NATURAL LANGUAGE PROCESSING CHATBOT INTERFACE

CAPSTONE PROJECT GROUP 4 - FINAL REPORT

2021-2022

SUBMITTED BY

Krishnakant Reddy B

Mahadeva Reddy

Pradeebha Benildus

Shyam Kumar

Vemula Ganesh

Vishal Verma

Project Mentor: Shyam Muralidharan

Great Lakes Institute of Management

CONTENTS

Abstract

Chapters

1. **INTRODUCTION**
   1.1. Problem Statement
   1.2. Objective
   1.3. Data Source
2. **OVERVIEW OF THE FINAL PROCESS**
3. **EXPLORATORY DATA ANALYSIS AND DATA PRE-PROCESSING**
   3.1. Data Pre-Processing
   3.2. Exploratory Data Analysis
4. **MODEL BUILDING AND EVALUATION**
   4.1. Models Explored and Building
   4.2. Model Summary
5. **BENCHMARKING- COMPARISON OF EXPERIMENTS**
   5.1. Model Performance and Comparisons
   5.2. Final Model Selection
6. **GUI BUILDING**
   6.1. Building API
   6.2. Chatbot
7. **IMPLICATION- BUSINESS VALUE DERIVED**
8. **LIMITATIONS AND SCOPE OF IMPROVEMENT**
   8.1. Limitations of Data
   8.2. Scope of Improvement
9. **CLOSING REFLECTION**
10. **FINAL NOTE**
11. **REFERENCES**

Abstract

A chatbot is a software or computer program that simulates human conversation or "chatter" through text or voice interactions.

Users in both business-to-consumer (B2C) and business-to-business (B2B) environments increasingly use chatbot virtual assistants to handle simple tasks. Adding chatbot assistants reduces overhead costs, uses support staff time better, and enables organizations to provide customer service during hours when live agents aren't available.

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 a text interface.

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

There are mainly two approaches used to design the chatbots, described as follows:

In a ***Rule-based*** approach, a bot answers questions based on some rules on which it is trained. The rules defined can be very simple to very complex. The bots can handle simple queries but fail to manage complex ones.

Self-learning bots are the ones that use some Machine Learning-based approaches and are definitely more efficient than rule-based bots. These bots can be further classified into two types: Retrieval Based or Generative

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

Traditional chatbots: Traditional chatbots are driven by system and automation, mainly through scripts with minimal functionality and the ability to maintain only system context.

Current chatbot: Current chatbots are driven by back and forth communication between the system and humans. They have the ability to maintain both system and task contexts.

Future chatbot: Future chatbots can communicate at multiple levels with automation at the system level. They have the ability to maintain the system, task, and people contexts. There is a possibility of the introduction of master bots and eventually a bot OS.

1. INTRODUCTION

This capstone project is based on designing an ML/DL-based chatbot utility that can help the professionals to highlight the safety risk as per the incident description.

1.1 Problem Statement

Domain: Industry Safety. NLP-based Chatbot.

Context:

The database comes from one of the biggest industries in Brazil and in the world. 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 an environment.

Data Description:

This database basically records accidents from 12 different plants in 03 different countries where every line in the data is an occurrence of an accident.

Columns Description:

- ***Data*****:** timestamp or time/date information
- ***Countries*****:** which country the accident occurred (anonymized)
- ***Local*****:** the city where the manufacturing plant is located (anonymized)
- **Industry sector*:*** which sector the plant belongs to
- ***Accident level*****:** from I to VI, it registers how severe was the accident (I mean not severe but VI means very severe)
- ***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)
- ***Gender*****:** if the person is male or female
- ***Employee or Third Party*****:** if the injured person is an employee or a third party
- ***Critical Risk*****:** some description of the risk involved in the accident
- ***Description*****:** Detailed description of how the accident happened.

1.2 Objective

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

1.3 Data sources

Link to download the dataset:

https://drive.google.com/file/d/1_GmrRP1S2OIa02KlfOBNkYa8uxazGbfE/view?usp=sharing

Original dataset link:

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

2. OVERVIEW OF THE FINAL PROCESS

The brief approach for the solution is given as follows:

1. The above objective defines that the end solution requires a model building based that predicts the Potential Accident Level by providing a description of the accident in text format
2. The first step is Data Cleansing and Data-Preprocessing to have good cleaned data for the input dataset for the model to predict the expected output
3. After the pre-processing of the dataset, the EDA (exploratory data analysis) has been given to understand the dataset more deeply and to identify trends and patterns which also help us to understand the structure of the dataset
4. After the EDA, the model buildings have been given in which we have built ML-based models, ANN-based models, LSTM, CNN + LSTM, BERT, Simple Transformer Model. A detailed summary and evaluation of all the models are given as well to compare the results and performance
5. The benchmarking of the outcome and comparison has been captured. The final selection of the model is also given
6. The layout of the GUI-based chatbot has been given with a sample prediction
7. The business value derived based on the outcome of the model is analyzed and covered
8. Limitations of the model and scope of improvement have been covered
9. The closing reflection and final note have been covered which includes the topics and lessons learned in each of the steps

3. DATA PRE-PROCESSING AND EXPLORATORY DATA ANALYSIS

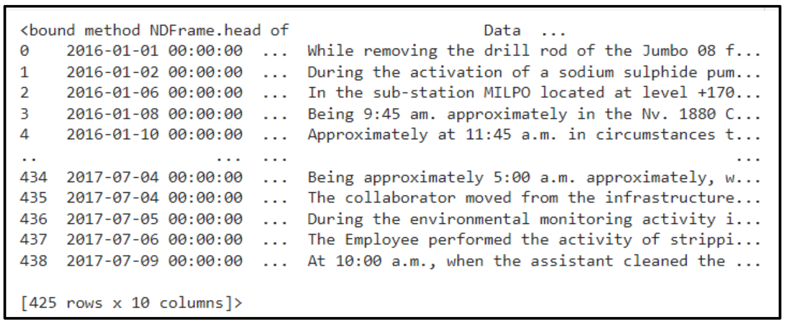
3.1 Data Pre-Processing

Data preprocessing is a data mining technique that is used to transform the raw data in a useful and efficient format.

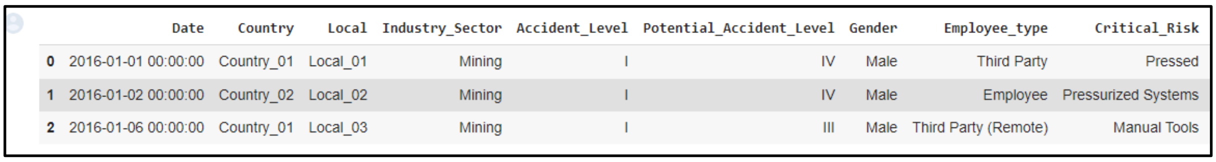
3.1.1. Importing data and Re-naming columns:

We Imported the Dataset from the given CSV file and the size of the dataset is 425 * 10. The Column names of imported data are not in the prescribed format for python libraries. So renaming it with the proper names.

Input data -“`DataSet-industrial_safety_and_health_database_with_accidents_description.csv`”



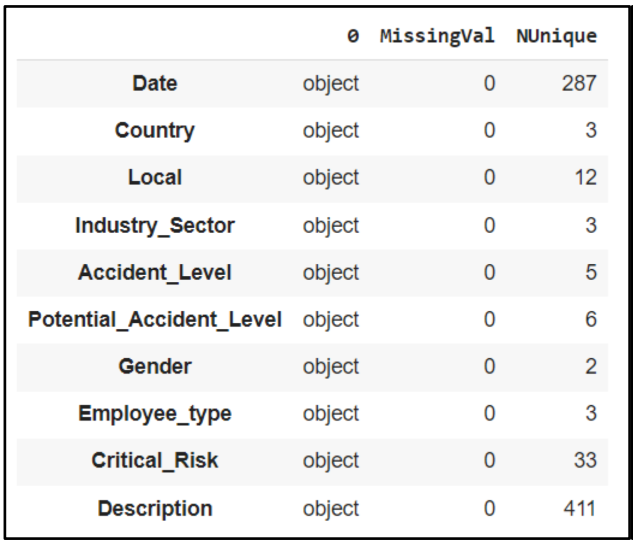
```
<bound method NDFrame.head of                        Data  ...
0    2016-01-01 00:00:00  ...  While removing the drill rod of the Jumbo 08 f...
1    2016-01-02 00:00:00  ...  During the activation of a sodium sulphide pum...
2    2016-01-06 00:00:00  ...  In the sub-station MILPO located at level +170...
3    2016-01-08 00:00:00  ...  Being 9:45 am. approximately in the Nv. 1880 C...
4    2016-01-10 00:00:00  ...  Approximately at 11:45 a.m. in circumstances t...
..                   ...  ...                                               ...
434  2017-07-04 00:00:00  ...  Being approximately 5:00 a.m. approximately, w...
435  2017-07-04 00:00:00  ...  The collaborator moved from the infrastructure...
436  2017-07-05 00:00:00  ...  During the environmental monitoring activity i...
437  2017-07-06 00:00:00  ...  The Employee performed the activity of strippi...
438  2017-07-09 00:00:00  ...  At 10:00 a.m., when the assistant cleaned the ...

[425 rows x 10 columns]>
```



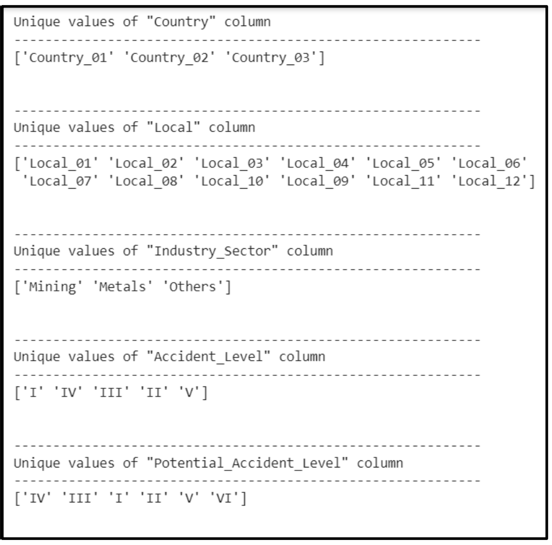
	Date	Country	Local	Industry_Sector	Accident_Level	Potential_Accident_Level	Gender	Employee_type	Critical_Risk
0	2016-01-01 00:00:00	Country_01	Local_01	Mining	I	IV	Male	Third Party	Pressed
1	2016-01-02 00:00:00	Country_02	Local_02	Mining	I	IV	Male	Employee	Pressurized Systems
2	2016-01-06 00:00:00	Country_01	Local_03	Mining	I	III	Male	Third Party (Remote)	Manual Tools

3.1.2. Checking Missing Values and outliers:

By Checking the missing values we found there are no null values and outliers in the dataset. We have only 3 countries, 12 Locals, 3 Industry sectors, 5 accident levels, 6 potential accident levels, 3 Employee types, and 33 Critical Risks including Not applicable.



	0	MissingVal	NUnique
Date	object	0	287
Country	object	0	3
Local	object	0	12
Industry_Sector	object	0	3
Accident_Level	object	0	5
Potential_Accident_Level	object	0	6
Gender	object	0	2
Employee_type	object	0	3
Critical_Risk	object	0	33
Description	object	0	411



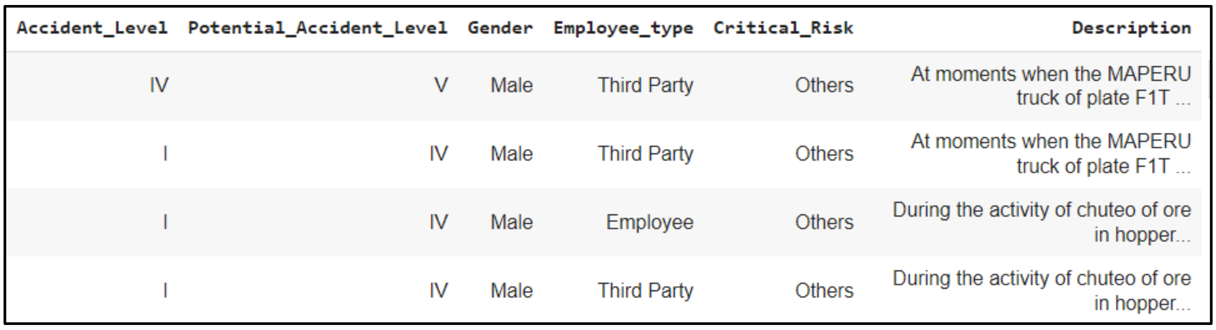
```
Unique values of "Country" column
----------------------------------------------------------------
['Country_01' 'Country_02' 'Country_03']


----------------------------------------------------------------
Unique values of "Local" column
----------------------------------------------------------------
['Local_01' 'Local_02' 'Local_03' 'Local_04' 'Local_05' 'Local_06'
 'Local_07' 'Local_08' 'Local_10' 'Local_09' 'Local_11' 'Local_12']


----------------------------------------------------------------
Unique values of "Industry_Sector" column
----------------------------------------------------------------
['Mining' 'Metals' 'Others']


----------------------------------------------------------------
Unique values of "Accident_Level" column
----------------------------------------------------------------
['I' 'IV' 'III' 'II' 'V']


----------------------------------------------------------------
Unique values of "Potential_Accident_Level" column
----------------------------------------------------------------
['IV' 'III' 'I' 'II' 'V' 'VI']
```

In Accident Level and Potential Accident Level, Five types of accident levels (1 to 5) are present. But, Six types of Potential Accident Levels (1 to 6) are there and there is only one value registered under 'Potential Accident level 6'. So Replace it with level 5.

3.1.3. Removing Duplicates:

By Checking the Duplicates, we have 7 duplicate values on the entire dataset, let us drop them.

By Checking the 'Description' Column alone as a subset we can see 14 Duplicates since the unique value in the Description is 411 and the total value is 418 after removing duplicates. So dropping the second instance(7 rows).



Accident_Level	Potential_Accident_Level	Gender	Employee_type	Critical_Risk	Description
IV	V	Male	Third Party	Others	At moments when the MAPERU truck of plate F1T ...
I	IV	Male	Third Party	Others	At moments when the MAPERU truck of plate F1T ...
I	IV	Male	Employee	Others	During the activity of chuteo of ore in hopper...
I	IV	Male	Third Party	Others	During the activity of chuteo of ore in hopper...

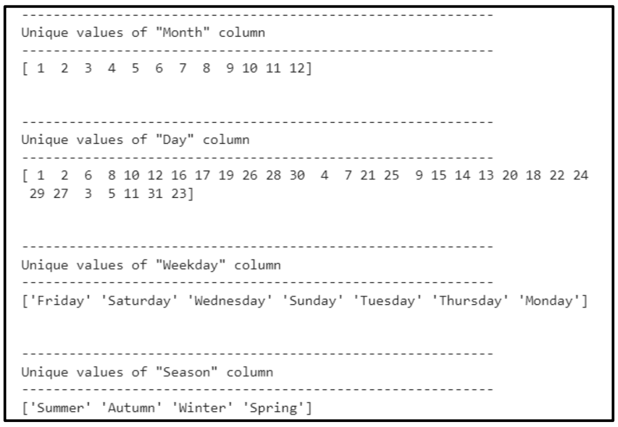
After removing the duplicates the size of the dataset is 411*10.

3.1.4. PreProcess - Time Series Data:

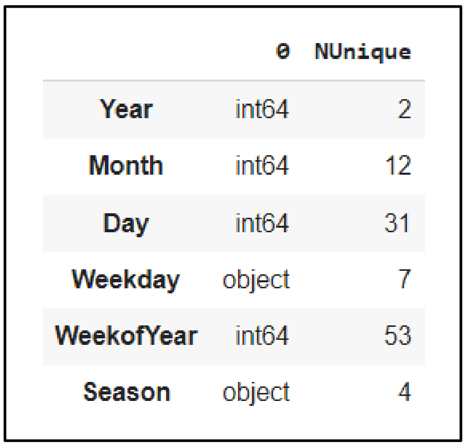
The Entire data was captured between January 2016 - September 2017. We split the Date columns into a date, month, year, day, weekday, and week of the year. The Countries where the dataset was collected are anonymized but they are all located in South America - Brazil. Brazil has four climatological seasons as below.

- ➢ Spring: September to November
- ➢ Summer: December to February
- ➢ Autumn: March to May
- ➢ Winter: June to August

Based on the month variable we created a new feature called a season. Then we checked unique values of all the time variables, based on the result we can conclude the accidents happen all days and months of the year.



```
------------------------------------------------------------
Unique values of "Month" column
------------------------------------------------------------
[ 1  2  3  4  5  6  7  8  9 10 11 12]


------------------------------------------------------------
Unique values of "Day" column
------------------------------------------------------------
[ 1  2  6  8 10 12 16 17 19 26 28 30  4  7 21 25  9 15 14 13 20 18 22 24
 29 27  3  5 11 31 23]


------------------------------------------------------------
Unique values of "Weekday" column
------------------------------------------------------------
['Friday' 'Saturday' 'Wednesday' 'Sunday' 'Tuesday' 'Thursday' 'Monday']


------------------------------------------------------------
Unique values of "Season" column
------------------------------------------------------------
['Summer' 'Autumn' 'Winter' 'Spring']
```



	0	NUnique
Year	int64	2
Month	int64	12
Day	int64	31
Weekday	object	7
WeekofYear	int64	53
Season	object	4

3.1.5. Text PreProcessing:

Text preprocessing is a method to clean the text data and make it ready to feed data to the model. Text data contains noise in various forms like emotions, punctuation, text in different cases.

For Text Preprocessing we created a separate python file called NLP_text_preprocess.py file and a Class called PreProcessing in it. We have several steps that need to be done for text Preprocessing. These steps are passed as arguments that can be controlled with the help of the config.py file which has all the step names as Boolean.

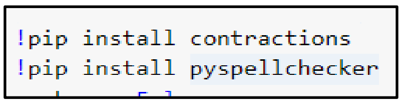
```
from NLP_text_preprocess import PreProcessing
```

We need to call our python file like above in our main file and the NLP_Text_preprocess file and config file need to be maintained in the same folder. In order to use this NLP_Text_preprocess we need to import two libraries beforehand, we are importing it.

The Steps Followed in Text Preprocessing in the same order:

- ➢ Convert into Lower case
- ➢ Removing the URL
- ➢ Removing the Special characters
- ➢ Remove Time Formats
- ➢ Expand Contradictions
- ➢ Removing Punctuation
- ➢ Removing Whitespaces
- ➢ Check the spelling mistakes
- ➢ Removing the Stop words
- ➢ Convert the words into lemma form

Libraries to install beforehand



```
!pip install contractions
!pip install pyspellchecker
```

Special Characters - 'å¼«¥ª°©ð±§µæ¹¢³¿®ä£' these characters are removed in our data.
Time Formats - 10:20.AM These time formats are not needed for our analysis, so we removed them.
Expand Contradictions - Contractions are words or combinations of words that are shortened by dropping letters and replacing them with an apostrophe.
Example:- Was not will be converted into wasn't. We are expanding the Contradictions.
Spelling Mistakes - We are using the Spell Checker library for commonly misspelled words.
Stop Words- Stop words are a set of commonly used words in a language. But, in our dataset, some stop words might play a vital role so we are not removing those words.

not_stop_words = ["not", "nor", "after", "before", "above", "below", "between"]

Lemmatization - 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. We use the WordNetLemmatizer library to convert the word into lemma form.

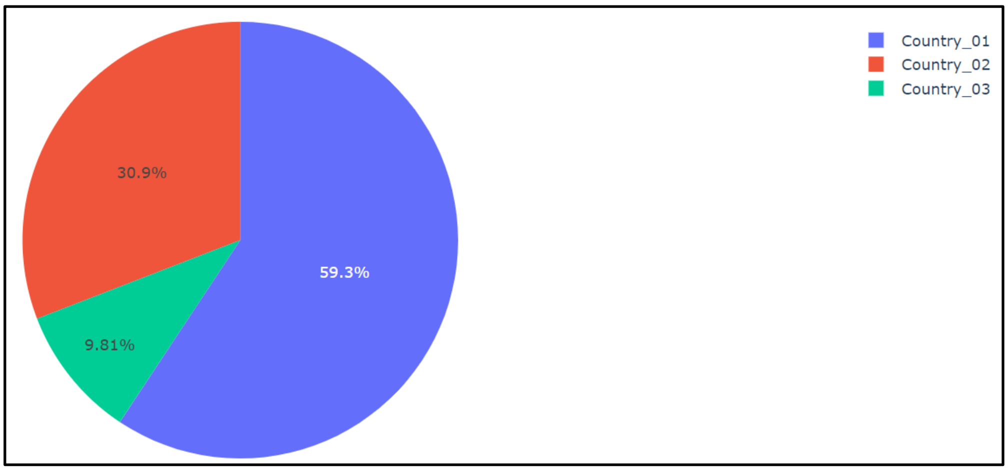
For Performing some operations we need to convert the sentences into tokens. After doing all the preprocessing steps we convert the tokens back into sentences and we are created a new column 'Description_Preprocessed' to have processed data.

Description	Year	Month	Day	Weekday	WeekofYear	Season	Description_preprocessed
Being 9:45 am. approximately in the Nv. 1880 C...	2016	1	8	Friday	1	Summer	approximately 1880 cx695 ob personnel begin ta...
Approximately at 11:45 a.m. in circumstances	2016	1	10	Sunday	1	Summer	approximately circumstance mechanic anthony gr...

3.2. Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, spot anomalies, test hypotheses, and check assumptions with the help of summary statistics and graphical representations.

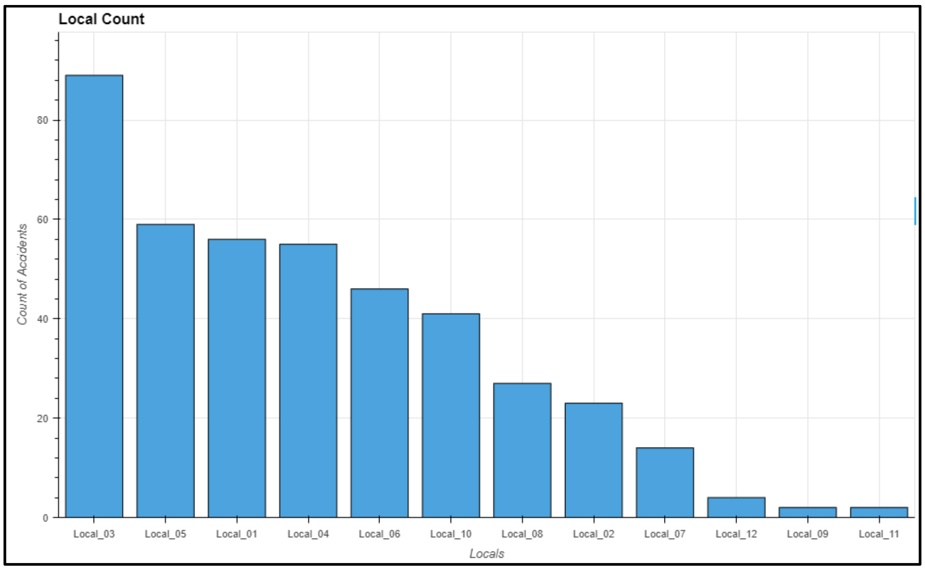
3.2.1. Distribution of Accidents Based on Country Level



In the given dataset, we have three countries, Country_01, Country_02, and Country_03. The proportion of accidents based on the Country level is as follows:

1. **Country_01**: Accidents- 248 i.e. **59.0%**
2. **Country_02**: Accidents- 129 i.e. **31.0%**
3. **Country_03**: Accidents- 41 i.e. **10.0%**

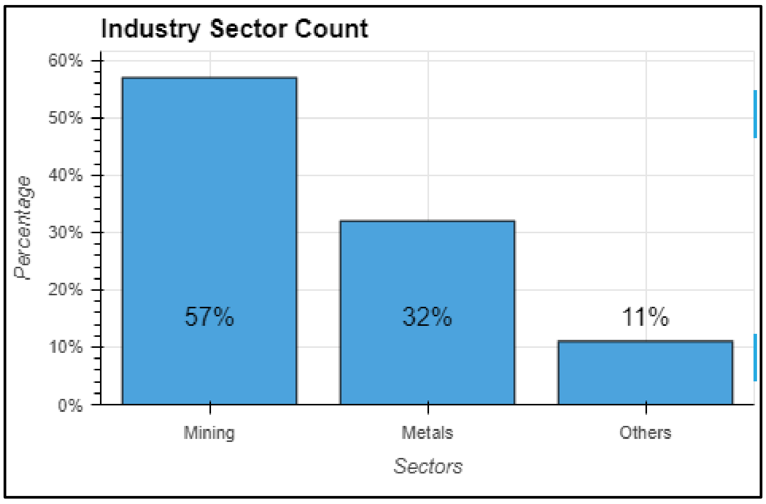
3.2.2. Distribution of Manufacturing Plants Based on Local



In the given dataset, we have a total of twelve locals where the manufacturing plants are located. The distribution of the manufacturing plants based on Local is shown in the above plot and the observations are as follows:

1. **A maximum** number of manufacturing plants are located in **Local_03**
2. **the Lowest** number of manufacturing plants are located in **Local_09**

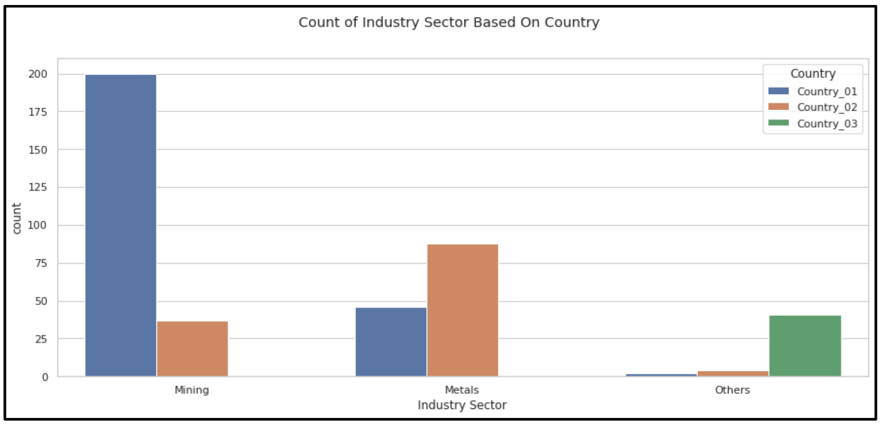
3.2.3. Distribution of Industry Based on Sector



The manufacturing plants are distributed into three types of Industry Sector, Mining, Metals, and Others. The distribution of these industries based on the sector is shown in the above plot and the observations are as follows:

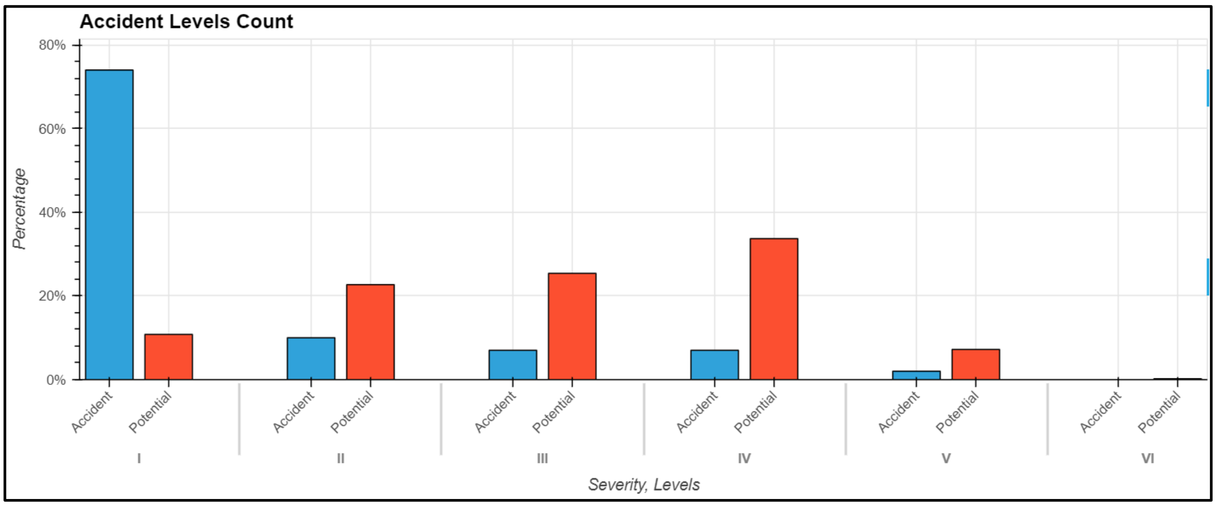
1. **Mining Sector**: Industry Count- 237 i.e. 57.0%
2. **Metals Sector**: Industry Count- 134 i.e. 32.0%
3. **Others Sector**: Industry Count- 47 i.e. 11.0%

3.2.4. Distribution of Industries Based on Country and Sector



The above distribution shows that Country_01 and Country_02 have only Mining and Metals sectors whereas Country_03 has manufacturing plants from all the three sectors.

3.2.5. Distribution of the Accident Level & Potential Accident Level

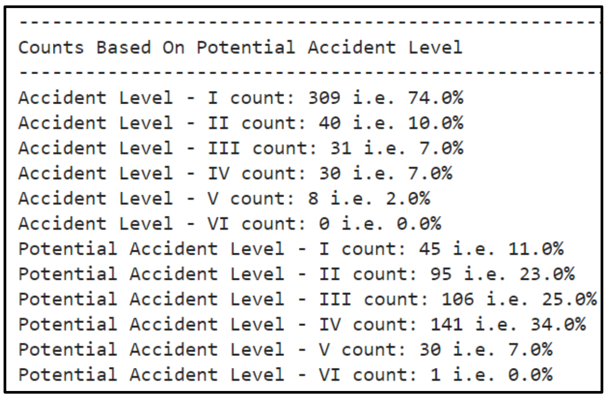


In the given dataset, the level of accidents is distributed into five groups and for each accident level, we have potential accident levels distributed into six groups.

The observations from the above plot are as follows:

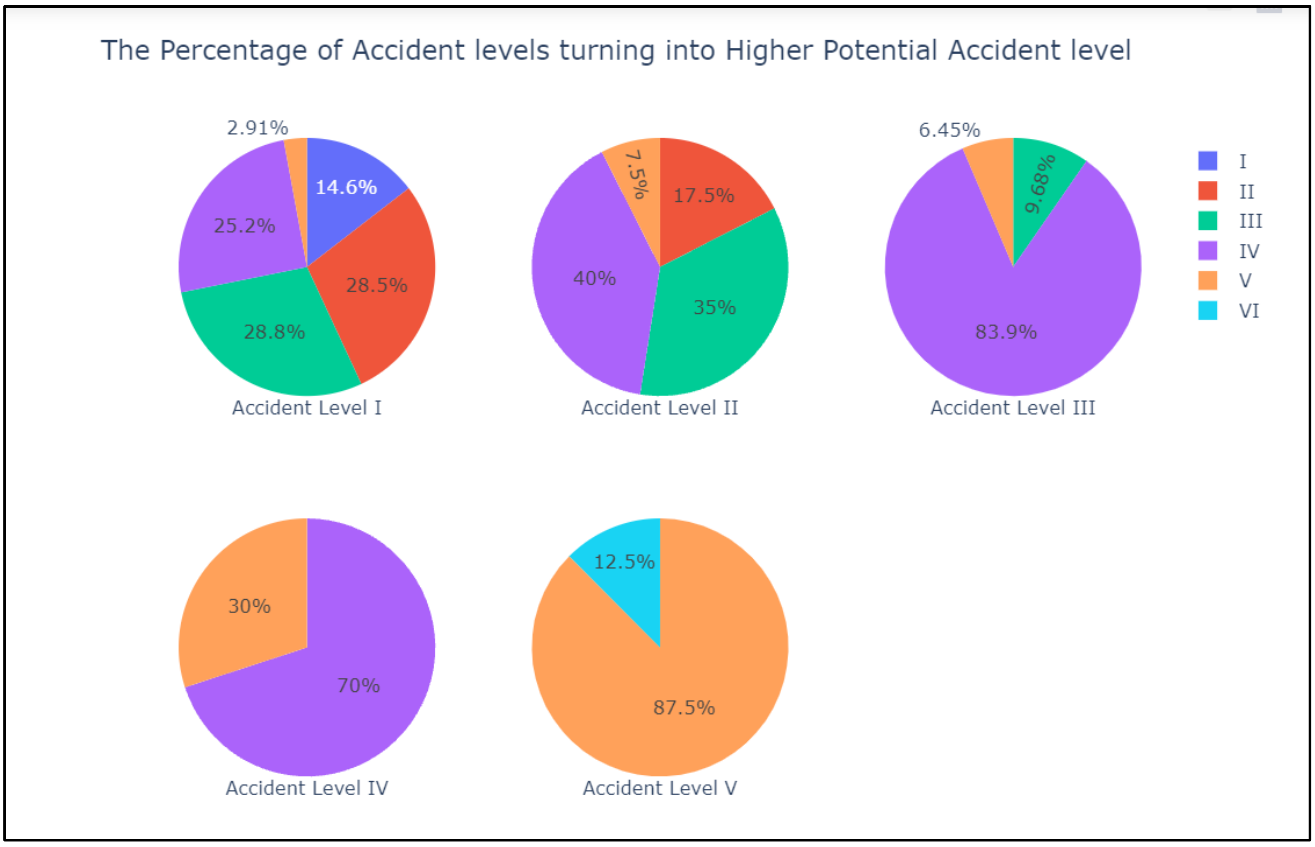
1. The number of accidents decreases as the Severity Level of Accident increases

2. The number of accidents increases as the Potential Accident Level increases



```
-----------------------------------------------------
Counts Based On Potential Accident Level
-----------------------------------------------------
Accident Level - I count: 309 i.e. 74.0%
Accident Level - II count: 40 i.e. 10.0%
Accident Level - III count: 31 i.e. 7.0%
Accident Level - IV count: 30 i.e. 7.0%
Accident Level - V count: 8 i.e. 2.0%
Accident Level - VI count: 0 i.e. 0.0%
Potential Accident Level - I count: 45 i.e. 11.0%
Potential Accident Level - II count: 95 i.e. 23.0%
Potential Accident Level - III count: 106 i.e. 25.0%
Potential Accident Level - IV count: 141 i.e. 34.0%
Potential Accident Level - V count: 30 i.e. 7.0%
Potential Accident Level - VI count: 1 i.e. 0.0%
```

3.2.6. The possibilities of lower-level accidents turning into higher-level accidents



These charts show the probabilities of any accident level turning into higher potential level accidents:

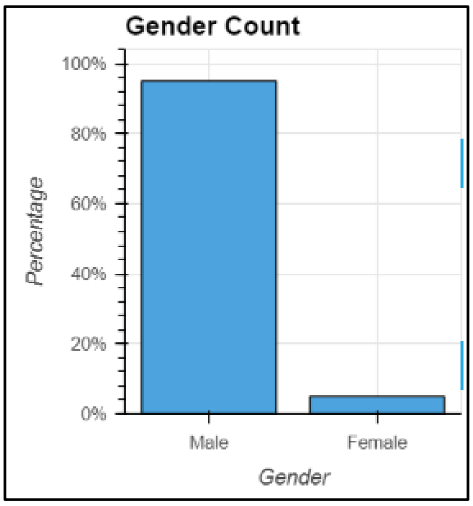
1. When the accident level is identified as level 1, there are higher chances (100-14.6 = 85.4%) of it turning into a potential accident level greater than 1, which should be taken care of.
2. For the accident level 2 also, it has higher chances (100-17.5=82.5%) of turning into major accidents
3. When the accident level is identified as 3, it is more likely to turn into higher Potential levels (100-9.68)=90.32
4. For the accident level 4, there is less chance (30%, not actually very less) to turn into higher potential level accidents
5. For the accident level 5, there is less chance (12.5%) to turn into higher potential level accidents

It can be concluded that safety measures have to be taken care of to make the lower level (I, II, III) accidents not turn into major accidents as there are high chances of 85.4%, 82.5%, and 90.32%

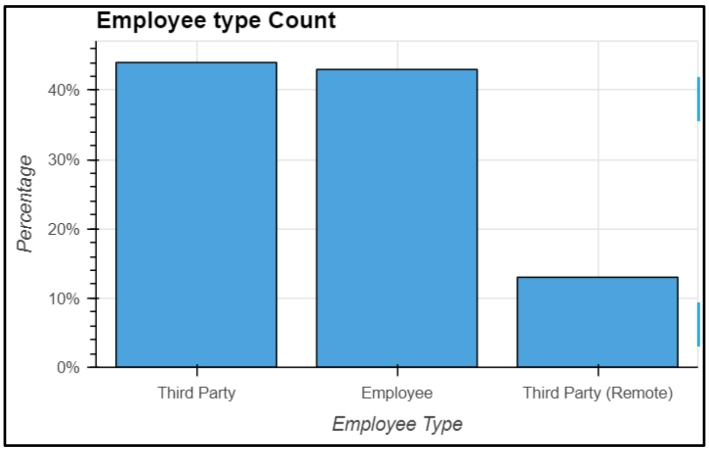
3.2.7. Distribution of Accidents Based on Gender

From the plot, we have observed that the proportion of Female workers is very less as compared to male workers.

1. **Male** Count- 396 i.e. **95.0%**
2. **Female** Count- 22 i.e. **5.0%**



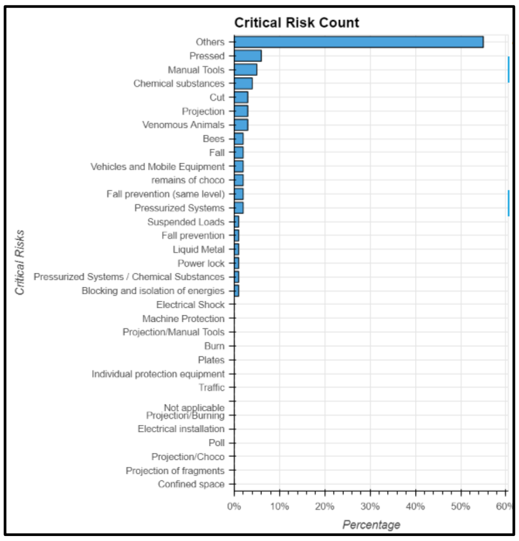
3.2.8. Distribution of Employees Based on Employee Type



The maximum number of employees is from the Third-party and the minimum is from the Third Party (Remote).

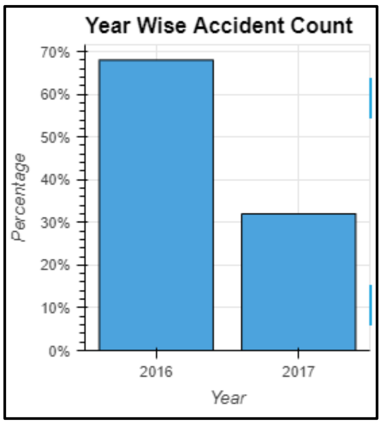
1. **44.0%** out of the total employees are from **Third Party** type
2. **43.0%** out of the total employees are from **Employee** type
3. **13.0%** out of the total employees are from **Third Party (Remote)** type

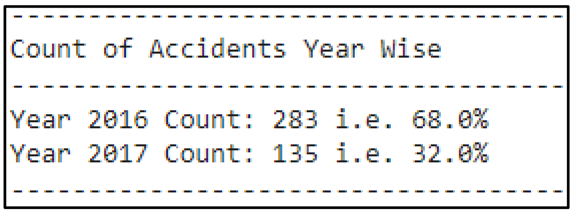
3.2.9. Distribution of Accidents Based on Critical Risk



The maximum number of accidents i.e., 55.0% of the total is falling under the Critical Risk 'Others'. Apart from 'Others', the second and third most Critical Risk counts are from 'Pressed' and 'Manual Tools'.

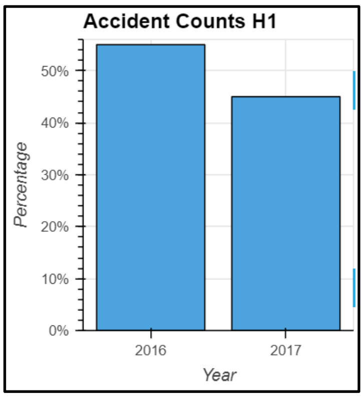
3.2.10. Accidents Counts Based on Year

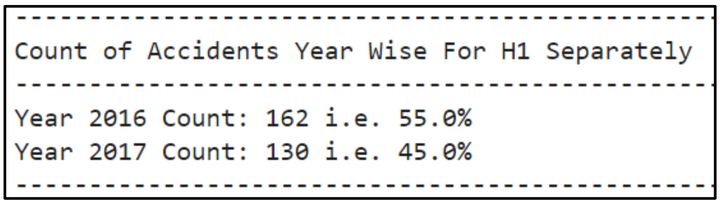




```
----------------------------------------
Count of Accidents Year Wise
----------------------------------------
Year 2016 Count: 283 i.e. 68.0%
Year 2017 Count: 135 i.e. 32.0%
----------------------------------------
```

The maximum number of accidents as per the given dataset is recorded in the year 2016 as compared to 2017. Although, this has to be noted that in the given dataset, data is not available from August-2017.Comparing data for H1 only for 2016 and 2017.

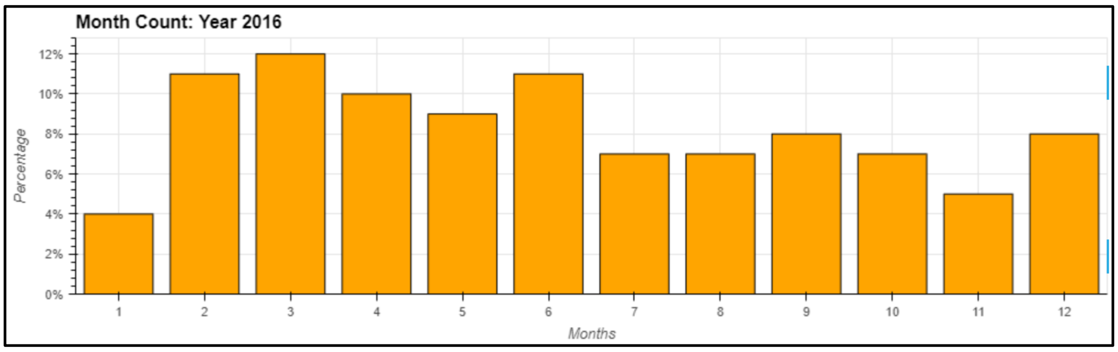




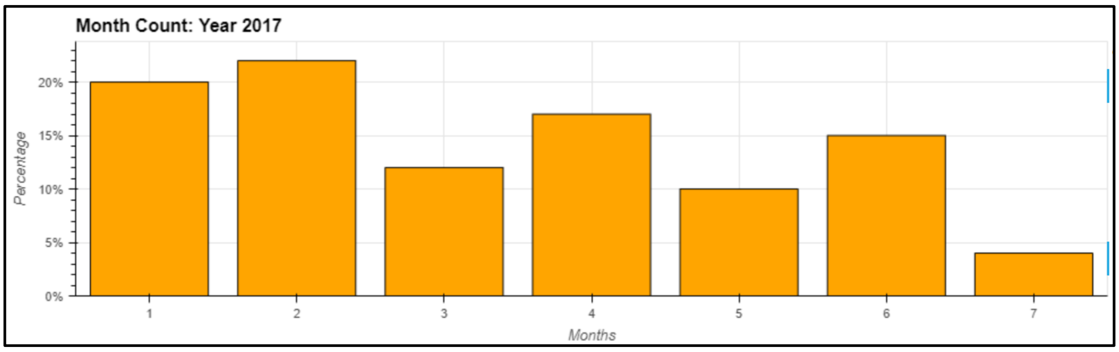
```
------------------------------------------------
Count of Accidents Year Wise For H1 Separately
------------------------------------------------
Year 2016 Count: 162 i.e. 55.0%
Year 2017 Count: 130 i.e. 45.0%
------------------------------------------------
```

On comparing the accidents count for the first six months for both 2016 and 2017, we have observed that the maximum number of accidents recorded is from the year 2016 as compared to 2017. The counts have decreased from last year by **10%**.

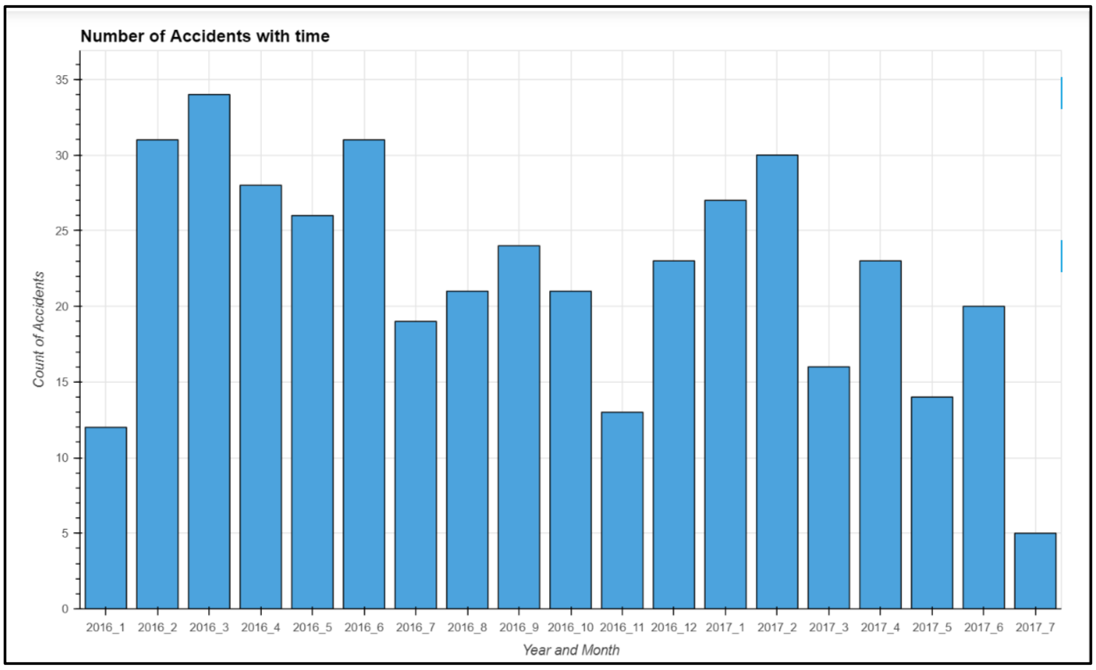
3.2.11. Distribution of Accidents Based on Month For 2016 and 2017 Separately



In the year 2016, the maximum number of accidents were recorded in the first six months as compared to the last six months. Also, the maximum accidents were recorded in the month of March and the minimum in January.



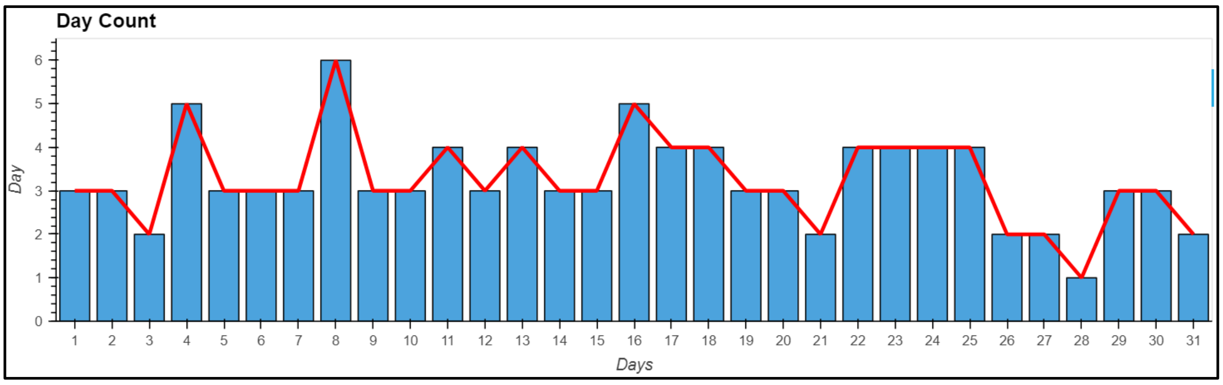
For the year 2017, if we compare data for the first six months, we have observed that the maximum number of accidents recorded in January and minimum in June.



- In the first half of both years, the number of accidents is more.

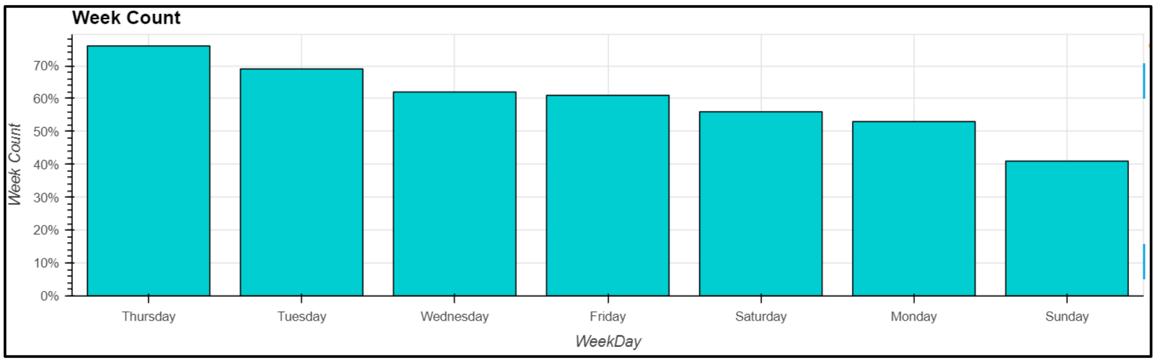
- But as the number of accidents is reducing with time (2016 to 2017), the 2017 first half has fewer accidents compared to the 2016 first half
- The number of accidents in the years 2016 and 2017, is fluctuating. But overall it is decreasing towards the end of the year. It can be concluded that precautions are taken to reduce accidents.

3.2.12. Count of Accidents Based on Days



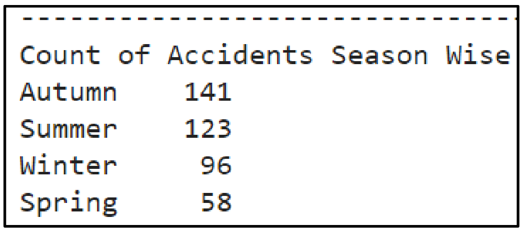
The maximum number of accidents was recorded on the 4th, 8th, and 16th day of each month.

3.2.13. Count of Accidents Based on WeekDay

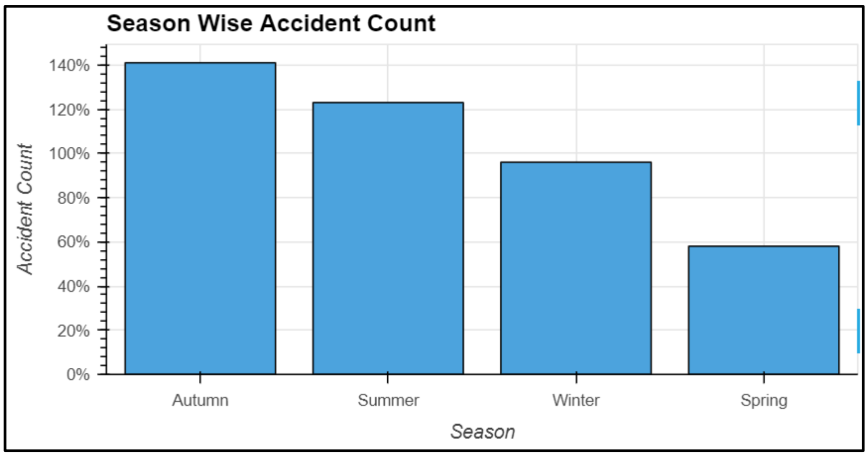


The maximum number of accidents increased during the middle of the week and declined after the middle of the week. Also, we have observed that accidents occurred on weekends also.

3.2.14. Count of Accidents Based on Season

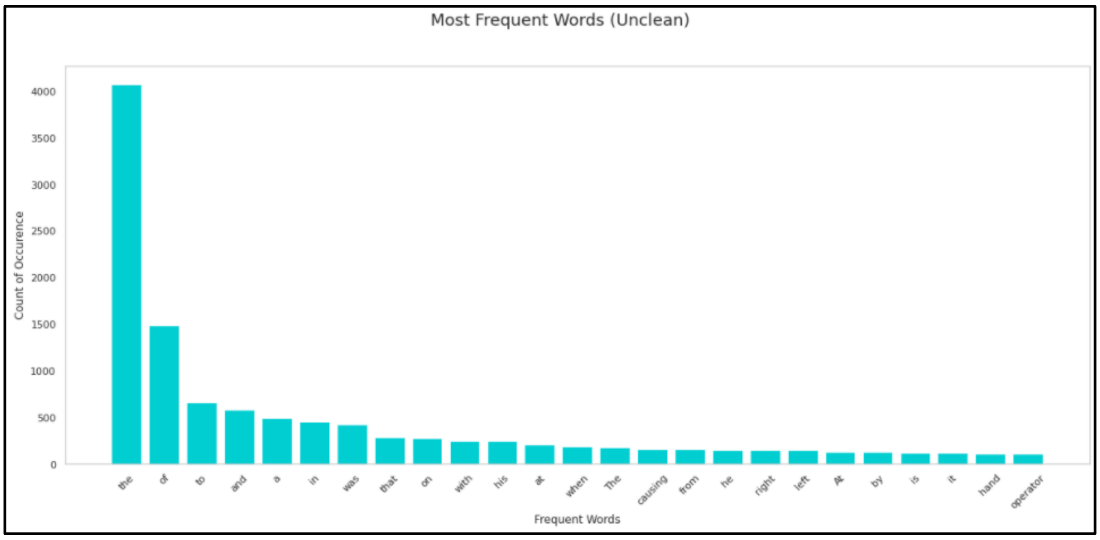


```
--------------------------------
Count of Accidents Season Wise
Autumn     141
Summer     123
Winter      96
Spring      58
```

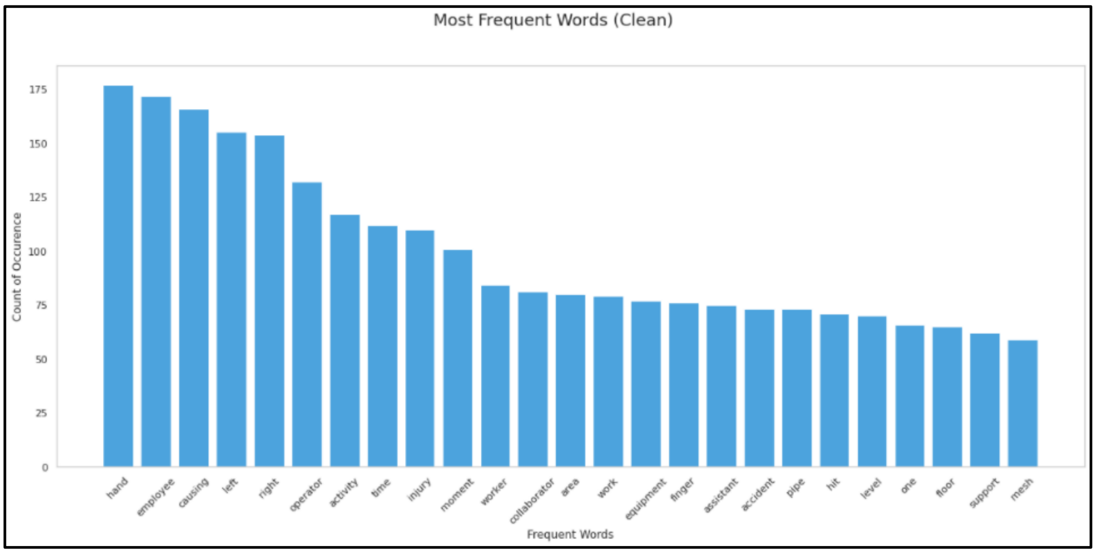


The maximum number of accidents reported in the Autumn and Summer seasons and the minimum in the Spring season. The occurrence of accidents is related to the climate (especially temperature).

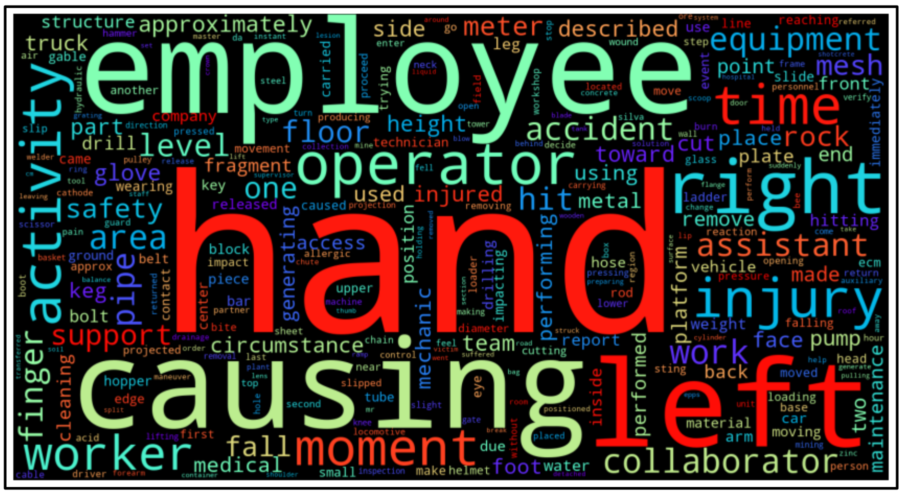
3.2.15. Most Frequent Words in the Description (Unclean and Cleaned Data).



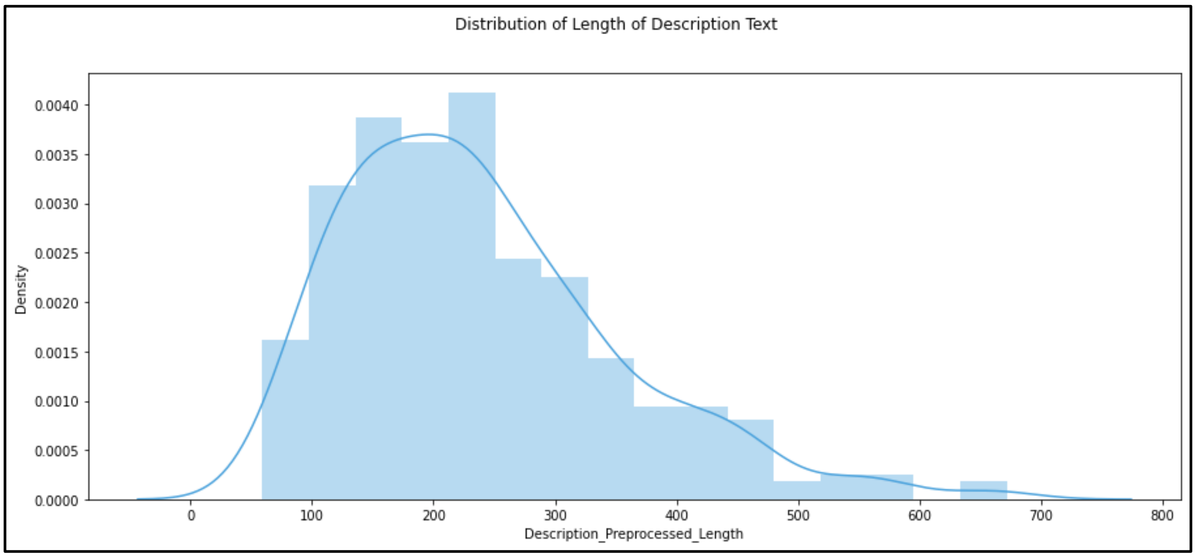
In the unclean data, the top three most frequent words are ‘the’, ’of’, and ’to’.



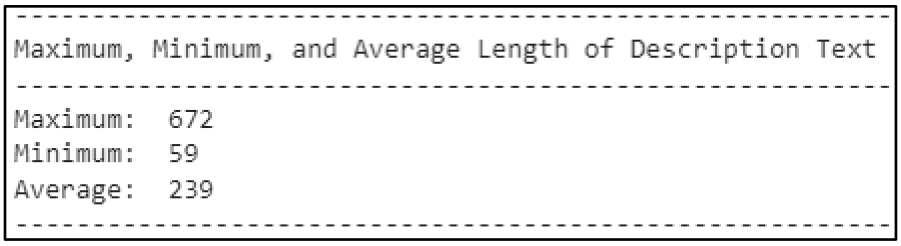
In the processed data, the top three most frequent words are 'hand', 'employee', and 'causing'. The above three words can also be visualized using the Word Cloud.



3.2.16. Distribution of Length of Description Text.

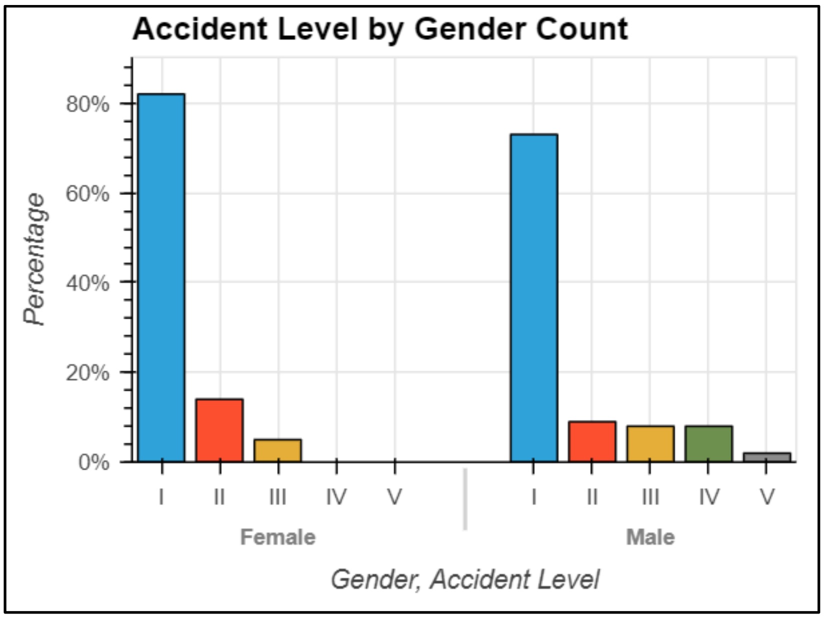


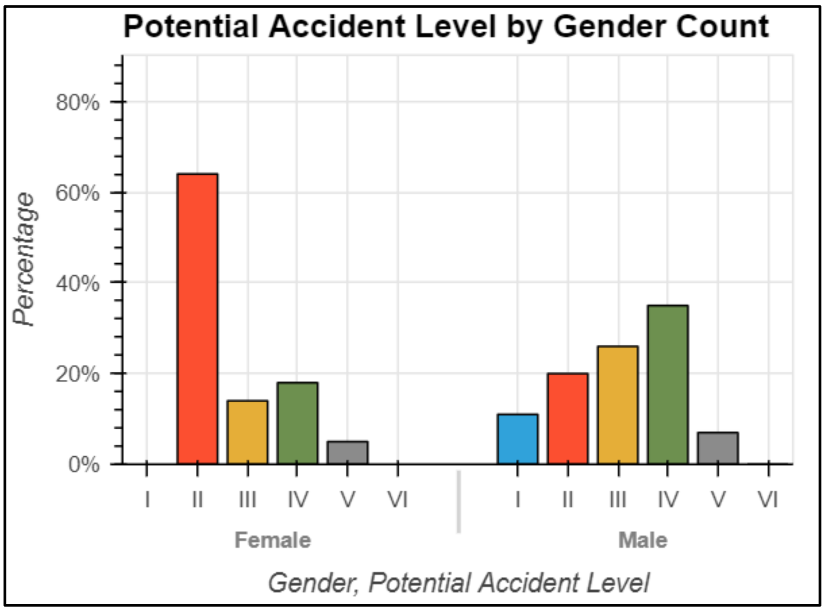
The distribution of the length of words is shown in the above distribution plot. We have observed that the distribution is positively skewed and most of the text is of the length of 200 to 300.



```
-----------------------------------------------------------------
Maximum, Minimum, and Average Length of Description Text
-----------------------------------------------------------------
Maximum:  672
Minimum:  59
Average:  239
-----------------------------------------------------------------
```

3.2.17. Accident Level and Potential Accident Level by Gender Count





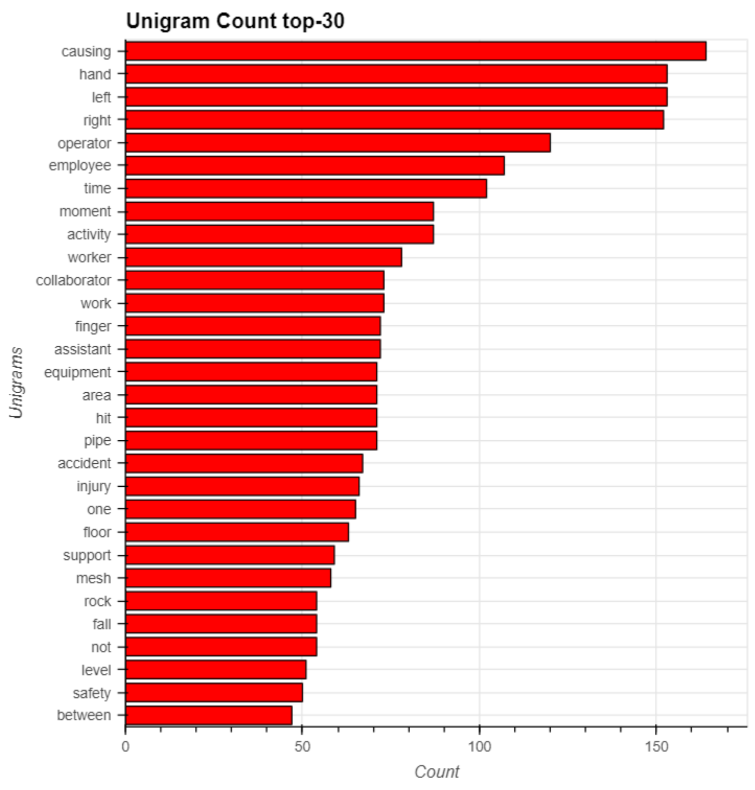
1. From the above plot, the proportion of accident levels in each gender is not equal and males have higher accident levels than females
2. Potential accident level IV is higher in Male as compared to females
3. Potential accident level II is higher in Females as compared to Male

3.2.18. NGram Analysis

We Performed Unigram, Bigram and Trigram analysis to check the most frequent words of the clean data and most frequent combinations of the data.

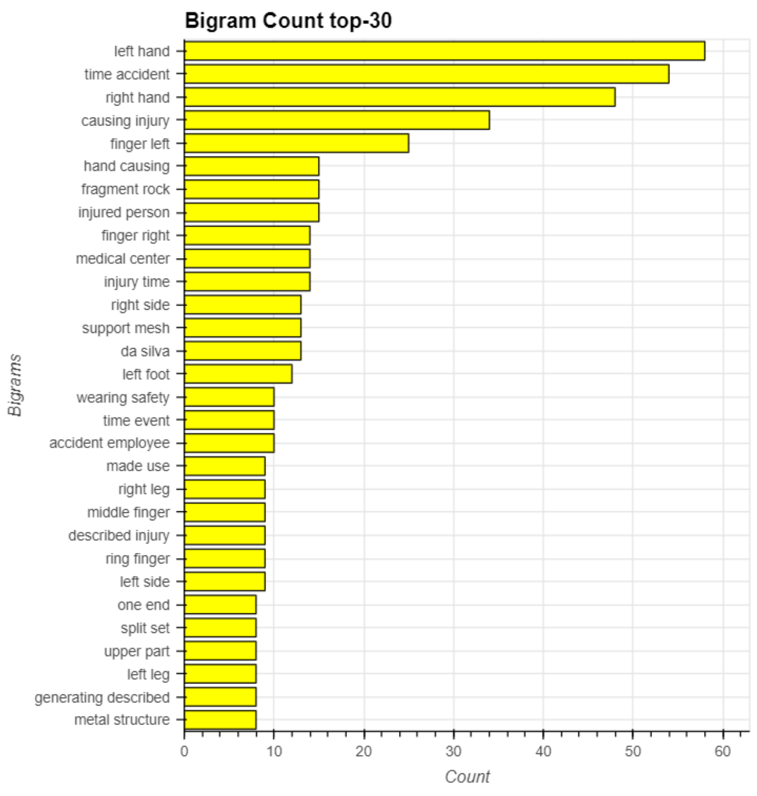
Unigram Analysis:

- There are several words which are related to hands. For example left, hand, right and finger.
- Moreover there are several words which are related to movement of something. For example hit, remov, fall and move.



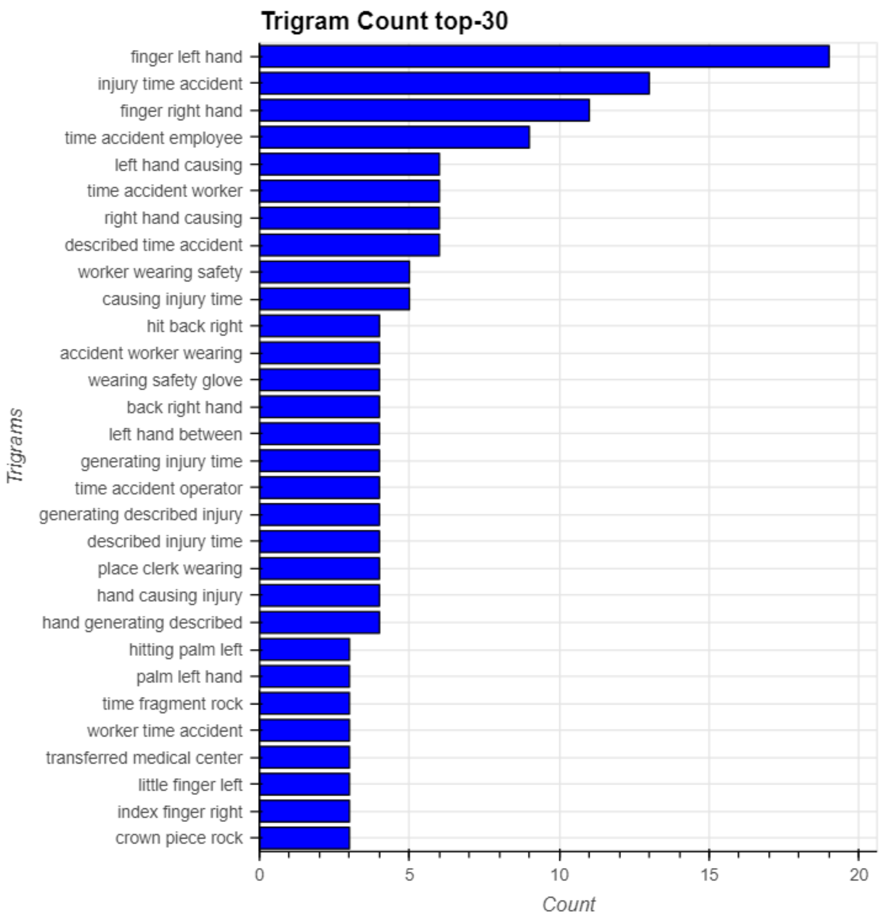
Bi-Gram Analysis:

- There are so many phrases which are related to hands. For example: left hand, right hand, finger left, finger right, middle finger and ring finger.
- There are also some phrases which are related to other body parts. For example, my left foot and right leg.



Tri-Gram Analysis:

- Like Unigram and Bigram, there are also many phrases which are related to hands or other body parts, but concreteness seems to increase.
- For example one hand glove, left arm uniform and wear safety uniform.



3.2.19. Hypothesis Testings

1. Check the proportion of metal, mining, and other sectors in Country_01 and whether the difference is statistically significant?

We have observed that the Metals and the Mining industry are not available in Country_03. Also, the distribution of each industry country-wise differs significantly.

State the Ho and Ha:

Ho = There is no difference in the proportion of the industry sector.

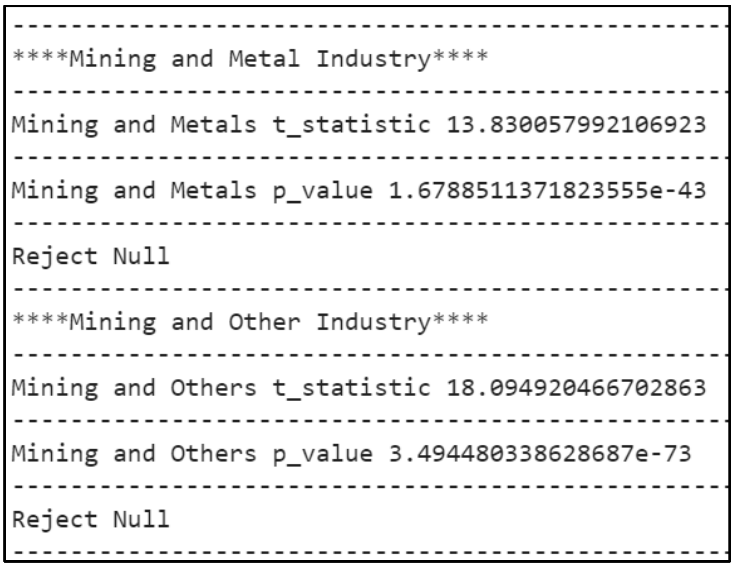
Ha = There is a difference in the proportion.

Decide the significance level: **alpha = 0.05**

Identify the test-statistic: **Z-test** of proportions

Proportions of mining, metals, others in country_01 = 81.0%, 19.0%, 1.0% respectively.

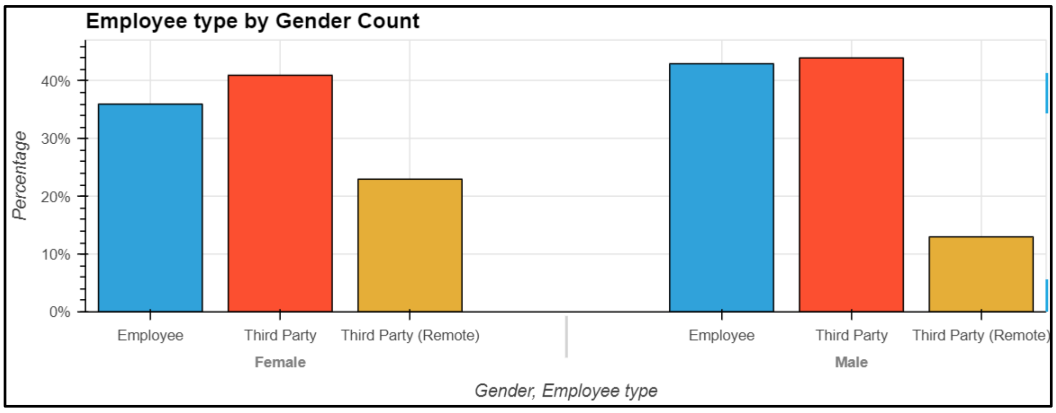
Calculate the p_value using test statistics.



```
--------------------------------------------------
****Mining and Metal Industry****
--------------------------------------------------
Mining and Metals t_statistic 13.830057992106923
--------------------------------------------------
Mining and Metals p_value 1.6788511371823555e-43
--------------------------------------------------
Reject Null
--------------------------------------------------
****Mining and Other Industry****
--------------------------------------------------
Mining and Others t_statistic 18.094920466702863
--------------------------------------------------
Mining and Others p_value 3.494480338628687e-73
--------------------------------------------------
Reject Null
--------------------------------------------------
```

From the above, we have enough (95% confident) evidence to prove that the proportion of industry sectors is ***different*** in country_01.

2. Employee type by Gender - Is the distribution of employee type differ significantly gender-wise?



1. Proportion of third-party employees in each gender is equal
2. The proportion of the third party(remote) employees in each gender are not equal
3. The proportion of own employees in each gender is not equal

Let's check if that difference is statistically significant?

State the Ho and Ha:

Ho = The proportions of own employees in each gender are equal.

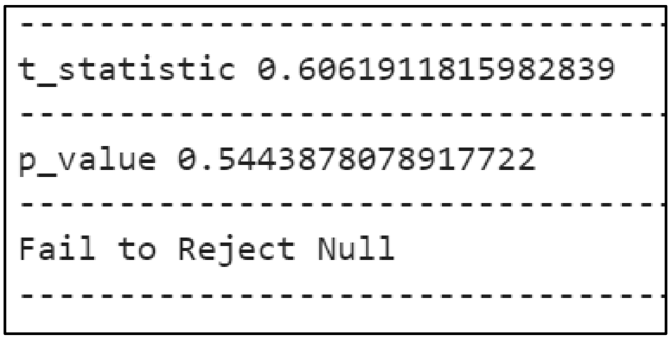
Ha = The proportions of own employees in each gender are not equal.

Decide the significance level: **alpha = 0.05**

Identify the test-statistic: **Z-test** of proportions

Proportion of own employee types in male, female = 43.0%, 36.0% respectively.

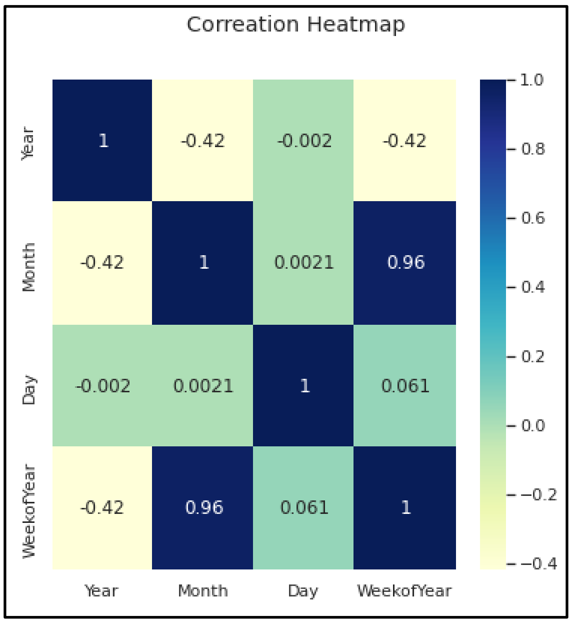
Calculate the p_value using test-statistic.



```
---------------------------------
t_statistic 0.6061911815982839
---------------------------------
p_value 0.5443878078917722
---------------------------------
Fail to Reject Null
---------------------------------
```

From the above, we fail to reject Null Hypothesis, we have enough (95% confident) evidence to prove that the proportion of own employees in each gender is ***equal***.

3.2.20. Correlation Between the Features



The above heatmap shows that the WeekofYear feature is having a very high positive correlation with the Month feature.

3.2.21. Summary of Exploratory Data Analysis

1. **Local**: Highest manufacturing plants are located in Local_03 city and lowest in Local_09 city.
2. **Country**: Percentage(%) of accidents occurred in respective countries: 59% in Country_01, 31% in Country_02 and 10% in Country_03.
3. **Industry Sector**: Percentage(%) of manufacturing plants belongs to respective sectors: 57% to the Mining sector, 32% to Metals sector and 11% to Others sector.
4. **Country and Industry Sector**: Metals and Mining industry sector plants are not available in Country_03. The distribution of industry sectors differs significantly in each country.
5. **Accident Levels**: The number of accidents decreases as the Accident Level increases and increases as the Potential Accident Level increases.
6. **Gender**: There are more men working in this industry as compared to women.
7. **Employee type**: 44% Third-party employees, 43% own employees, and 13% Third-party(Remote) employees working in this industry.

8. **Gender and Employee type**: Proportion of third party employees in each gender is equal, third party(remote) employees in each gender are not equal and own employees in each gender are not equal.
9. **Gender and Accident Levels**: Males have a higher accident level than females. There are many low risks at the general accident level, but many high risks at the potential accident level.
10. **Correlation**: WeekOfYear feature is showing a very high positive correlation with the Month Feature.

4. MODEL BUILDING

4.1. Models explored and building

We have explored and built the following models:

1. Text Data Augmentation
2. ML Models
    a. Logistic Regression
    b. SVM
    c. Gradient Boost
    d. XGBoost
3. Hyper Parameter tuning for the models
4. Artificial Neural Networks
5. LSTM - original and Augmented data
6. CNN +LSTM Multi-Input Models
7. LSTM + LSTM Multi-Input Model
8. BERT
9. Simple Transformer- Using Pre-processed Data
10. Simple Transformer- Using Augmented Training Data
11. Simple Transformer- Using Augmented Training Data With Target Class Balanced
12. Simple Transformers with Distill BERT (Custom)

4.1.1. Text Data Augmentation

NLP AUG - is a python library that helps you with augmenting nlp for your machine learning projects. This Library generates synthetic data for improving model performance without manual effort.

Importing Augmentation library

```
import nlpaug.augmenter.word as naw
```

In NLP Aug we are using Context-based augmentation using Bert-base-uncased. BERT aug is designed to perform insertion and substitution. Different from previous word embeddings, insertion is predicted by the BERT language model rather than picking one word randomly. Substitution uses surrounding words as a feature to predict the target word.

```
aug_bert = naw.ContextualWordEmbsAug(

    model_path='bert-base-uncased', action="substitute")

Example:
```

```
Original:
during the activation of a sodium sulphide pump the piping wa uncoupled and the sulfide solution wa designed in the
Augmented Text:
during overnight activation of a sodium sulphide pump the piping wa uncoupled and the sulfide solution wa designed
```

We Augmented only the train data, Tokensied and padded it. We added both the normal train data and augmented train data for the text and targets. Shape of Train and test data

```
X_text_train.shape,y_text_train.shape,X_text_test.shape, y_text_test.shape
```

```
((656, 200), (656, 5), (83, 200), (83, 5))
```

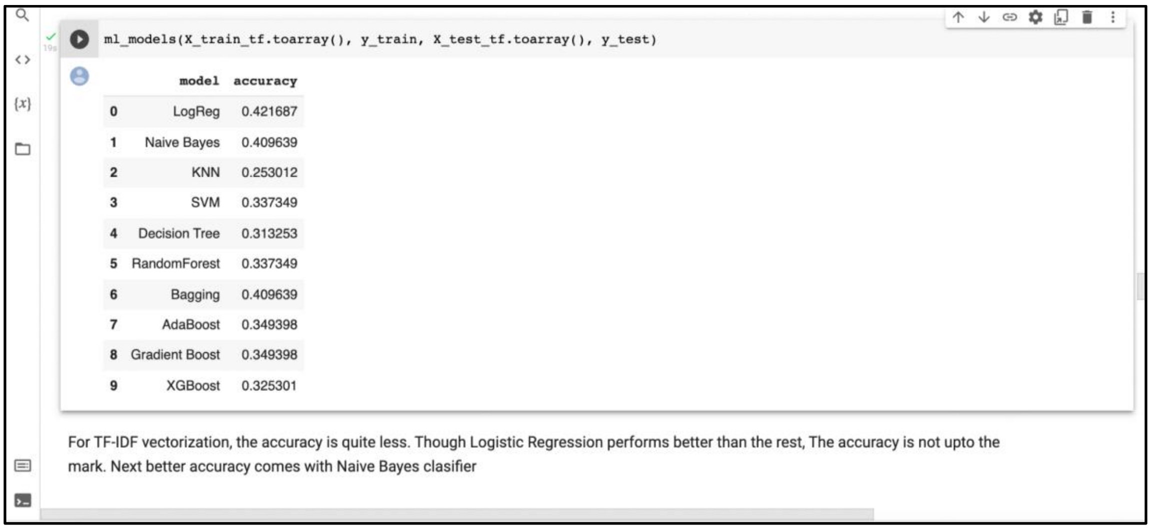
We One hot encoded the targets and pickled the data. This Augmented data is directly used in all the models.

```
import pickle
infile = open('/content/drive/MyDrive/AIML/Capstone/Train_Test_augment.pickle','rb')
X_train,y_train,X_test,y_test = pickle.load(infile)
```

4.1.2. Machine Learning Models

Using ML models: Have built the basic ML models with Processed description Vs Potential Accident Levels and observed the below:

Using TF-IDF:

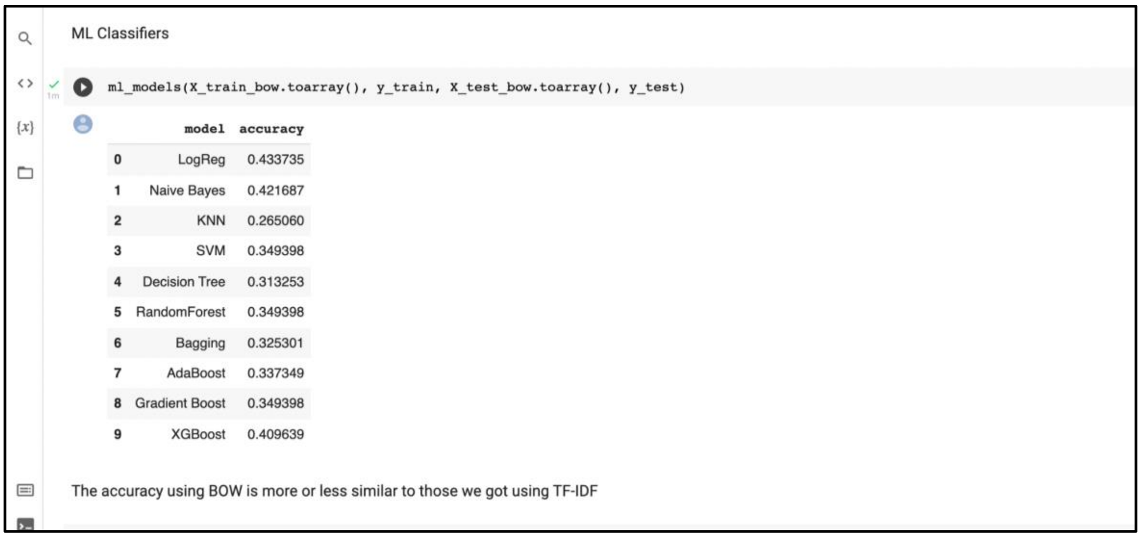


```
ml_models(X_train_tf.toarray(), y_train, X_test_tf.toarray(), y_test)
```

	model	accuracy
0	LogReg	0.421687
1	Naive Bayes	0.409639
2	KNN	0.253012
3	SVM	0.337349
4	Decision Tree	0.313253
5	RandomForest	0.337349
6	Bagging	0.409639
7	AdaBoost	0.349398
8	Gradient Boost	0.349398
9	XGBoost	0.325301

For TF-IDF vectorization, the accuracy is quite less. Though Logistic Regression performs better than the rest, The accuracy is not upto the mark. Next better accuracy comes with Naive Bayes clasifier

Using Bag of Words:



ML Classifiers

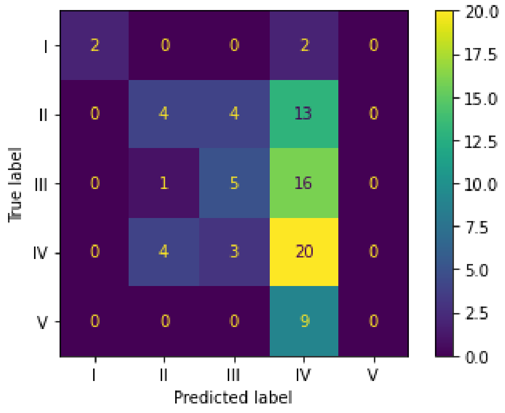
```
ml_models(X_train_bow.toarray(), y_train, X_test_bow.toarray(), y_test)
```

	model	accuracy
0	LogReg	0.433735
1	Naive Bayes	0.421687
2	KNN	0.265060
3	SVM	0.349398
4	Decision Tree	0.313253
5	RandomForest	0.349398
6	Bagging	0.325301
7	AdaBoost	0.337349
8	Gradient Boost	0.349398
9	XGBoost	0.409639

The accuracy using BOW is more or less similar to those we got using TF-IDF

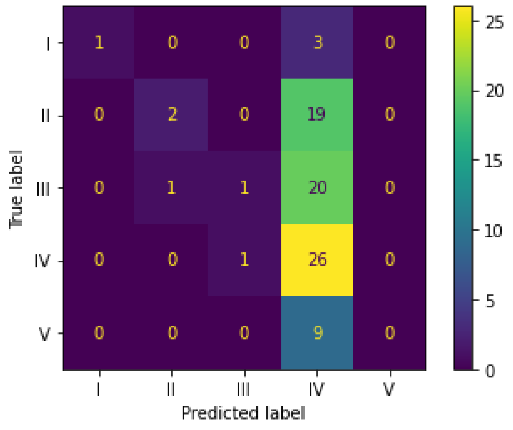
4.1.2.a. Logistic Regression

```
LogReg
              precision    recall  f1-score   support

           I       1.00      0.50      0.67         4
          II       0.44      0.19      0.27        21
         III       0.42      0.23      0.29        22
          IV       0.33      0.74      0.46        27
           V       0.00      0.00      0.00         9

    accuracy                           0.37        83
   macro avg       0.44      0.33      0.34        83
weighted avg       0.38      0.37      0.33        83
```

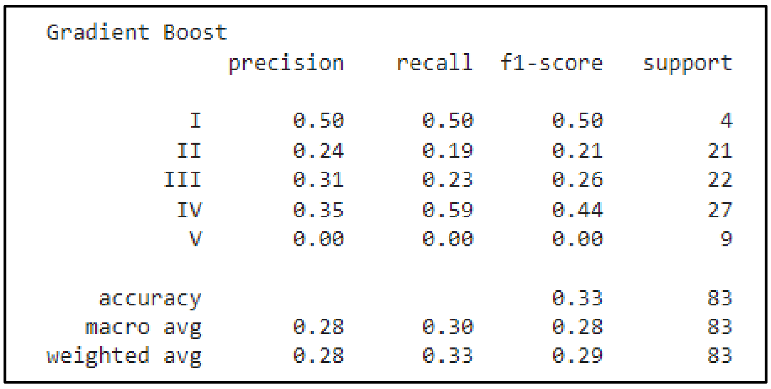


4.1.2.b. Support Vector Machine (SVM)

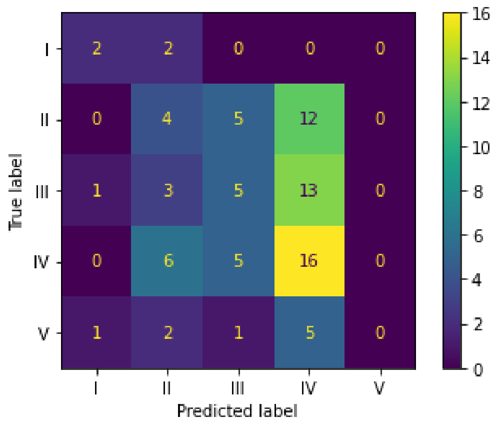
```
SVM
              precision    recall  f1-score   support

           I       1.00      0.25      0.40         4
          II       0.67      0.10      0.17        21
         III       0.50      0.05      0.08        22
          IV       0.34      0.96      0.50        27
           V       0.00      0.00      0.00         9

    accuracy                           0.36        83
   macro avg       0.50      0.27      0.23        83
weighted avg       0.46      0.36      0.25        83
```



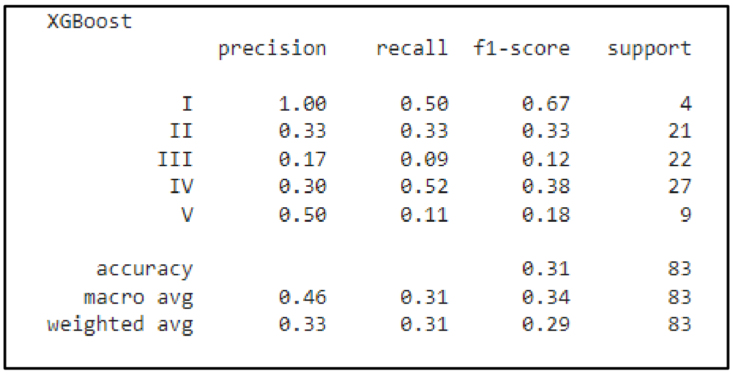
4.1.2.c. Gradient Boosting



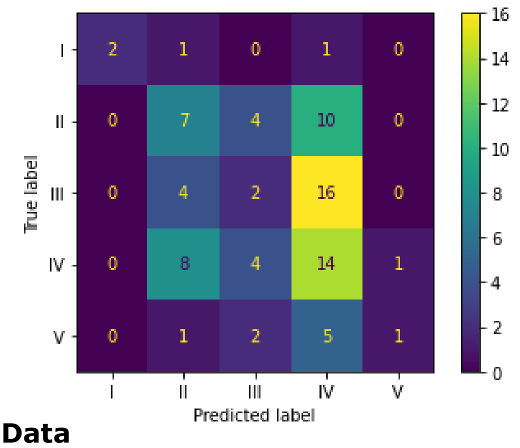
```
Gradient Boost
              precision    recall  f1-score   support

           I       0.50      0.50      0.50         4
          II       0.24      0.19      0.21        21
         III       0.31      0.23      0.26        22
          IV       0.35      0.59      0.44        27
           V       0.00      0.00      0.00         9

    accuracy                           0.33        83
   macro avg       0.28      0.30      0.28        83
weighted avg       0.28      0.33      0.29        83
```



4.1.2.d. XG Boost



```
XGBoost
              precision    recall  f1-score   support

           I       1.00      0.50      0.67         4
          II       0.33      0.33      0.33        21
         III       0.17      0.09      0.12        22
          IV       0.30      0.52      0.38        27
           V       0.50      0.11      0.18         9

    accuracy                           0.31        83
   macro avg       0.46      0.31      0.34        83
weighted avg       0.33      0.31      0.29        83
```



4.1.2.e. Machine Learning Models using Augmented Data

Next we try to build Machine Learning models using Augmented Data and then we re-run the model with hyperparameter tuning.

4.1.3 Hyperparameter tuning for all models:

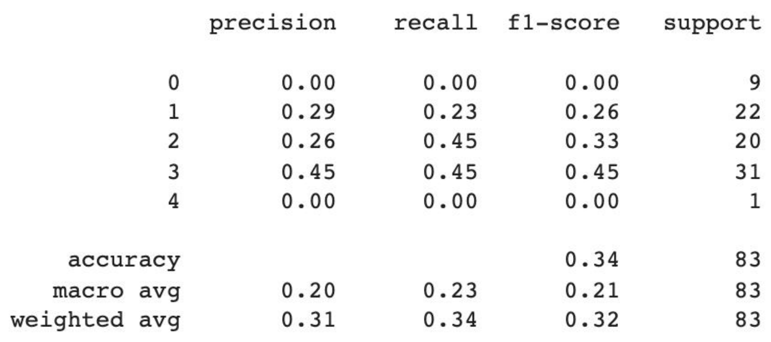
With Original Data: We can see that using Augmented data, the accuracy we have obtained is less when compared to running the hyperparameter tuning on original data. Let's try to print on individual models using Augmented data.

	model	accuracy
0	LogReg	0.168675
1	Naive Bayes	0.240964
2	KNN	0.204819
3	SVM	0.337349
4	Decision Tree	0.313253
5	RandomForest	0.325301
6	Bagging	0.301205
7	AdaBoost	0.277108
8	Gradient Boost	0.325301
9	XGBoost	0.301205

4.1.3.1.a Hyperparameter tuning for RF classifier on Augmented Data:

Classification Report:

Here, the accuracy obtained is 33.73% and the classification report obtained is given below.

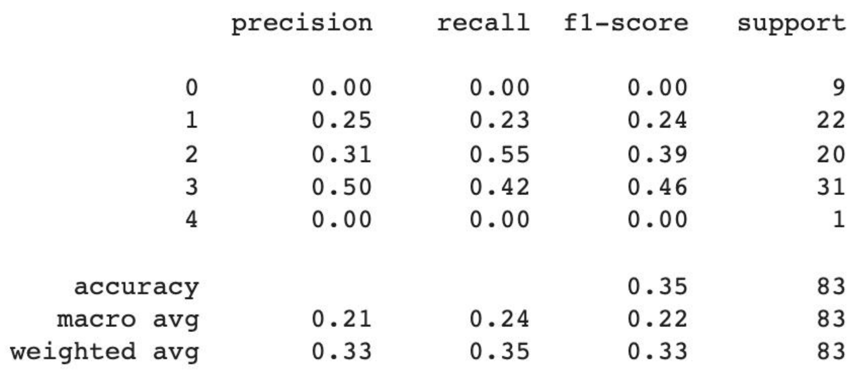


```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         9
           1       0.29      0.23      0.26        22
           2       0.26      0.45      0.33        20
           3       0.45      0.45      0.45        31
           4       0.00      0.00      0.00         1

    accuracy                           0.34        83
   macro avg       0.20      0.23      0.21        83
weighted avg       0.31      0.34      0.32        83
```

4.1.3.1.b Hyperparameter tuning for Bagging classifier on Augmented Data:

Classification Report:

Here, the accuracy obtained is 34.95% and the classification report obtained is given below.

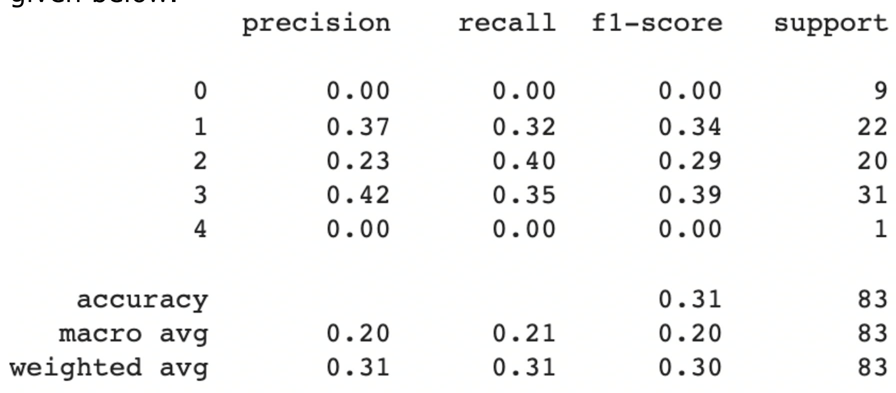


```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         9
           1       0.25      0.23      0.24        22
           2       0.31      0.55      0.39        20
           3       0.50      0.42      0.46        31
           4       0.00      0.00      0.00         1

    accuracy                           0.35        83
   macro avg       0.21      0.24      0.22        83
weighted avg       0.33      0.35      0.33        83
```

4.1.3.1.c Hyperparameter tuning for XGBoost on Augmented Data:

Classification Report:

Here, the accuracy obtained is 31.33% and the classification report obtained is given below.



```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         9
           1       0.37      0.32      0.34        22
           2       0.23      0.40      0.29        20
           3       0.42      0.35      0.39        31
           4       0.00      0.00      0.00         1

    accuracy                           0.31        83
   macro avg       0.20      0.21      0.20        83
weighted avg       0.31      0.31      0.30        83
```

4.1.3.1.d Hyperparameter tuning for Logistic Regression on Augmented Data:

Classification Report:

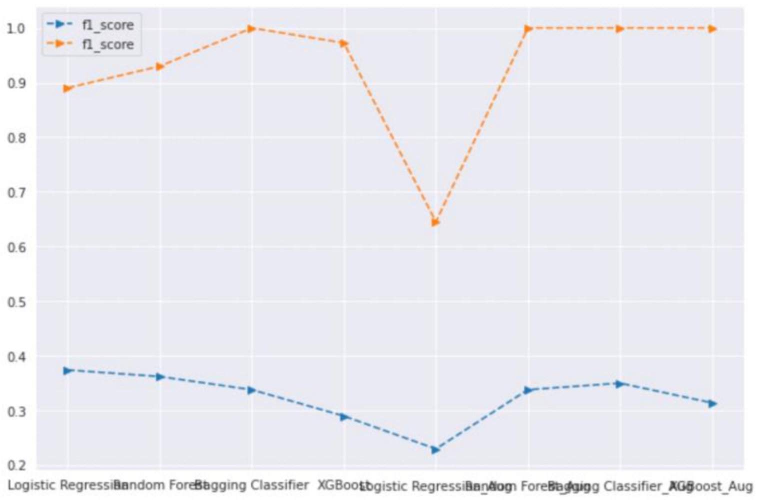
Here, the accuracy obtained is 22.89% and the classification report obtained is given below.

```
              precision    recall  f1-score   support

           0       0.17      0.22      0.19         9
           1       0.33      0.36      0.35        22
           2       0.12      0.10      0.11        20
           3       0.32      0.23      0.26        31
           4       0.00      0.00      0.00         1

    accuracy                           0.23        83
   macro avg       0.19      0.18      0.18        83
weighted avg       0.26      0.23      0.24        83
```

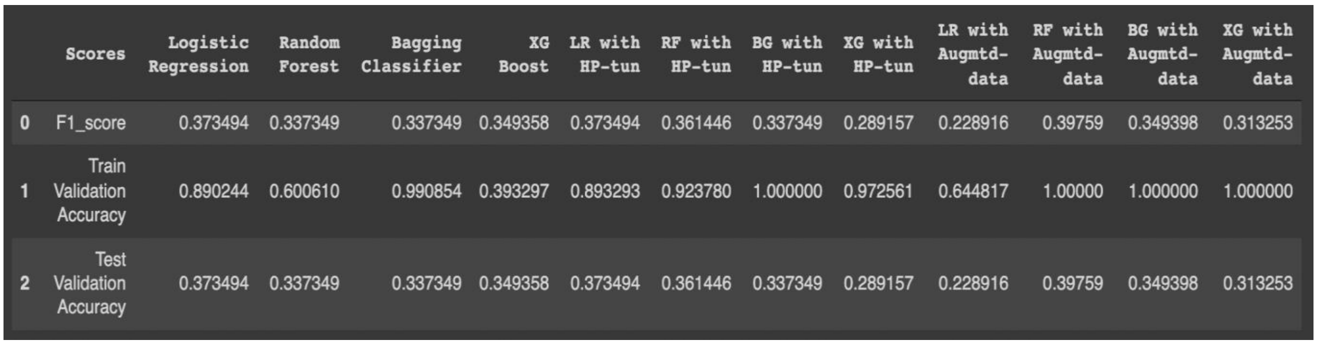
4.1.3.2 Comparing the hyperparameter tuning models on Original data and Augmented Data:

	Model	F1_score	Train Validation Accuracy
0	Logistic Regression	0.373494	0.890244
1	Random Forest	0.361446	0.929878
2	Bagging Classifier	0.337349	1.000000
3	XGBoost	0.289157	0.972561
4	Logistic Regression_Aug	0.228916	0.644817
5	Random Forest_Aug	0.337349	1.000000
6	Bagging Classifier_Aug	0.349398	1.000000
7	XGBoost_Aug	0.313253	1.000000

Then we draw the plot of the above data for F1_score as below:



4.1.3.3 Comparing all the models:



	Scores	Logistic Regression	Random Forest	Bagging Classifier	XG Boost	LR with HP-tun	RF with HP-tun	BG with HP-tun	XG with HP-tun	LR with Augmtd-data	RF with Augmtd-data	BG with Augmtd-data	XG with Augmtd-data
0	F1_score	0.373494	0.337349	0.337349	0.349358	0.373494	0.361446	0.337349	0.289157	0.228916	0.39759	0.349398	0.313253
1	Train Validation Accuracy	0.890244	0.600610	0.990854	0.393297	0.893293	0.923780	1.000000	0.972561	0.644817	1.00000	1.000000	1.000000
2	Test Validation Accuracy	0.373494	0.337349	0.337349	0.349358	0.373494	0.361446	0.337349	0.289157	0.228916	0.39759	0.349398	0.313253

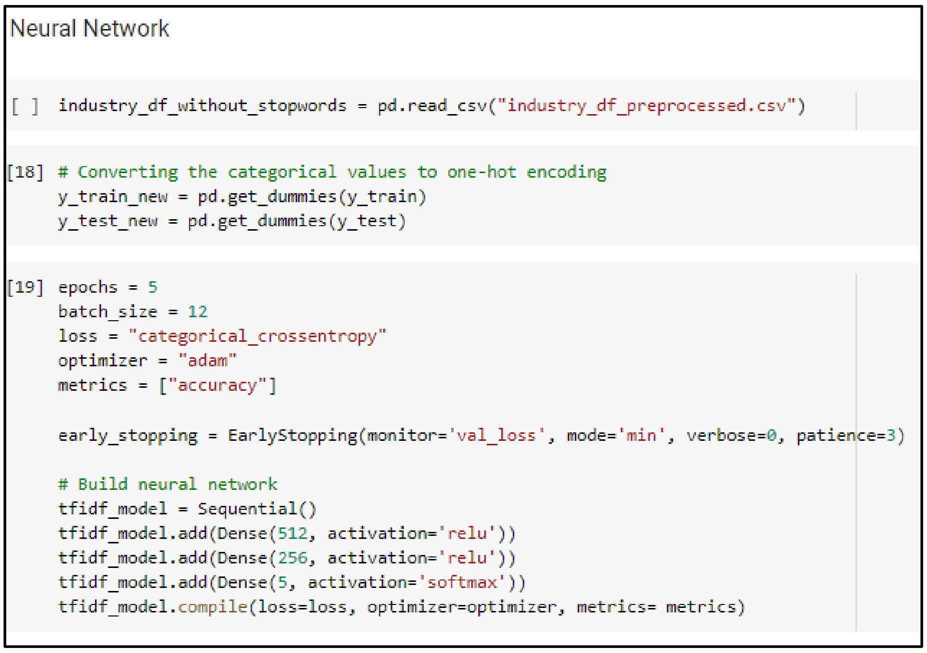
Observations:

- We could see that accuracy obtained on Normal data is higher than using hyperparameters on Augmented data.
- Logistic Regression gave higher accuracy of around 37% while using hyperparameters tuning.

4.1.4. Using Artificial Neural Networks:

Have built the basic ANN models with Processed description Vs Potential Accident Levels and observed the below:

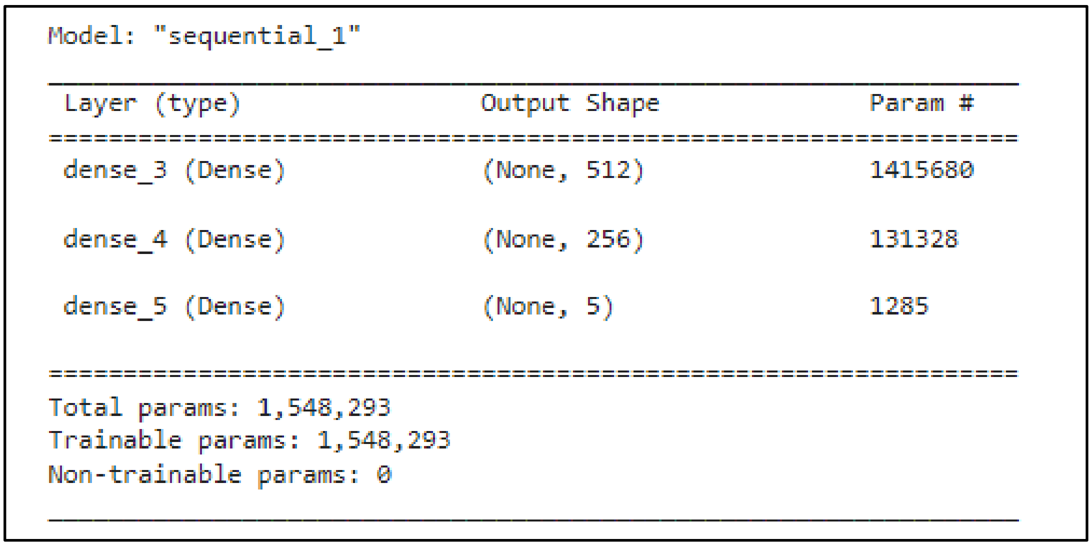
4.1.4.1 ANN Layer and Summary



Neural Network

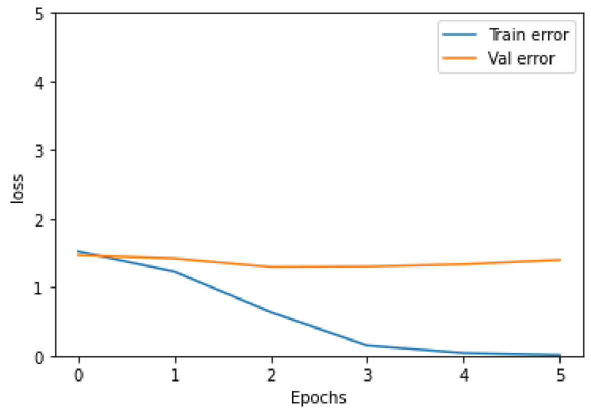
```
[ ]  industry_df_without_stopwords = pd.read_csv("industry_df_preprocessed.csv")

[18] # Converting the categorical values to one-hot encoding
     y_train_new = pd.get_dummies(y_train)
     y_test_new = pd.get_dummies(y_test)

[19] epochs = 5
     batch_size = 12
     loss = "categorical_crossentropy"
     optimizer = "adam"
     metrics = ["accuracy"]

     early_stopping = EarlyStopping(monitor='val_loss', mode='min', verbose=0, patience=3)

     # Build neural network
     tfidf_model = Sequential()
     tfidf_model.add(Dense(512, activation='relu'))
     tfidf_model.add(Dense(256, activation='relu'))
     tfidf_model.add(Dense(5, activation='softmax'))
     tfidf_model.compile(loss=loss, optimizer=optimizer, metrics= metrics)
```

ANN Layers

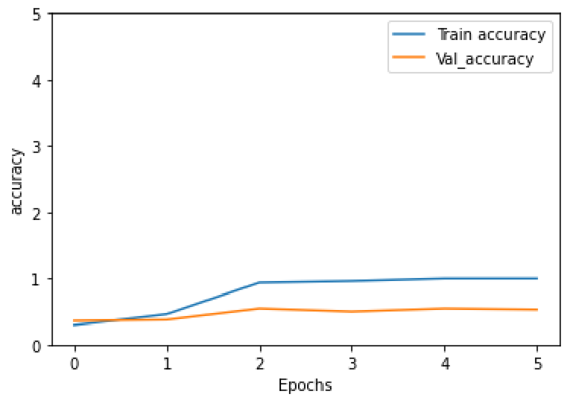


```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 512)               1415680

 dense_4 (Dense)             (None, 256)               131328

 dense_5 (Dense)             (None, 5)                 1285

=================================================================
Total params: 1,548,293
Trainable params: 1,548,293
Non-trainable params: 0
_________________________________________________________________
```

Model Layout Summary

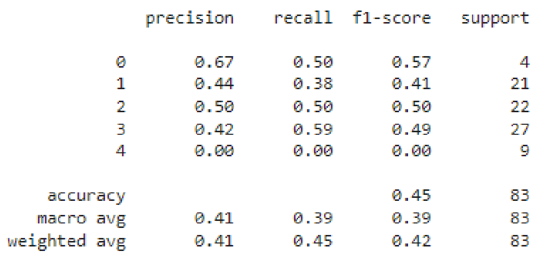


Training loss vs Validation loss



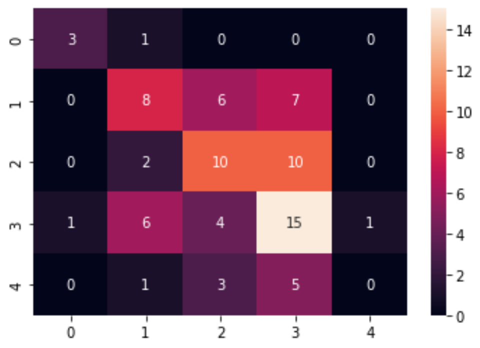
Train accuracy vs Validation accuracy

4.1.4.2. Confusion Matrix & Classification Report



```
              precision    recall  f1-score   support

           0       0.67      0.50      0.57         4
           1       0.44      0.38      0.41        21
           2       0.50      0.50      0.50        22
           3       0.42      0.59      0.49        27
           4       0.00      0.00      0.00         9

    accuracy                           0.45        83
   macro avg       0.41      0.39      0.39        83
weighted avg       0.41      0.45      0.42        83
```

Classification report



	0	1	2	3	4
0	3	1	0	0	0
1	0	8	6	7	0
2	0	2	10	10	0
3	1	6	4	15	1
4	0	1	3	5	0

Confusion Matrix

4.1.5. LSTM

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning.

In our model, we took the processed Descriptions(text data) as our features and the Potential Accident Level as our Targets.

4.1.5.1. Targets [Y] - We label encode the targets to convert the Roman letters into numbers ranging from 0 to 4, then we one-hot encode it. The targets will be converted into a size of 411*5.

4.1.5.2. Features[X] - For LSTM we don't remove the stopwords while preprocessing and we used Tokeniser from Keras library to convert the words into numbers with maximum words of 3000.

Vocabulary size - 2841

Maximum number of words in the description - 183

The Maximum length of our features is kept as 185 and we pad all our data in our description. With this, the data size will be 411*185.

4.1.5.3. Glove Embeddings- For Embedding our features we use Glove Embeddings 6B to 200D. We created a Dictionary for our Glove embeddings. Then we created an Embedding matrix with our Vocabulary as our row size and 200D as our columns. The Embedding Matrix size will be 2841*200 which will act as the embedding weights for our LSTM model.

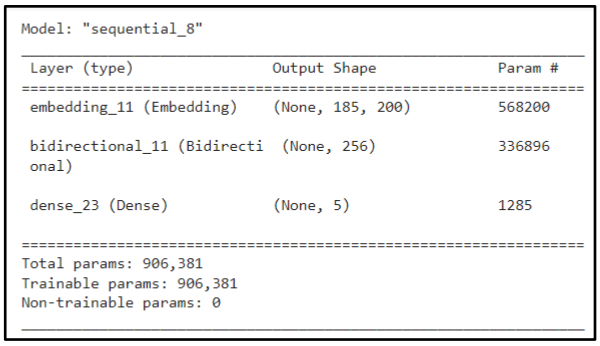
4.1.5.4. Models: Then we created two models,

The *Sequential Model*- In this model, we used an Embedding layer with Bidirectional LSTM then finally a dense layer with a softmax activation function. The Final Layer is used as Sigmoid since we have multiple targets as our output.

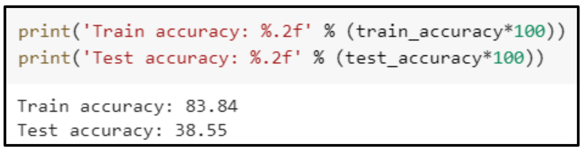
With batch size 8 and running for 10 epochs we got,

Optimizer - Adam

Loss - Categorical Cross Entropy



```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_11 (Embedding)    (None, 185, 200)          568200

 bidirectional_11 (Bidirecti  (None, 256)              336896
 onal)

 dense_23 (Dense)            (None, 5)                 1285

=================================================================
Total params: 906,381
Trainable params: 906,381
Non-trainable params: 0
_________________________________________________________________
```



```
print('Train accuracy: %.2f' % (train_accuracy*100))
print('Test accuracy: %.2f' % (test_accuracy*100))

Train accuracy: 83.84
Test accuracy: 38.55
```

We can clearly see that the base model overfits the data.

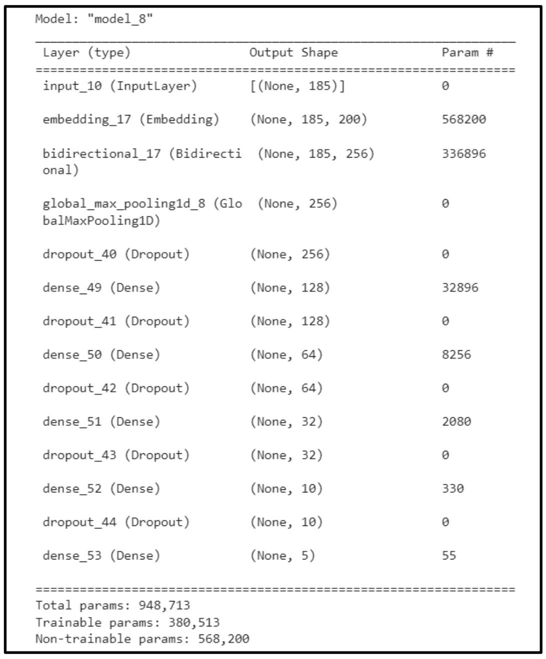
LSTM model with Dense Layers- In this model, we used the Embedding layer with a Bidirectional LSTM Layer along with 5 dense layers and Drop out layers to gradually reduce the networks from 256 to 5. The first four layers use the 'Relu' activation function and the last layer uses the 'Softmax' activation function.

With batch size 8 and running for 30 epochs we got,

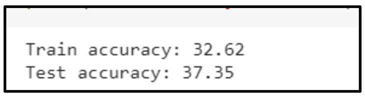
Optimizer - Adam/SGD

Loss - Categorical Cross Entropy

Learning rate - 0.001/0.0001



```
Model: "model_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_10 (InputLayer)       [(None, 185)]             0

 embedding_17 (Embedding)    (None, 185, 200)          568200

 bidirectional_17 (Bidirecti  (None, 185, 256)         336896
 onal)

 global_max_pooling1d_8 (Glo  (None, 256)              0
 balMaxPooling1D)

 dropout_40 (Dropout)        (None, 256)               0

 dense_49 (Dense)            (None, 128)               32896

 dropout_41 (Dropout)        (None, 128)               0

 dense_50 (Dense)            (None, 64)                8256

 dropout_42 (Dropout)        (None, 64)                0

 dense_51 (Dense)            (None, 32)                2080

 dropout_43 (Dropout)        (None, 32)                0

 dense_52 (Dense)            (None, 10)                330

 dropout_44 (Dropout)        (None, 10)                0

 dense_53 (Dense)            (None, 5)                 55

=================================================================
Total params: 948,713
Trainable params: 380,513
Non-trainable params: 568,200
```



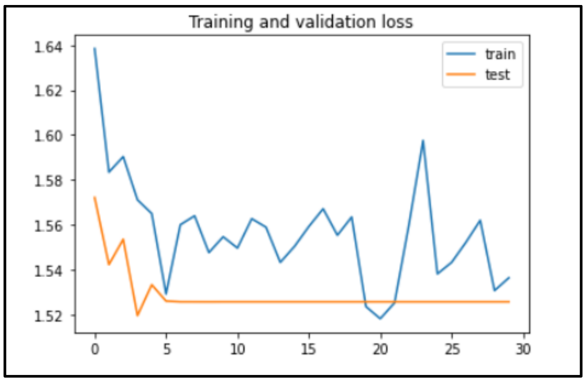
```
Train accuracy: 32.62
Test accuracy: 37.35
```

This Model doesn't overfit but gives very little test and training accuracy.

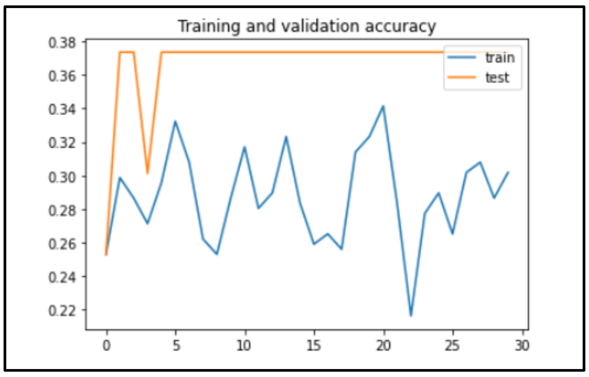
This LSTM Model has been run on various combinations such as

Models	Accuracy
With Stop words	32 - 36 %
Without Stop Words	33 - 37%
SGD Optimizer	28 - 34%
Adam Optimizer	33 - 37%
Learning Rate 0.001	34 - 37%
Learning Rate 0.0001	32 - 36%

With all these combinations we got the accuracy ranges from 32 - 37%. Clearly, LSTM doesn't work well with this data. Since the number of data points is very less we got very low accuracy, we need more data to improve the model performance.



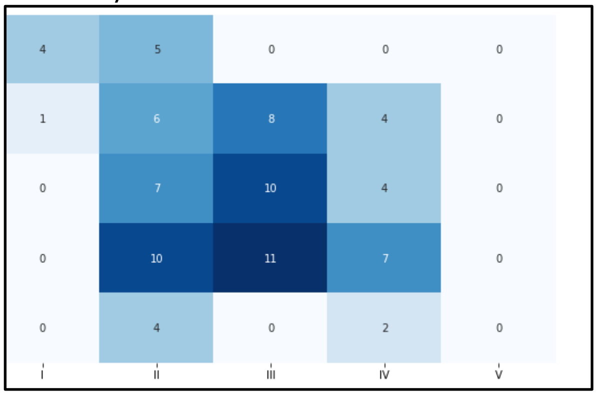
Loss Graph



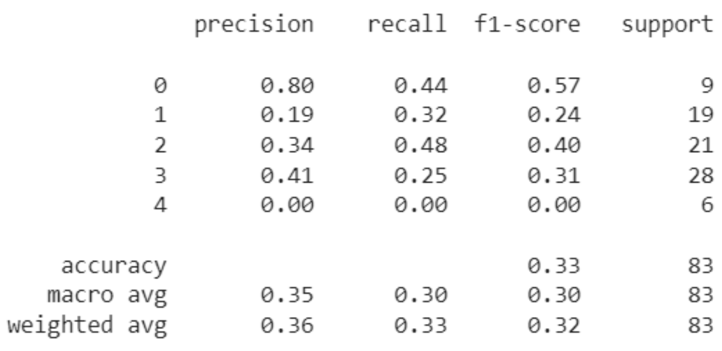
Accuracy Graph

4.1.5.5. Confusion Matrix & Classification Report:

This model only predicts potential accident level 4 for all our test data and we got a very less weighted average F1 score. Need further analysis or resample the data to get better accuracy.



Confusion Matrix

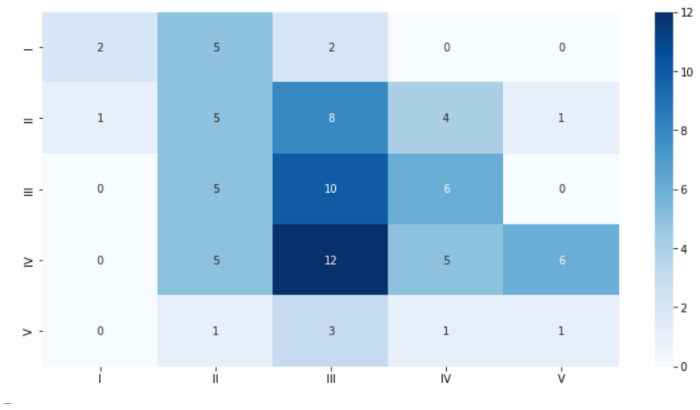


```
              precision    recall  f1-score   support

           0       0.80      0.44      0.57         9
           1       0.19      0.32      0.24        19
           2       0.34      0.48      0.40        21
           3       0.41      0.25      0.31        28
           4       0.00      0.00      0.00         6

    accuracy                           0.33        83
   macro avg       0.35      0.30      0.30        83
weighted avg       0.36      0.33      0.32        83
```

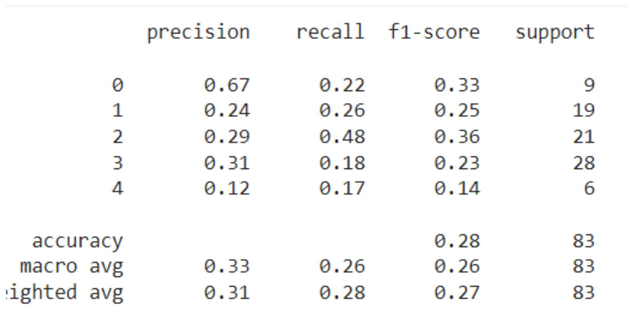
Classification Report

4.1.5.6. LSTM- Using Augmented Data

Next we tried the same Model with augmented data created by nlp aug.



Confusion Matrix



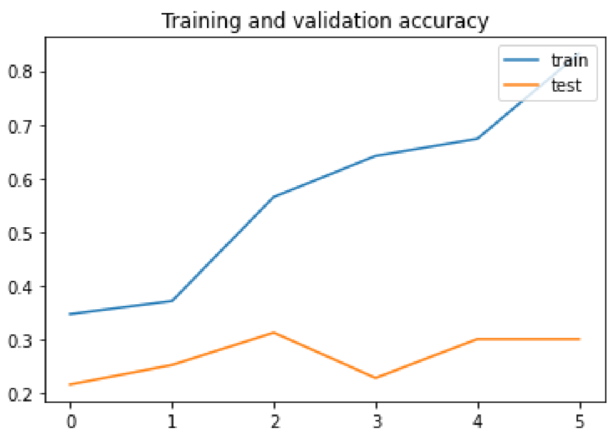
```
              precision    recall  f1-score   support

           0       0.67      0.22      0.33         9
           1       0.24      0.26      0.25        19
           2       0.29      0.48      0.36        21
           3       0.31      0.18      0.23        28
           4       0.12      0.17      0.14         6

    accuracy                           0.28        83
   macro avg       0.33      0.26      0.26        83
ighted avg       0.31      0.28      0.27        83
```

Classification Report

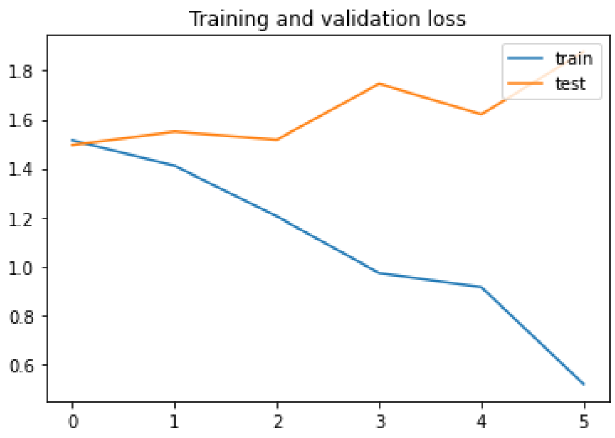
We Augment the train data and try to run the same model, there is slight increase in train accuracy but decrease in test accuracy. still the model is highly overfit.

```
print('Train accuracy: %.2f' % (train_accuracy*100))
print('Test accuracy: %.2f' % (test_accuracy*100))
```

```
Train accuracy: 94.05
Test accuracy: 30.12
```



Accuracy Graph

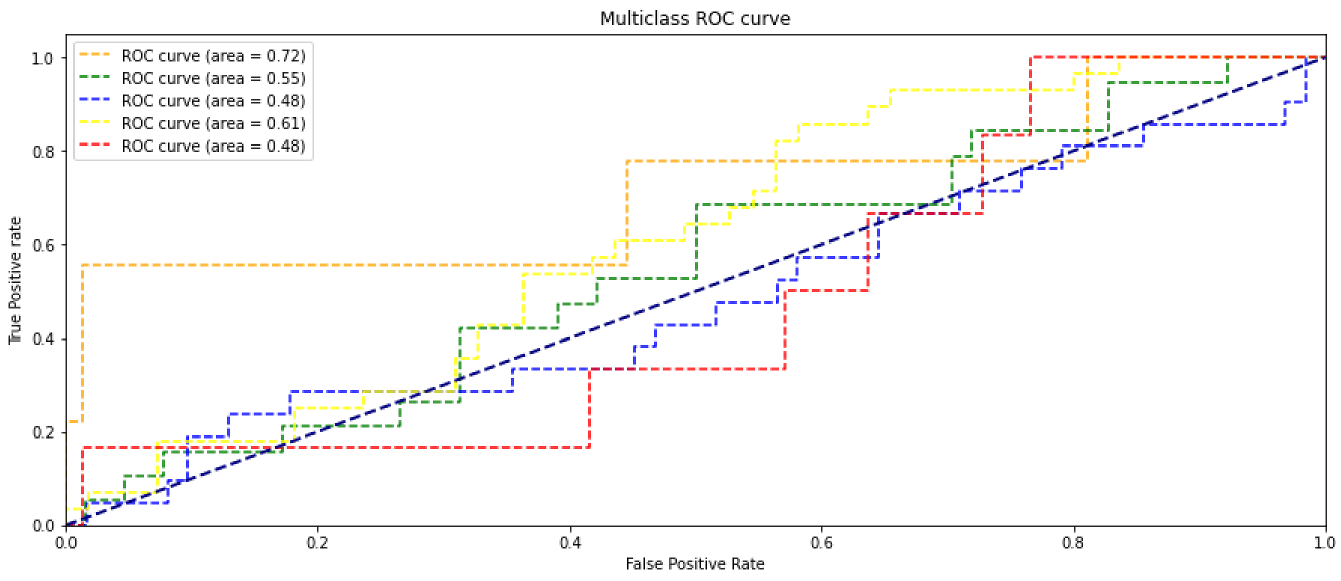


Loss Graph

4.1.5.7. ROC Curve

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds.

From the ROC curves we are getting good accuracy for potential accident level 1 and moderate accuracy for remaining accident levels.



4.1.6. CNN + LSTM Multi-Input Models

This multi-input model is the concatenation of the CNN model and the LSTM model. CNN model is given the categorical data to process and the LSTM model is given the text data (Sequential data) to process. The output layers from both models are concatenated and given to the dense layers to get the output.

In this model, processed description (text data) and categorical data are given as features and the Potential Accident Level is taken as the target variable.

4.1.6.1. Targets [Y]- We label encode the targets to convert the Roman letters into numbers ranging from 0 to 4, then we one-hot encode it. The targets will be converted into a size of 411*5.

4.1.6.2. Features[X]- Two types of inputs are used in this model.

4.1.6.3. Categorical data: The accident date, country, and location are used as categorical inputs. Location and country values are one-hot encoded. The date is split into columns of Year, Month, Day, WeekofYear, Season, and Weekday variables. Categorical data is of size 411*19

4.1.6.4. Model Building and Training

In this model, two branches are there:

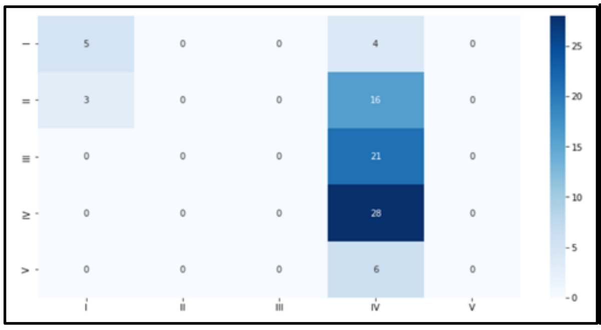
- One branch contains the Embedding layer followed by Bidirectional LSTM, max-pooling layer and dense layers along with dropout layers.

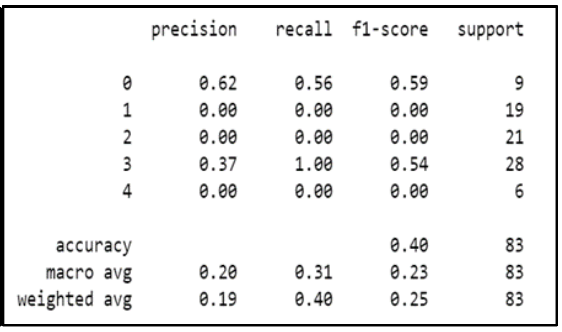
- The second branch contains one-dimensional convolution layers followed by max-pooling layer, flatten layer, and Dense layers along with batch normalization and dropout layers.

These two branches are concatenated, and it is given as input to another dense layer with 'relu' as an activation function. The last layer is a dense layer with 'softmax' as an activation function.

With batch size 8, running for 30 epochs, Optimizer - Adam, Loss - Categorical Cross-Entropy, and Learning rate – 0.01.

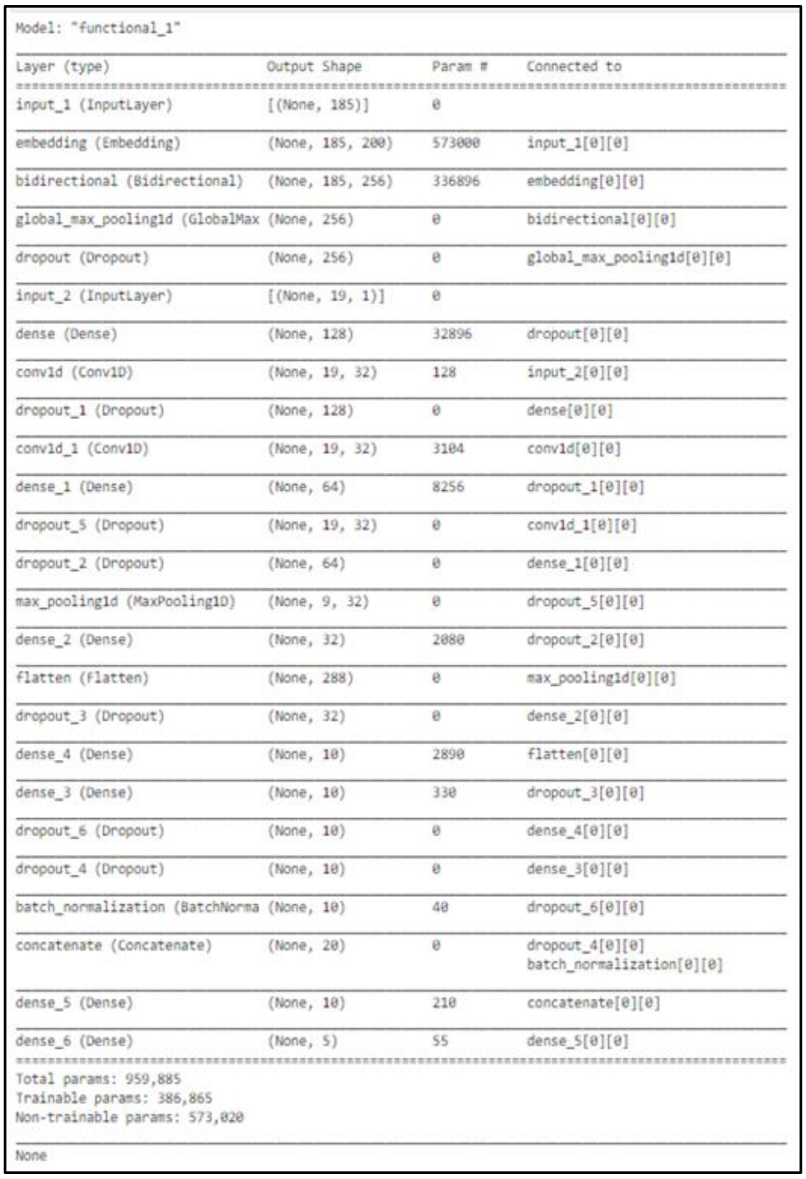
4.1.6.5. Confusion Matrix & Classification Report:





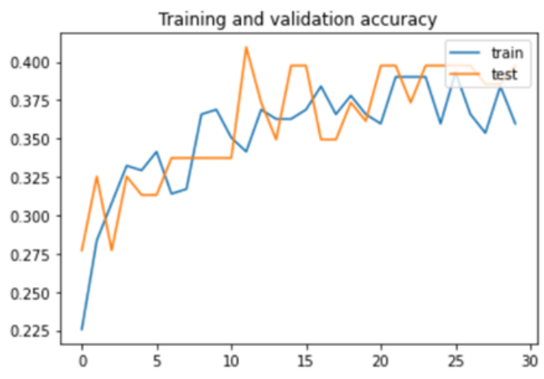
```
              precision    recall  f1-score   support

           0       0.62      0.56      0.59         9
           1       0.00      0.00      0.00        19
           2       0.00      0.00      0.00        21
           3       0.37      1.00      0.54        28
           4       0.00      0.00      0.00         6

    accuracy                           0.40        83
   macro avg       0.20      0.31      0.23        83
weighted avg       0.19      0.40      0.25        83
```

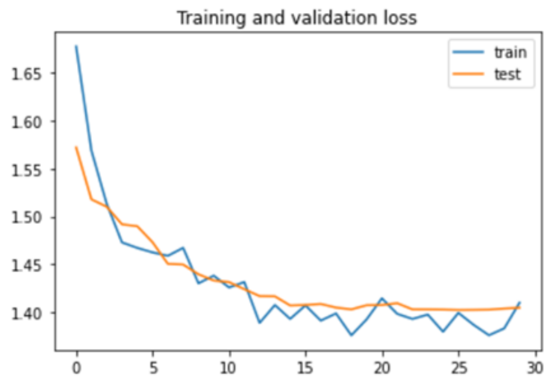
The training and testing accuracy achieved is 36.89% and 39.76%. Also, it can be observed that the model is predicting all the classes as IV except a few.



```
Model: "functional_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 185)]        0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 185, 200)     573000      input_1[0][0]
__________________________________________________________________________________________________
bidirectional (Bidirectional)   (None, 185, 256)     336896      embedding[0][0]
__________________________________________________________________________________________________
global_max_pooling1d (GlobalMax (None, 256)          0           bidirectional[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 256)          0           global_max_pooling1d[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 19, 1)]      0
__________________________________________________________________________________________________
dense (Dense)                   (None, 128)          32896       dropout[0][0]
__________________________________________________________________________________________________
conv1d (Conv1D)                 (None, 19, 32)       128         input_2[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 128)          0           dense[0][0]
__________________________________________________________________________________________________
conv1d_1 (Conv1D)               (None, 19, 32)       3104        conv1d[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           8256        dropout_1[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 19, 32)       0           conv1d_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 64)           0           dense_1[0][0]
__________________________________________________________________________________________________
max_pooling1d (MaxPooling1D)    (None, 9, 32)        0           dropout_5[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 32)           2080        dropout_2[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 288)          0           max_pooling1d[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 32)           0           dense_2[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 10)           2890        flatten[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 10)           330         dropout_3[0][0]
__________________________________________________________________________________________________
dropout_6 (Dropout)             (None, 10)           0           dense_4[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 10)           0           dense_3[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 10)           40          dropout_6[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 20)           0           dropout_4[0][0]
                                                                 batch_normalization[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 10)           210         concatenate[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 5)            55          dense_5[0][0]
==================================================================================================
Total params: 959,885
Trainable params: 386,865
Non-trainable params: 573,020
__________________________________________________________________________________________________
None
```

4.1.6.6. The Model Accuracy and Model Loss graph:

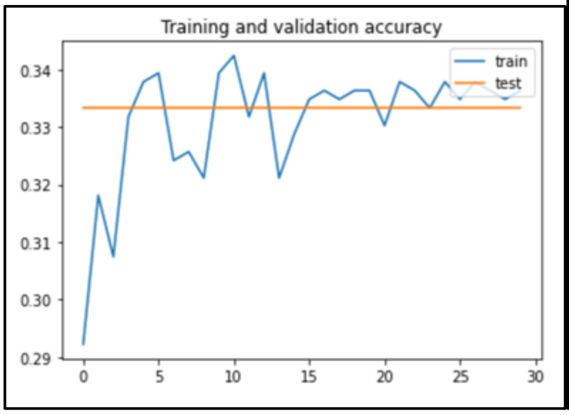


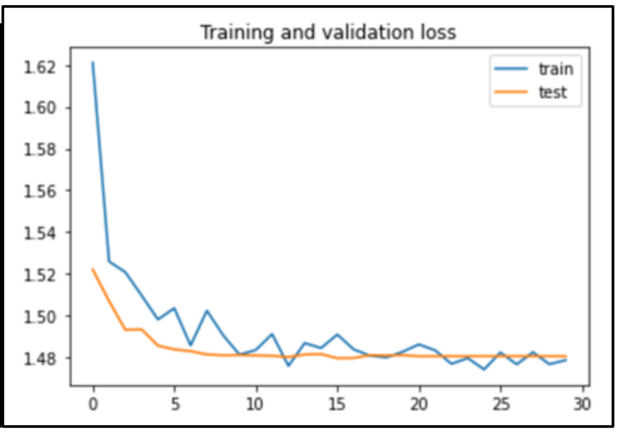


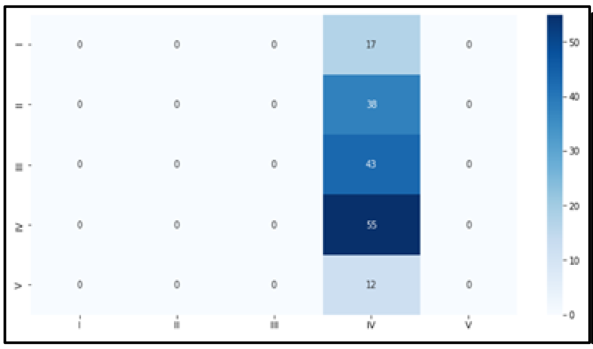
The training loss is being reduced to 15 epochs, after that, it is fluctuating around 1.4 but is not reduced further. Validation is loss is consistent after 15 epochs, no further reduction is observed.

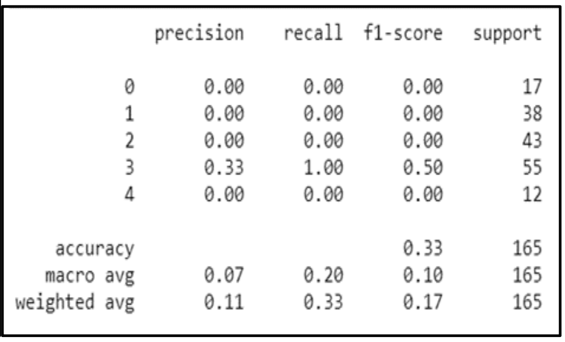
Conclusion- This multi-input model doesn't work well with this data. Since the number of data points is very less, we got very low accuracy, we need more data to improve the model performance

4.1.6.7. CNN + LSTM Multi-Input Model with augmented data









```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        17
           1       0.00      0.00      0.00        38
           2       0.00      0.00      0.00        43
           3       0.33      1.00      0.50        55
           4       0.00      0.00      0.00        12

    accuracy                           0.33       165
   macro avg       0.07      0.20      0.10       165
weighted avg       0.11      0.33      0.17       165
```

The CNN+LSTM model is used with augmented data. It has been observed that validation accuracy is consistent and no learning over each epoch.

From the confusion matrix, it can be observed that all are predicted as class IV.

4.1.7 LSTM + LSTM Multi-Input Model

This multi-input model is the concatenation of the two LSTM models. One LSTM model is given the categorical data to process and the other LSTM model is given the text data (Sequential data) to process. The output layers from both models are concatenated and given to the dense layers to get the output.

In this model, processed description (text data) and categorical data are given as features and the Potential Accident Level is taken as the target variable similar to CNN+LSTM Multi-Input model

4.1.7.1. Model Building and Training

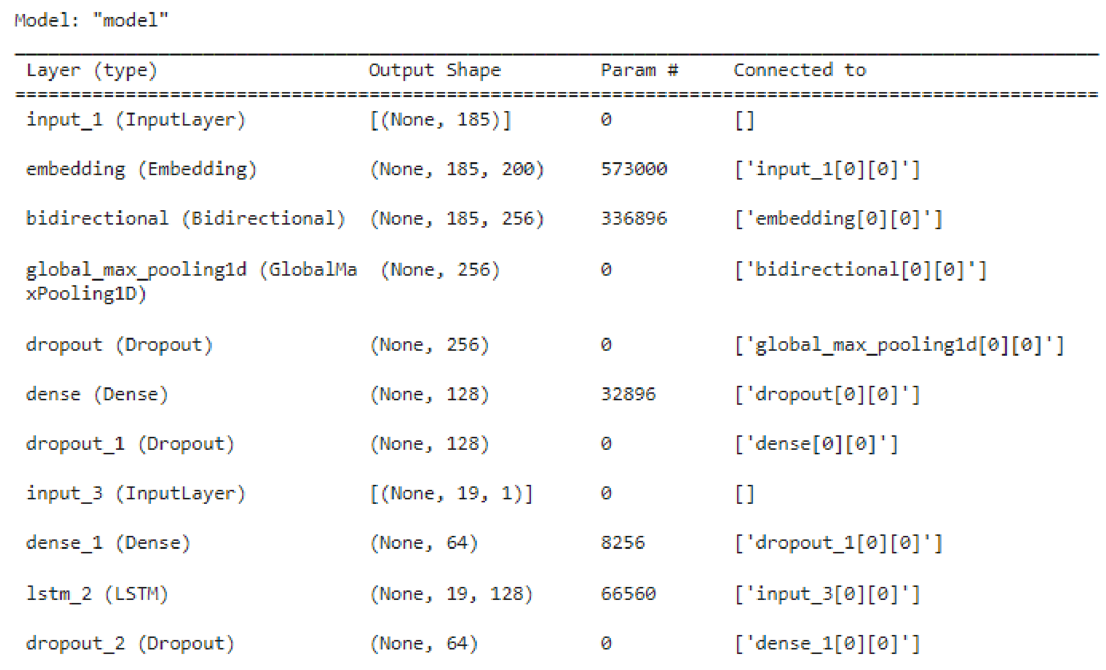
In this model, two branches are there:

- One branch contains the Embedding layer followed by Bidirectional LSTM, max-pooling layer and dense layers along with dropout layers.

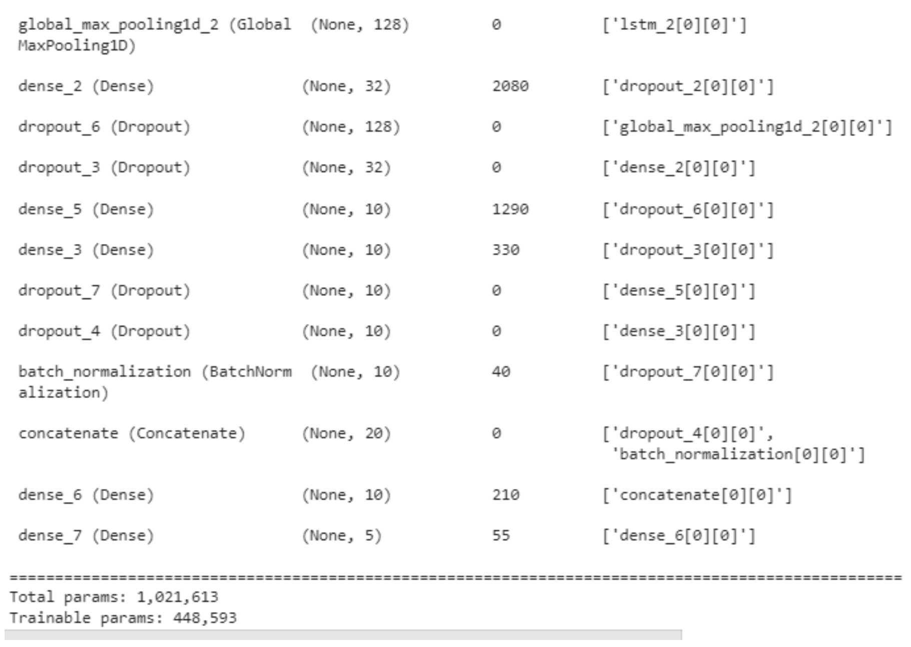
- The second branch contains LSTM followed by Dense layers along with batch normalization and dropout layers.

These two branches are concatenated, and it is given as input to another dense layer with ‘relu’ as an activation function. The last layer is a dense layer with ‘softmax’ as an activation function.

With batch size 8, running for 30 epochs, Optimizer - Adam, Loss - Categorical Cross-Entropy, and Learning rate – 0.01.

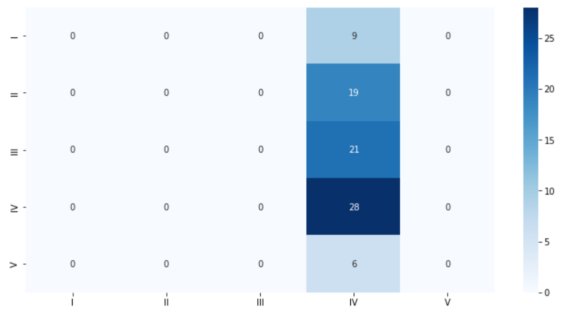


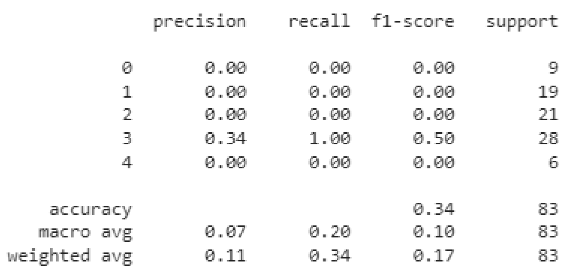
```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 185)]        0           []

 embedding (Embedding)          (None, 185, 200)     573000      ['input_1[0][0]']

 bidirectional (Bidirectional)  (None, 185, 256)     336896      ['embedding[0][0]']

 global_max_pooling1d (GlobalMa  (None, 256)         0           ['bidirectional[0][0]']
 xPooling1D)

 dropout (Dropout)              (None, 256)          0           ['global_max_pooling1d[0][0]']

 dense (Dense)                  (None, 128)          32896       ['dropout[0][0]']

 dropout_1 (Dropout)            (None, 128)          0           ['dense[0][0]']

 input_3 (InputLayer)           [(None, 19, 1)]      0           []

 dense_1 (Dense)                (None, 64)           8256        ['dropout_1[0][0]']

 lstm_2 (LSTM)                  (None, 19, 128)      66560       ['input_3[0][0]']

 dropout_2 (Dropout)            (None, 64)           0           ['dense_1[0][0]']
```



```
 global_max_pooling1d_2 (Global  (None, 128)         0           ['lstm_2[0][0]']
 MaxPooling1D)

 dense_2 (Dense)                (None, 32)           2080        ['dropout_2[0][0]']

 dropout_6 (Dropout)            (None, 128)          0           ['global_max_pooling1d_2[0][0]']

 dropout_3 (Dropout)            (None, 32)           0           ['dense_2[0][0]']

 dense_5 (Dense)                (None, 10)           1290        ['dropout_6[0][0]']

 dense_3 (Dense)                (None, 10)           330         ['dropout_3[0][0]']

 dropout_7 (Dropout)            (None, 10)           0           ['dense_5[0][0]']

 dropout_4 (Dropout)            (None, 10)           0           ['dense_3[0][0]']

 batch_normalization (BatchNorm  (None, 10)          40          ['dropout_7[0][0]']
 alization)

 concatenate (Concatenate)      (None, 20)           0           ['dropout_4[0][0]',
                                                                  'batch_normalization[0][0]']

 dense_6 (Dense)                (None, 10)           210         ['concatenate[0][0]']

 dense_7 (Dense)                (None, 5)            55          ['dense_6[0][0]']

==================================================================================================
Total params: 1,021,613
Trainable params: 448,593
```

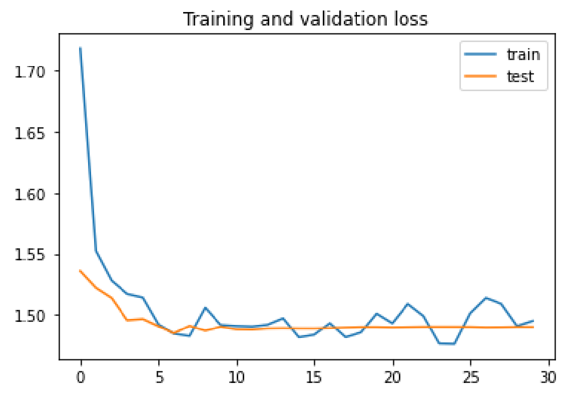
4.1.7.2. Confusion Matrix & Classification Report:

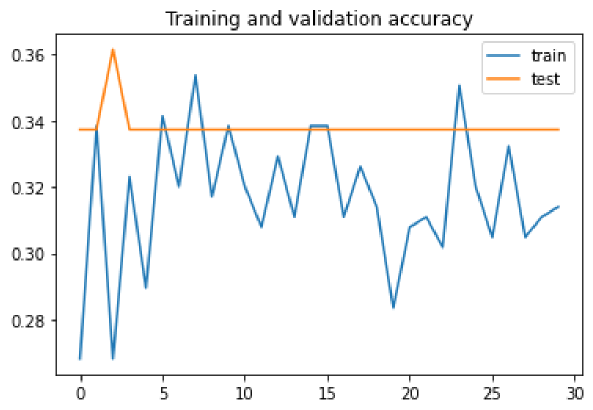




```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         9
           1       0.00      0.00      0.00        19
           2       0.00      0.00      0.00        21
           3       0.34      1.00      0.50        28
           4       0.00      0.00      0.00         6

    accuracy                           0.34        83
   macro avg       0.07      0.20      0.10        83
weighted avg       0.11      0.34      0.17        83
```

4.1.7.3. The Model Accuracy and Model Loss graph:





The training and testing accuracy
achieved is 33% and 34%. Also, it can be observed that the model is predicting all the classes as IV

As the accuracies are not much, it can be said that this multi-input model doesn't work well with this augmented data also.

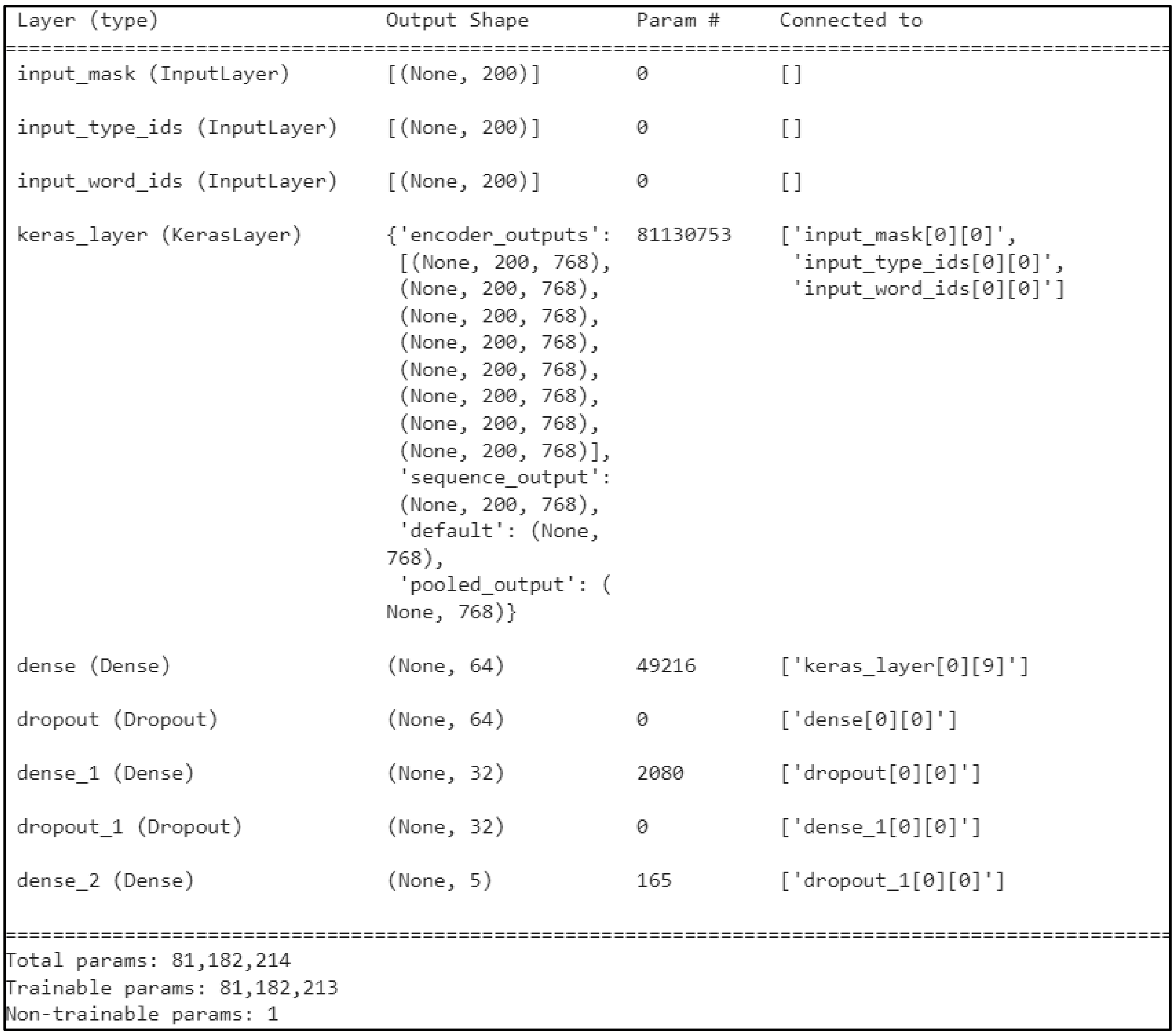
4.1.8. BERT

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training.

4.1.8.1: BERT model to fine-tune:

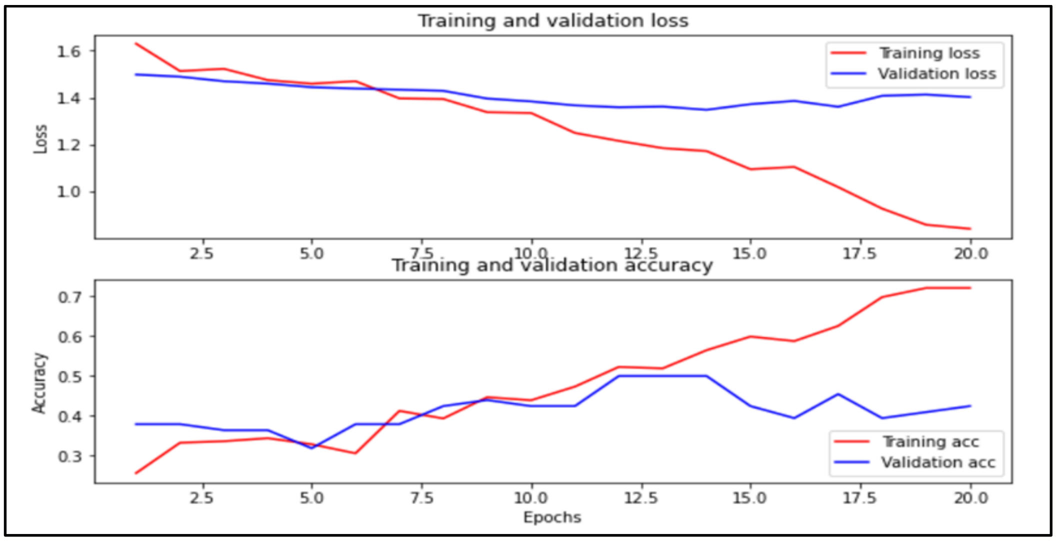
bert_model_name = 'small_bert/bert_en_uncased_L-8_H-768_A-12'

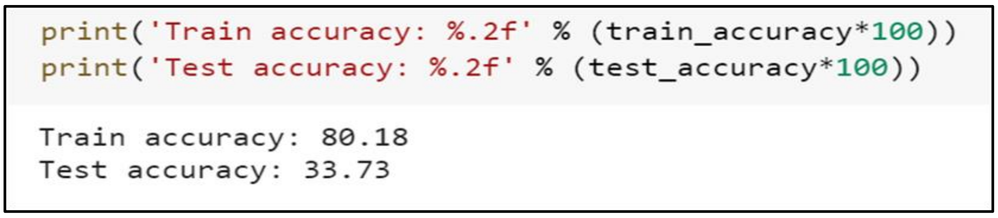
4.1.8.2: Model Building:



```
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_mask (InputLayer)        [(None, 200)]        0           []

 input_type_ids (InputLayer)    [(None, 200)]        0           []

 input_word_ids (InputLayer)    [(None, 200)]        0           []

 keras_layer (KerasLayer)       {'encoder_outputs':  81130753    ['input_mask[0][0]',
                                 [(None, 200, 768),               'input_type_ids[0][0]',
                                 (None, 200, 768),                'input_word_ids[0][0]']
                                 (None, 200, 768),
                                 (None, 200, 768),
                                 (None, 200, 768),
                                 (None, 200, 768),
                                 (None, 200, 768),
                                 (None, 200, 768)],
                                 'sequence_output':
                                 (None, 200, 768),
                                 'default': (None,
                                768),
                                 'pooled_output': (
                                None, 768)}

 dense (Dense)                  (None, 64)           49216       ['keras_layer[0][9]']

 dropout (Dropout)              (None, 64)           0           ['dense[0][0]']

 dense_1 (Dense)                (None, 32)           2080        ['dropout[0][0]']

 dropout_1 (Dropout)            (None, 32)           0           ['dense_1[0][0]']

 dense_2 (Dense)                (None, 5)            165         ['dropout_1[0][0]']

==================================================================================================
Total params: 81,182,214
Trainable params: 81,182,213
Non-trainable params: 1
```

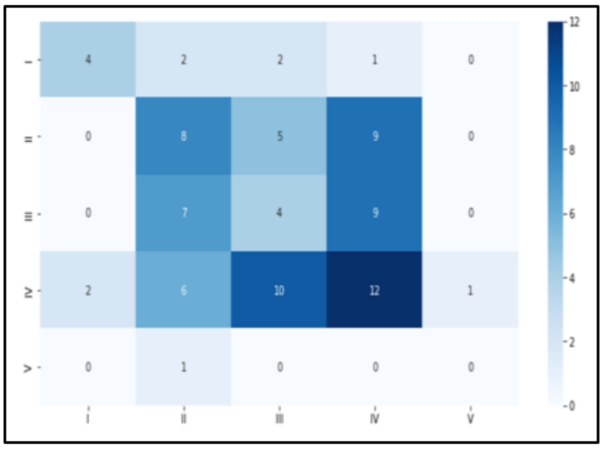
4.1.8.3: The Model Accuracy and Model Loss graph:

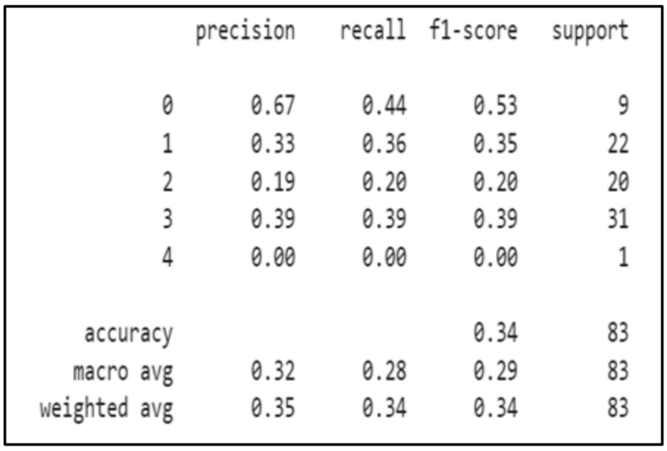




```
print('Train accuracy: %.2f' % (train_accuracy*100))
print('Test accuracy: %.2f' % (test_accuracy*100))

Train accuracy: 80.18
Test accuracy: 33.73
```

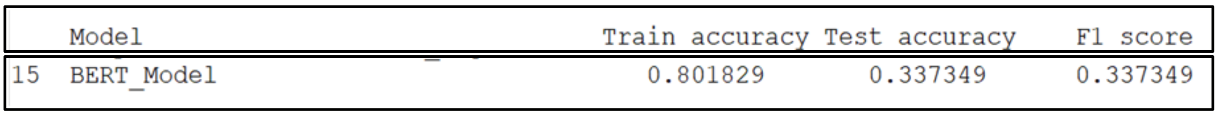
4.1.8.4: Confusion Matrix   & Classification Report





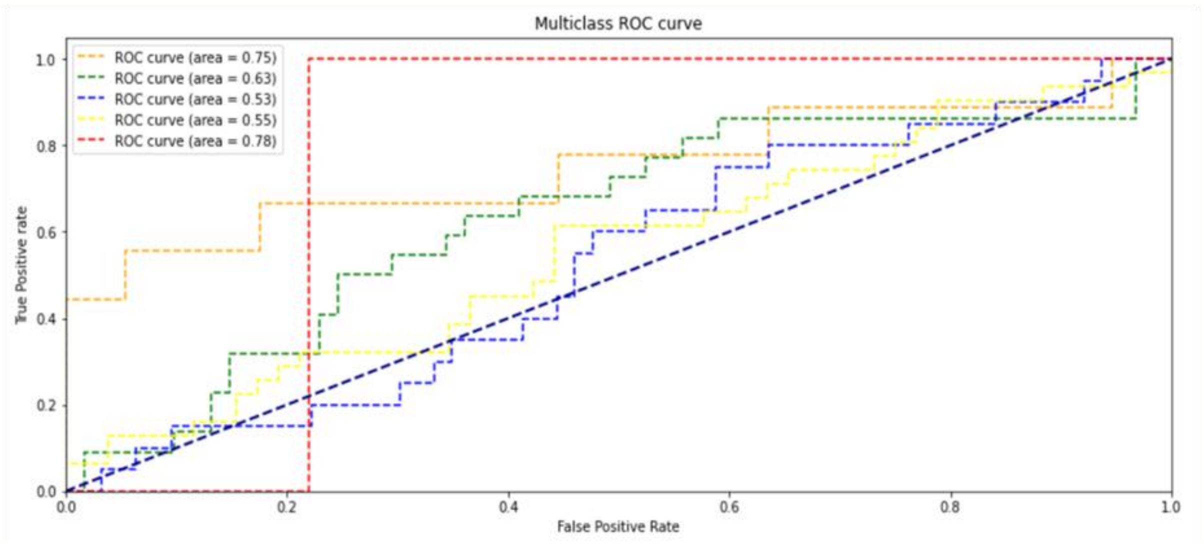
```
              precision    recall  f1-score   support

           0       0.67      0.44      0.53         9
           1       0.33      0.36      0.35        22
           2       0.19      0.20      0.20        20
           3       0.39      0.39      0.39        31
           4       0.00      0.00      0.00         1

    accuracy                           0.34        83
   macro avg       0.32      0.28      0.29        83
weighted avg       0.35      0.34      0.34        83
```

4.1.8.5: Train Accuracy, Test Accuracy, and F1 score:



```
    Model                          Train accuracy Test accuracy   F1 score
15  BERT_Model                           0.801829      0.337349   0.337349
```

4.1.8.6: ROC Curve:



4.1.9. Simple Transformer- Using Pre-Processed Data

A transformer is a deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data. It is used primarily in the field of natural language processing (NLP) and in computer vision (CV).

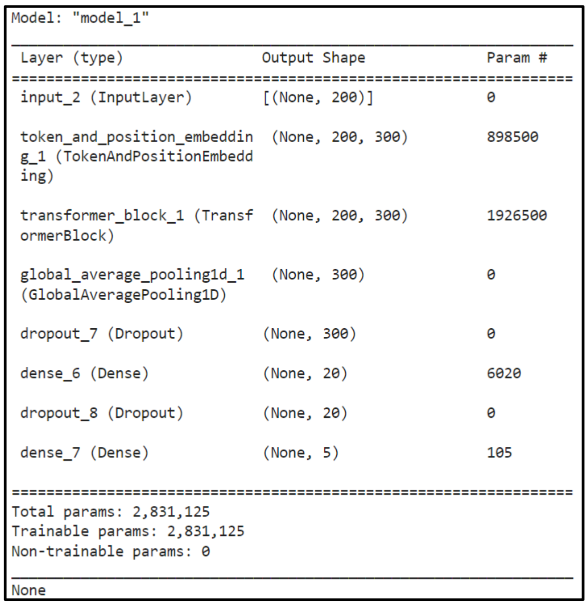
In our model, we took the processed Descriptions(text data) as our features and the Potential Accident Level as our Targets.

4.1.9.1. Targets [Y]- We label encode the targets to convert the Roman letters into numbers ranging from 0 to 4, then we one-hot encode it. The targets will be converted into a size of 411*5.

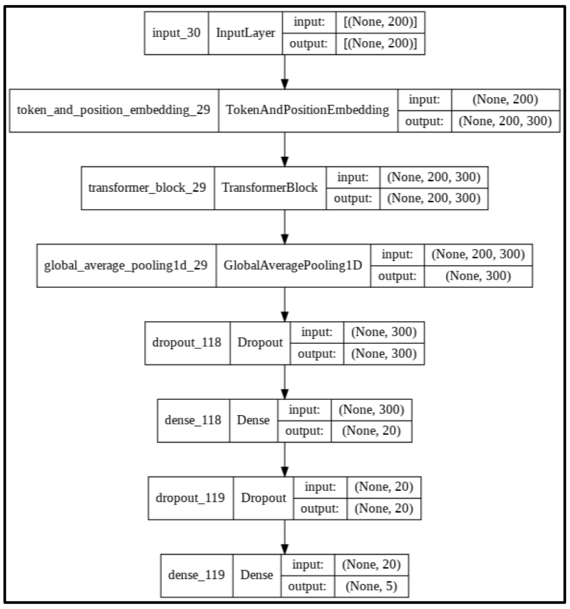
4.1.9.2. Features[X]- We have used Tokeniser from Keras library to convert the words into numbers with maximum words of 10,000.

4.1.9.3. Model Building and Training

Simple Transformer Model with Dense Layer- In this model we used Embedding layer along with 1 dense layer and Drop out layers to gradually reduce the networks from 300 to 20. The first dense layer uses the 'Relu' activation function and the last layer uses the 'Softmax' activation function. With batch size 32, running for 4 epochs, Optimizer - Adam, Loss - Categorical Cross-Entropy, and Learning rate - 1e-03, 1e-04.

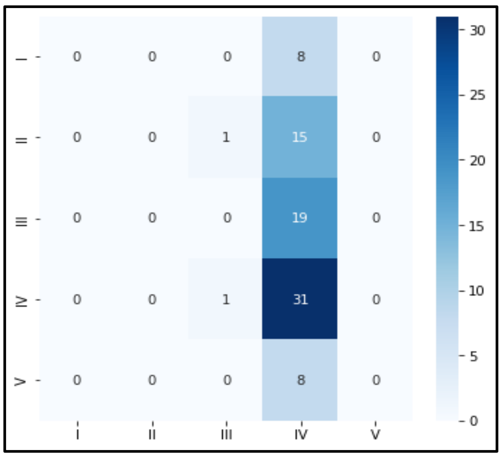


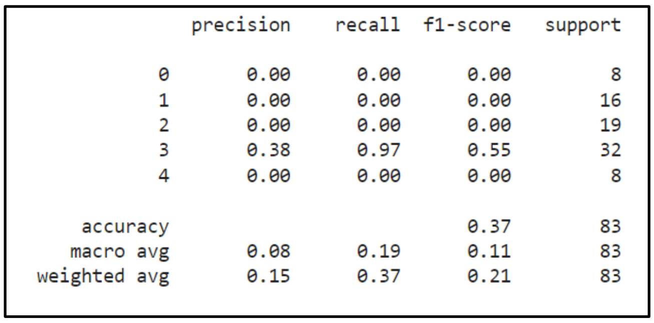
```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 200)]             0

 token_and_position_embeddin  (None, 200, 300)         898500
 g_1 (TokenAndPositionEmbedd
 ing)

 transformer_block_1 (Transf  (None, 200, 300)         1926500
 ormerBlock)

 global_average_pooling1d_1   (None, 300)              0
 (GlobalAveragePooling1D)

 dropout_7 (Dropout)         (None, 300)               0

 dense_6 (Dense)             (None, 20)                6020

 dropout_8 (Dropout)         (None, 20)                0

 dense_7 (Dense)             (None, 5)                 105

=================================================================
Total params: 2,831,125
Trainable params: 2,831,125
Non-trainable params: 0
_________________________________________________________________
None
```



input_30 | InputLayer | input: [(None, 200)] | output: [(None, 200)]
token_and_position_embedding_29 | TokenAndPositionEmbedding | input: (None, 200) | output: (None, 200, 300)
transformer_block_29 | TransformerBlock | input: (None, 200, 300) | output: (None, 200, 300)
global_average_pooling1d_29 | GlobalAveragePooling1D | input: (None, 200, 300) | output: (None, 300)
dropout_118 | Dropout | input: (None, 300) | output: (None, 300)
dense_118 | Dense | input: (None, 300) | output: (None, 20)
dropout_119 | Dropout | input: (None, 20) | output: (None, 20)
dense_119 | Dense | input: (None, 20) | output: (None, 5)

4.1.9.5. Confusion Matrix & Classification Report:

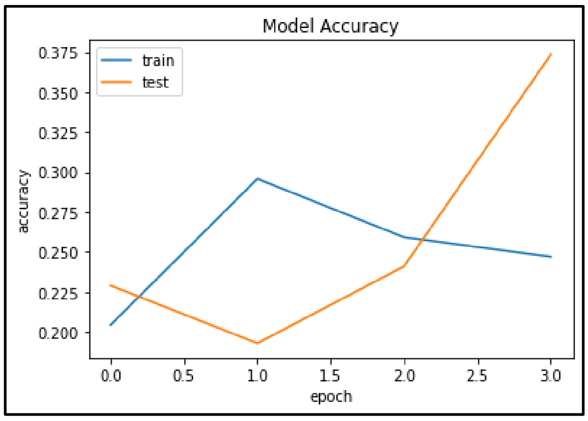


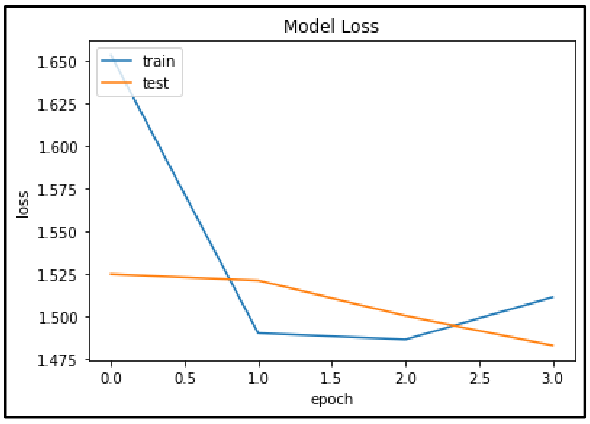


```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.00      0.00      0.00        16
           2       0.00      0.00      0.00        19
           3       0.38      0.97      0.55        32
           4       0.00      0.00      0.00         8

    accuracy                           0.37        83
   macro avg       0.08      0.19      0.11        83
weighted avg       0.15      0.37      0.21        83
```

The training and testing accuracy achieved is 34.14% and 37.34%. Also, We can observe that the model is predicting all the classes as IV.

4.1.9.6. The Model Accuracy and Model Loss graph:



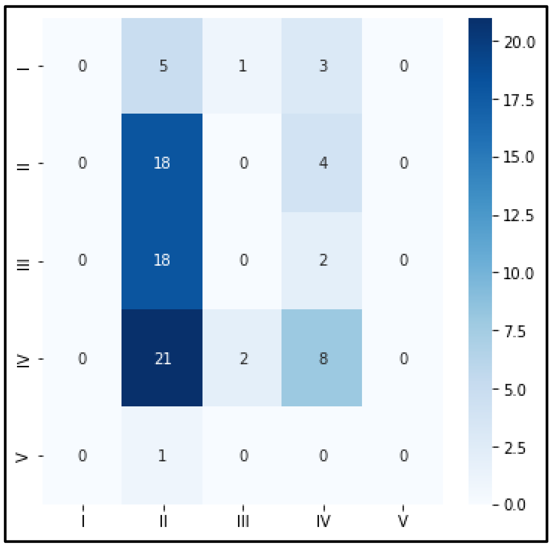


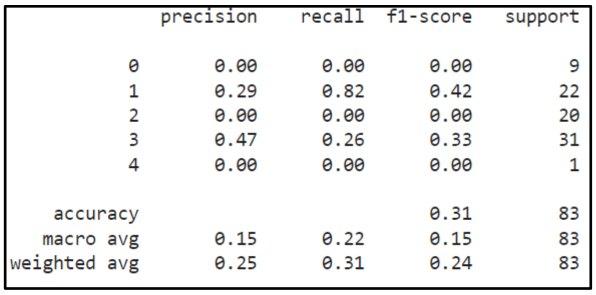
Conclusion- Simple Transformer- With Pre-Processed data is not performing well with the provided dataset.

4.1.10. Simple Transformer- Using Augmented Training Data

In this model, we have used the same Simple Transformer Model architecture as before but the training with augmented training data. For data augmentation, we have used the library ‘nlpaug’.

4.1.10.1. Confusion Matrix & Classification Report:

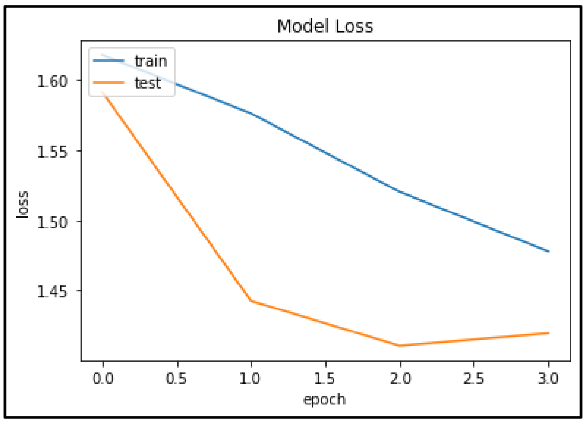


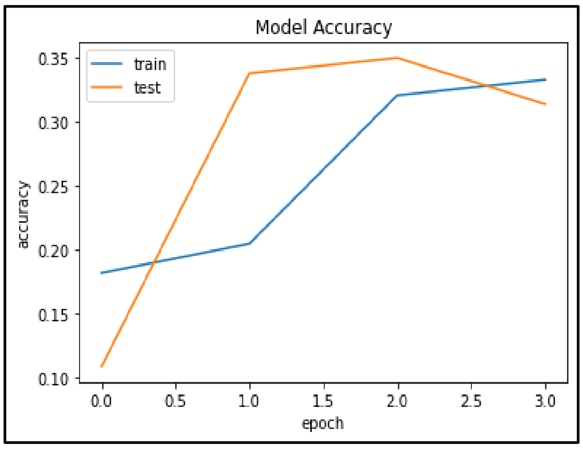


```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         9
           1       0.29      0.82      0.42        22
           2       0.00      0.00      0.00        20
           3       0.47      0.26      0.33        31
           4       0.00      0.00      0.00         1

    accuracy                           0.31        83
   macro avg       0.15      0.22      0.15        83
weighted avg       0.25      0.31      0.24        83
```

The training and testing accuracy achieved is 37.04% and 31.32%. Also, We can observe that the model is predicting all the classes as II and IV.

4.1.10.2. The Model Accuracy and Model Loss graph:

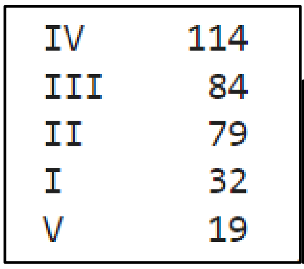




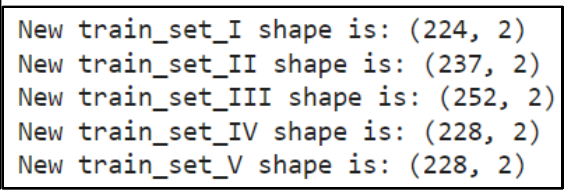
Conclusion- Simple Transformer- With Augmented data is not performing well with the provided dataset.

4.1.11. Simple Transformer- Using Augmented Training Data With Target Class Balanced

In this model, we have used the same Simple Transformer Model architecture as before but the training with augmented training data. For data augmentation, we have used the library 'nlpaug'. The target label i.e., Potential_Accident_Level has been balanced.



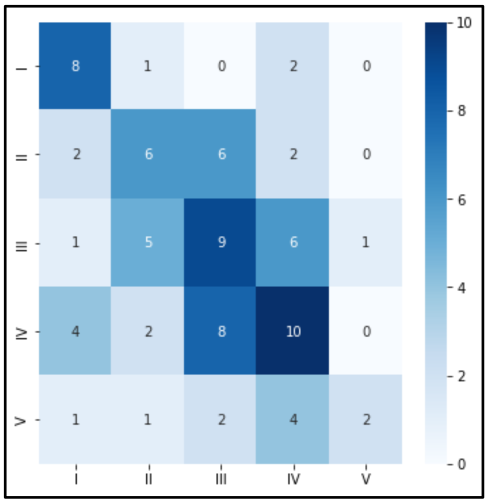
```
IV      114
III      84
II       79
I        32
V        19
```

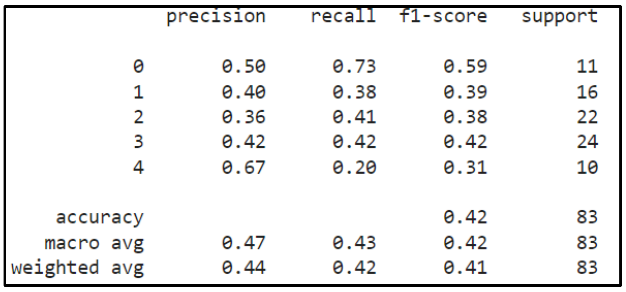


```
New train_set_I shape is: (224, 2)
New train_set_II shape is: (237, 2)
New train_set_III shape is: (252, 2)
New train_set_IV shape is: (228, 2)
New train_set_V shape is: (228, 2)
```

Before Balancing After Balancing

4.1.11.1. Confusion Matrix & Classification Report:

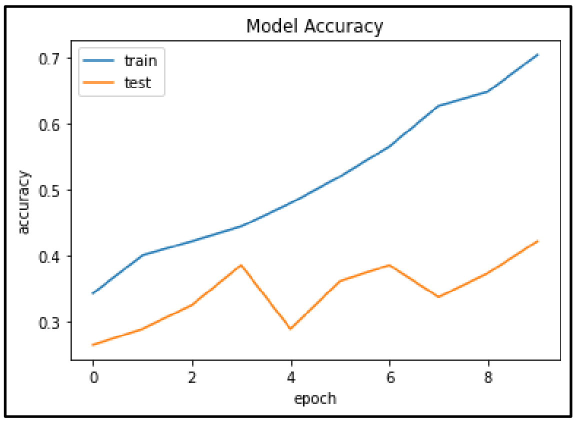


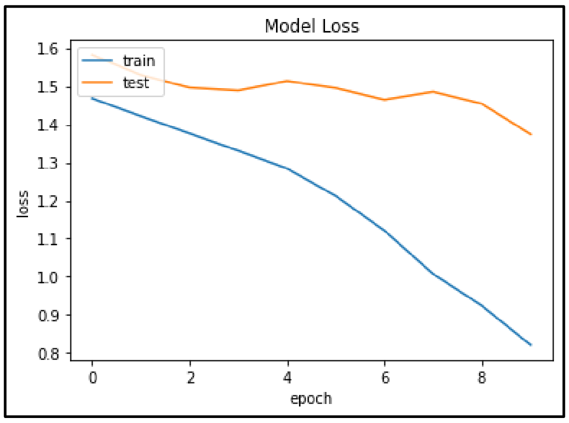


```
              precision    recall  f1-score   support

           0       0.50      0.73      0.59        11
           1       0.40      0.38      0.39        16
           2       0.36      0.41      0.38        22
           3       0.42      0.42      0.42        24
           4       0.67      0.20      0.31        10

    accuracy                           0.42        83
   macro avg       0.47      0.43      0.42        83
weighted avg       0.44      0.42      0.41        83
```

The training and testing accuracy achieved is 82.20% and 42.16%. The model is clearly an overfit model. Also, We can observe that the model is predicting all the classes for the test data set which is an improvement over the previous Transformer Models used.

4.1.11.2. The Model Accuracy and Model Loss graph:





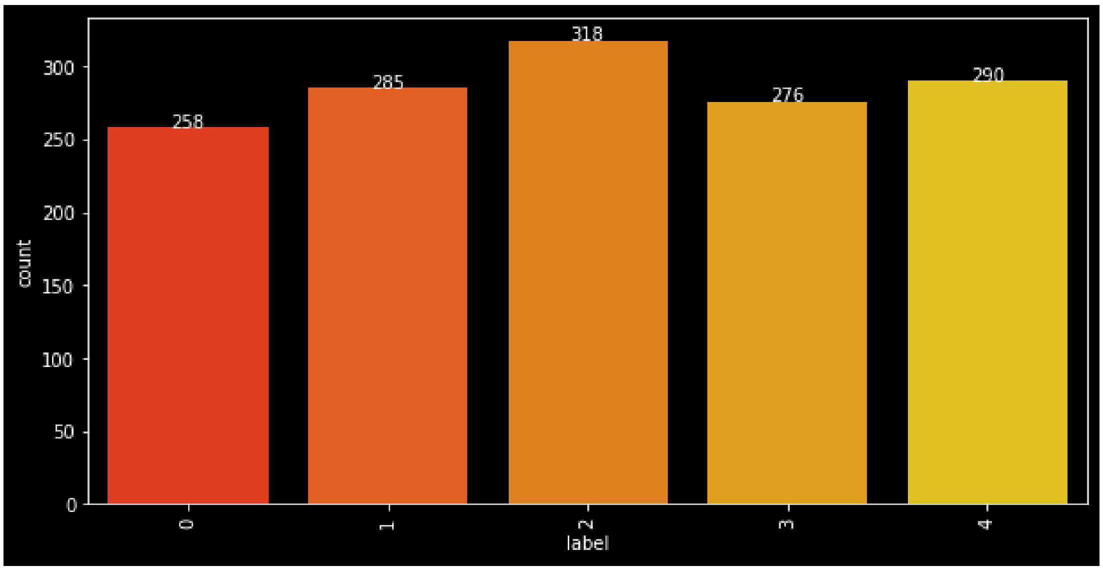
Conclusion: Simple Transformer- Using Augmented Training Data With Target Class Balanced is performing well in the case of the Train Data set but not well in Test. It's an overfit model and not fit for the provided dataset.

4.1.12. Simple Transformers with Distill BERT (Custom)

Simple Transformers is a Natural Language Processing (NLP) library designed to simplify the usage of Transformer models without having to compromise on utility. It is built on the amazing work of Hugging Face and its Transformers library.

```
For installation - !pip install --upgrade simpletransformers
```

From this library we can customize the inputs, augmentation, and target balancing. We Augmentation is based on synonym replacement for the entire dataset and split it into test and train. We augment the data based on the target size and the final result is shown below.



4.1.12.1. Target balancing

```
df_train.label.value_counts()
```

```
2    254
4    232
1    228
3    221
0    206
Name: label, dtype: int64
```

```
df_test.label.value_counts()
```

```
2    64
4    58
1    57
3    55
0    52
Name: label, dtype: int64
```

We need to enable the GPU in colab and Cuda is available to do parallel processing and fast computation and we are using Wandb to plot the graphs.

```
import torch
torch.cuda.is_available()
```

```
True
```

```
import wandb
wandb.login()
```

```
wandb: Appending key for api
True
```

4.1.12.2. Model Building:

```
model_type = 'distilbert'
model_name = 'distilbert-base-cased'


train_args = {
    "reprocess_input_data": True,
    "overwrite_output_dir": True,
    "use_cached_eval_features": True,

    #early stopping
    "early_stopping_delta": 0.01,
    "early_stopping_metric": "loss",
    "use_early_stopping": True,
    "early_stopping_metric_minimize": False,
    "early_stopping_patience": 3,
    "evaluate_during_training_steps": 500,
```

```
# size
"train_batch_size": 32, #
"max_seq_length": 200, # use small value f
"num_train_epochs": 20,

"wandb_project": "Chatbot_Distillbert-2",
```

Model Parameters

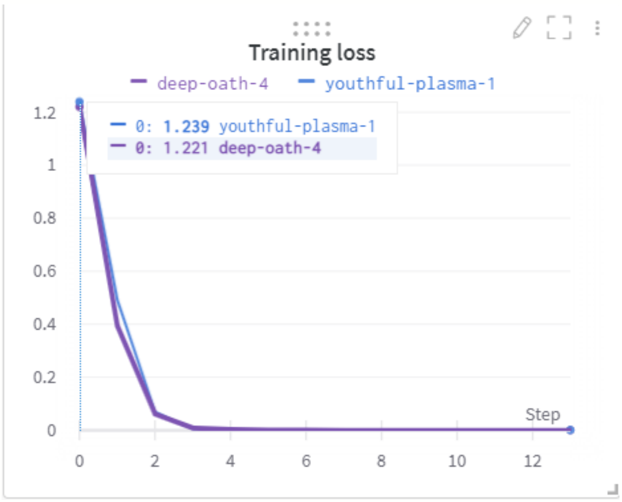
After running the model the train and test accuracy we got is

```
result_distilbert

{'acc': 0.9825174825174825,
 'eval_loss': 0.163581965686558,
 'mcc': 0.9783924226872274}
```

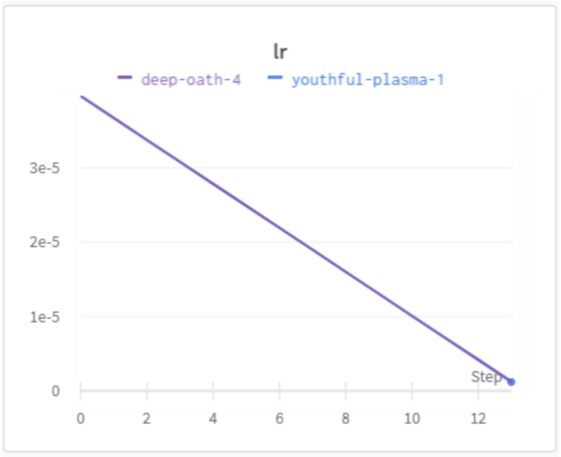
Test Accuracy

```
result_distilbert_train

{'acc': 0.9825174825174825,
 'eval_loss': 0.163581965686558,
 'mcc': 0.9783924226872274}
```

Train Accuracy

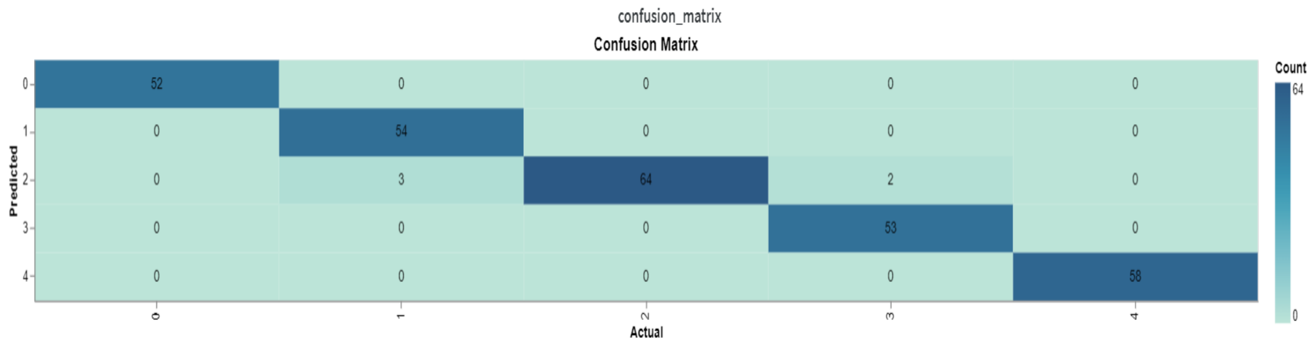


Training loss

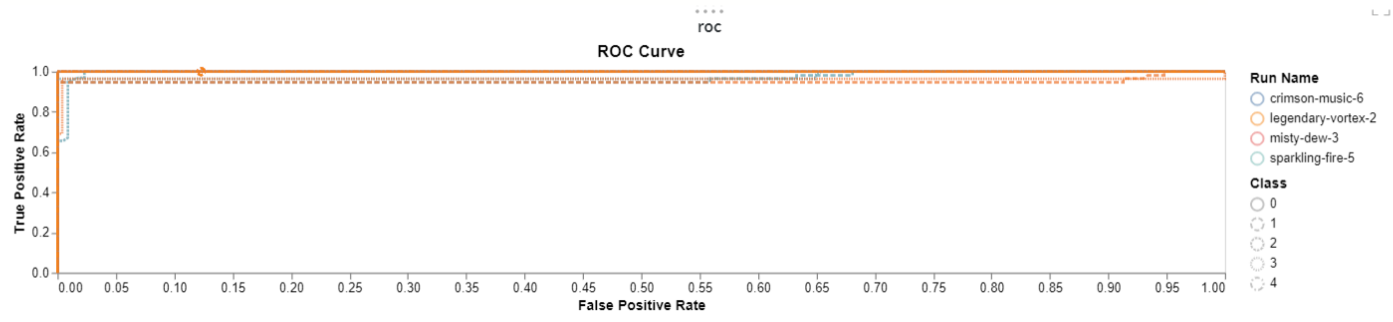


Learning Rate

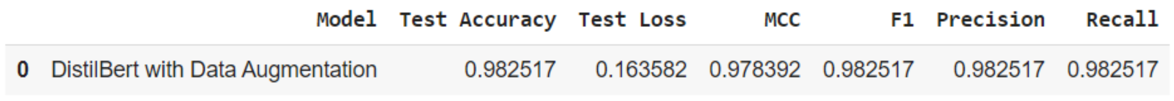
4.1.12.3. Confusion Matrix



4.1.12.4. ROC Curve



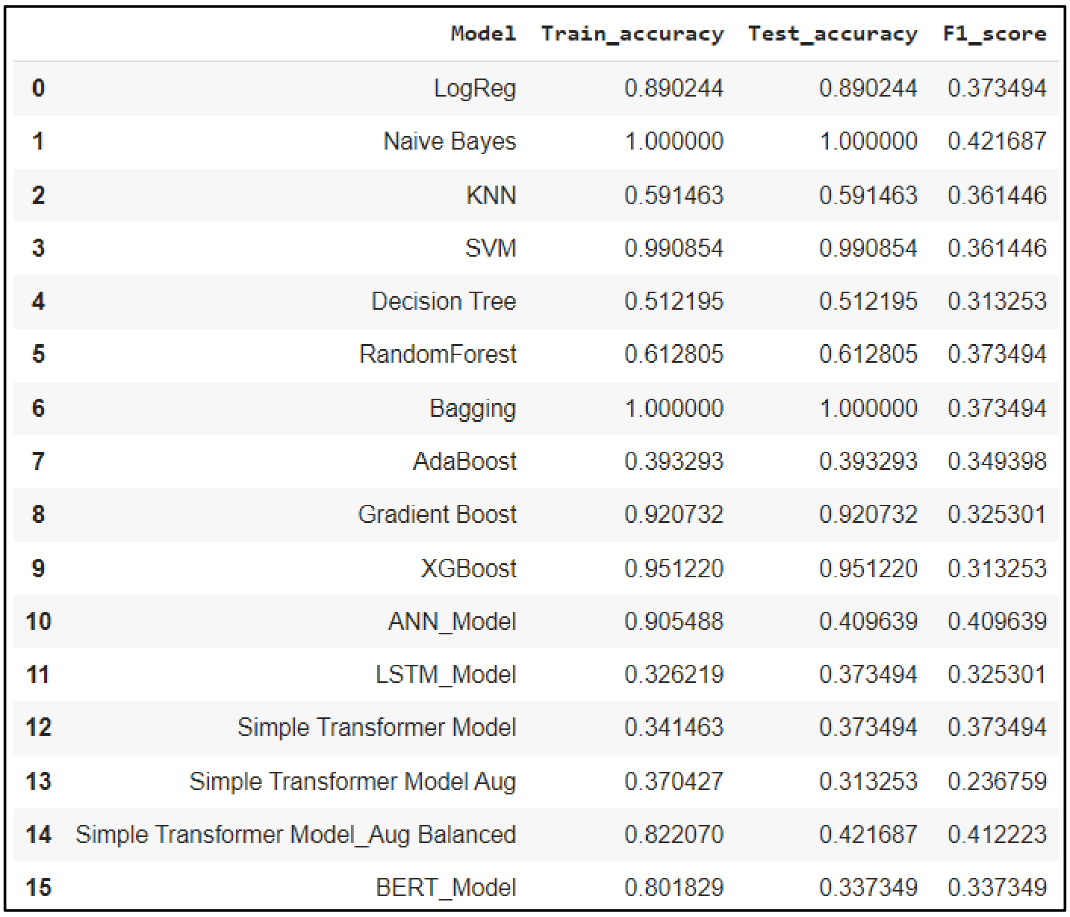
4.1.12.5. Accuracy and F1 score



	Model	Test Accuracy	Test Loss	MCC	F1	Precision	Recall
0	DistilBert with Data Augmentation	0.982517	0.163582	0.978392	0.982517	0.982517	0.982517

4.2 MODEL SUMMARY

The summary of the above-explored models is shown in the below table.



	Model	Train_accuracy	Test_accuracy	F1_score
0	LogReg	0.890244	0.890244	0.373494
1	Naive Bayes	1.000000	1.000000	0.421687
2	KNN	0.591463	0.591463	0.361446
3	SVM	0.990854	0.990854	0.361446
4	Decision Tree	0.512195	0.512195	0.313253
5	RandomForest	0.612805	0.612805	0.373494
6	Bagging	1.000000	1.000000	0.373494
7	AdaBoost	0.393293	0.393293	0.349398
8	Gradient Boost	0.920732	0.920732	0.325301
9	XGBoost	0.951220	0.951220	0.313253
10	ANN_Model	0.905488	0.409639	0.409639
11	LSTM_Model	0.326219	0.373494	0.325301
12	Simple Transformer Model	0.341463	0.373494	0.373494
13	Simple Transformer Model Aug	0.370427	0.313253	0.236759
14	Simple Transformer Model_Aug Balanced	0.822070	0.421687	0.412223
15	BERT_Model	0.801829	0.337349	0.337349

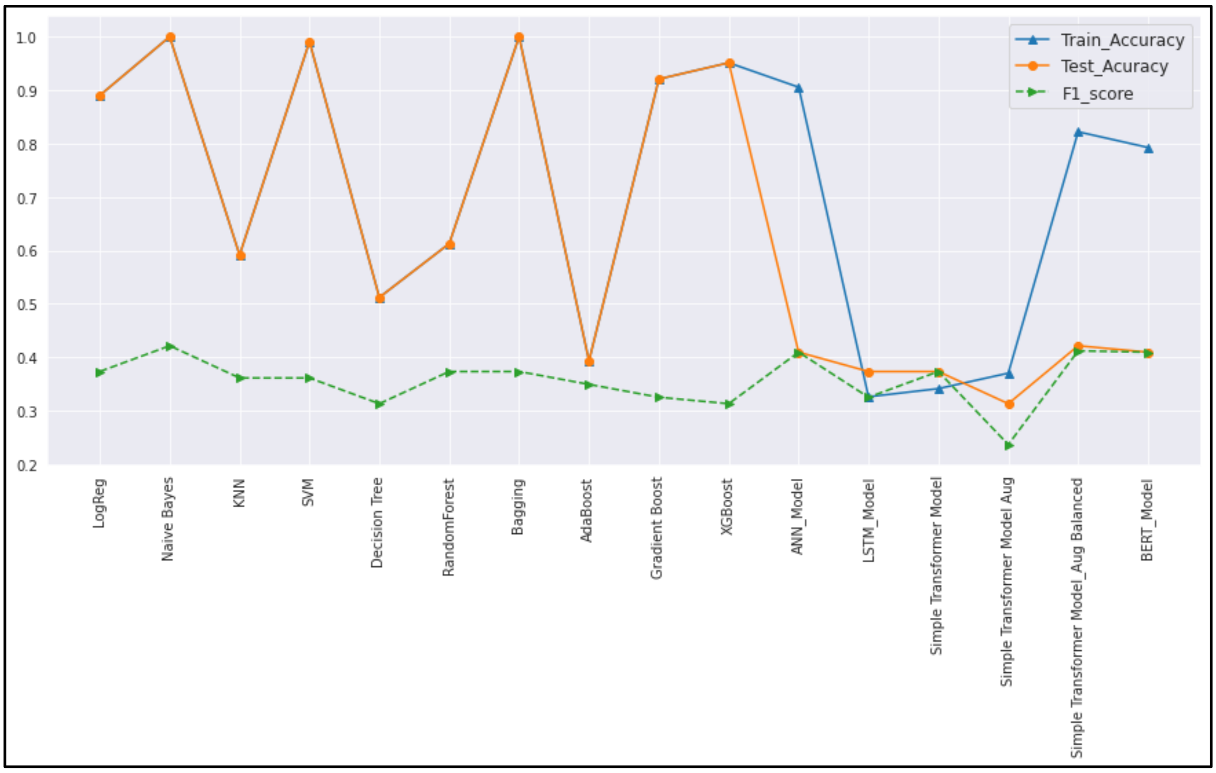
5. BENCHMARKING- COMPARISON OF EXPERIMENTS

5.1. Model Performance and Comparisons

Most model-performance measures are based on the comparison of the model's predictions with the (known) values of the dependent variable in a dataset. We have built 10 different Machine learning models, an Artificial Neural Network model, an LSTM model, and a Simple Transformer Model for analysis. The Naive Bayes and Bagging model clearly overfits the data. We have good train and test accuracy for logistic regression but a low F1 score so we can neglect it. The F1 scores are very less for all the models.

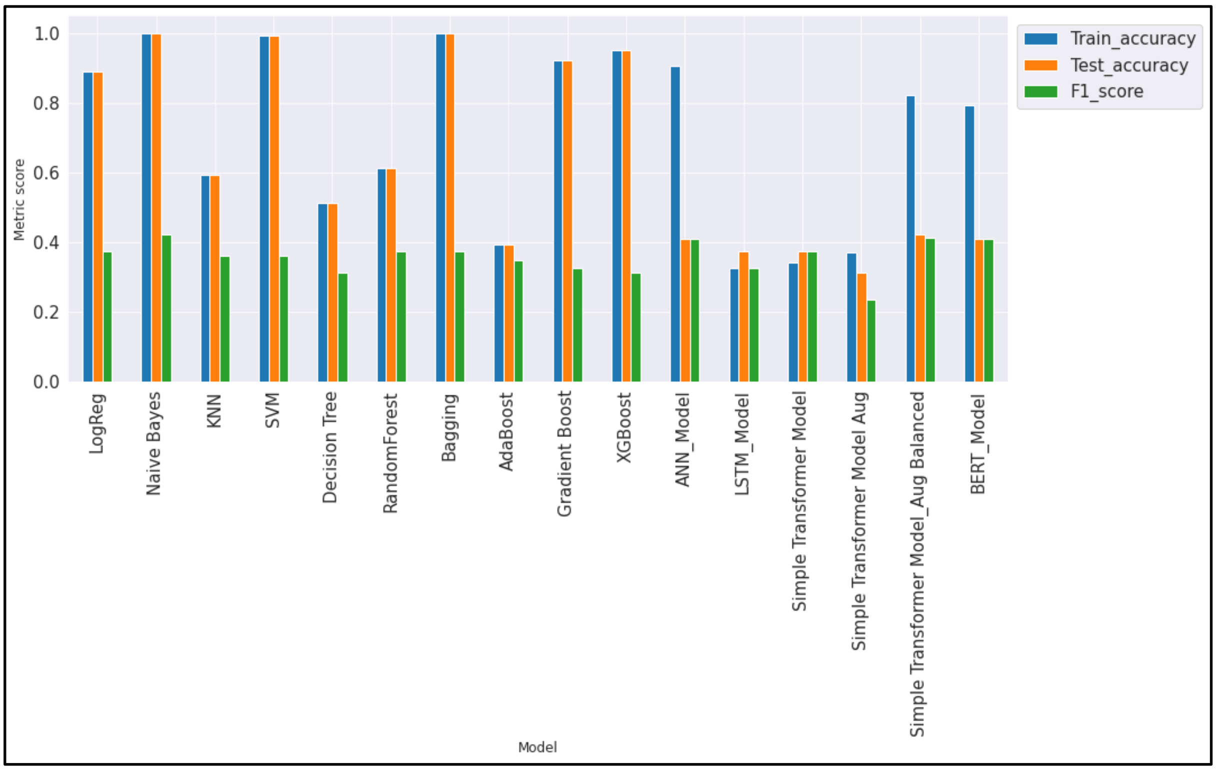
The model with a Higher F1 Score is the ANN model and the Simple Transformer model with balanced training class labels but still, the model overfits with higher train accuracy and low test accuracy.

Model Performance - Train Accuracy, Test Accuracy, and F1 Score



Model Performance graphs

From the Graphs we can see that most of the ML models are overfit, the ANN, LSTM, and Simple Transformer with balanced training labels have decent test accuracy but again low F1 Score.



Metric Score for all the models

5.2. Final Model Selection

The above summary of all models shows that all ML Models are overfit and ANN, LSTM, Simple Transformer Models, BERT have good training accuracy but less test accuracy.

If we compare the confusion matrix and F1-score for ANN, LSTM, Simple Transformer, and BERT models, then we have decent test accuracy in **Simple Transformer Model with Balanced Target Class and BERT model.**

So, the final model we have used in our chatbot for the prediction of Potential_Accident_Level is **Simple_Transformer_Model with Balanced Target Class** having F1-score of 41.2% which is higher than that of **BERT**.

6. GUI BUILDING:

GUI means **Graphical User Interface**. It is the common user Interface that includes Graphical representation like buttons and icons, and communication can be performed by interacting with these icons rather than the usual text-based or command-based communication.

6.1. Chatbot

The **Chatbot** is a Machine Learning based conversational dialog engine built using Python which makes it possible to generate responses based on collections of known conversations. The language-independent design of ChatterBot allows it to be trained to speak any language.

6.1.1. Using Flask - Web Services

Flask is a web framework, it's a Python module that lets you develop web applications easily. It has a small and easy-to-extend core: it's a microframework that doesn't include an ORM (Object Relational Manager) or such features. It does have many cool features like url routing, template engine. It is a WSGI web app framework.

We used Flask library to create a chatbot using web services. We created a style.css file for our webpage styling and index.html file for creating a webpage.

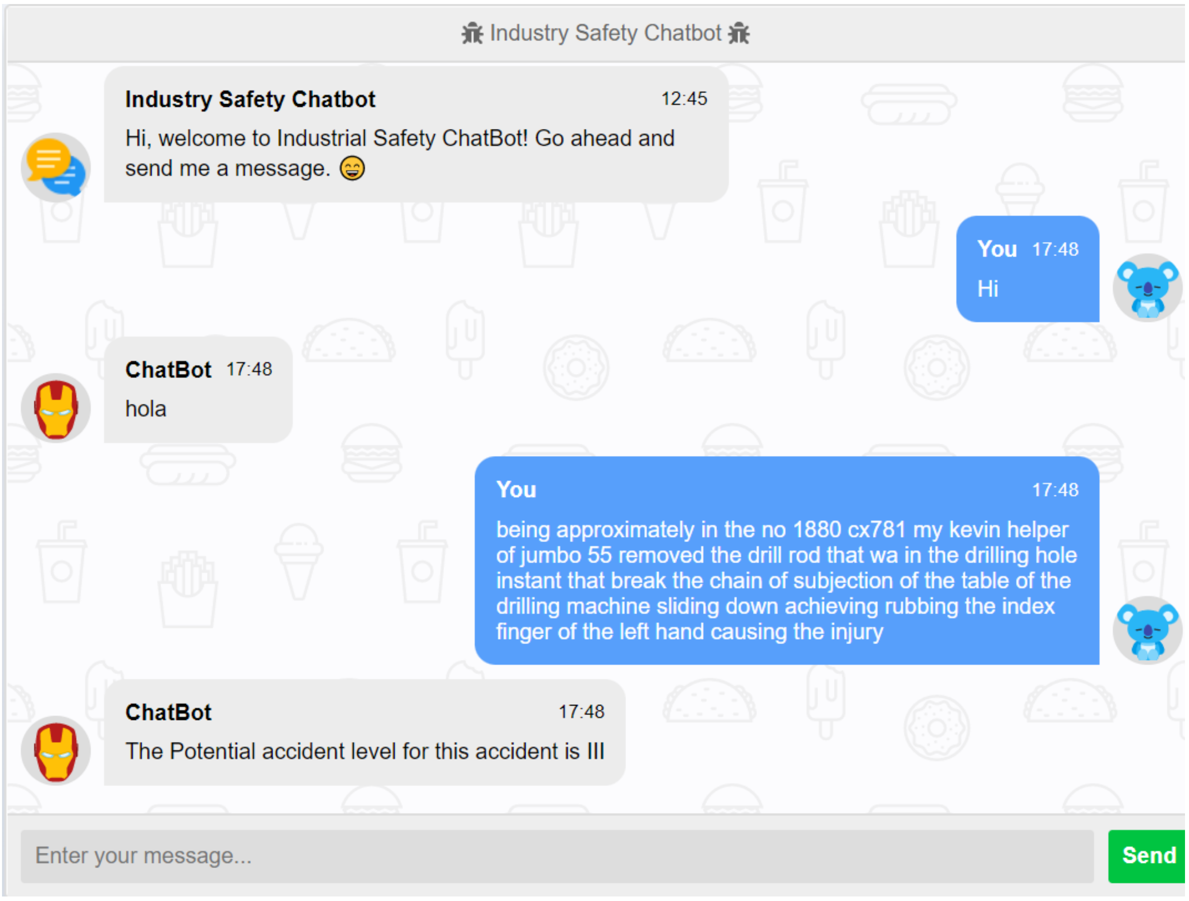
Libraries to install beforehand for flask ngrok

```
!wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.tgz

!tar -xvf /content/drive/MyDrive/AIML/Capstone/Jan-G4---NLP-
Chatbot/ngrok-stable-linux-amd64.tgz

!./ngrok authtoken 23H0IY10fqeKMIW7kG05JhKZMae_3Zabr2iqkU9AUcZ7CrRTP
```

6.1.1.2. Code for Running flask

```
from flask import Flask, render_template, request
from flask_ngrok import run_with_ngrok

app = Flask(__name__)
app.static_folder = 'static'
run_with_ngrok(app)

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/get")
def get_bot_response():
    userText = request.args.get('msg')
    return chat_res(userText)


if __name__ == "__main__":
    app.run()
```

6.1.1.3. Chatbot using FLASK



Industry Safety Chatbot

Industry Safety Chatbot 12:45
Hi, welcome to Industrial Safety ChatBot! Go ahead and send me a message. 😄

You 17:48
Hi

ChatBot 17:48
hola

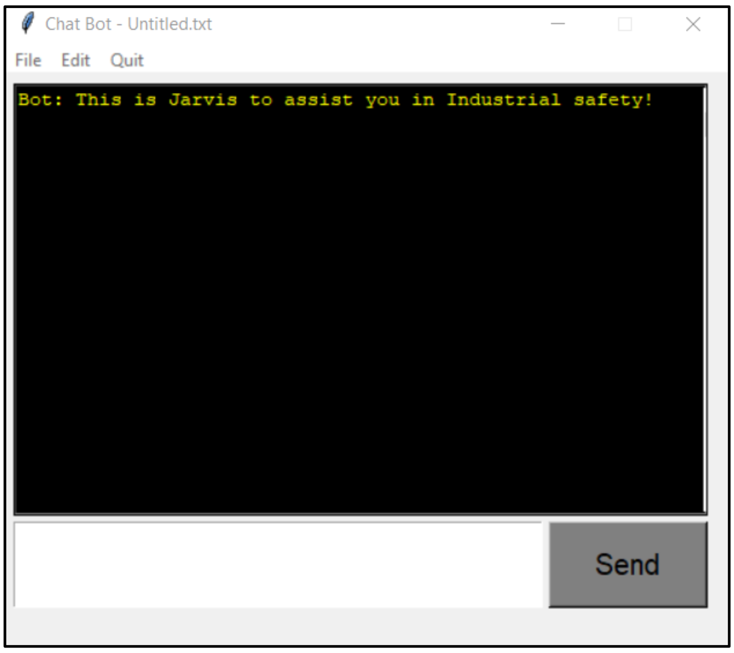
You 17:48
being approximately in the no 1880 cx781 my kevin helper of jumbo 55 removed the drill rod that wa in the drilling hole instant that break the chain of subjection of the table of the drilling machine sliding down achieving rubbing the index finger of the left hand causing the injury

ChatBot 17:48
The Potential accident level for this accident is III

6.1.2. Using Tkinter - Clickable UI

Our Project Objective is to design a Clickable UI-based chatbot interface that accepts text as input and replies back with relevant answers.AI chatbot with GUI using Python Tkinter. This chatbot uses NLP(Natural language Processing) and takes an Article as input and responds to user commands based on that Article.

When receiving the greeting message from the User our industry safety Chatbot greets the User. After that, while entering the description it will process the text with our finalized model and predict the Potential accident level and return it back to the user.



```
Bot: This is Jarvis to assist you in Industrial safety!
```

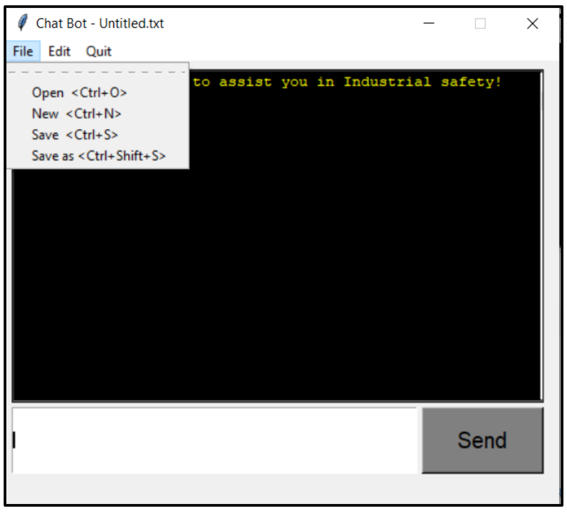
Chatbot Layout

The followings are the layout of Chatbot:

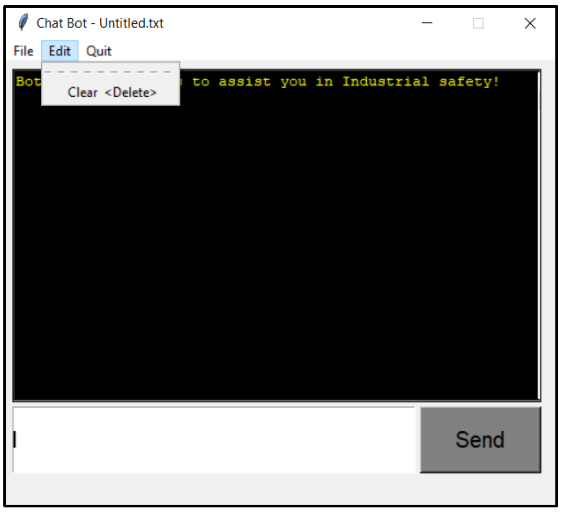
- ➢ The black window is the text window - where all our conversations are displayed
- ➢ The White Box is the Chat Window - where the User enters the text
- ➢ Send Button - To send the chat to the bot.
- ➢ Menu bar - contains File, Edit, Quit menus.

The Menu bars have some events like

- ➢ New - Open a new window to start the conversation
- ➢ Open - To open already saved conversation
- ➢ Save as - Create a new .txt file and save our conversation in our local drive
- ➢ Save - Save the current Conversation
- ➢ Clear - To clear the conversation

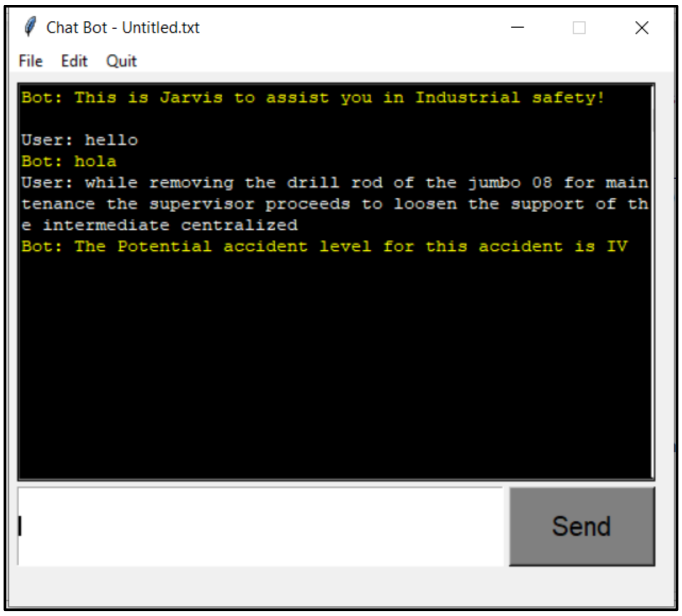


```
to assist you in Industrial safety!
```



```
Bot                  to assist you in Industrial safety!
```

Menus on chatbot



```
Bot: This is Jarvis to assist you in Industrial safety!

User: hello
Bot: hola
User: while removing the drill rod of the jumbo 08 for main
tenance the supervisor proceeds to loosen the support of th
e intermediate centralized
Bot: The Potential accident level for this accident is IV
```

Chatbot with sample data and prediction

7. IMPLICATION- BUSINESS VALUE DERIVED

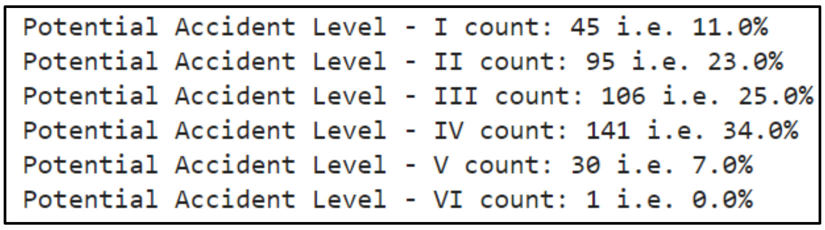
As the increase in the number of injuries/accidents in industrial plants sometimes leads to the death of the employees, there should be a methodology to analyze the risks of the accidents. For example, Accident Level and Potential Accident Level.

Applying the NLP and Machine Learning techniques to classify the potential risk of the accidents by analyzing the incident description, helps the SME or professionals to highlight the safety risk and take proper measures to prevent such risks and damage.

8. LIMITATIONS AND SCOPE OF IMPROVEMENT

8.1. Limitations of Data

The target label i.e., 'Potential_Accident_Level' is imbalanced in the dataset due to less number of observations. Also, there is a lack of quality data.



```
Potential Accident Level - I count: 45 i.e. 11.0%
Potential Accident Level - II count: 95 i.e. 23.0%
Potential Accident Level - III count: 106 i.e. 25.0%
Potential Accident Level - IV count: 141 i.e. 34.0%
Potential Accident Level - V count: 30 i.e. 7.0%
Potential Accident Level - VI count: 1 i.e. 0.0%
```

As the number of observations in Potential_Accident_Level VI is very less so we have merged the Potential_Accident_Level V and VI in one group.

There are fewer features available in the dataset.

8.2. Scope of Improvement

1. Ask for properly balanced data or quality data between the groups
2. Collecting more data for other Classes and trying to minimize the difference between each class
3. Combine similar classes with business decisions
4. Use of more advanced techniques for generating synthetic data

9. CLOSING REFLECTION

1. In a total of 6 target classes of 'Potential_Level_Accident', the imbalance data for class III and class IV consists of almost 59% of the total observations. Hence, the model accuracy was less and lead to an overfit model
2. The Simple Transformer Model with balanced class data with data augmentation gives an accuracy of 82% in the case of training but 42% in testing. Also, it should be noted that the F1 score of 42% is also higher in all the explored models

10. FINAL NOTE

1. In this project, we have learned about the pre-processing of text using various modules
2. Learned about different techniques of Exploratory Data Analysis and derived informative conclusions
3. We have applied the Statistical approach while solving business problems like Hypothesis testing
4. Learned to handle imbalance dataset for target class
5. Learned and used the various methods of Data Augmentation techniques such as 'nlpaug'
6. Built more than 20 models and explored their results
7. Built Hybrid models such as CNN + LSTM with multi-input model
8. Learned to handle multi-class distribution and multi-label text classification
9. Learned to build API and GUI-based chatbot application

11. REFERENCES

1. **Chatbot Definition:** https://searchcustomerexperience.techtarget.com/definition/chatbot
2. **Types of Chatbot:** https://www.mygreatlearning.com/blog/basics-of-building-an-artificial-intelligence-chatbot/#types of chatbot
3. **Exploratory Data Analysis:** https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15
4. **Transformer Model:** https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)
5. **Holoviews Plot Tips:** http://holoviews.org/user_guide/Customizing_Plots.html
6. **NLP Aug:** https://github.com/makcedward/nlpaug
7. **Simple Transformers:** https://simpletransformers.ai/docs/multi-class-classification/