department. User shares all relevant information in predefined format. Based on the information provided by user, safety department record the accident and analyze the underlying critical risk of the accident with accident and potential accident levels. They update the safety database and also implement the safety improvement plan based on the critical risk. This is how the safety is improved on ongoing basis.

However, in this process user does not get any details of the underlying critical risk details which he can use to minimize the impact of the accident. These details come to the user department only at the time when safety improvement plans are being implemented.

Figure below provides high level overview of the **existing** business process for recording Safety Accidents in the Industry and implementing safety improvement plans.



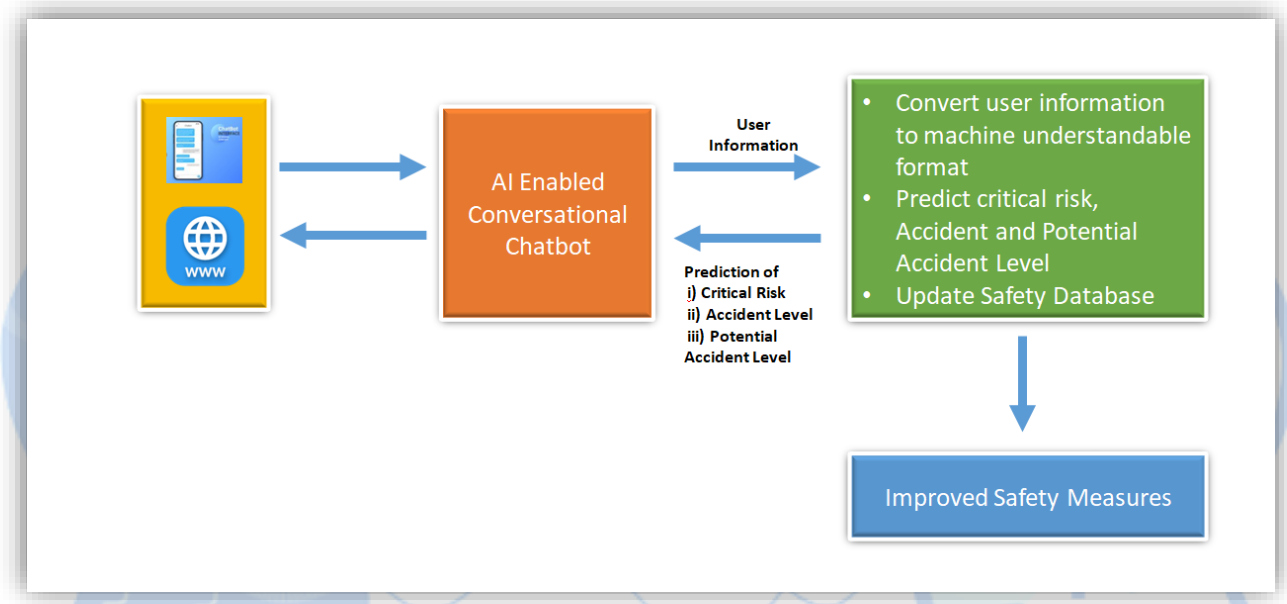
As said above, there are many limitations in the current system which prevents users to get timely support and provide inputs to prevent accidents / decrease of impact of such accidents.

5.2 Proposed Business Process

In the proposed business process, it is envisioned to provide information of the underlying critical risk to the user department at run time on immediate basis and round the clock. This means, safety department will need to provide such support round the clock and train its staff to do the analysis so

that immediate response can be provided to user department. This will need huge cost of manpower and training to implement this revised process. It is proposed to use Artificial Intelligence based solution to be implemented for this purpose which will not require manpower round the clock and will work at its own level. Artificial intelligence based Chatbot provides human like support round the clock at the faster pace.

In this system, Chatbot will be implemented and provided to users vide different channels viz. web page or mobile application, that they can use to interact with the Chatbot. Chatbot in turn will provide the critical risk, accident level and potential accident level to end user on immediate basis. **As Chatbot are digital machines and hence available round the clock.**

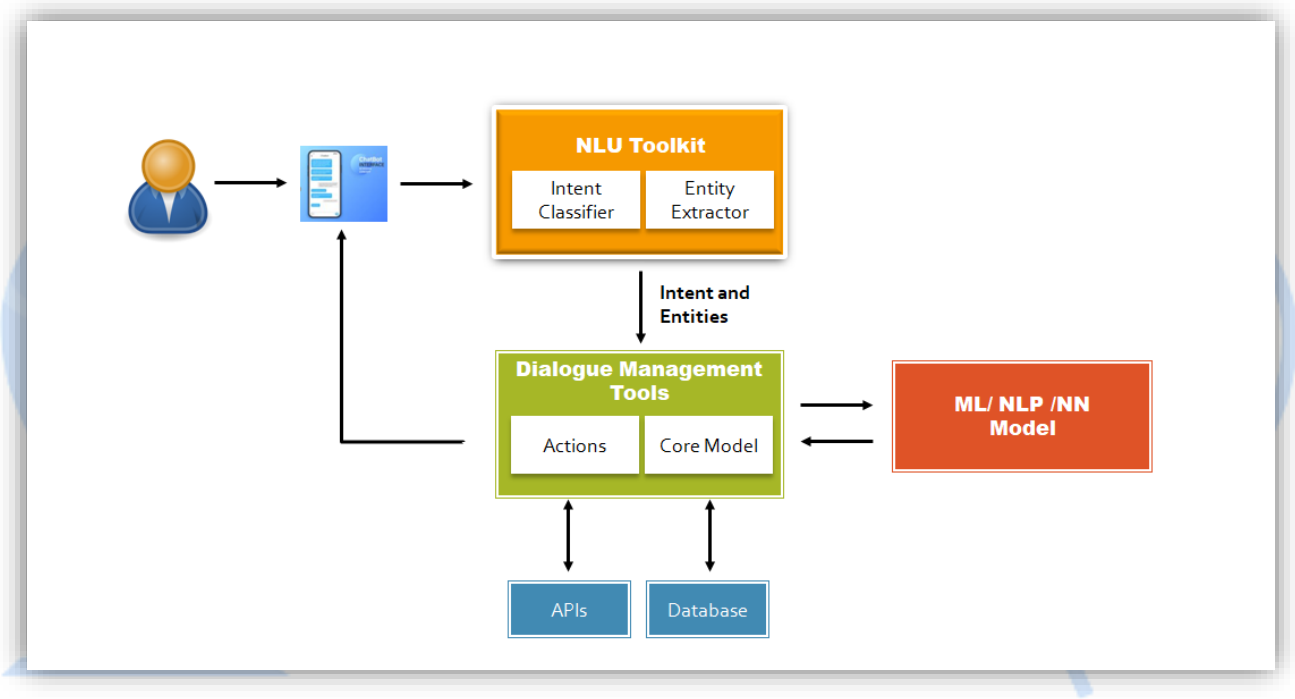


6 High level Architecture of Chatbot

Chatbot architecture consists of the following:

- Chat window/Session/Front end application interface
- Natural Language Processing [NLP] Toolkit
- Dialogue Management Tools
- Models based on Machine Learning, Neural network, Natural Language Processing
- Application Programming Interface (API)
- Database

Please refer the below figure to understand the architectural interface



6.1.1 Chatbot Interface

Chatbot interface is the user interface which is being published to user via multiple channels. In the current project, we plan to publish the user interface using python “Tkinter” libraries which will be integrated on a HTML page.

6.1.2 NLU Toolkit

Natural Language Processing is the important library which is being used to develop conversational Chatbot. It helps user to provide their input in user understandable language. NLP component extracts the intent and entities from the conversation and plan for actions and Chatbot responses based on the user inputs. This is a paradigm shift from old system where user had to inform safety helpdesk and fill up a form to provide the inputs. In addition to this, user also provides free text description of accidents which helps the Chatbot for better predictions.

6.1.3 Intent

Intents are used to define what business want a Chatbot to respond with when it picks up the intention of a user, or when Chatbot want to trigger a response based off of some other event.

For example, if a user says 'Hi', we want Bot to respond with 'Hello' / 'Hi'.

Here the intent of customer is "Greetings" and Chatbot understands the user intent as greetings.

6.1.3.1 Entities

Entities are knowledge repositories used by the Chatbot to provide personalized and accurate responses. With Entities we can easily extract important information from the ongoing conversation, such as country, gender, Industry type or anything we want. Use them when we want our Chatbot to catch important data.

6.1.4 Dialogue Management Tools

Chatbot are known for their human-like conversational abilities. To generate better user experiences, businesses have been working hard to make human Chatbot conversations more humane. Dialogue management is the crucial aspect that makes Chatbot to conduct contextual communications.

Here Chatbot reads the input from user; parse the 'pattern' of the user and finds the 'intent' from that pattern based on the NLP based neural network model on which intent/response model is built. Based on the intent, response is predicted and sent back to user. Here the predefined pattern in the model may not always match the user inputs, but due to the advanced NLP based models, it identifies the intent and response as per prediction.

6.1.5 Models (ML/NN/NLP)

We have used models at three places:

- At first place, it is used to predict the response of Chatbot to user based on the intent
- Secondly, the named entity recognition to identify the named entities
- Lastly, model is used to predict the critical risk, accident level & Potential accident Level of an accident based on the information provided by the user

6.1.6 Application programming Interface (API)

Once the model is trained, it is saved to publish using on a **HTML** page using F**lask**. Then interface is being used to initialize the Chatbot on a single click where user can provide inputs related to the accident and get the immediate response.

6.1.7 Database

Pandas data frame is being used to save runtime dataset and offline it is exported in csv files, it will help analyzing the different type of user inputs and thereby fine-tuning in model for better accuracy on preidction

6.1.8 Chatbot Window and Session

Chatbot window will be an input window where user will put the details including the description of the accident and user will get the output of Critical Risk, accident level and Potential accident level.

7 Import the Data

For this project we have received dataset from the below link:

https://www.kaggle.com/ihmstefanini/industrial-safety-and-health-analytics-database

Dataset filename:

"IHMStefanini_industrial_safety_and_health_database_with_accidents_description.csv"

7.1 Import the file

Dataset is imported using pandas read_csv command. The dataset is loaded to pandas dataframe for further analysis.

```
safety_df=pd.read_csv('IHMStefanini_industrial_safety_and_health_database_with_accidents_description.csv')
```

7.2 Check first few rows of DataFrame

```
safety_df.head()
```

	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...

7.3 Check the Data types of different attribute of Data Frame

```
safety_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 425 entries, 0 to 424
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                425 non-null    int64
 1   Data                      425 non-null    object
 2   Countries                 425 non-null    object
 3   Local                     425 non-null    object
 4   Industry Sector           425 non-null    object
 5   Accident Level            425 non-null    object
 6   Potential Accident Level  425 non-null    object
 7   Genre                     425 non-null    object
 8   Employee or Third Party   425 non-null    object
 9   Critical Risk             425 non-null    object
 10  Description               425 non-null    object
dtypes: int64(1), object(10)
memory usage: 36.6+ KB
```

It is observed that except attribute " Unnamed: 0" all others have data type as object which means they contain text values.

7.4 Checking the Shape of Data frame

```
safety_df.shape
```

```
(425, 11)
```

7.5 5 Point Summary

```
safety_df.describe(include='all')
```

	Unnamed: 0	Data	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party	Critical Risk	Description
count	425.000000	425	425	425	425	425	425	425	425	425	425
unique	NaN	287	3	12	3	5	6	2	3	33	411
top	NaN	2017-02-08 00:00:00	Country_01	Local_03	Mining	I	IV	Male	Third Party	Others	During the activity of chuteo of ore in hopper...
freq	NaN	6	251	90	241	316	143	403	189	232	3
mean	224.084706	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	125.526786	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	118.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	226.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	332.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	438.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

8 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is employed to analyze and summarize the main characteristics of the safety dataset, it's the process of performing initial understanding on data so as to discover patterns, to spot anomalies, check assumptions with the help of summary statistics and graphical representations. During of the data, several techniques are implemented including basic feature analysis to better understand the dataset provided.

- Dataset consist of 425 records and 11 attributes.
- No null values in the Dataset
- 10 attributes are of object type and 1 integer

8.1 Dataset Description

The given dataset consists of the attributes related to safety incidents reported by the users, along with description of the accidents with information of Country, Location, and Accident Level etc. Below are the complete details of all 11 attributes:

1. **Unnamed** : 0 : contain values from 0 to 438
2. **Data**: timestamp or time/date information of the accident
3. **Countries**: which country the accident occurred (3 Different Countries)
4. **Local**: The city where the manufacturing plant is located (12 Different Locations)
5. **Industry sector**: which sector the plant belongs to (3 Sectors, Mining, Metals and Others)
6. **Accident level**: from I to VI, it registers how severe was the accident (I means not severe but VI means very severe)
7. **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)
8. **Genre**: The injured person is male of female
9. **Employee or Third Party**: The injured person is an employee or a third party
10. **Critical Risk**: Category / Type of the risk e.g. Manual Tools, Pressed etc.
11. **Description**: Detailed description of how the accident happened

8.2 Attribute "Unnamed: 0"

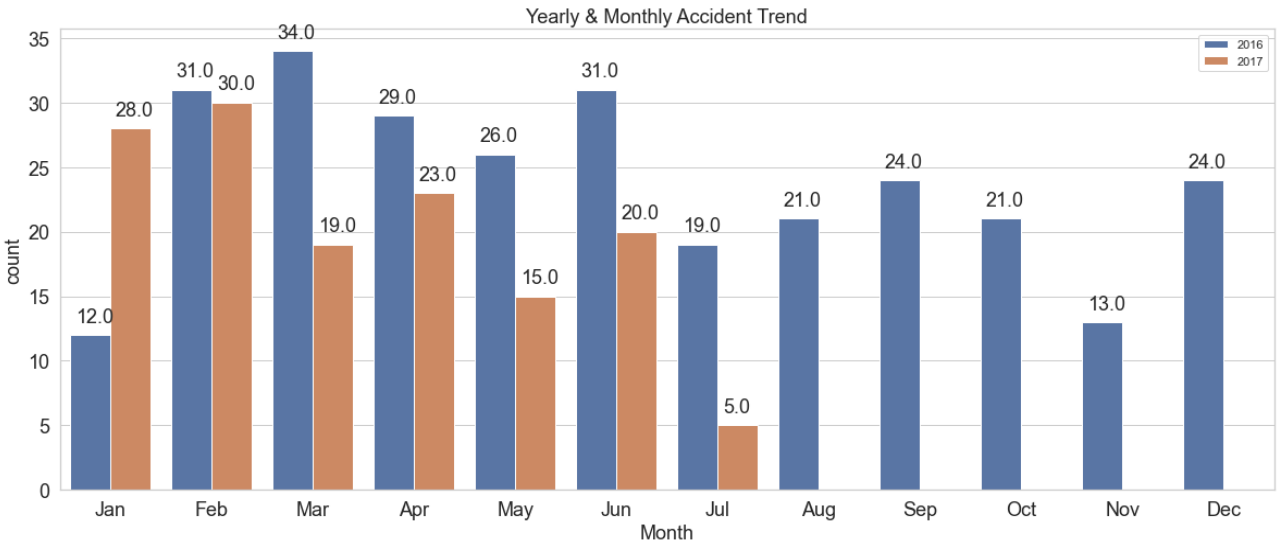
The attribute "Unnamed: 0" has values e.g. 1, 2, 3 starting from 0 to 438. it has no duplicate values, however the number are not in sequence which is observed in the bottom 10 rows e.g. index 415 having a corresponding value as 429 and last index 424 having a corresponding values as 438. it looks more to be a index columns without proper indexing. Therefore we will drop this attribute as it will not add any values in the analysis

8.3 Attribute "Data"

This attribute provides information about the time when the accident information was logged.

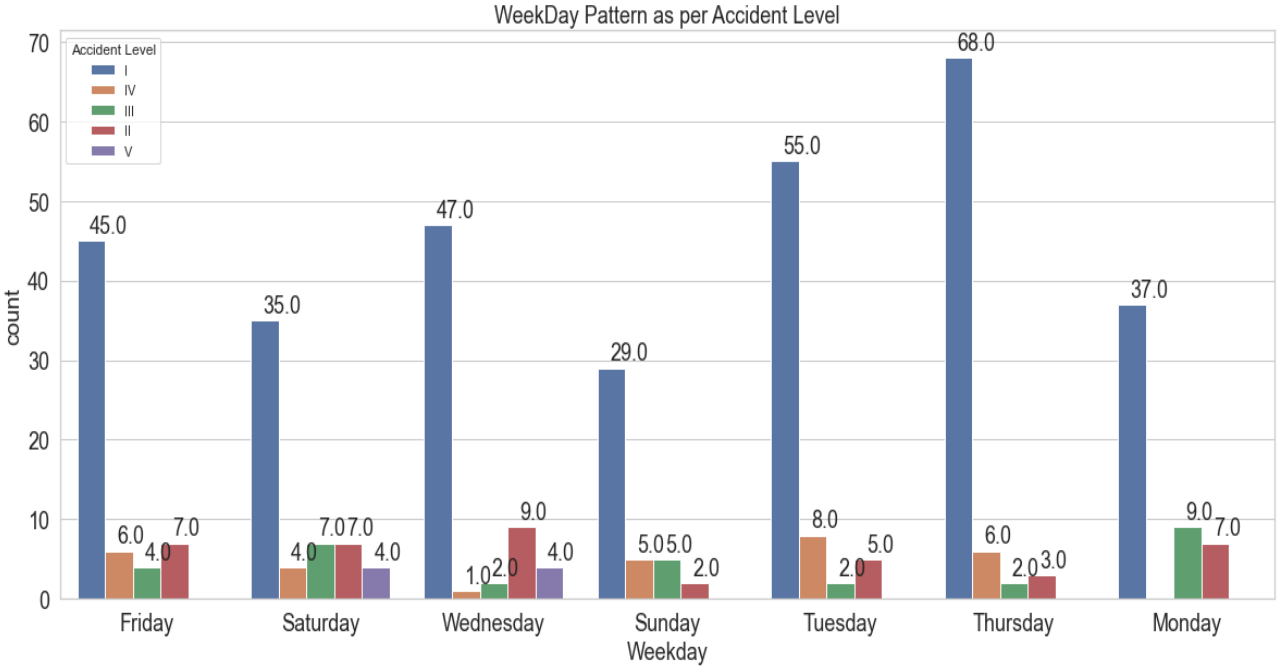
Monthly Trend: We observe the following:

- During Mar 2016 maximum accidents were logged followed by Feb 2016 & Jun 2016
- In 2017 less accidents were recorded in specific month compare to 2016 except Jan 2017
- Data is recorded only for first 7 months in 2017



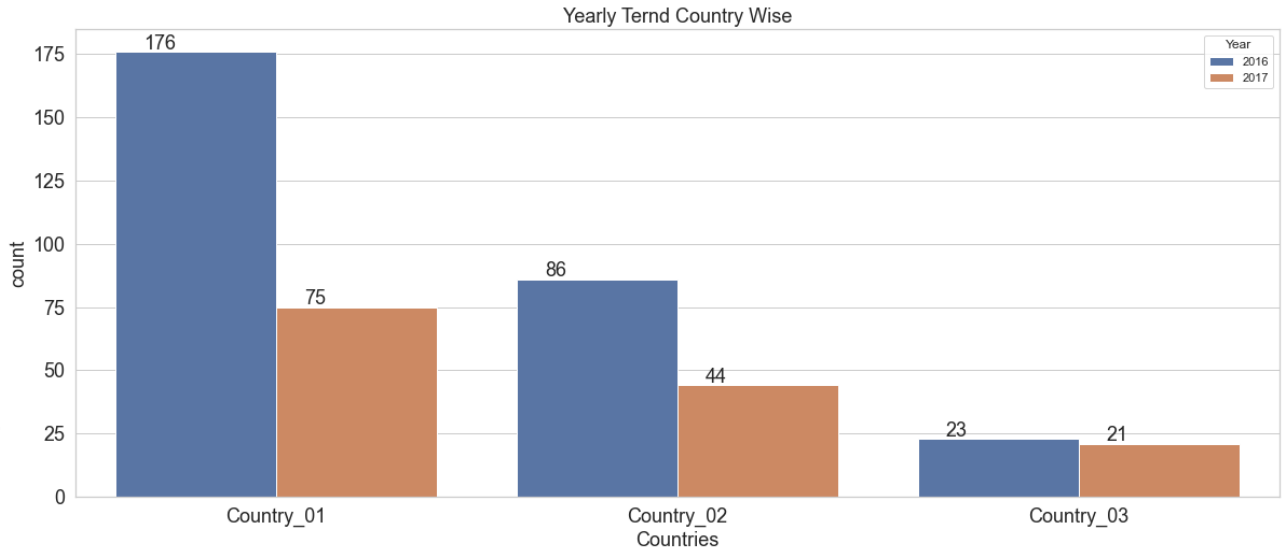
Weekday wise Trend:

We observe that maximum accidents were happened on Thursday followed by Tuesday and Wednesday.



8.4 Attribute "Countries"

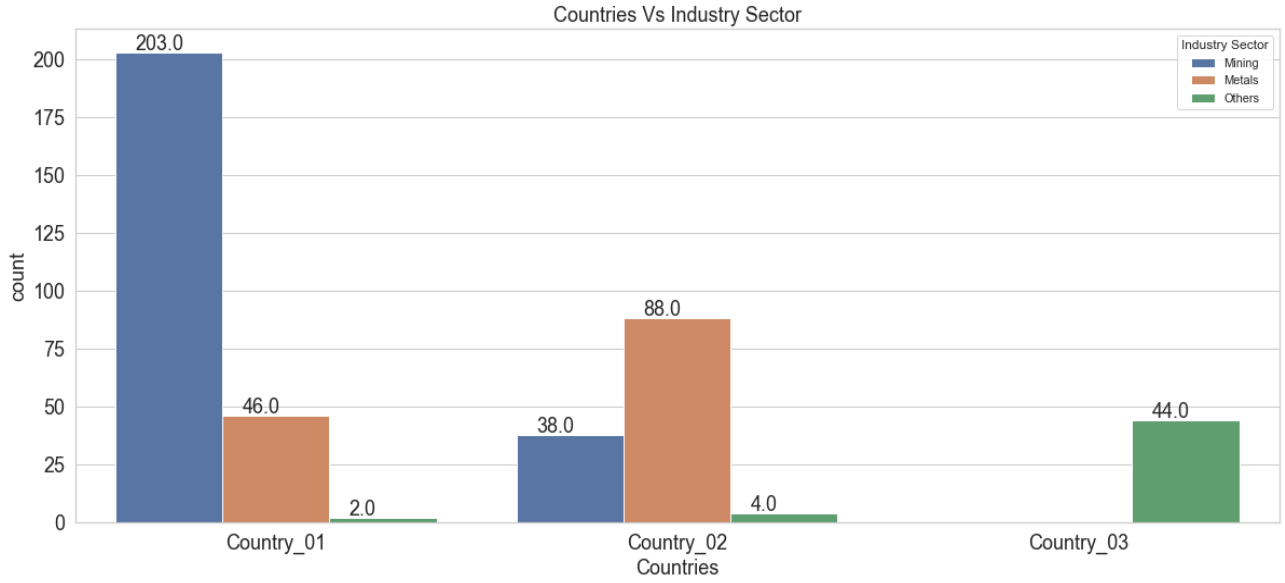
This attribute provides the information about the country where the accident happened. There are total 3 countries from where the data is captured. We analyze the trend based on the countries as below:



We can see that maximum accidents were recorded in country_01 followed up by Country_02 and Country_03

Country wise Industry wise Trend: We observed that:

- Country_01 has 80%of the accidents recorded from the mining industry
- Country_02 has 68% of the accidents recorded from the Metal industry
- Country_03 has all 44 accidents from the other industry sector

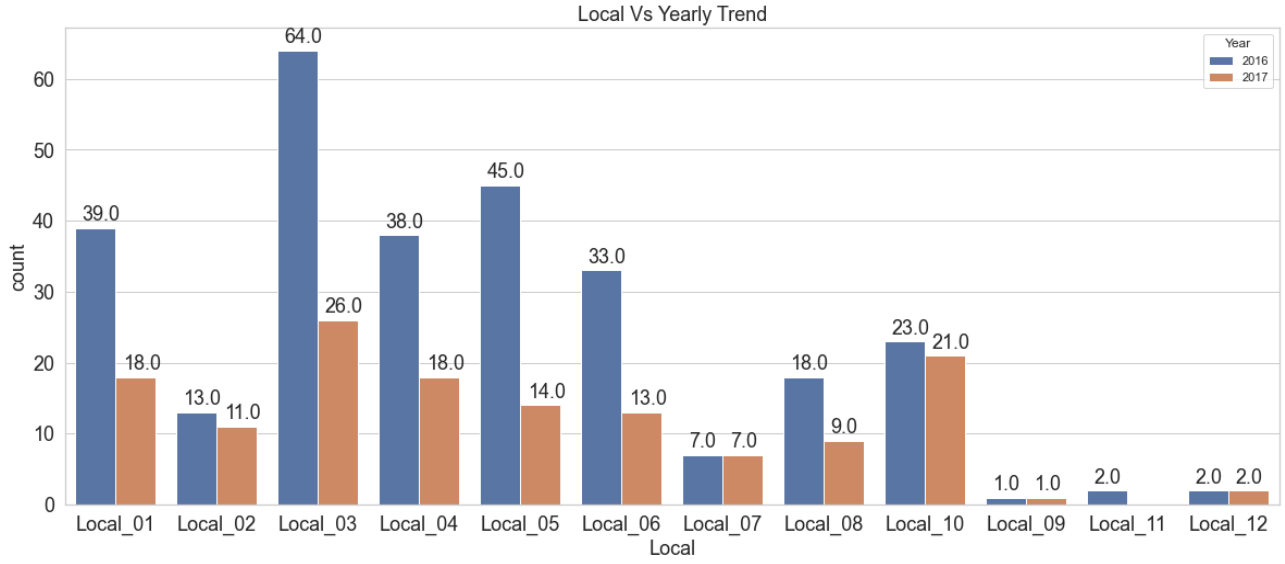


8.5 Attribute "Local"

This attribute provides the information about the city where the accident was recorded. There are total 12 cities.

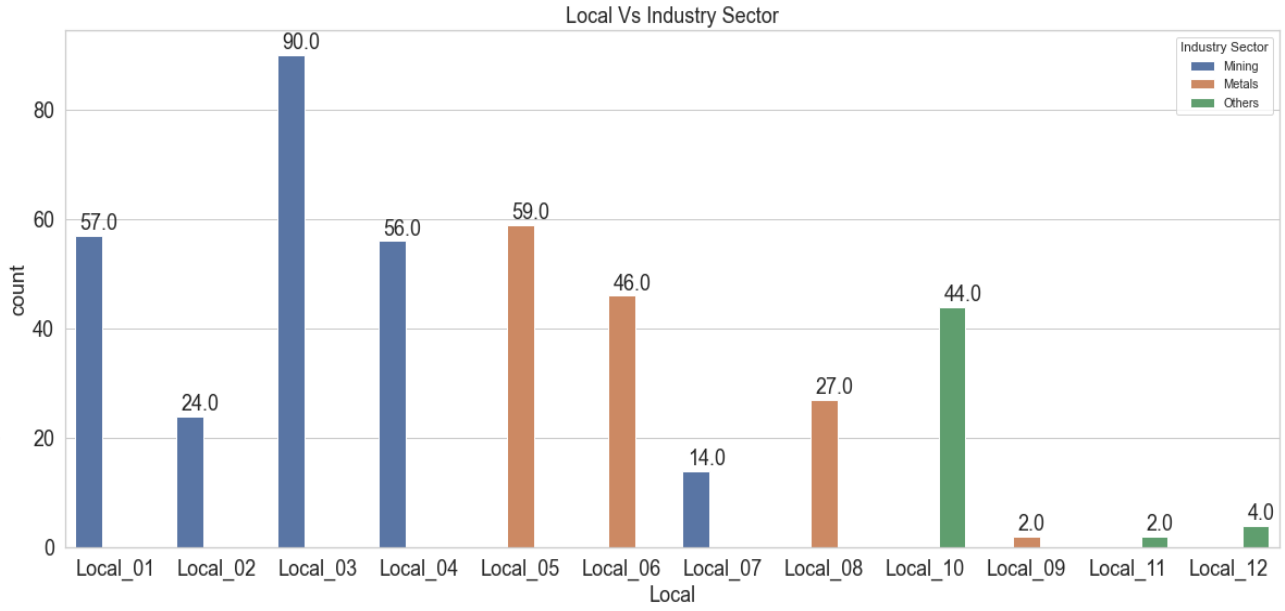
Local wise yearly Trend: We observed that:

- Local_03 has 22% of the accidents recorded during 2016, followed by Local_05 and Local_01
- There is a reduction in accidents recorded during 2017compare to 2016 in almost all Local



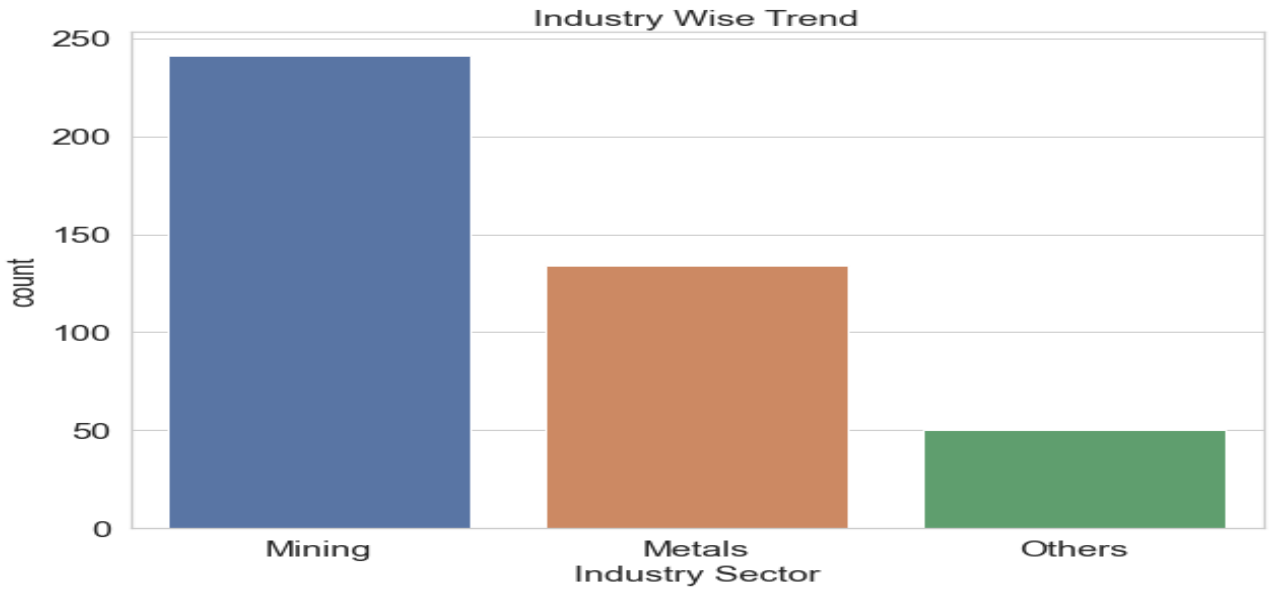
Local wise Industry Trend: We observed that:

- Specific Locals have specific industries
- Mining industry is available in 5 locals, Metal industry is in 4 locals and others in 3 locals



8.6 Attribute "Industry sector"

This section provides the information about the industry in which accident were recorded.



We observed that maximum accidents were recorded from Mining industry followed by Metals.

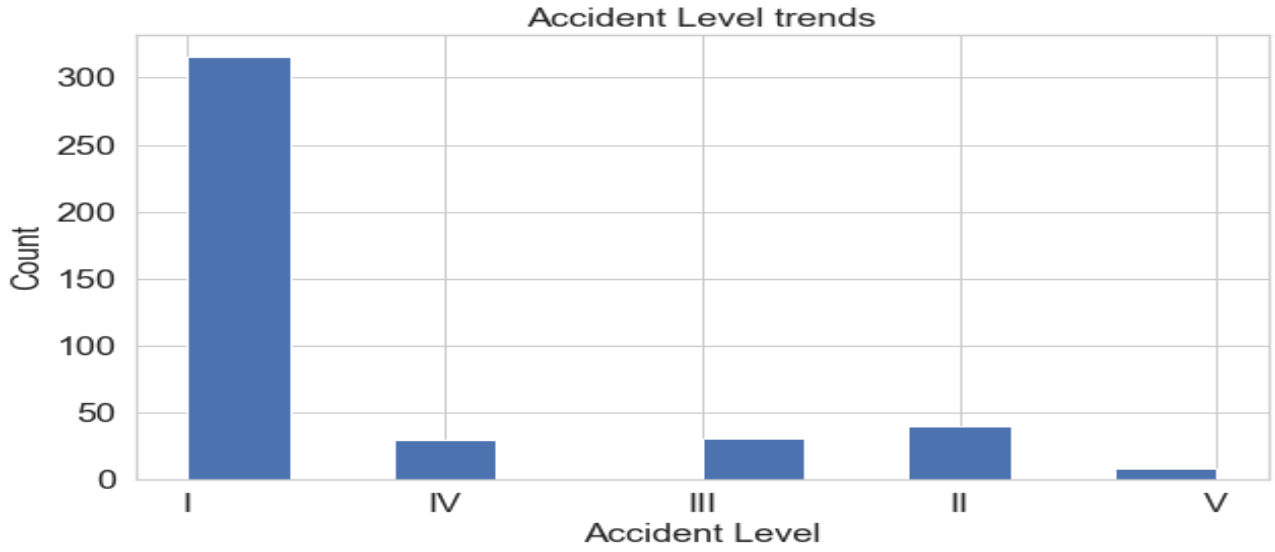
Accident Level Trend Industry Sector Wise: We observed that:

- Maximum Level 1 accidents were recorded in Mining industry, followed by metals
- Mining industry is available in 5 locals, Metal industry is in 4 locals and others in 3 locals

8.7 Attribute "Accident level"

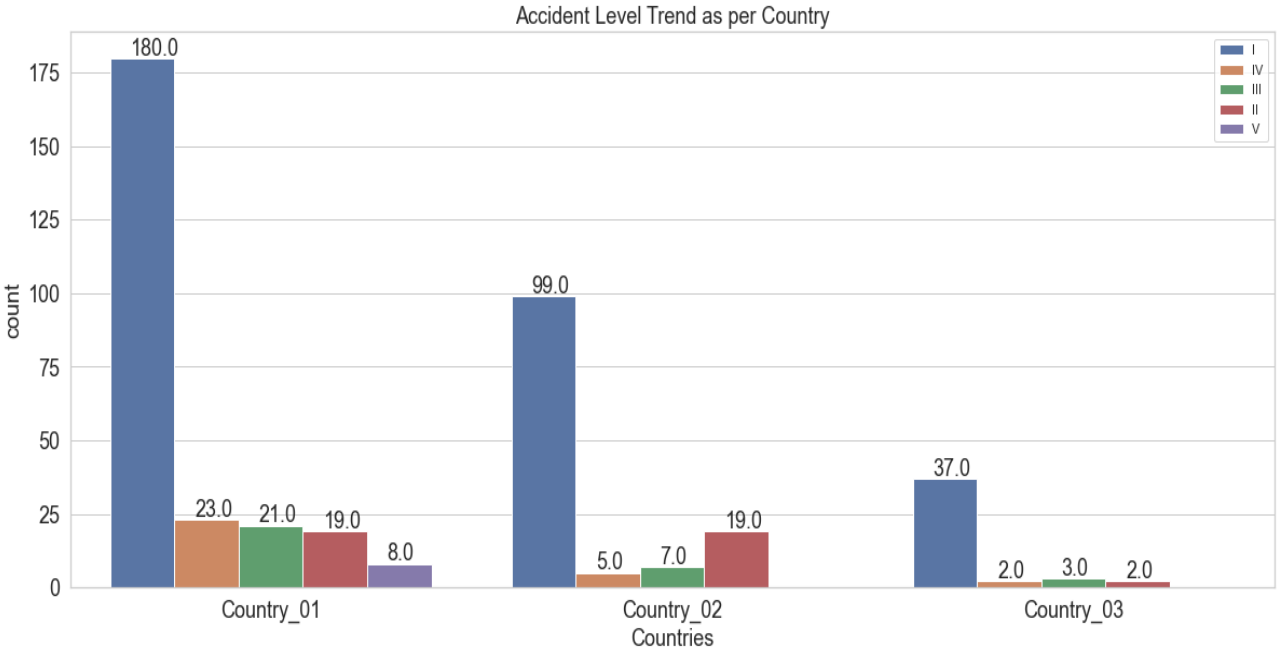
This attribute provides information about the severity of the accident where accident level 1 has the lowest severity and level V being the highest severity.

We can observer 74% of the accidents were of recorded as level I followed by 9.4% as level II



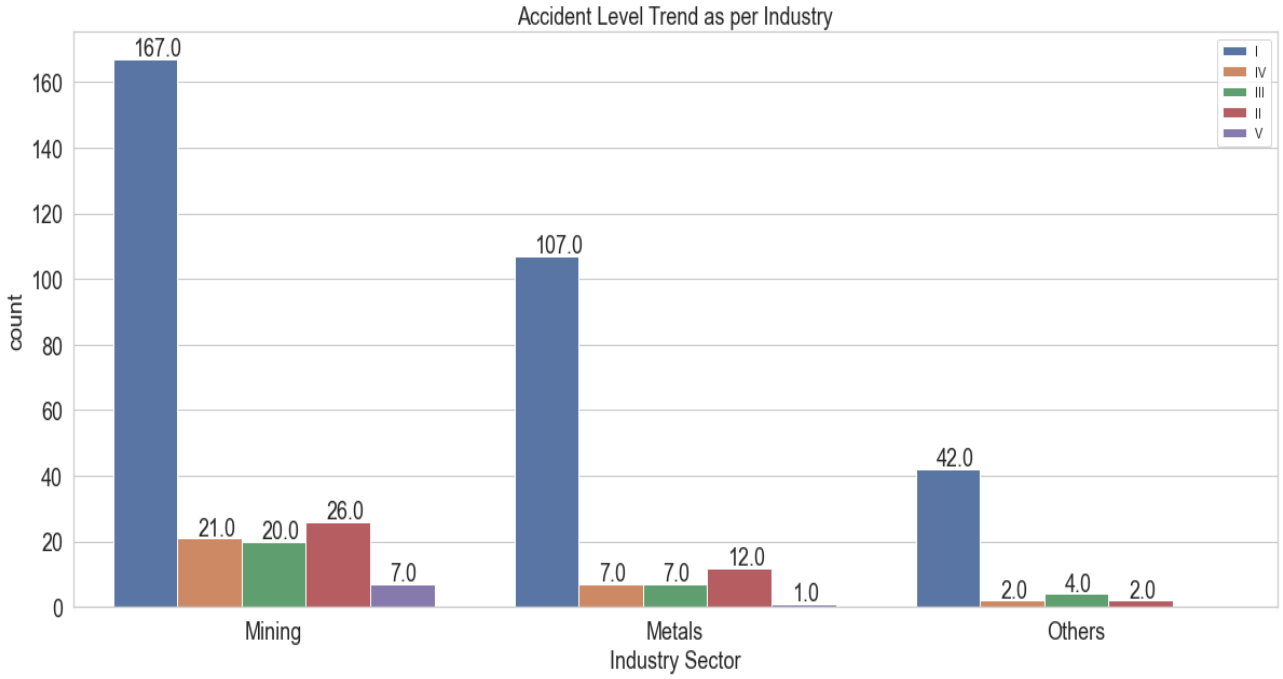
Accident Level Trend Country Wise: We observed that:

- Country_01 has maximum accident recorded as level I but as an exception level IV & III are recorded more than Level II. This is not a general trend
- In Country_03, 85% of accidents were logged as level I
- Level V accidents were only logged in country_01



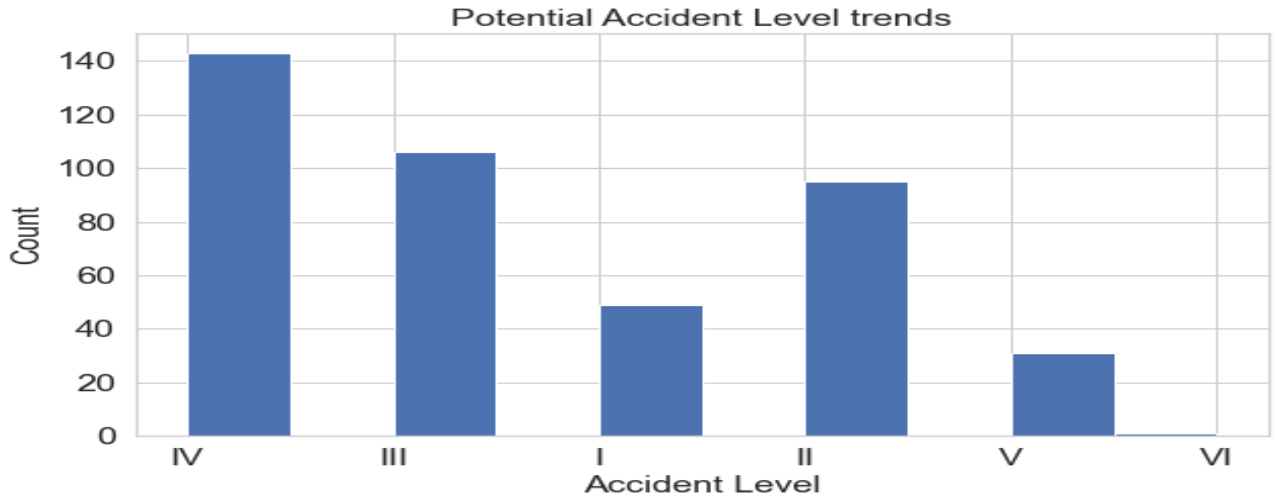
Accident Level Trend Industry Wise: We observed that:

- Almost 56% of the accidents were logged in mining industry, followed by 31% in Metal industry
- Level V accidents were only recorded in Metal and Mining industry



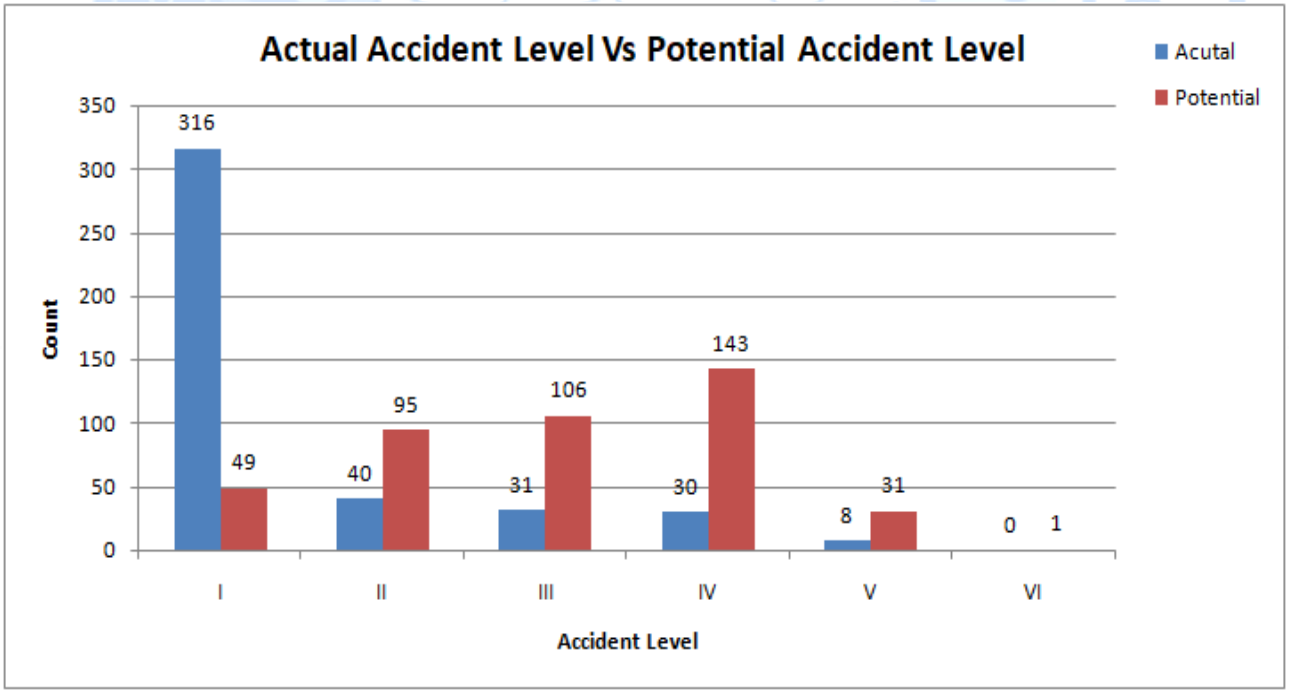
8.8 Attribute "Potential Accident Level"

This attribute provides information about the severity of the Potential accident where accident level 1 has the lowest severity and level VI being the highest severity.



Accident Level Vs Potential Accident Level: We observed that:

- Only 11% of the accidents were of Level I, however in reality 74% were limited to Level I
- ~34% of the accidents were logged as Potential Level IV accidents where in reality only 7% were logged as level IV



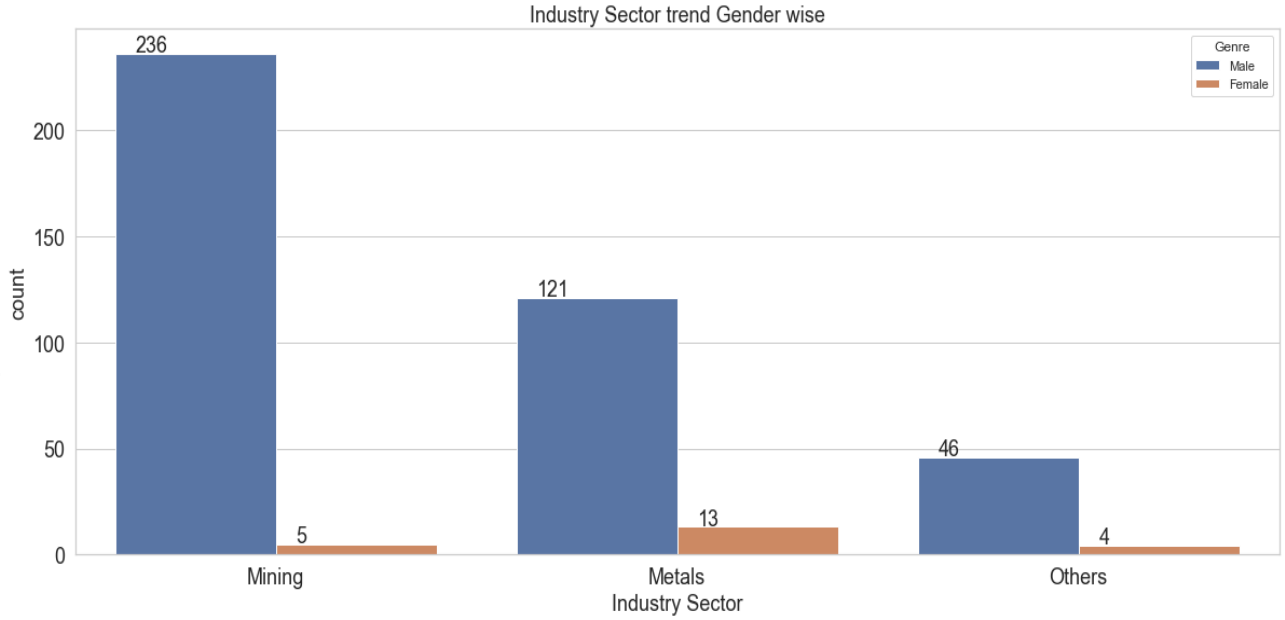
8.9 Attribute "Genre"

This attribute provides information about the gender of the personal who was affected by the accident.

- ~95% of the accidents were of Male gender.

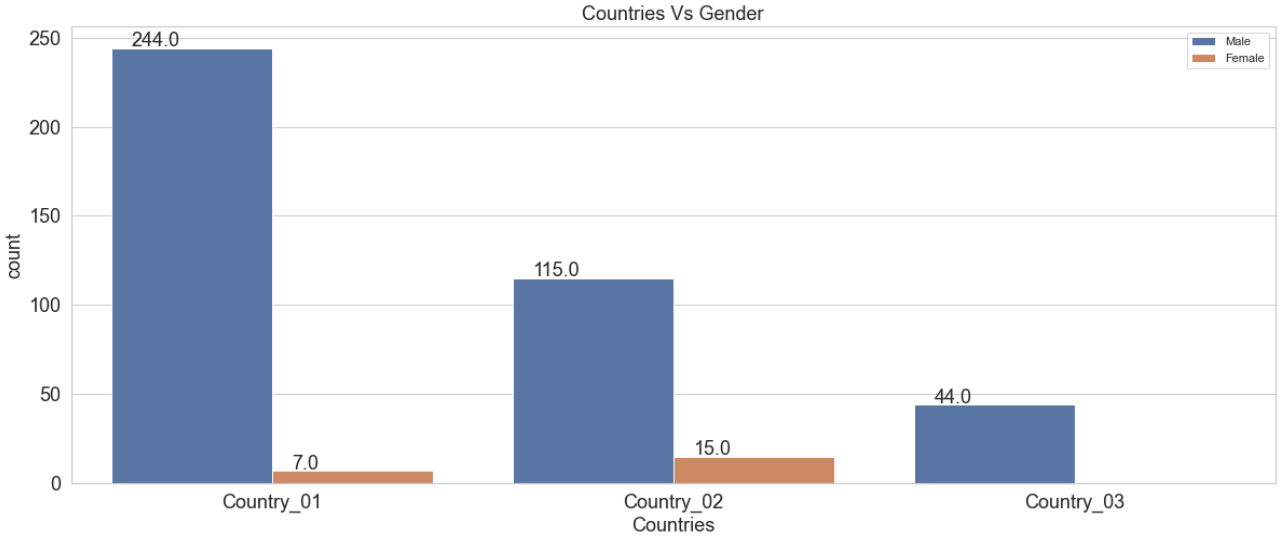
Industry Sector wise Gender Trend: We observed that:

We observed that in metal industry 9% of accidents were related to female. This is four times higher than mining industry where only 2% accidents were related to females



Country wise Gender Trend: We observed that:

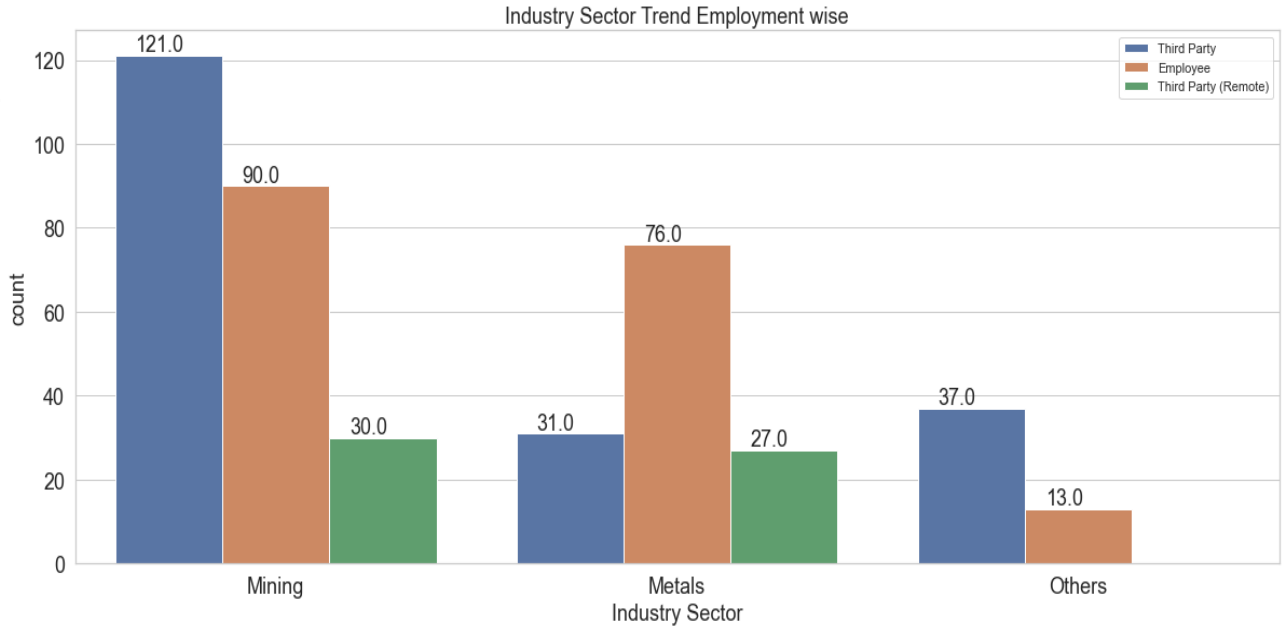
- Country_03 has not recorded any accidents related to females, possible country_03 don't have female employees. We don't have much information about that



8.10 Attribute "Employee or Third Party"

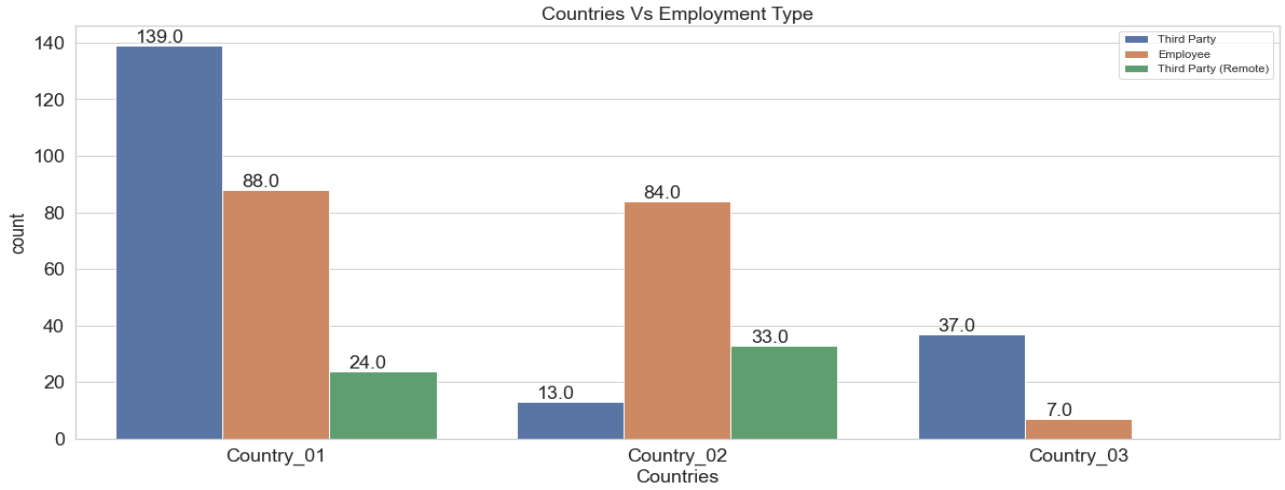
This attribute provides information about the employment type about the person affected by the accident. We observe the following:

- Mining industry records maximum accidents for third party
- In metal industry maximum accidents were recorded for employees
- In other industries no accidents were recorded for third party (Remote)



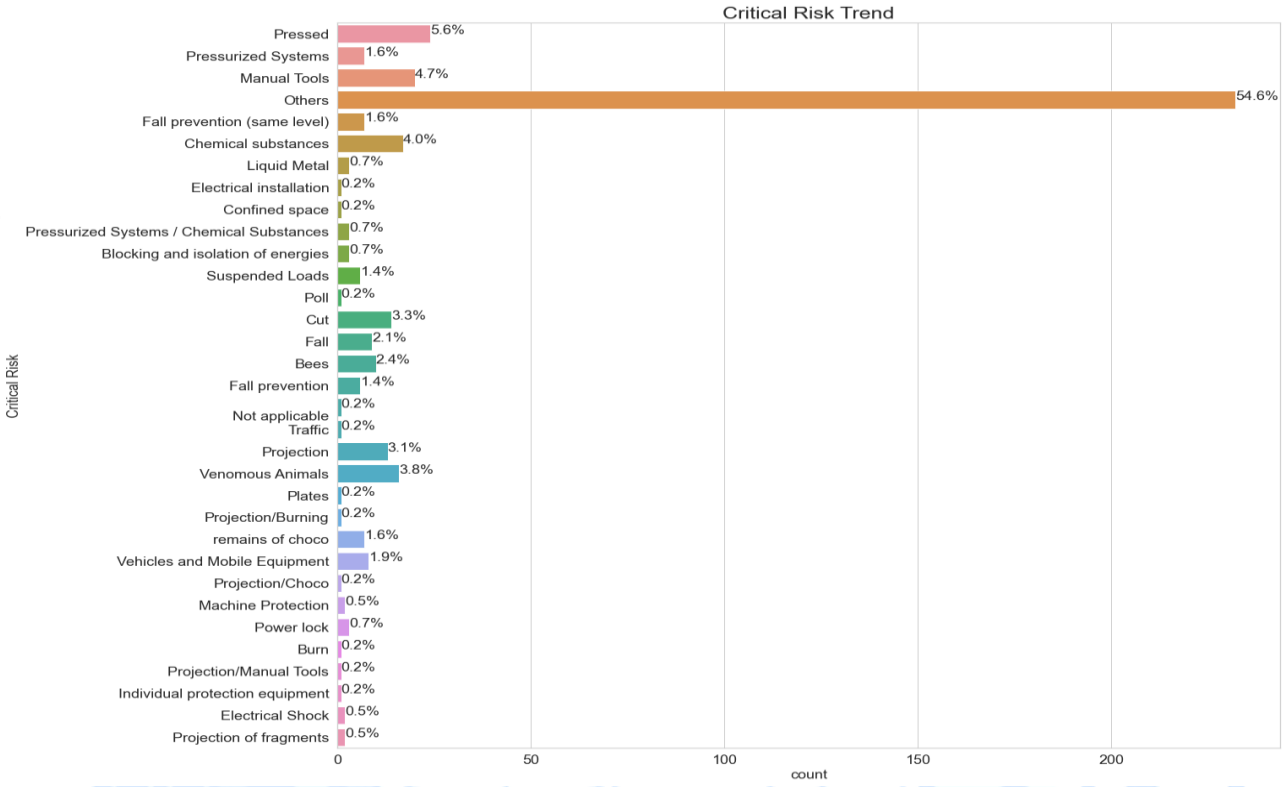
Country wise Employment Trend: We observed that:

- Country_03 has not recorded any accidents related to Third Party (Remote) workers
- Country_01 records maximum accidents for third party
- In Country_02 maximum accidents were recorded for employees



8.11 Attribute "Critical Risk"

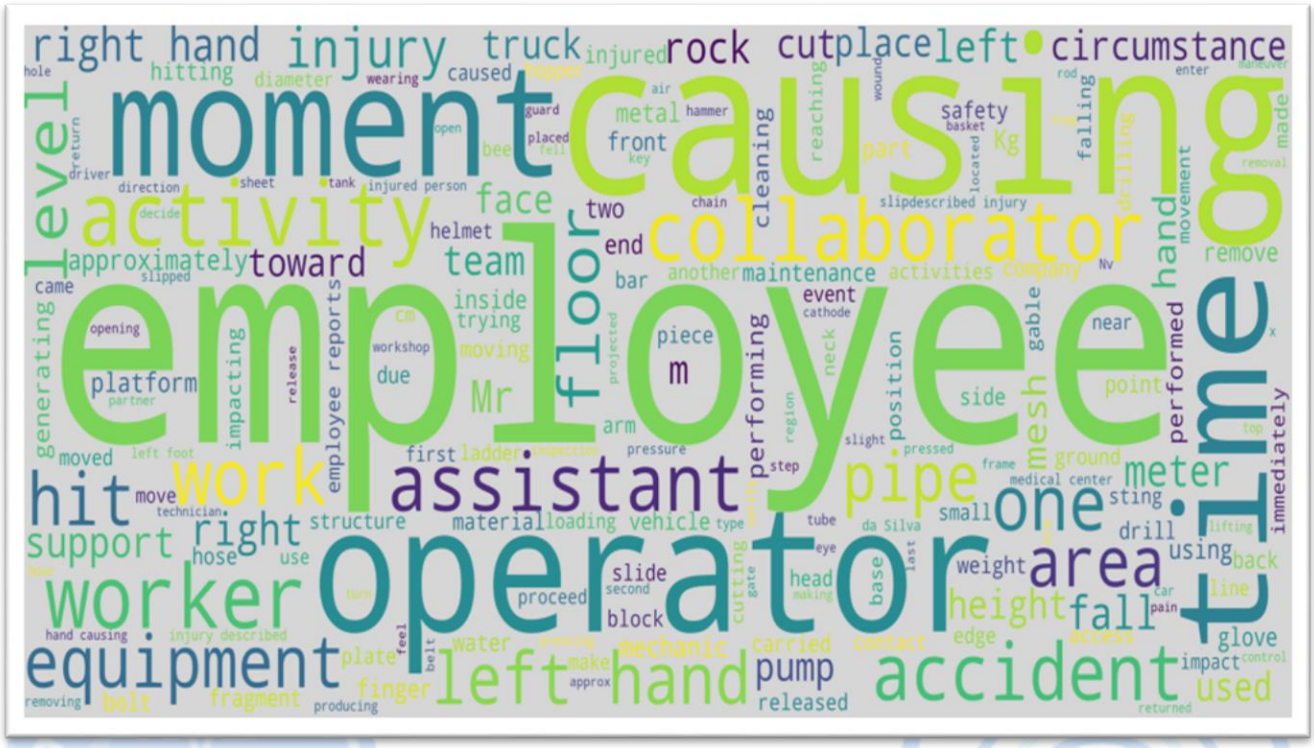
This attribute provide details about the Critical Risk which was the cause of the accident. Under Critical Risk attribute there are 33 different categories, others category has maximum contribution of 54% followed by Pressed ~6% and manual tools as 4.7%



8.12 Attribute "Description"

This attribute provides the information about the accident description in free text. The minimum length of description in the dataset is 94characters whereas maximum length is 1029 characters.

We created the word cloud to observe the words which are repeated maximum number of times



From the word cloud we observe that following words are repeated maximum number of times:

- Employee
- Operator
- Causing
- Movement
- Left Hand
- Time
- Right Hand
- Injury

9 Data Cleansing

Data cleansing is done to remove non-essential attributes, remove NAN value, remove special characters etc. This section covers the data cleansing steps.

9.1 Removal of Non-essential attribute

Attribute name “Unnamed: 0” has values e.g. 1,2,3 starting from 0 to 438. Also it has no duplicate values, however the number are not in sequence which is observed in the bottom 10 rows e.g. index 415 having a corresponding value as 429 and last index 424 having a corresponding values as 438. It looks more to be a index columns without proper indexing. Therefore we will drop this attribute as it will not add any values in the analysis

9.2 Identify and Remove Duplicates Records

We identified, post removal of “Unnamed: 0” there are 7 duplicate rows, however as of now we are not removing these rows as we have such a small dataset, during model building we will observe the performance on removal and while keep these rows in the dataframe

```
aa=sum(safety_df.duplicated())
if aa > 0:
    print("\033[1m""There are {:.0f} duplicates rows in the DataFrame""\033[0m".format(aa))
else:
    print("\033[1m""There are no duplicates rows in the DataFrame""\033[0m")
```

```
There are 7 duplicates rows in the DataFrame
```

9.3 Remove NaN values

We observe that there is no Nan records are there in data frame:

```
def myfunc(dfname):`
    lst=list(dfname.columns)
    nullcheck=dfname.isnull().values.any()
    if nullcheck == True:
        print('There are null values in the DataFrame')
    else:
        print('No Null Values in the DataFrame')
    for ax in lst:
        ze=len(dfname[dfname[ax]==0])
        ze1=len(dfname[dfname[ax]== '#'])
        ze2=len(dfname[dfname[ax]=='@'])
        ze3=len(dfname[dfname[ax]=='NaN'])
        ze4=len(dfname[dfname[ax] == " "])
        ze5=len(dfname[dfname[ax].isnull()])
        ze6=len(dfname[dfname[ax] == '?'])
        chkobj=dfname[ax].dtypes

        if ze > 0:
            print('No of Zeros in attribute',ax,'is:', ze)

        if ze1 > 0:
            print('No of # in attribute',ax,'is:', ze1)

        if ze2 > 0:
             print('No of @ in attribute',ax,'is:', ze2)
        if ze3 > 0:
             print('Not a number in attribute',ax,'is:', ze3)
        if ze4 > 0:
              print('No of blank in attribute',ax,'is:', ze4)
        if ze5 > 0:
              print('No of null in attribute',ax,'is:', ze5)
        if ze6 > 0:
              print('No of Question Marks in attribute',ax,'is:', ze6)

        if chkobj == float:
            ze7=len(dfname[dfname[ax] < 0])
            if ze7 > 0:
                print('Negative Values in',ax,'is:', ze7)
        if chkobj == int:
            ze8=len(dfname[dfname[ax] < 0])
            if ze8 > 0:
                print('Negative Values in',ax,'is:', ze8)

myfunc(safety_df)
```

```
No Null Values in the DataFrame
No of Zeros in attribute Unnamed: 0 is: 1
```

9.4 Label Encoding

We have observed that except “Unnamed: 0” attribute we have all the attributes as object type therefore we need to convert “Country, Local, Industry Type, Accident Level, Potential Accident Level, Gender & Employment Type them into numeric values. For Target class we will do the conversion at modeling stage as we need to try multiple option i) keeping all classes ii) removing classes which were having low count iii) removal of others class etc.

2.3 Attribute : Industry Sector

```
safety_df['Industry Sector']=safety_df['Industry Sector'].map({'Mining' : 0, 'Metals' : 1,'Others' : 2})
```

2.4 Attribute : Accident Level

```
safety_df['Accident Level']=safety_df['Accident Level'].map({'I' : 0, 'II' : 1,'III' : 2,'IV' : 3,'V' : 4})
```

10 Data / NLP Pre-Processing

Data pre-processing is a onetime effort required during the model training phase. The purpose of this pre-processing is to reduce noise in the training data. We reduce noise and enrich training data using the following techniques:

10.1 Removal of stop words& Lemmatization

10.1.1 Removal of Stop Words

When computers process natural language, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called stop words. Stop-words removal is a technique that can increase performance of a ML or deep learning model. In this section, we determine whether the increase is significant enough to warrant its usage for subsequent comparisons of models. Removal of these stop words is will help in extraction of key feature words and enhance the model accuracy. NLTK has

a collection of these stopwords which we can use to remove these from any given sentence. This is inside the NLTK.corpus module. NLTK has a list of stopwords stored in 16 different languages. We can use that to filter out stop words from the sentences

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should',

"should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

10.1.2 Lemmatize the words

Lemmatization is one of the most common text pre-processing techniques used in Natural Language Processing (NLP) and machine learning in general.Lemmatization is the process of grouping together the inflected forms of a word so they can be analyzed as a single item.

Lemmatization technique is like stemming. The output we will get after lemmatization is called 'lemma', which is a root word rather than root stem, the output of stemming. After lemmatization, we will be getting a valid word that means the same thing.

Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. If confronted with the token saw, stemming might return just s, whereas lemmatization would attempt to return either see or saw depending on whether the use of the token was as a verb or a noun. The two may also differ in that stemming most commonly collapses derivationally related words, whereas lemmatization commonly only collapses the different inflectional forms of a lemma.

```
wl=WordNetLemmatizer()

def clean_up(x):
    check=re.compile('[^A-Za-z0-9\s]')
    text=check.sub('',x)
    text=text.lower()
    text = re.sub(r"[,.\"\'!@#$%^&*(){}?/;`~:<>+=-]", "", text)
    text=text.split()
    text=[wl.lemmatize(word) for word in text if not word in stopwords.words('english')]
    text=' '.join(text)
    return text
```

10.2 Tokenization, Sequencing & Padding

Tokenization is the task of taking a text or set of text and breaking it up into its individual tokens. Tensor flow provides the Tokenize class for preparing text documents for deep learning. The Tokenize class allows to vectorize a text corpus, by turning each text into a sequence of integers. It must be constructed and then fit on either raw text documents or integer encoded text documents.

Tokenize from "tensorflow.keras.preprocessing.text" package is applied to the raw data from the Description label. And then function "fit_on_texts" is applied to update internal vocabulary based on the list of texts. Once we obtain the dictionary with tokenize configuration, function

"texts_to_sequences" is applied to transform each text into a sequence of integers. We have taken into account the top 1000 words and the method returns list of sequences which are then sent to model.

LSTMs take inputs of same length, input sequences are padded to 100 lengths while testing and training.

```
max_features = 1000 # Number of words to take from tokenizer(most frequent words)
maxlen = 100 # Maximum length of each sentence to be limited to
embedding_size = 200 # size of embedding vector

# Tokenizer
tokenizer = Tokenizer(num_words=max_features, split=' ')
tokenizer.fit_on_texts(safety_df['Clean_description'].values)
X_desc = tokenizer.texts_to_sequences(safety_df['Clean_description'].values)

# Word Indexing
word_index=tokenizer.word_index
vocab_size = len(tokenizer.word_index) + 1

#Padding
X_desc=sequence.pad_sequences(X_desc,maxlen=maxlen)
```

11 Data Preparation for AIML model learning

Before we feed the data into classification model, we will merge the description vector with other features

11.1 Merging description vector with other features

At high level, free text description is encapsulated so that it gets converted to the machine learning understandable feature set. These features are added to the pre-defined features (like country, industry etc) so that full feature set is created.

Currently, description text is encapsulated with 100 dimensions so that every word is converted to 100 features using NLP libraries. These 100 feature vectors are added to 07 features which were originally provided in the dataset. Finally, 37 features are fed into the models to predict the 'Critical Risk' Label.

11.2 Target Class analysis and way forward

Initially we have got 33 target classes in the given dataset, it was found during the EDA that the data is completely biased toward one category which is "Others" contributing to 55% of the total data. It was clear that by carry these biased classes will not make us effective in prediction. So some working was initiated to understand the data further.

Class wise contribution details are given below:

Target Class	Count	%
Others	232	55%
Pressed	24	6%
Manual Tools	20	5%
Chemical substances	17	4%
Venomous Animals	16	4%
Cut	14	3%
Projection	13	3%
Bees	10	2%
Fall	9	2%
Vehicles and Mobile Equipment	8	2%
Pressurized Systems	7	2%
remains of choco	7	2%
Fall prevention (same level)	7	2%
....		

Moreover when have further read the given accident description it is also observe that not only other, there are many more classes which we not correctly classified. Serial no 15 accident description which is more related to Burn categorized as “Liquid metal” even though we have burn category available. Below is accident description given:

Description	Given Critical Risk	Correct Class
The employee was working in the When a thermal shock caused a splash of zinc in his direction, the employee, despite using all the indicated PPE, was hit by small spatters that passed between the facila and the hood. small burn in the face region.	Liquid Metal	Burn

So to overcome this challenge we have started working to reclassify the class based on the actual description and this is based on best of our understanding and the information we could gather from the open source community. Thereafter below are the revised categories:

```
ML_DF['Critical Risk'].value_counts()
```

```
fall                           61
Others                         41
projection                     37
Manual Tools                   36
sting by something             32
remains of choco               31
Chemical substances            30
Mechanical Failure             27
Vehicles and Mobile Equipment  20
Pressurized Systems            18
Pressed                        18
hit by something               16
Cut                            14
Machine Protection             13
mesh                            9
cut                             8
burn                            8
Loud Sound                      4
Electrical Shock                2
```

11.3 Removal of target classes having <15 count and others

Even after reclassifications there are 7 categories which were having accident records < 15 in numbers, as these categories will have only 58 accident records we will exclude these records while feeding the data into the model.

Similarly we will do it for “other” categories which are now reduced to 10% against 55% earlier in the original dataset. As others can’t be further categorized due to limitation of the captured description, we will exclude those records while feeding the data into classifiers.

```
#Remove low count classes and others
ML_DF = ML_DF.groupby('Critical Risk').filter(lambda x: len(x) > 15)
ML_DF = ML_DF[ML_DF['Critical Risk'] != 'others']
```

11.4 Label Encoder for target class

Label encoder is being now used for the target to convert the remaining classes into numeric values:

```
# Lable encode the Class
lb1 = LabelEncoder()
ML_DF['Critical Risk'] = lb1.fit_transform(ML_DF[['Critical Risk']])
```

11.5 Creating labels

We are now creating the X (Dependent Attributes) & y (Target Class). We have removed few of the attributes we are redundant in nature.

```
# Define X & Y
ML_X=ML_DF.drop(['Critical Risk','Accident Level','Potential Accident Level'], axis=1)
ML_y=ML_DF['Critical Risk']


#Split the Data
X_train,X_test,y_train,y_test=train_test_split(ML_X,ML_y,test_size=0.20,random_state=1)
```

11.6 Standard Scaler

Standard Scaler standardizes the features by subtracting the mean and then scaling to unit variance. Unit variance means dividing all the values by the standard deviation thereby eliminating the units. This dataset is being used for the machine learning models

12 Design, train and Test Machine Learning Classifier

Now we have data prepared and we are going ahead with multiple machine learning classification models:

Below are the machine learning models which were tried on the dataset using pipeline function of the sklearn:

```
from sklearn.pipeline import make_pipeline

from sklearn.linear_model import LogisticRegression # for Logistic regression model
from sklearn.naive_bayes import GaussianNB # for Gaussina Naive Bayes model
from sklearn.neighbors import KNeighborsClassifier # for KNN model
from sklearn.svm import SVC # for SVM Model
from sklearn.tree import DecisionTreeClassifier # for Decision Tree Classifier
from sklearn.ensemble import BaggingClassifier # for Bagging Classifier
from sklearn.ensemble import AdaBoostClassifier # for Ada Boosting Classifier
from sklearn.ensemble import GradientBoostingClassifier # for Graident Boosting Classifier
from sklearn.ensemble import RandomForestClassifier # For Random Forest Classifier

pipe=[]

pipe.append(make_pipeline(StandardScaler(),LogisticRegression()))
pipe.append(make_pipeline(StandardScaler(),GaussianNB()))
pipe.append(make_pipeline(StandardScaler(),KNeighborsClassifier()))
pipe.append(make_pipeline(StandardScaler(),SVC()))
pipe.append(make_pipeline(StandardScaler(),DecisionTreeClassifier()))
pipe.append(make_pipeline(StandardScaler(),BaggingClassifier()))
pipe.append(make_pipeline(StandardScaler(),AdaBoostClassifier()))
pipe.append(make_pipeline(StandardScaler(),GradientBoostingClassifier()))
pipe.append(make_pipeline(StandardScaler(),RandomForestClassifier()))
```

Below are the results of the Machine learning classifier, maximum test accuracy is of Random Forest classifier, which is also looks to be a over-fit model. We will not try the neural network and LSTM model and observe their accuracy.

	Model_Name	Train_Accuracy	Test_Accuracy
0	LogisticRegression	0.692833	0.148649
1	GaussianNB	0.208191	0.135135
2	KNeighborsClassifier	0.368601	0.175676
3	SVC	0.552901	0.270270
4	DecisionTreeClassifier	1.000000	0.229730
5	BaggingClassifier	0.996587	0.283784
6	AdaBoostClassifier	0.204778	0.121622
7	GradientBoostingClassifier	1.000000	0.256757
8	RandomForestClassifier	1.000000	0.310811

13 Design, train and Test Neural Network Classifier

Before we designed the neural network architecture we need to perform below activities:

13.1 Up Sampling

Post exclusion of classes having < 15 and others we are only left with 77% of data which is 326 records that is very less numbers for a neural network moreover there is still biasness in the output classes, so to manage this we will upscale the data.

SMOTE technique is used for up sampling the dataset and remove the biasness from the classes.

```
X_train, y_train = SMOTE(k_neighbors=5,random_state=1).fit_resample(X_train, y_train)
```

13.2 Conversion of Target Class to binary class matrix

to_categorical converts a class vector (integers) to binary class matrix which we will feed into the neural network

```
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

13.3 Word Embedding

A word embedding is a class of approaches for representing words and documents using a dense vector representation. Word embedding can be simply put as representation for words that capture their meaning, semantic relationships and the different types of contexts they are used in.
It is an improvement over more the traditional bag-of-word model encoding schemes where large sparse vectors were used to represent each word or to score each word within a vector to represent an entire vocabulary. These representations were sparse because the vocabularies were vast, and a given word or document would be represented by a large vector comprised mostly of zero values.

Instead, in an embedding, words are represented by dense vectors where a vector represents the projection of the word into a continuous vector space. The position of a word within the vector space is learned from text and is based on the words that surround the word when it is used. The position of a word in the learned vector space is referred to as its embedding. Two popular examples of methods of learning word embedding from text include:

- Word2Vec
- GloVe

We have adopted GloVe (Global Vectors for Word Representation) embedding, one of the commonly used and popular vectorization processes, the counts matrix is pre-processed by normalizing the counts and log-smoothing them.

In our model, we used word embeddings for model LSTM with Glove embeddings. Data was prepared by extracting the word embeddings using a pre-trained embedding file: glove.6B.200d.txt. Using the embedding matrix, weights matrix is created. For each word from tokenizer.word_index, if it is in the Glove vocabulary, a pre-trained word vector is loaded. This is then passed to our LSTM and neural network model.

13.4 Neural Network architecture

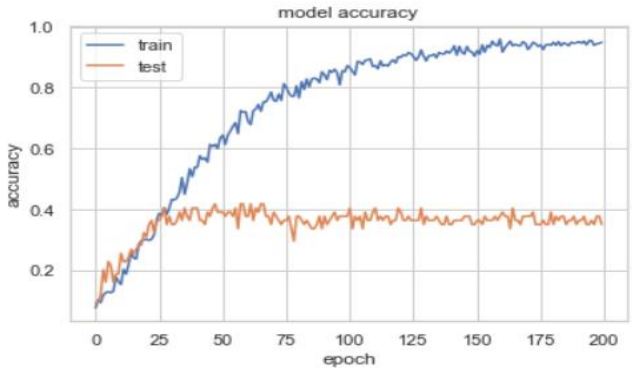
Below is the neural network architecture where we have used the glove embedding of 6b.200 dimensions with trainable weights.

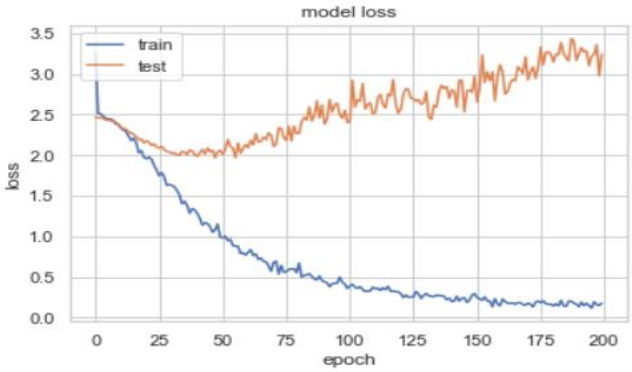
```
embeddings_index = dict()
f = open('glove.6B.200d.txt',encoding="utf-8")
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
#print('Loaded %s word vectors.' % len(embeddings_index))

embedding_matrix = np.zeros((len(word_index) + 1, 200))
for word, i in tokenizer.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

model1 = Sequential()
model1.add(Embedding(vocab_size, embedding_size,weights=[embedding_matrix], input_length=maxlen,trainable=True))
model1.add(Conv1D(64, 3, activation='relu'))
model1.add(Dropout(0.7))
model1.add(GlobalMaxPooling1D())
model1.add(Dense(64, activation='relu'))
model1.add(Dropout(0.5))
model1.add(Dense(16, activation='relu'))
model1.add(Dense(len(NN_y.unique()), activation='softmax'))
model1.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
history = model1.fit(X_train, y_train, batch_size=16, epochs=200, validation_data=(X_test, y_test), verbose=0)
```

3






```
Out[33]: ('Neural Network Classifier',
          0.9948979616165161,
          0.3513513505458832,
```

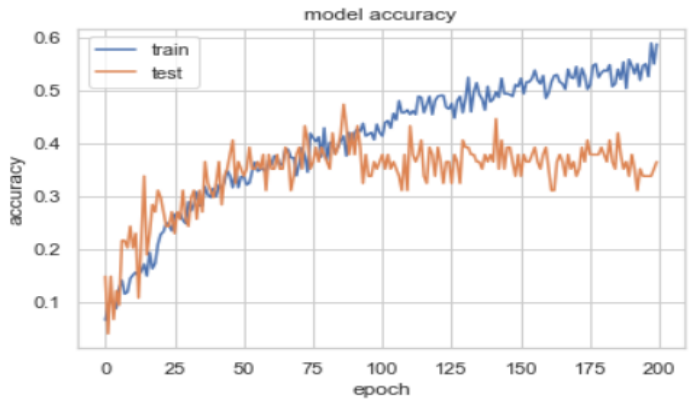
14 Design, Train and Test LSTM Classifier

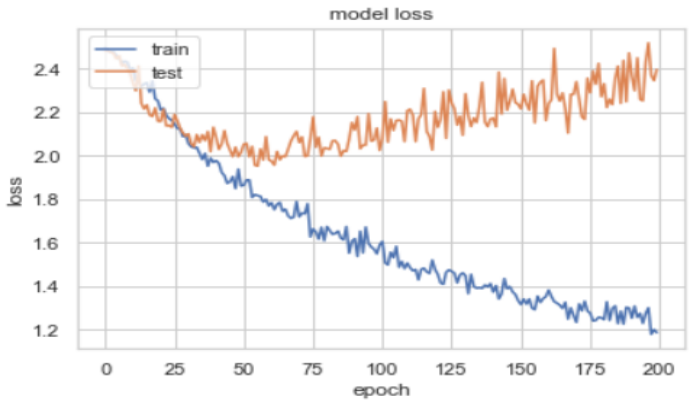
We have used the same dataset in LTSM as well which we feed into neural network classifier. Below is the architecture

```
embeddings_index = dict()
f = open('glove.6B.200d.txt',encoding="utf-8")
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))

embedding_matrix = np.zeros((len(word_index) + 1, 200))
for word, i in tokenizer.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector


model2 = Sequential()
model2.add(Embedding(vocab_size, embedding_size,weights=[embedding_matrix], input_length=maxlen,trainable=False))
model2.add((LSTM(56,dropout=0.6, recurrent_dropout=0.3)))
model2.add(Dense(10, activation='relu'))
model2.add(Dropout(0.5))
model2.add(Dense(len(LSTM_y.unique()), activation='softmax'))
model2.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
history = model2.fit(X_train, y_train, batch_size=32, epochs=200, validation_data=(X_test, y_test), verbose=0)
```

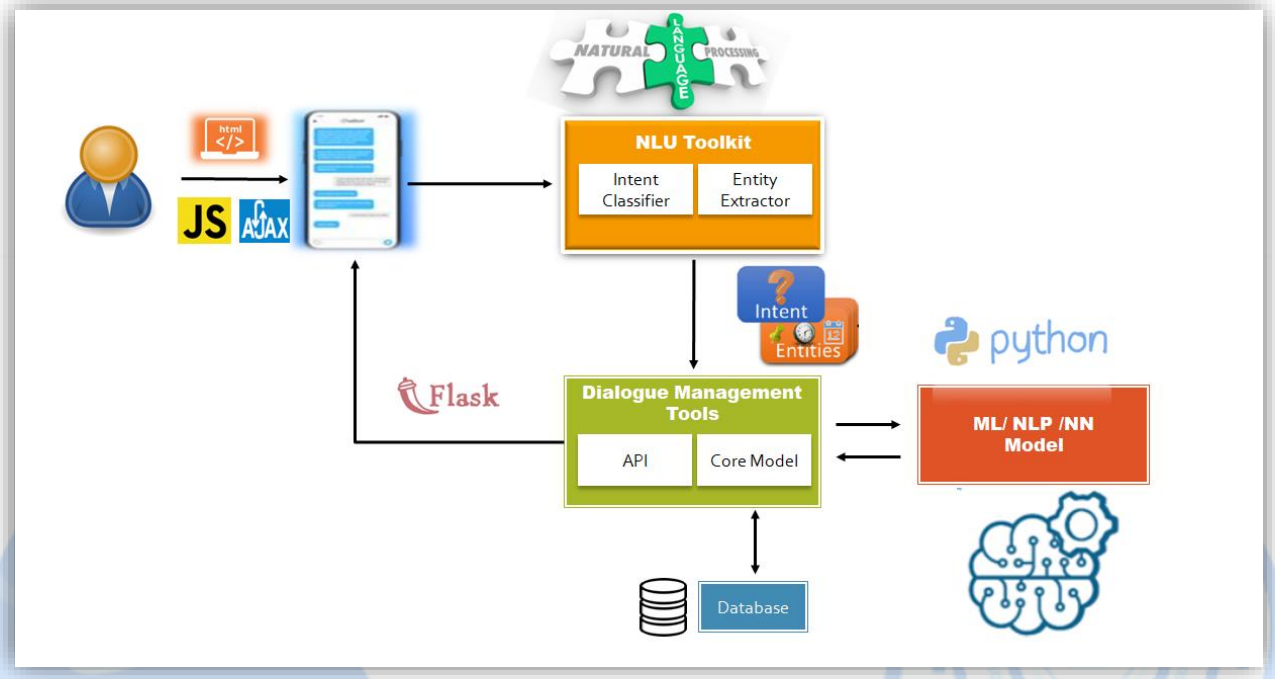





```
Out[31]: ('LSTM_Classifier',
          0.9132652878761292,
          0.36486485600471497,
```

15 Architecture of Solution

This section provides specific architecture of the solution that we designed, developed and deployed.



Following are the high-level components of the Chatbot architecture that are included for this project.

1. User
2. HTML / Java Script / Ajax
3. Graphical User Interface (GUI) / Chatbot
4. Natural Language Understanding (NLU)
5. Machine Learning / Neural Network and NLP Model
6. Flask
7. Database

15.1 User

User can be anyone who wish to predict the Critical Risk, Accident Level and Potential Accident Level of the given accident description, however user should be authorized from the respective department of the organizations to get those details from the Chatbot. It can't be publically accessible. However for the scope of the project, user management module is out of scope.

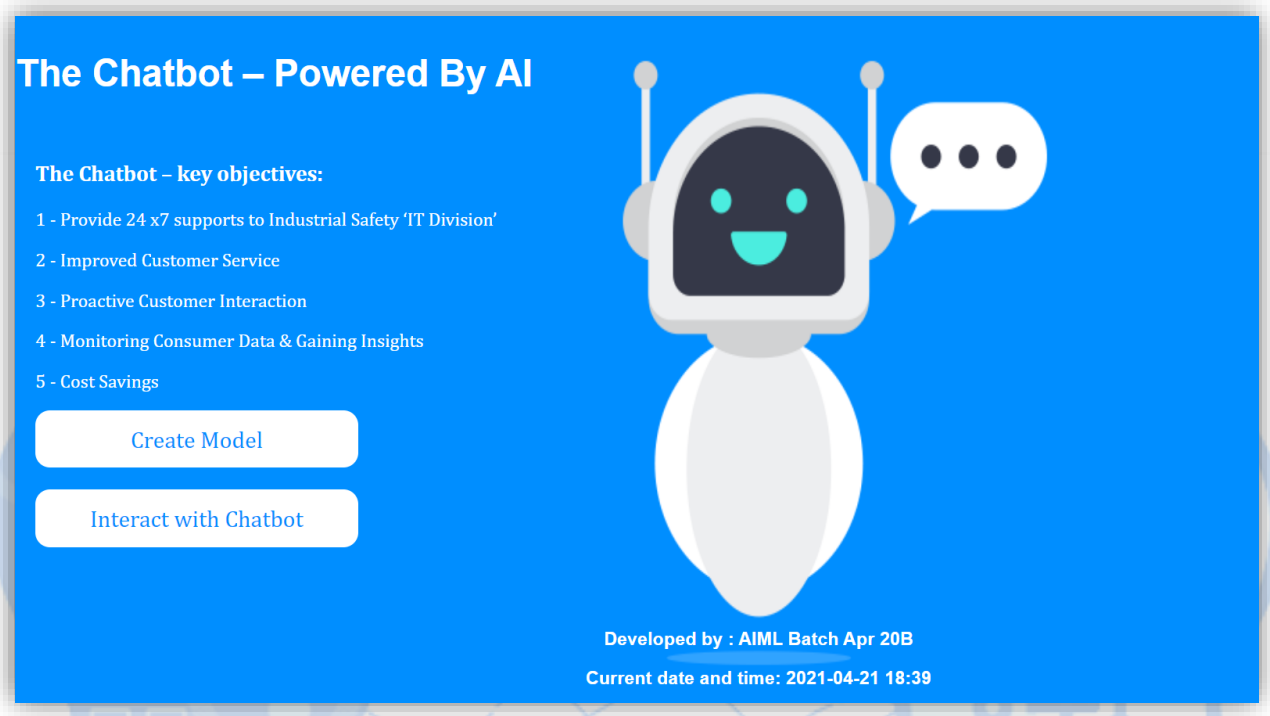
15.2 HTML / Java Script / Ajax

Chatbot can be accessed from the home page of respective company / organization. We have created an HTML default page which shows:

1. Benefits of the Chatbot

2. Create Model Button
3. Interact with Chatbot Button
4. Current Time
5. Developed by

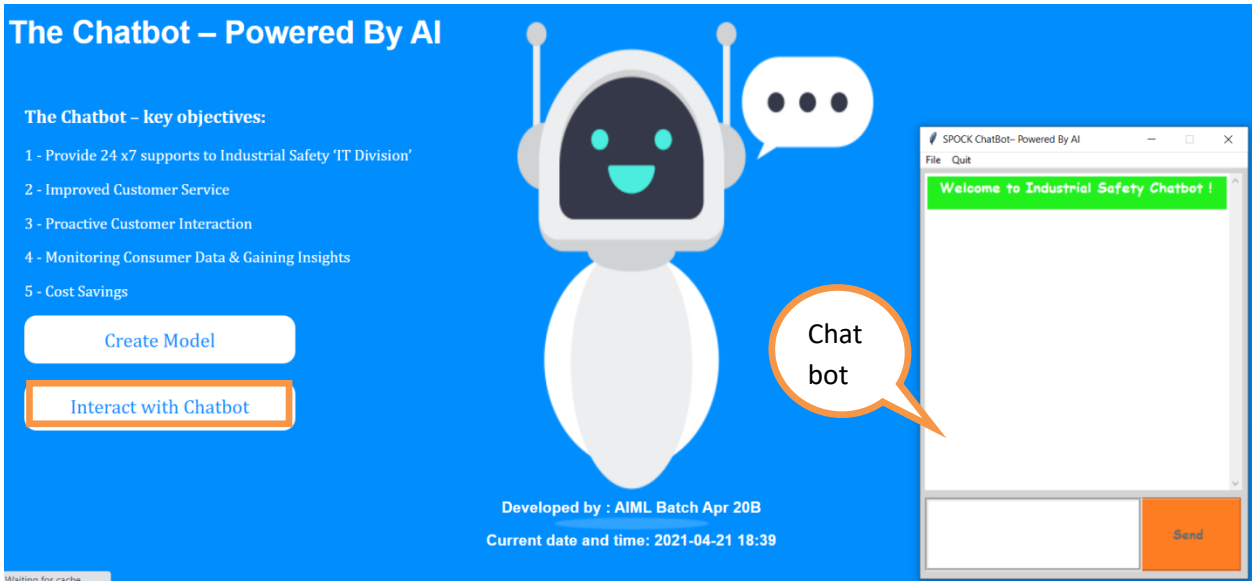
Below is the Home screen designed through HTML:



15.3 Chatbot GUI

User can click on the "**Interact with Chatbot**" button given on the home page. On click, a new window will open of a Chatbot where user can start the interaction with the Chatbot. Chatbot window shows:

- **Welcome Message:** Which is static text
- **Text Box:** User can enter in the inputs in this box
- **Interaction Window:** It is the section where the user and bot interaction will be catpured
- **Send Button:** Send button will be activated post entering the text in the textbox and on the button click event user will get the response from the Bot.
- **Menu options - File:** It has New sub menu which can be used by user to start a fresh interaction and Exit sub menu will be used to exit from the chat window
- **Menu Option - Quit:** User can quit anytime from the chat window by click on quit

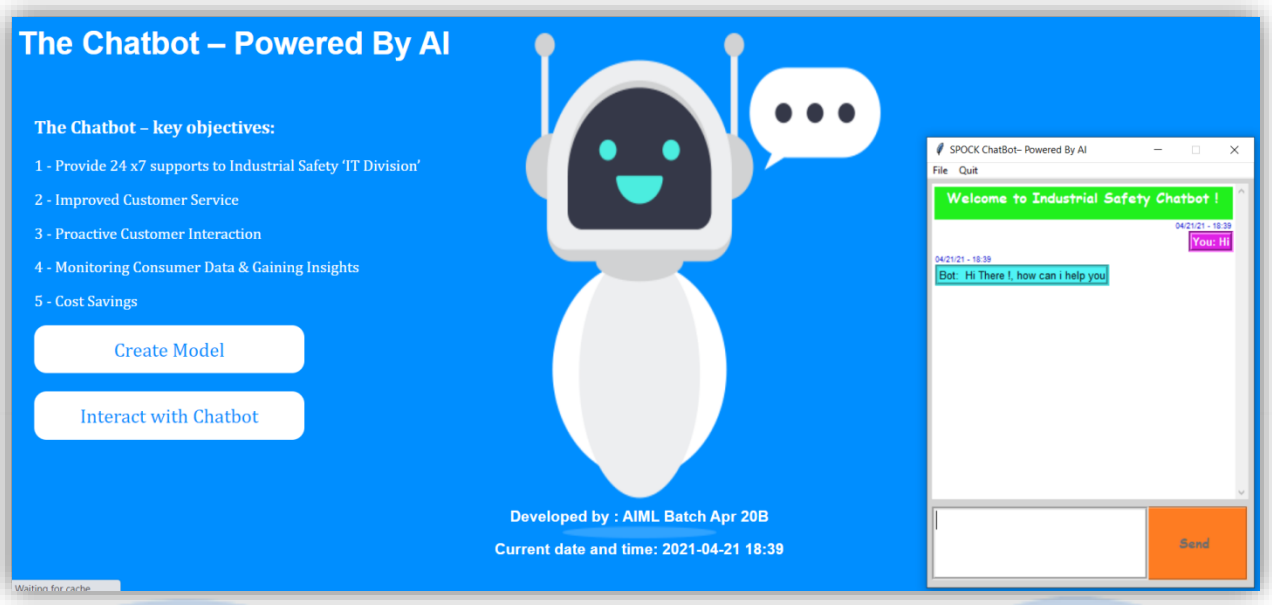


15.4 User Interaction with Chatbot

15.4.1 Greeting

User can type Hi / Hello in the textbox and Chatbot will take this as input and at the backend predict the intent of the given input basis which it will choose a random response and revert back to the user by one of the following:

“Hello”, ”Hey”, ”Good to see you again”, ”Hi There !”

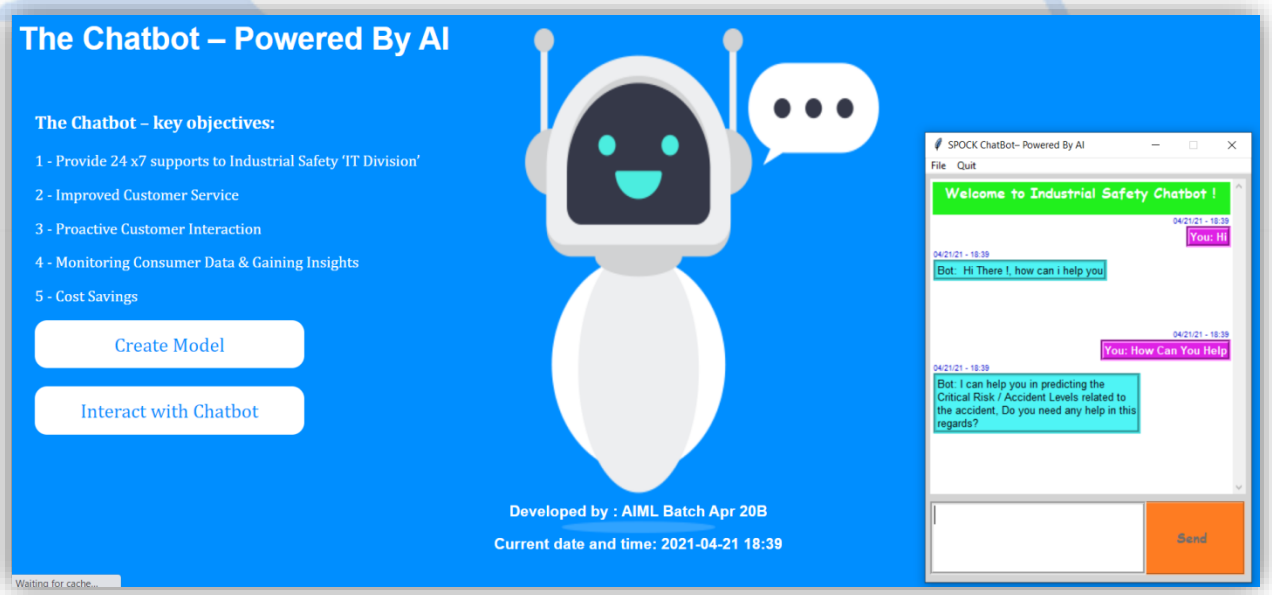


15.4.2 How can Chatbot Help

Post greeting interaction, now user can ask the Chatbot:

- What help you can provide
- How can you Help
- What can you do
- Help

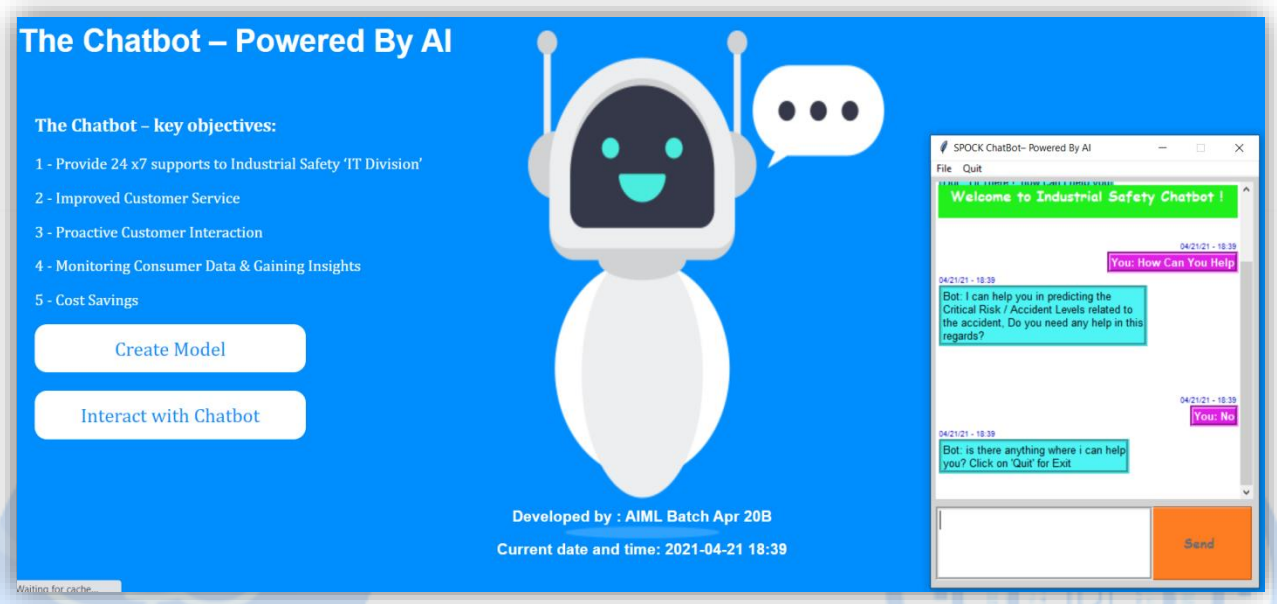
Upon understanding the above inputs, Bot will revert as: "I can help you in predicting the Critical Risk / Accident Levels related to the accident, do you need any help in this regards?"



If user says**: “No”**

Bot will revert: "is there anything where I can help you? Click on 'Quit' for Exit"

If user wishes to discontinue the interaction, can click on “Quit” menu option from the top

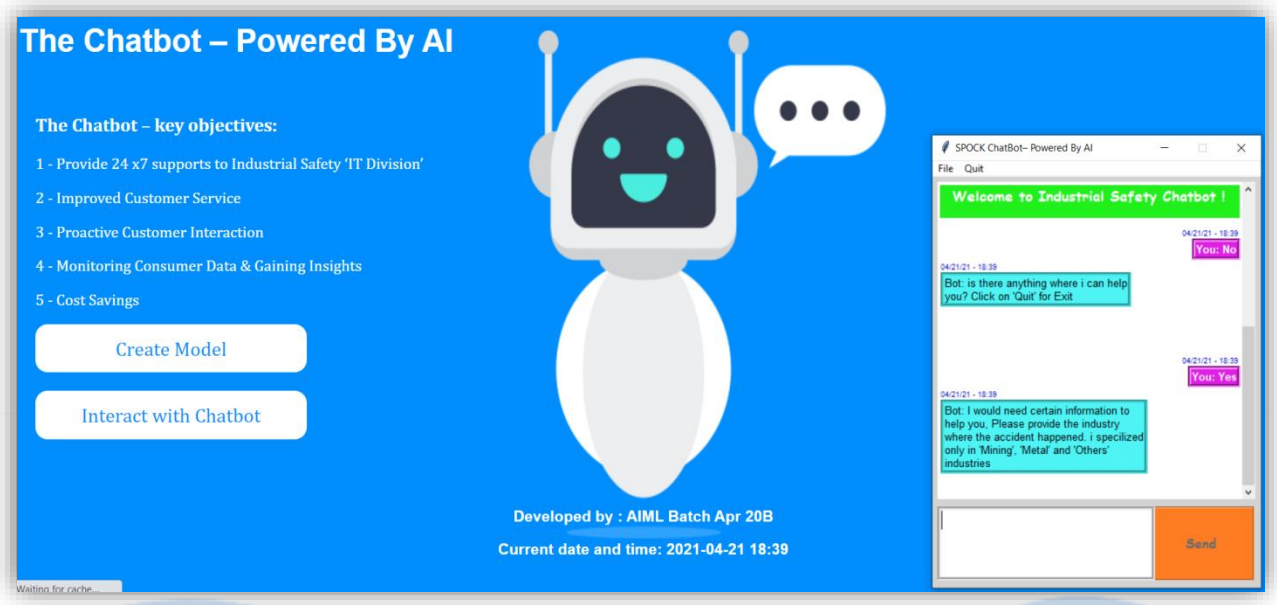


If the answer is: any of these **“Yes” / “Yeah” / ”Sure”**

Bot will revert: I would need certain information to help you, please provide the industry where the accident happened. I specialized only in 'Mining', 'Metal' and 'Others' industries".

Such inputs are required to predict the critical risk, accident type and potential accident type. Bot wukk take some more inputs from the user and store it in a pandas data frame and feed into the network for the prediction.

Therefore, we will see below few more interaction screens asking user inputs.

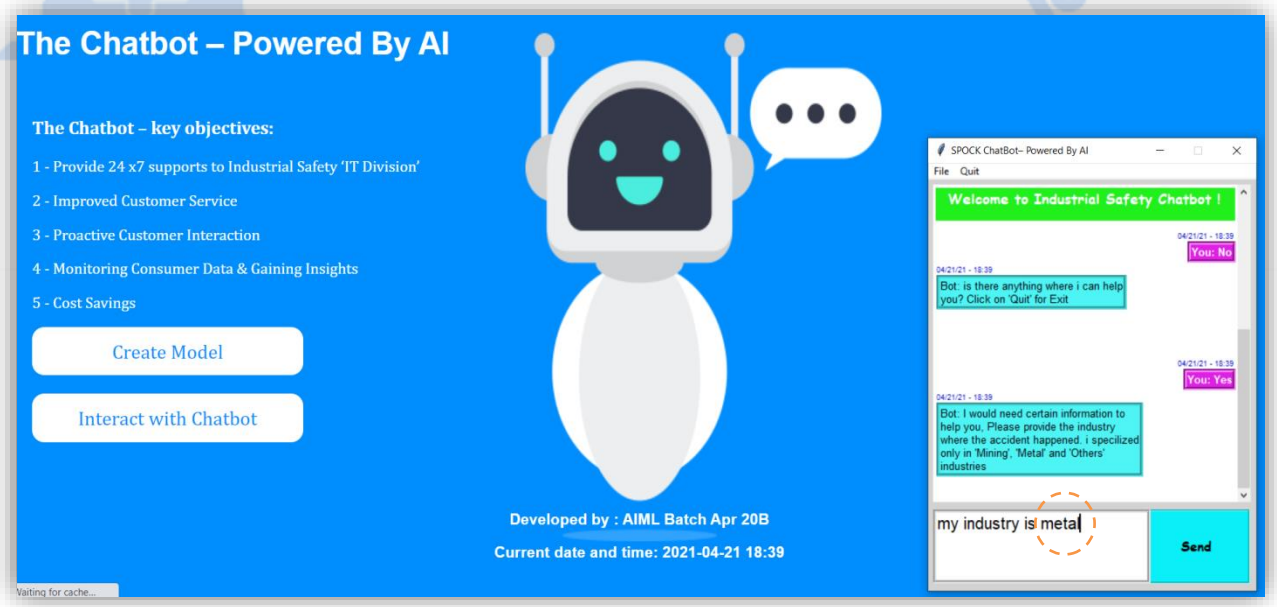


15.4.3 Enter the Industry Type

User now can type any of the 3 industries as a free text e.g.

- My industry is Metal
- I works in metal industry
- Metal is my industry

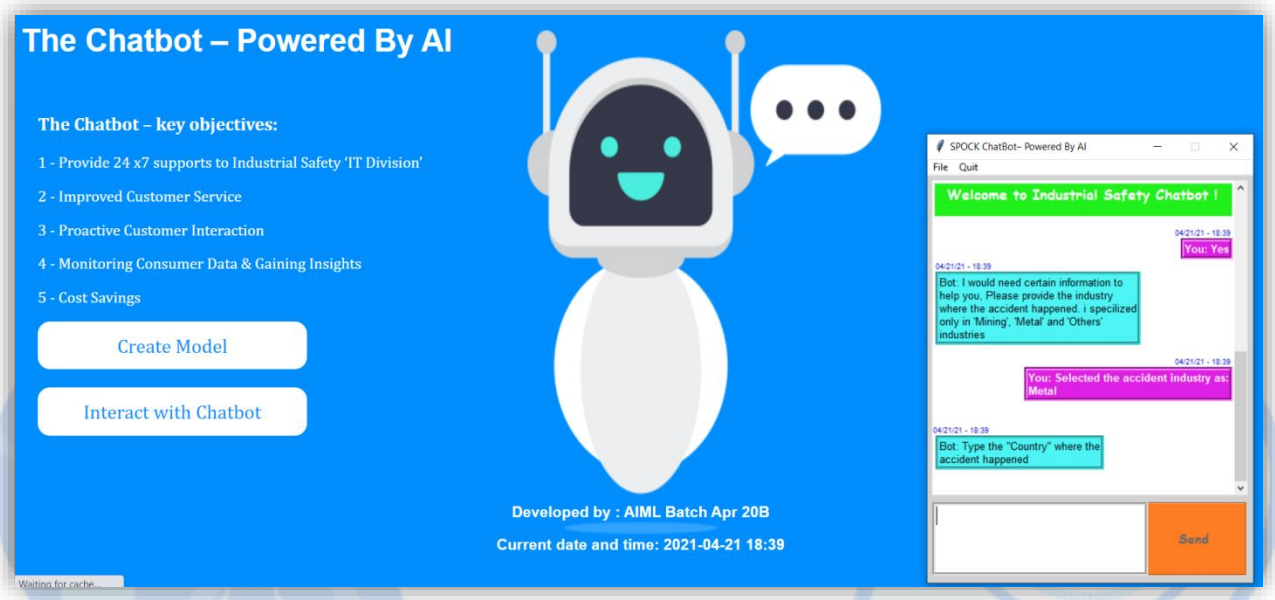
If user put any other text apart from 3 (Mining, Metal and Others) bot will not accept the input and wait for the correct input from the user.



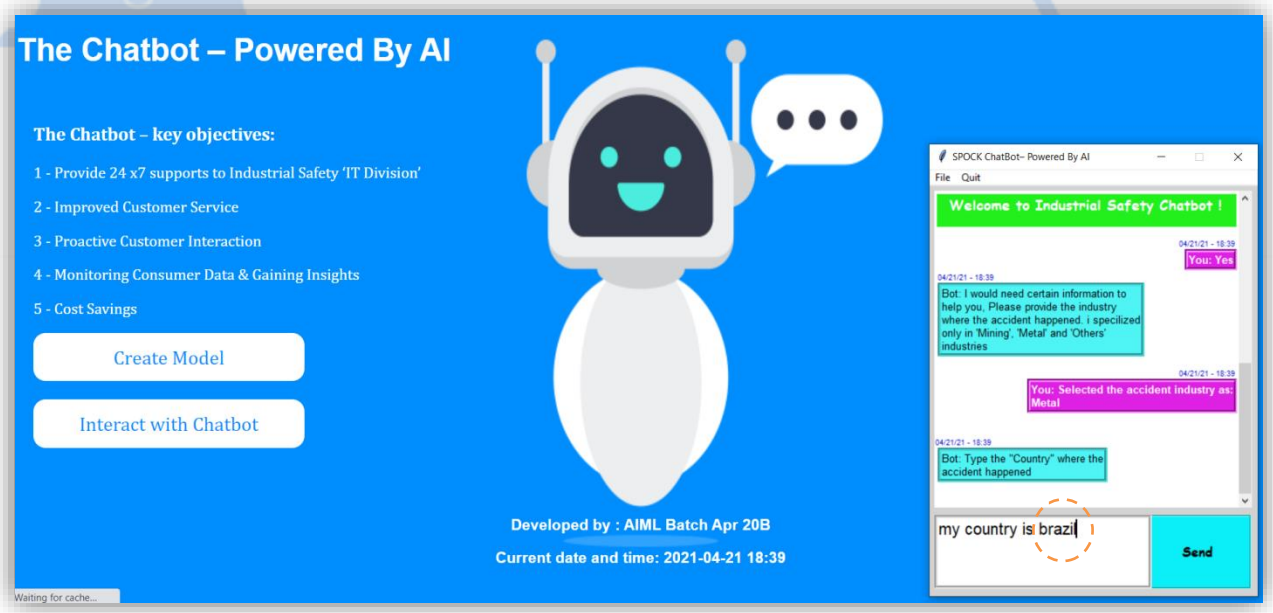
15.4.4 Enter the Country

From the user provided input bot will extract the industry type as Mining / Metal / Others and revert as message "**Selected the industry as Metal**"

Next, it will ask user to enter the country where the accident happened.



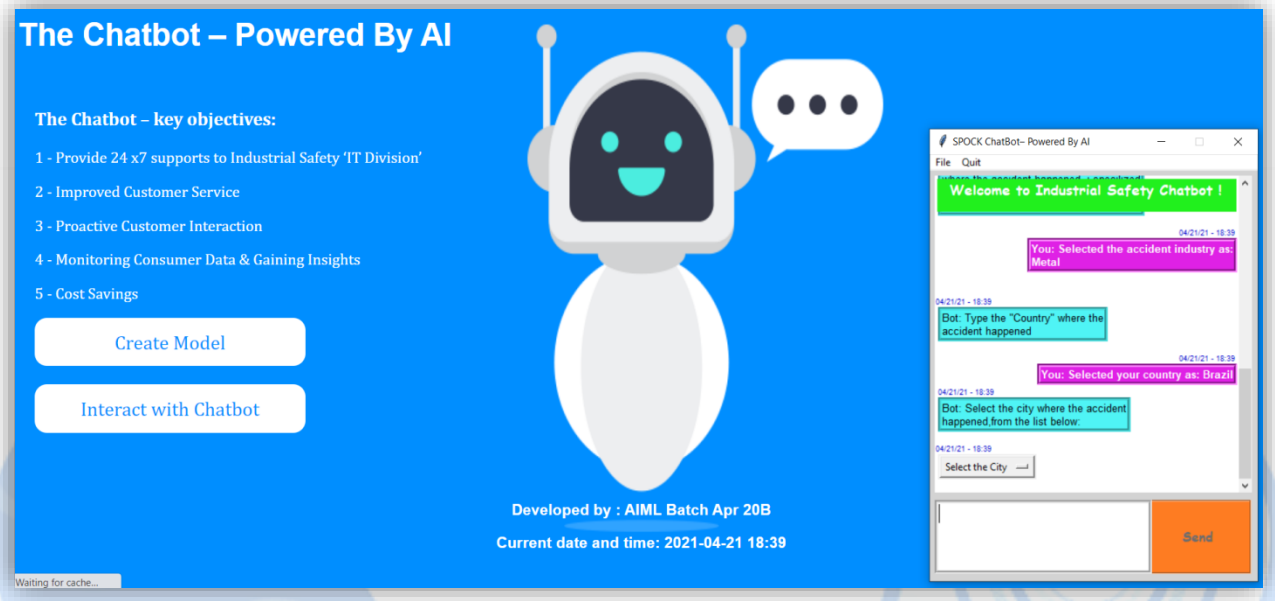
User now writes the country as a free text as shown below: "**my country is brazil**". We have used the NER (**Name Entity Recognition**) and it will extract the country name from the text.



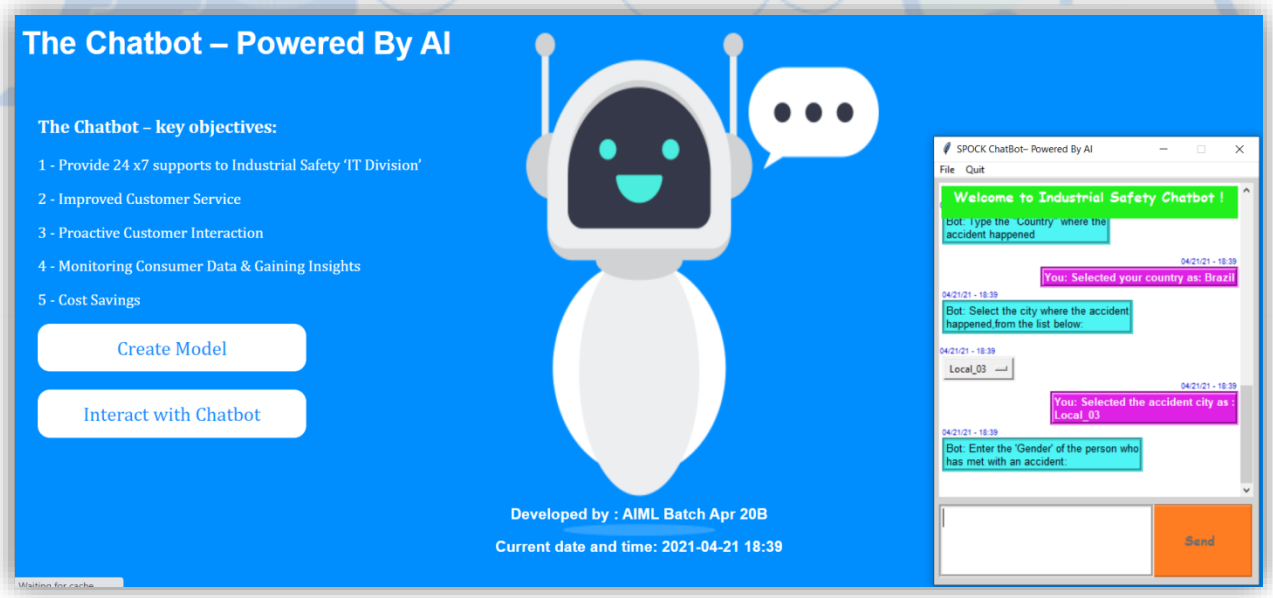
User will now get a response “Selected the country as “Brazil”

15.4.5 Select the City

Upon selection of the country user will now be asked to choose the city where the accident happened. It will be drop down menu and user can select it from any of the 12 cities mentioned in the drop down.

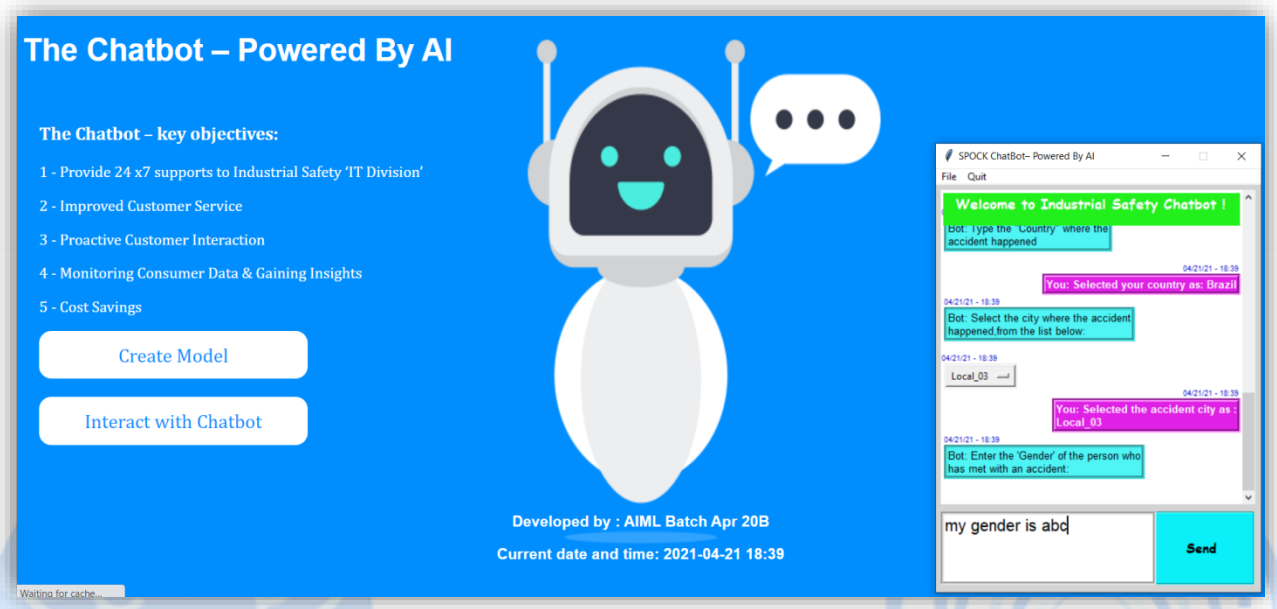


As we can see in the below picture user have selected the city as local_03, now bot will ask user to enter the gender of the person who has met with the accident.

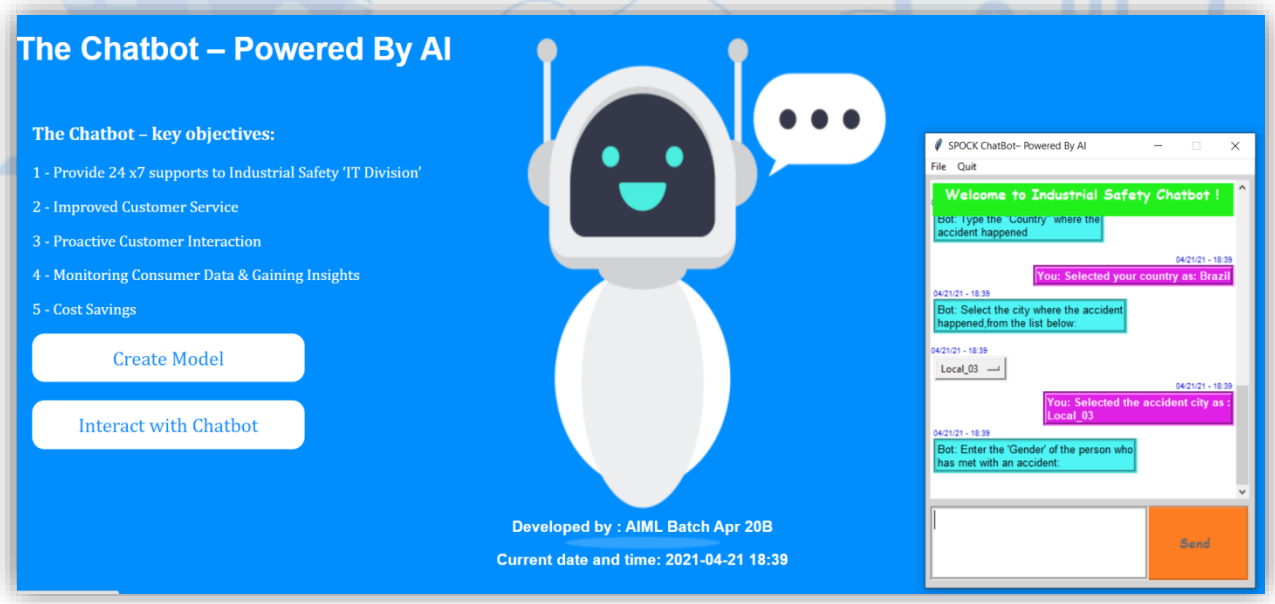


15.4.6 Enter the Gender Type

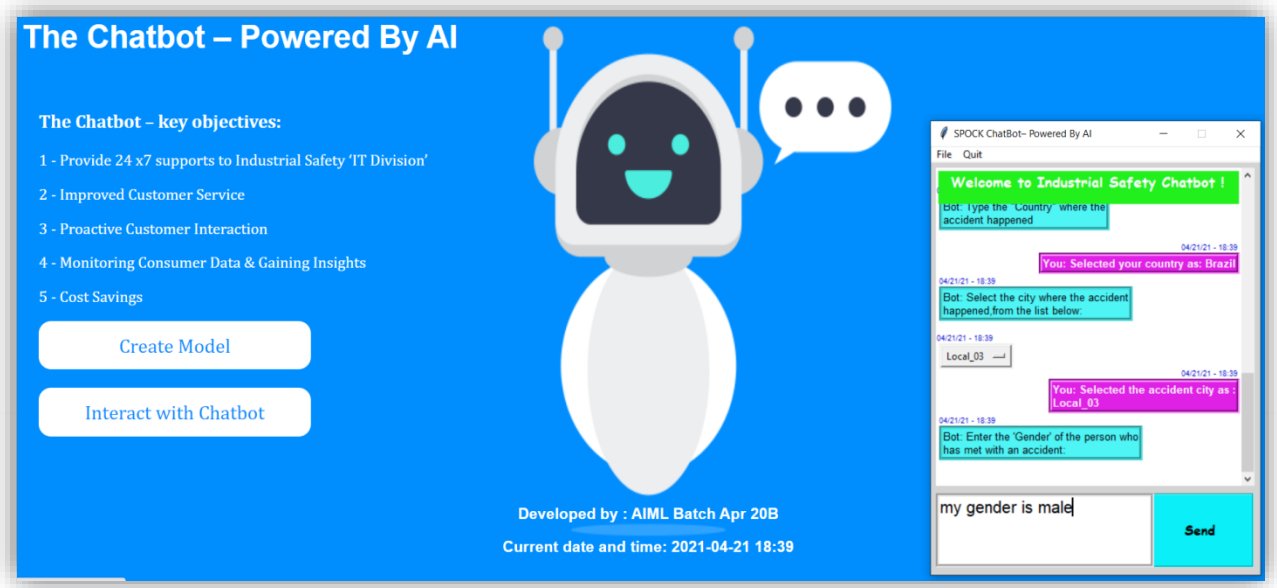
User now can enter in the text box e.g. my gender is male / female. However, if user put anything else apart from male or female bot will not accept the input.



Bot will wait for the correct input from the user, wrong information keyed by user will be removed and input box will be open for the user inputs.

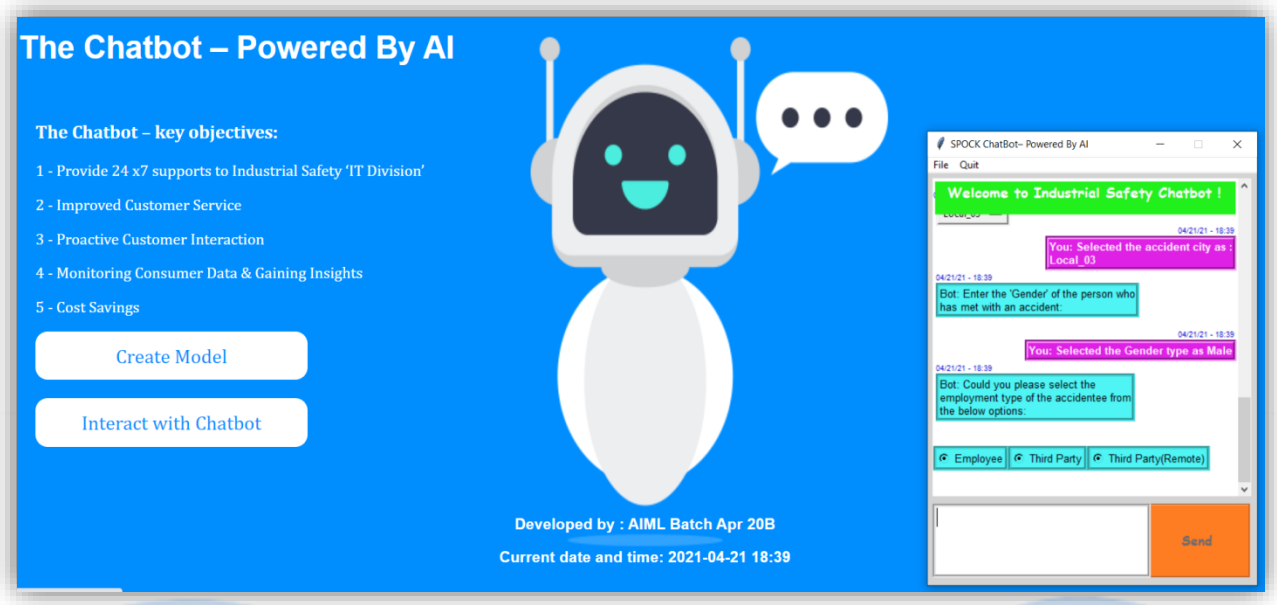


Upon correct information from the user, Bot will proceed further



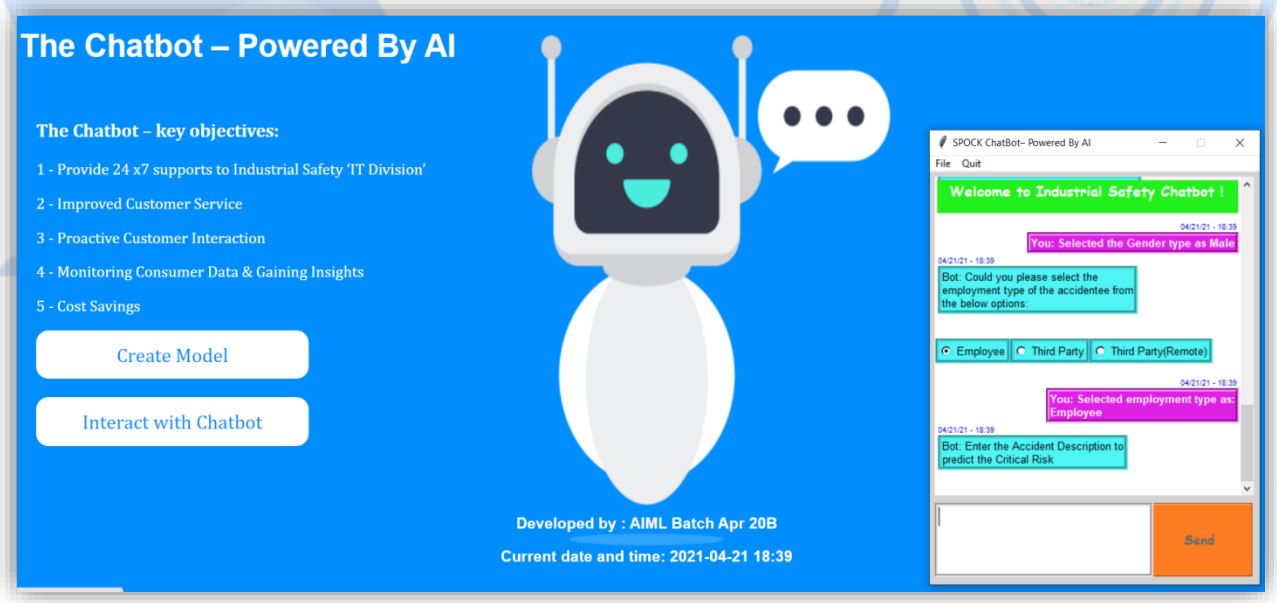
Bot will not extract the gender type from the user text as **Male** / **Female and** shows "Selected the gender type as Male / Female based on the inputs.

Now Bot will ask user to choose the employment type from the radio button out of 3 options e.g "Employee", "Third Party", "Third Party (Remote)"



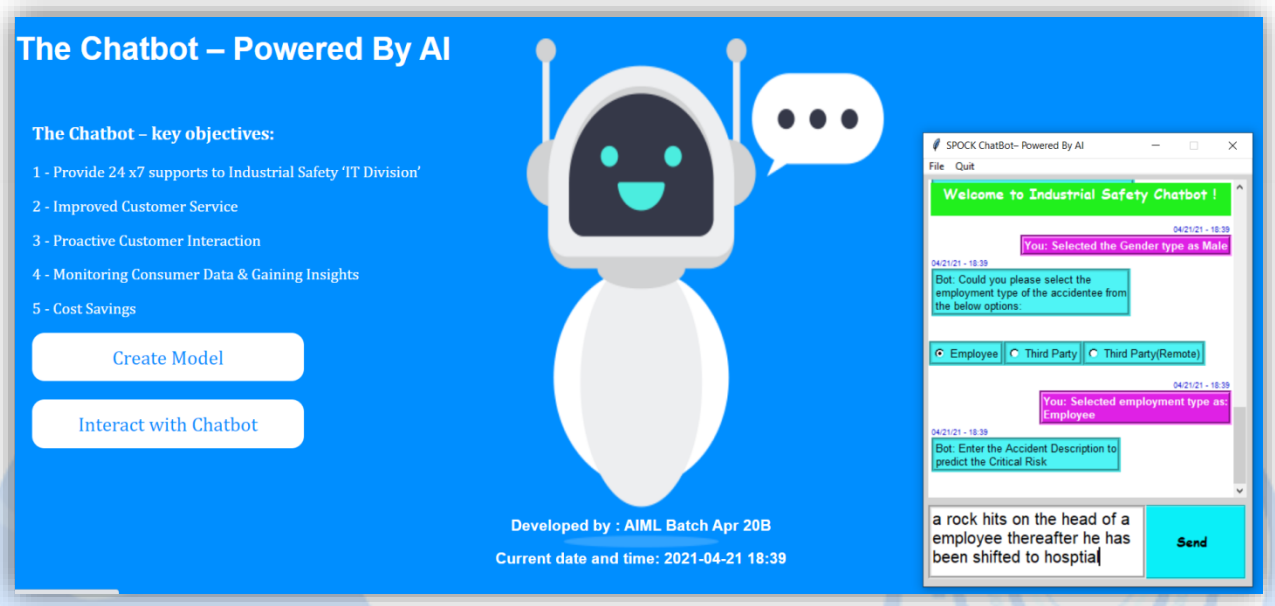
User now can select the employment type and basis which a response will be send by bot as “Selected employment type as “Employee” or Third Party or Third Party(Remote)

Now Bot will ask user to enter the Accident description.



15.4.7 Enter the Accident Description

User now can put the accident description in own words in the text box and click on send button to predict.

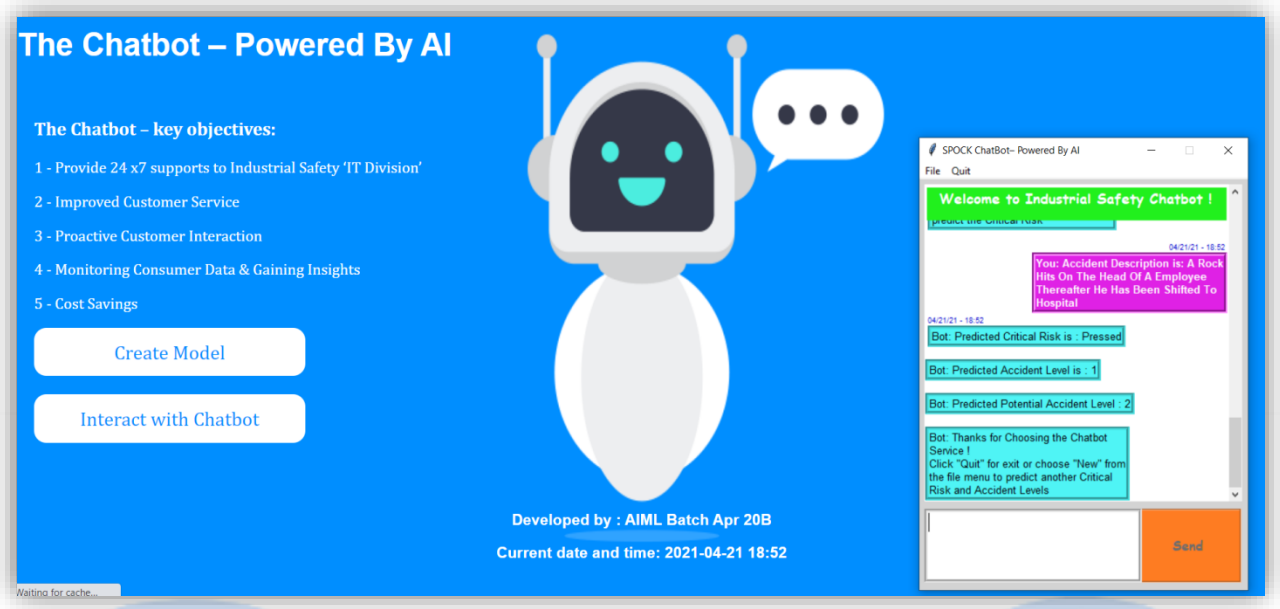


On the accident description inputs Bot will not predict:

- Critical Risk
- Accident Level
- Potential Accident Level

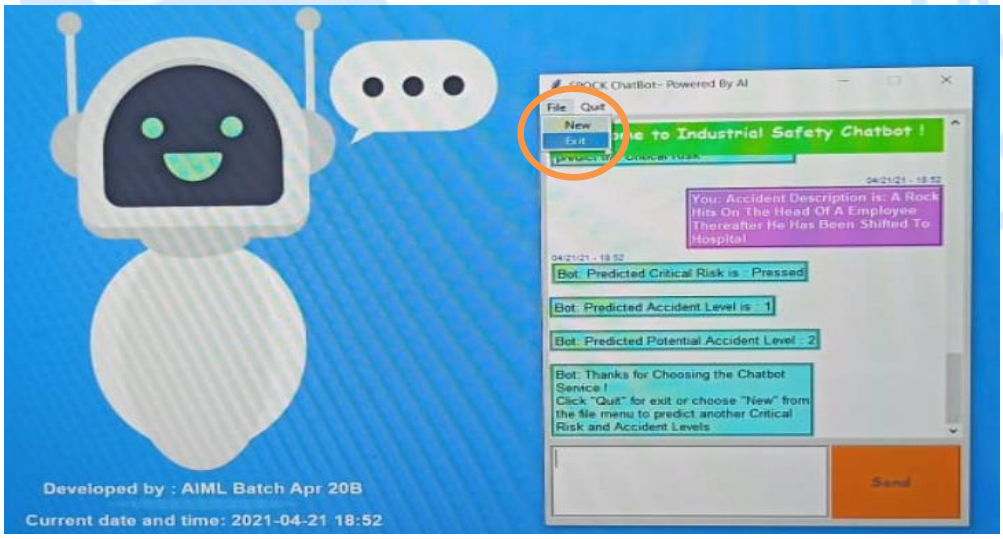
And show it on the chat interaction window. There will be thanks message to the user saying

"Thanks for choosing the Chatbot Service!"



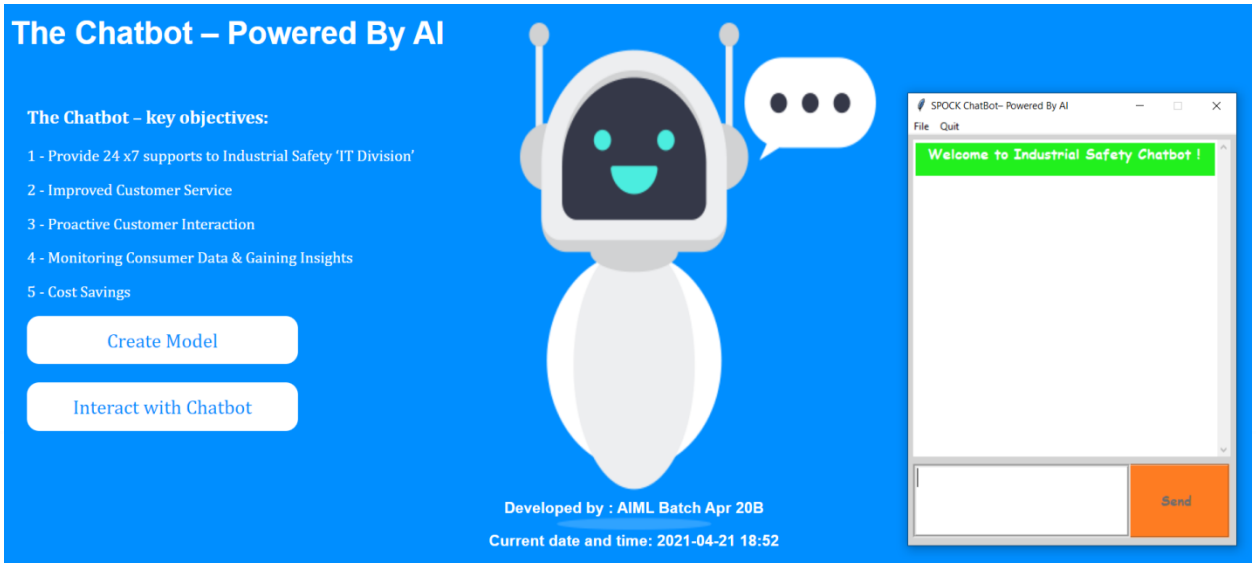
15.4.8 Menu Options

Now user has an option to quit from the menu option or click on the file menu to start a new interaction with Chatbot.



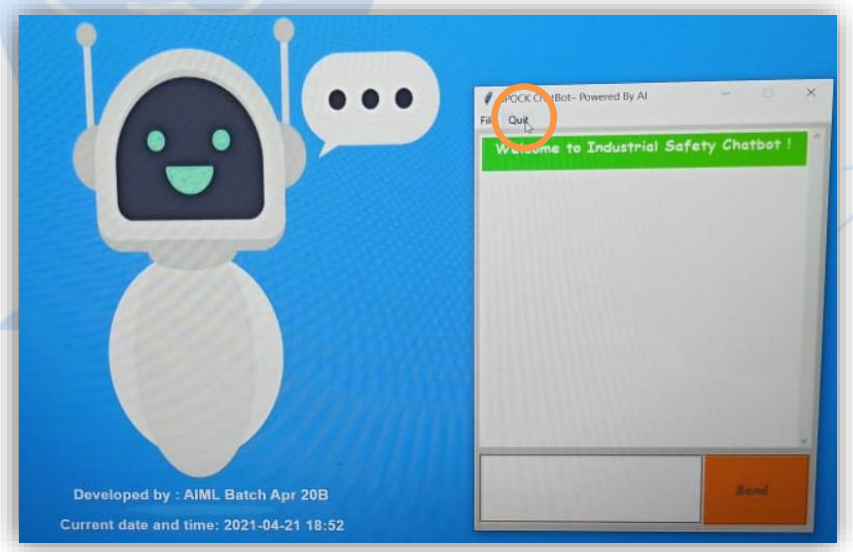
15.4.9 New Chat Session

One click on New a new session will be started

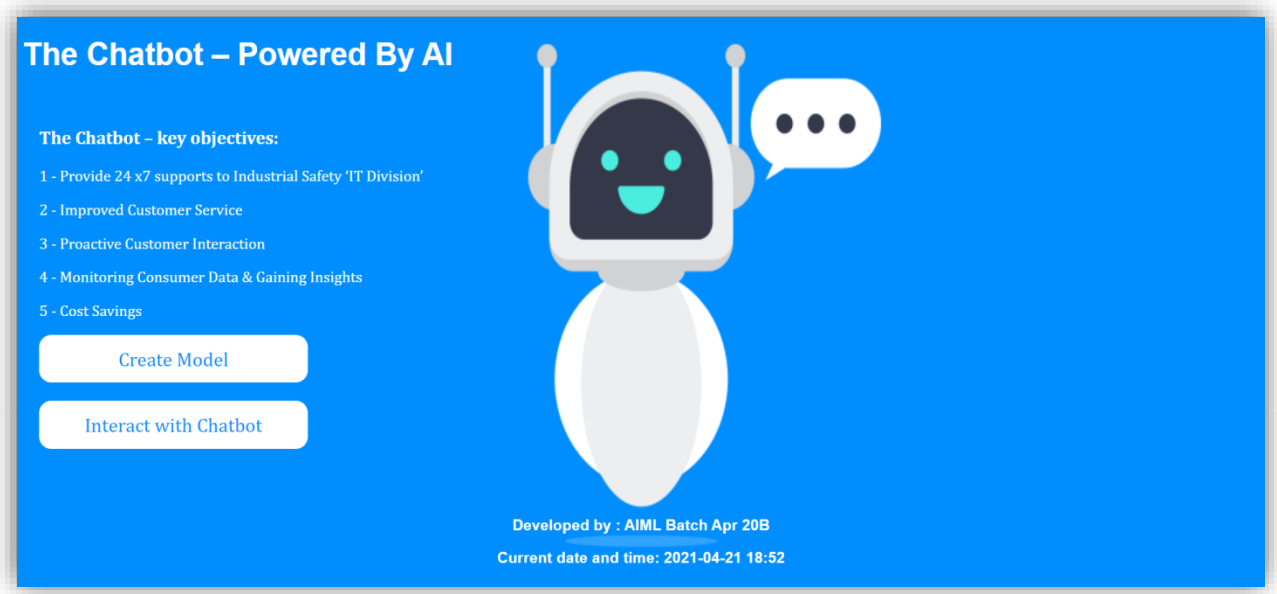


15.4.10 Quit

If user wishes to quit the Chatbot, he / she can simply click on quit button from the menu options.



Now the Chatbot window will be disappeared.



15.5 Natural Language Understanding (NLU)

Natural Language understanding is the important library which is being used to develop conversational Chatbot. It helps user to provide their input in user understandable language. NLP component extracts the intent and entities from the conversation and plan for actions and Chatbot responses based on the user inputs. This is a paradigm shift from old system where user had to inform safety helpdesk and fill up a form to provide the inputs. In addition to this, user also provides free text description of accidents which helps the Chatbot for better predictions

15.5.1 Intent

Intents are used to define what business want a Chatbot to respond with when it picks up the intention of a user, or when Chatbot want to trigger a response based off of some other event.

For example, if a user says 'Hi', we want Bot to respond with 'Hello' / 'Hi'.

Here the intent of customer is "Greetings" and Chatbot understands the user intent as greetings.

15.5.2 Entities

Entities are knowledge repositories used by the Chatbot to provide personalized and accurate responses. With Entities we can easily extract important information from the ongoing conversation, such as country, gender, Industry type or anything we want. Use them when we want our Chatbot to catch important data.

15.6 Machine Learning / Neural Network and NLP Model

We have used models at three places:

- Neural network model is being used to predict the intent of the user and respond based on the predicted intent
- Name entity recognition is being used from "**spacy**" to recognize the country etc.
- For Critical Risk, Accident Level and Potential Accident Level prediction we have used following models:
    - Machine Learning Models:
        - LogisticRegression
        - GaussianNB
        - KNeighborsClassifier
        - SVC
        - DecisionTreeClassifier
        - BaggingClassifier
        - AdaBoostClassifier
        - GradientBoostingClassifier
        - RandomForestClassifier
    - Neural Network Model with glove.6B.200d embedding
    - LSTM Classifier with glove.6B.200d embedding

All of above model have been tried and the performance is being evaluated to improve the accuracy. More detailing of the models is being done in section 11 & 12

15.7 Flask

Once the model is trained, it is saved to publish using on a **HTML** page using F**lask**. Then interface is being used to initialize the Chatbot on a single click where user can provide inputs related to the accident and get the immediate response.

15.8 Database

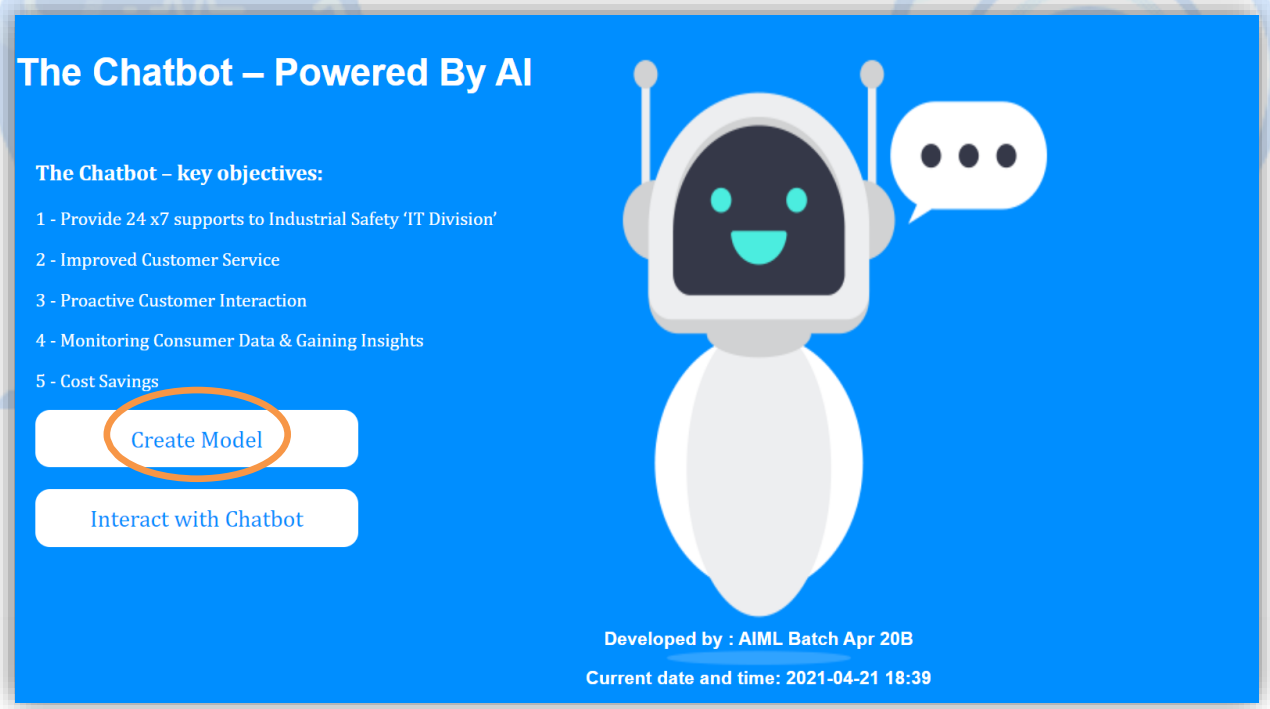
Pandas data frame is being used to save runtime dataset and offline it is exported in csv files, it will help analyzing the different type of user inputs and thereby fine-tuning in model for better accuracy on prediction

16 Create Model - Auto

We will now talk about another functionality of auto modeling. Where we have tried to automate few of the steps like:

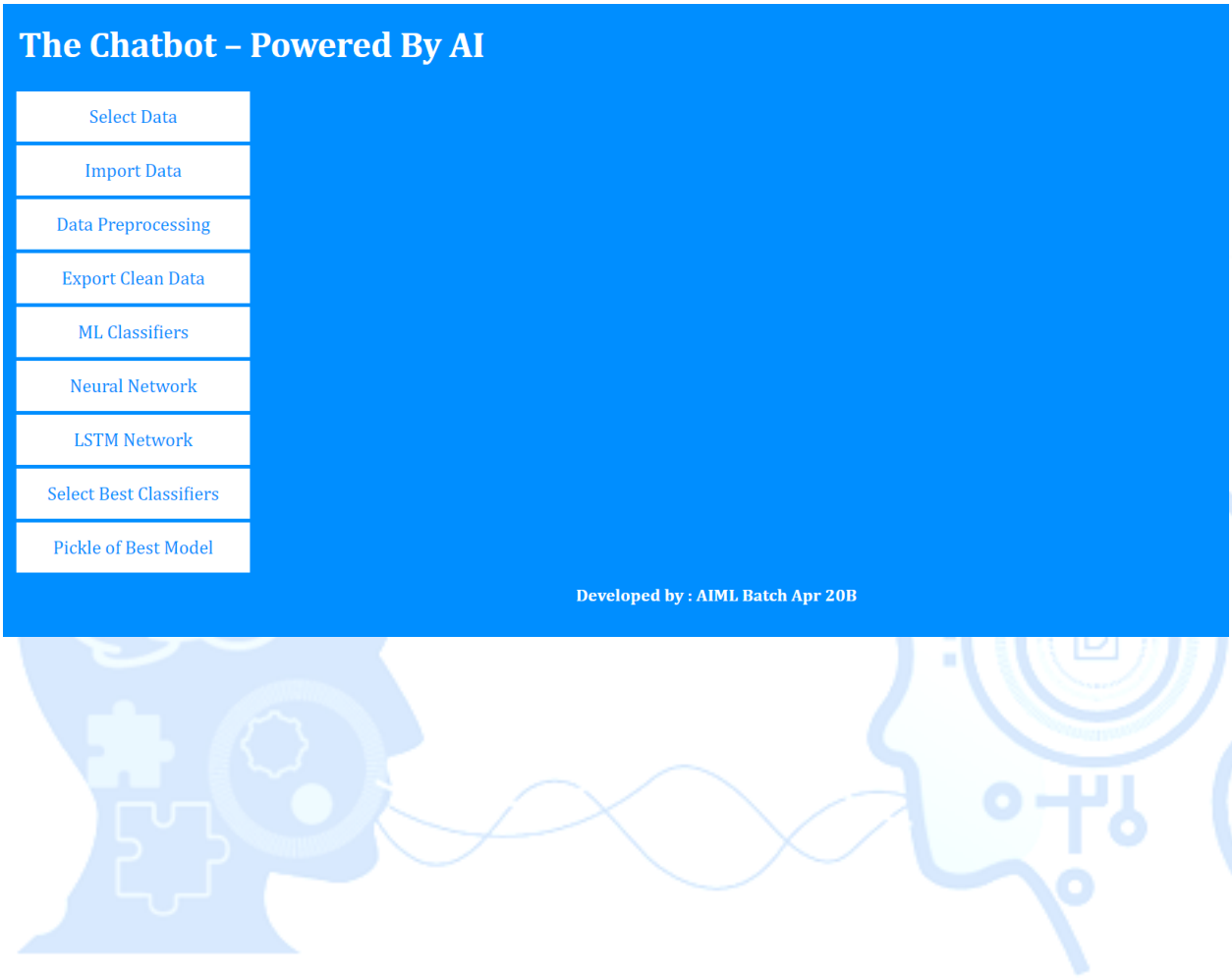
- Selection of the data from the drive
- Import the data and save into a dataframe
- Data Preprocessing
- Export Clean Data
- ML Classifier
- Neural Network Classifier
- LSTM classifier
- Selection of Best Classifier
- Pickling the Model

On the home screen there is another button (Marked in below image) called “Create Model”. On click of this button it will be redirected to another HTML page.



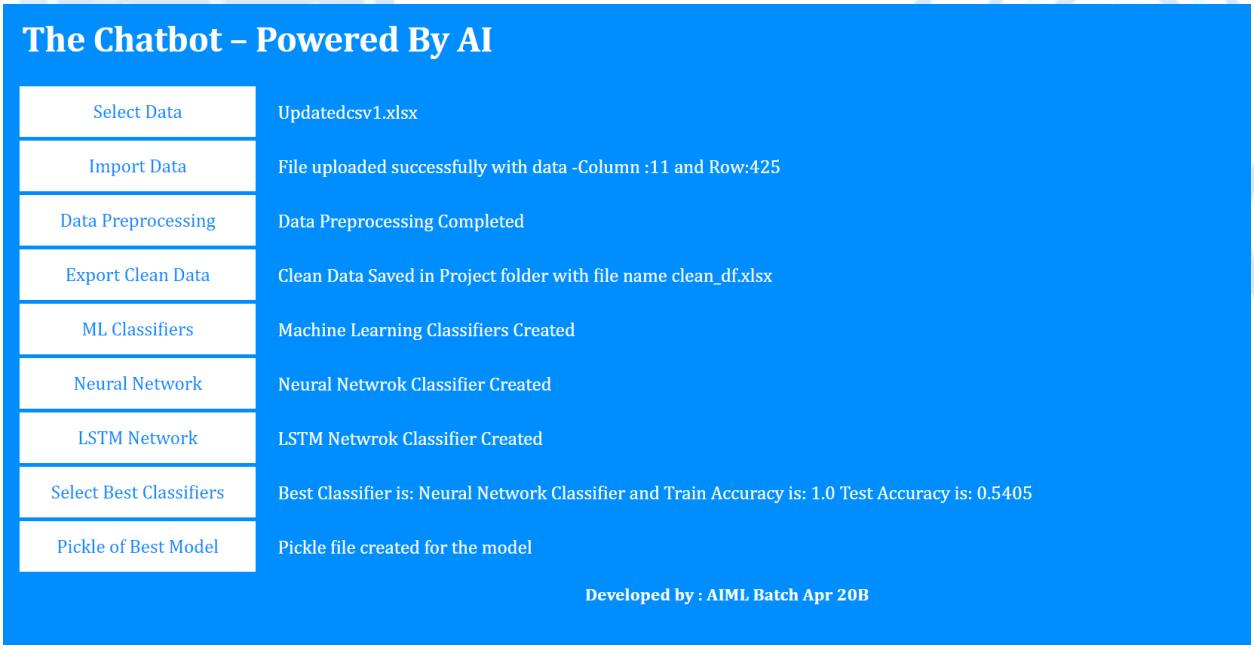
16.1 Create - Auto Model

Below screen is page of Create auto model page where user will see multiple buttons to create a prediction model in few clicks.



16.2 Steps to follow

- Step 1: User to click on the **Select Data** button where user needs to select the data file from the drive, On selection user will get a message in front of the button itself with the name of the file selected e.g. “Updated_Data.xlsx”
- Step 2: **Import Data,** On the click user will see a corresponding message “File Uploaded successfully with Data – Columns : xx and Row : xx
- Step 3: **Data Preprocessing**, in this EDA activities will be performed e.g. Null Value Check, Duplicate Value check etc.
- Step 4: **Export Clean Data,** Now the clean data will be saved in the project folder
- Step 5: **ML Classifier,** Here the machine learning algorithms will be created
- Step 6: **Neural Network Classifier,** on the click of this button Neural Network classifier will be created
- Step 7: **LTSM Network,** LSTM Classifier will be created on the click of this button
- Step 8: **Select Best Classifier**, Here it will check the Test accuracy of all the models and best accuracy model will be displayed in message
- Step 9: **Pickle the Best Model,** it will now pickle the best model



17 Installation Guide

- Create new folder in Jupyter installation directory. Example- Chatbot , Will use Chatbot in further documentation
- Copy the following file in Chatbot directory folder :
    - glove.6B.200d.txt
    - intents.json
    - ChatBot_Prediction_Model.ipynb
    - Chatbot_Intenface.ipynb
    - Install the entire required library. List of required library are mentioned in Jupyter notebook
- Crate two new sub folder in Chatbot directory with the name of “static” and “templates”
- Copy the image file(Chat1.png and loader.gif) in static folder
- Copy the html file(homepage.html and importdata.html) in templates folder
- We are using the JS library on importdata.html, therefore internet should be mandatory requirement. Url is mentioned below :

  https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js">

- Copy the updated data file with .xlsx extension
- Run the flask code given at the end of the notebook

18 Way Forwards

There are still a scope of improvement in few of the concepts which can be in done in the V2.0 and not being covered in V1.0 due to time limitations:

- Validation on Train model page
- <to be updated>