INDUSTRIAL SAFETY NLP BASED CHATBOT

FINAL REPORT GROUP 4

Team mentor:

Sumit Kumar

Team members:

1. Soumita Chowdhury
2. Latha M.S.
3. Devina Parmar

Table of Contents

Project Details:

In this dataset, the information about accidents in 12 manufacturing plants in 3 countries are given. We need to use this dataset to understand why accidents occur, and discover clues to reduce tragedy and accidents.

Import Dataset

```
Index(['Unnamed: 0', 'Data', 'Countries', 'Local', 'Industry Sector',
       'Accident Level', 'Potential Accident Level', 'Genre',
       'Employee or Third Party', 'Critical Risk', 'Description'],
      dtype='object')
```

Dataset columns are below:

- **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 means 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)
- **Genre** : if the person is male of 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

The dataset is in csv format. Basic exploration of the data is as below

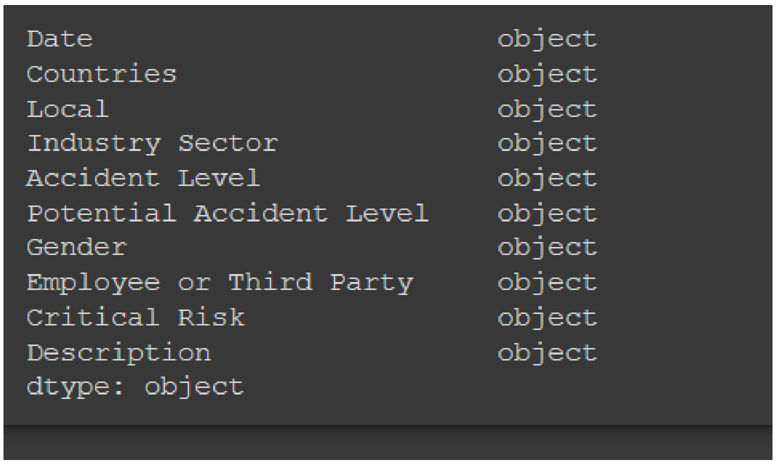
Shape of the dataset

```
(425, 10)
```

Rename columns
The field Unnamed: 0", is dropped and columns are renamed following

- ‘Data':'Date',
- 'Genre':'Gender'
- 'Employee or Third Party':'Employee Type'

Display datatypes

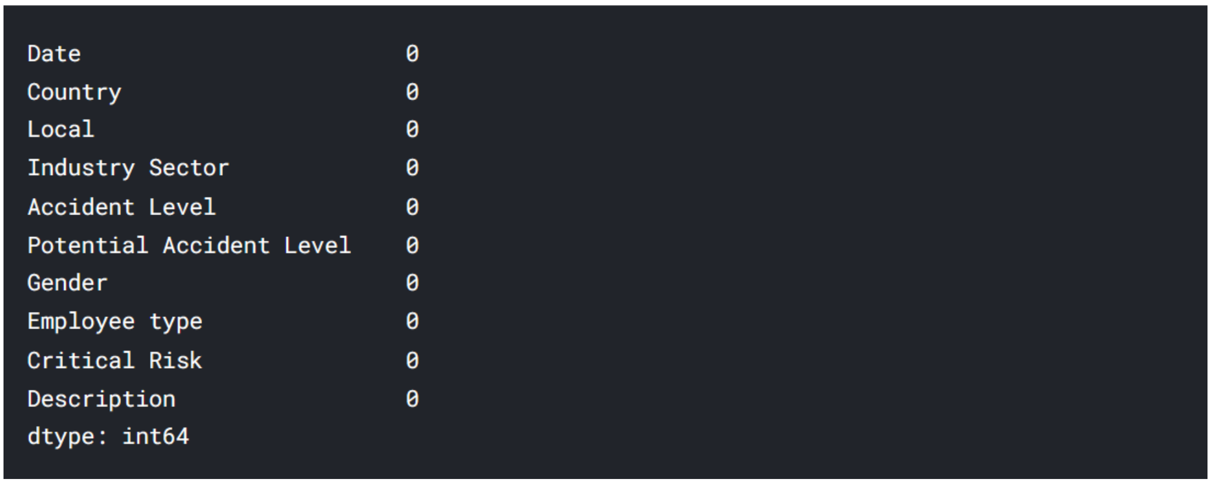


```
Date                          object
Countries                     object
Local                         object
Industry Sector               object
Accident Level                object
Potential Accident Level      object
Gender                        object
Employee or Third Party       object
Critical Risk                 object
Description                   object
dtype: object
```

Here, we can see that all the columns of the dataset are of "object" datatype. Coming to the type of data present in each column, we can see that there is a column "Date", which means it holds time series data. All other columns except "Description" are of categorical datatype.

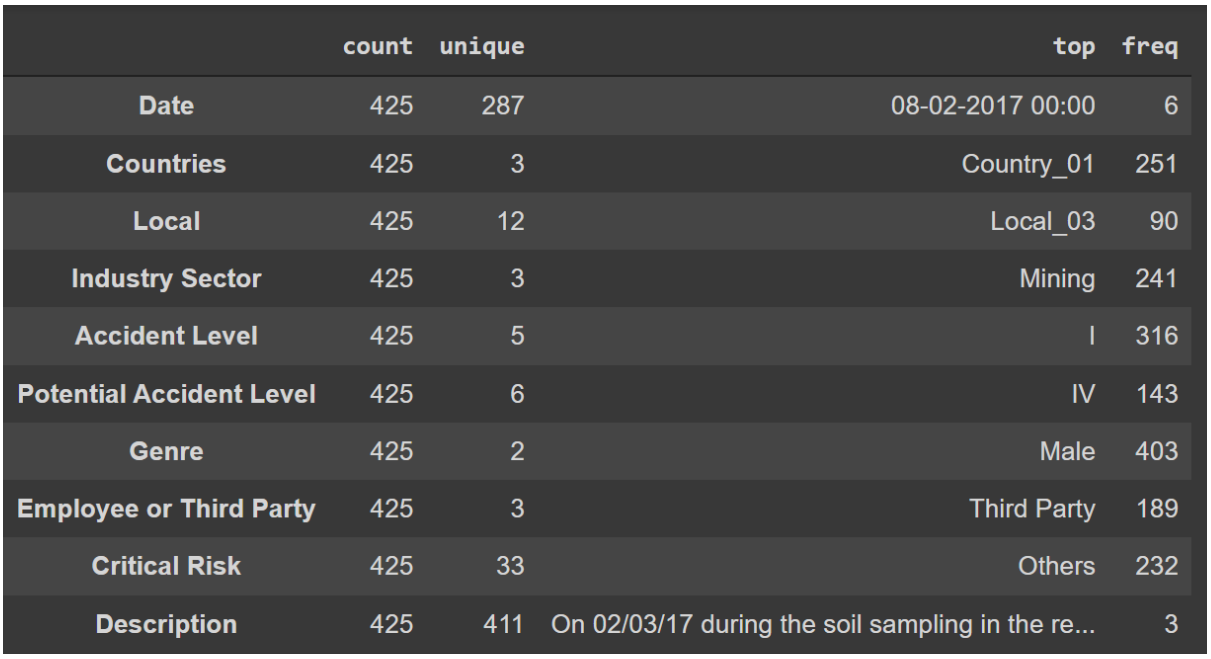
Check for null values

There are no null values present



```
Date                        0
Country                     0
Local                       0
Industry Sector             0
Accident Level              0
Potential Accident Level    0
Gender                      0
Employee type               0
Critical Risk               0
Description                 0
dtype: int64
```

Dataset Description



	count	unique	top	freq
Date	425	287	08-02-2017 00:00	6
Countries	425	3	Country_01	251
Local	425	12	Local_03	90
Industry Sector	425	3	Mining	241
Accident Level	425	5	I	316
Potential Accident Level	425	6	IV	143
Genre	425	2	Male	403
Employee or Third Party	425	3	Third Party	189
Critical Risk	425	33	Others	232
Description	425	411	On 02/03/17 during the soil sampling in the re...	3

From the above table, we can infer the below:

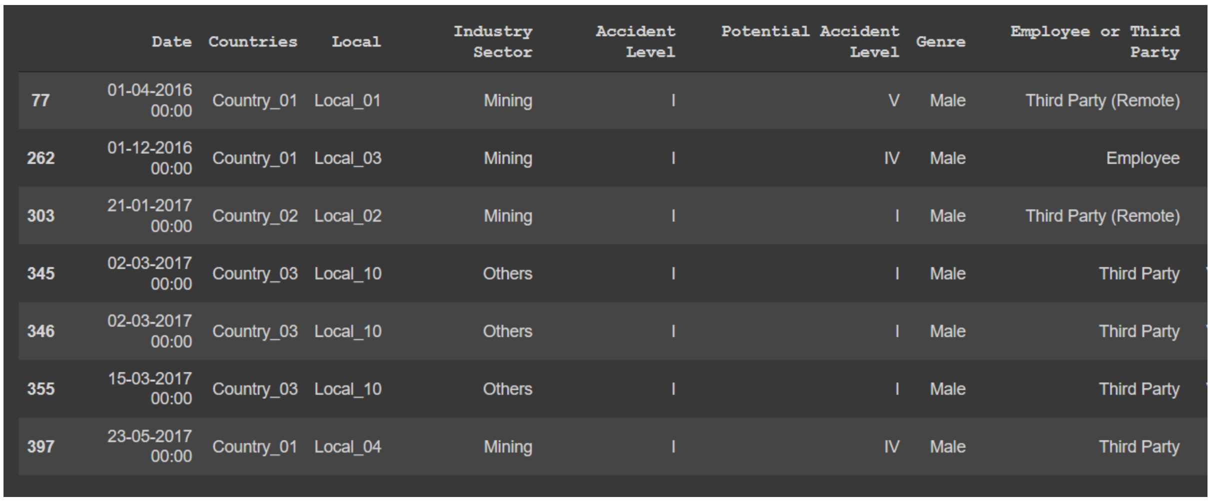
1. This dataset contains accident data of 3 countries, out of which Country1 has the most number of accidents.
2. The data is collected from 3 types of industry sectors.Local_3 has the most number of accidents.
3. There are 5 major accident levels in which this dataset has been classified.316 accidents are of accident level 1, making it the most frequent accident type. This also means that the data is not distributed evenly.
4. The data is a consolidation of accidents faced by employees as well as third party vendors and others. Third party employees have faced the most number of accidents according to this dataset.
5. 403 male employees have been reported to have accidents, which mean the distribution of data in this case is also not evenly balanced.
6. 33 different types of critical risks have been identified in the dataset.

The Categorical Variables that can be encoded to Numerical Values from the dataset

1. Local
2. Accident Level
3. Potential Accident Level

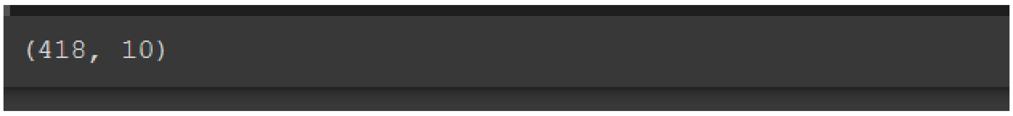
Duplicate Records in the Dataset

There are 7 duplicates records found in the dataset



	Date	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party
77	01-04-2016 00:00	Country_01	Local_01	Mining	I	V	Male	Third Party (Remote)
262	01-12-2016 00:00	Country_01	Local_03	Mining	I	IV	Male	Employee
303	21-01-2017 00:00	Country_02	Local_02	Mining	I	I	Male	Third Party (Remote)
345	02-03-2017 00:00	Country_03	Local_10	Others	I	I	Male	Third Party
346	02-03-2017 00:00	Country_03	Local_10	Others	I	I	Male	Third Party
355	15-03-2017 00:00	Country_03	Local_10	Others	I	I	Male	Third Party
397	23-05-2017 00:00	Country_01	Local_04	Mining	I	IV	Male	Third Party

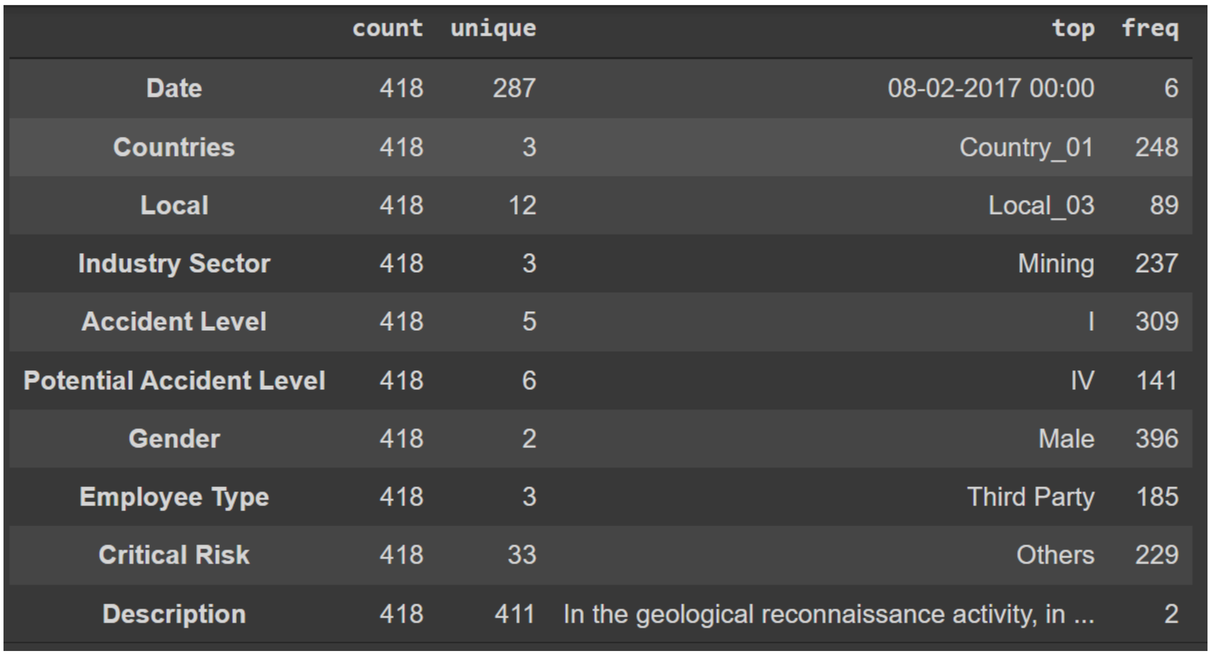
Shape of the dataset after duplicates are removed



```
(418, 10)
```

The dataset has 418 rows and 10 columns

Dataset description after duplicates removal and renaming



	count	unique	top	freq
Date	418	287	08-02-2017 00:00	6
Countries	418	3	Country_01	248
Local	418	12	Local_03	89
Industry Sector	418	3	Mining	237
Accident Level	418	5	I	309
Potential Accident Level	418	6	IV	141
Gender	418	2	Male	396
Employee Type	418	3	Third Party	185
Critical Risk	418	33	Others	229
Description	418	411	In the geological reconnaissance activity, in ...	2

From the above table, we can infer the below:

1. This dataset contains accident data of 3 countries, out of which Country1 has the most number of accidents.

2. The data is collected from 3 types of industry sectors. Local_3 has the most number of accidents.

3. There are 5 major accident levels in which this dataset has been classified.316 accidents are of accident level 1, making it the most frequent accident type. This also means that the data is not distributed evenly.

4. The data is a consolidation of accidents faced by employees as well as third party vendors and others. Third party employees have faced the most number of accidents according to this dataset.

5. 396 male employees have been reported to have accidents, which mean the distribution of data in this case is also not evenly balanced.

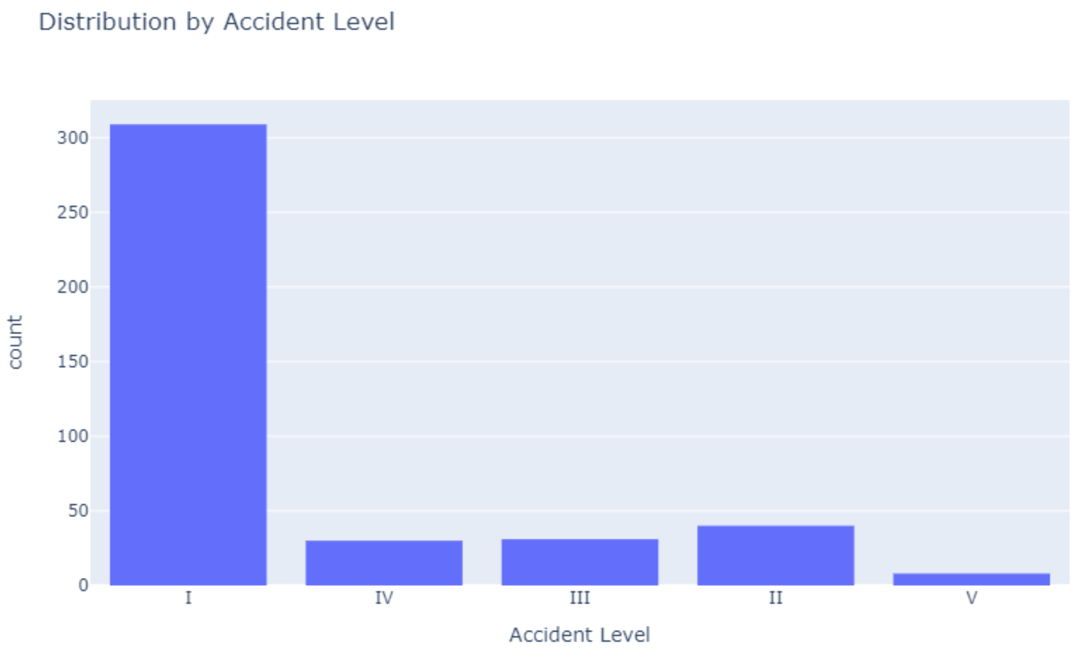
6. 33 different types of critical risks have been identified in the dataset.

The Categorical Variables that can be encoded to Numerical Values from the dataset

1. Local
2. Accident Level
3. Potential Accident Level

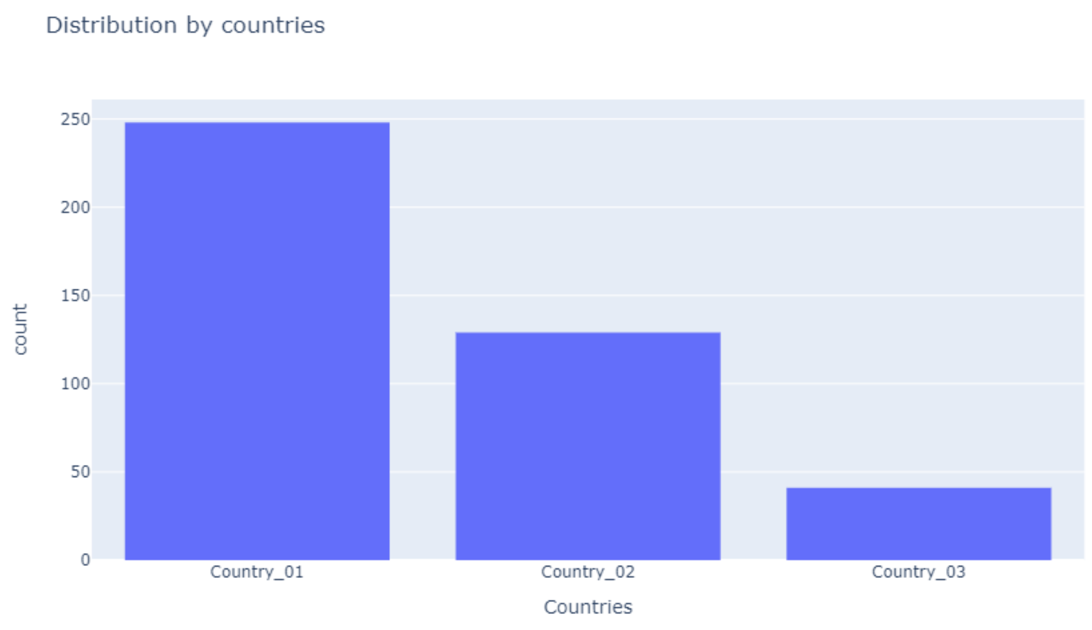
Univariate Analysis:

Checking the distribution of data based on accident levels



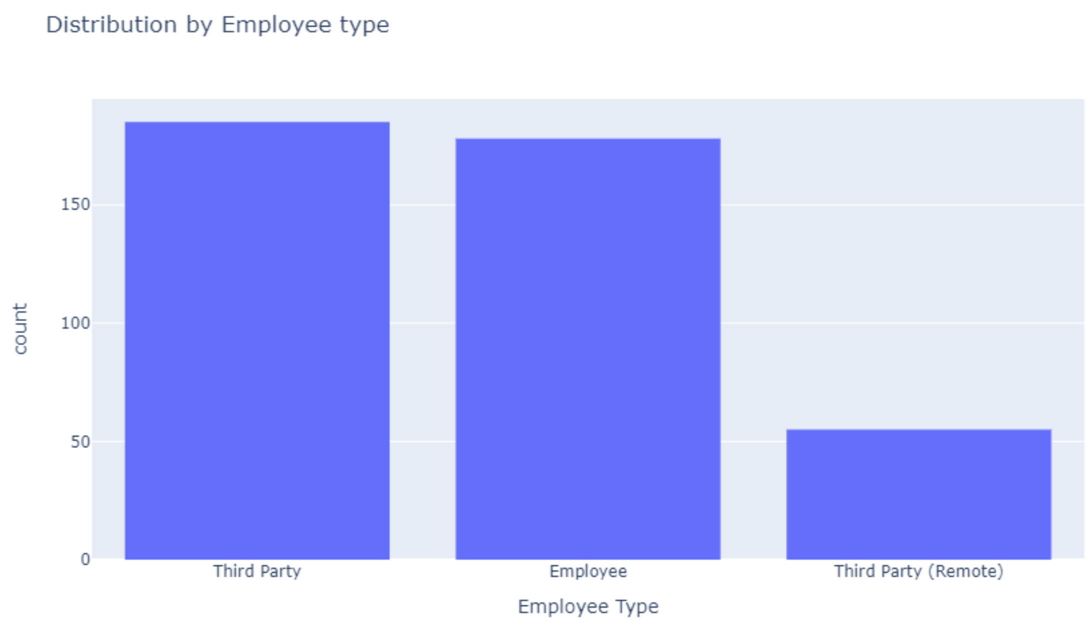
The distribution of Accident Levels is highly imbalanced in the dataset

Distribution of the data based on country wise



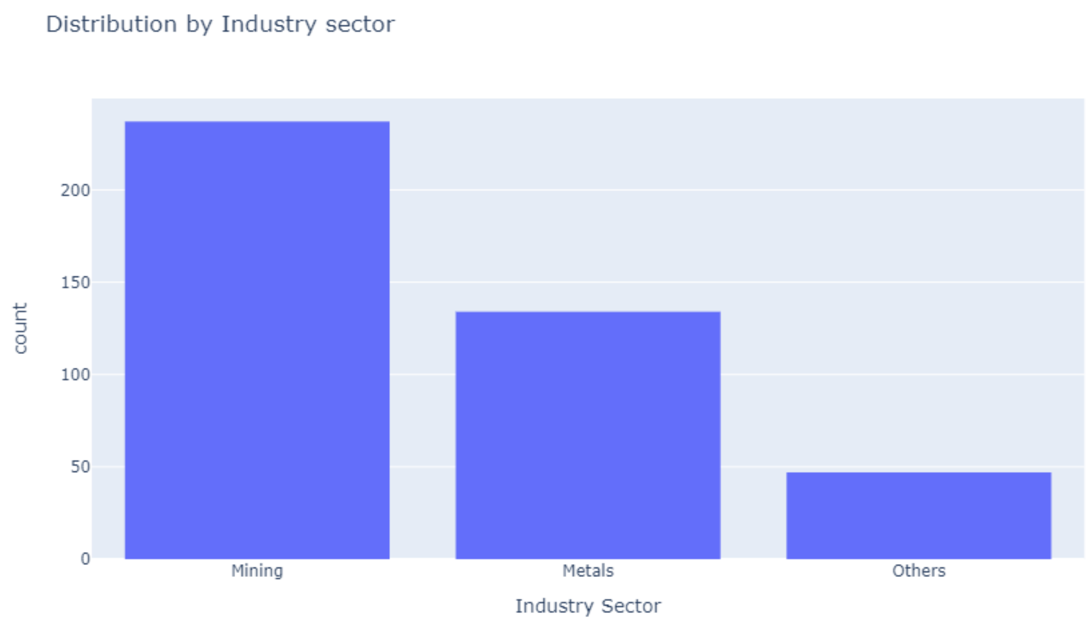
"Country_01" has the most number of accident cases

Distribution of accidents by Employee Types:



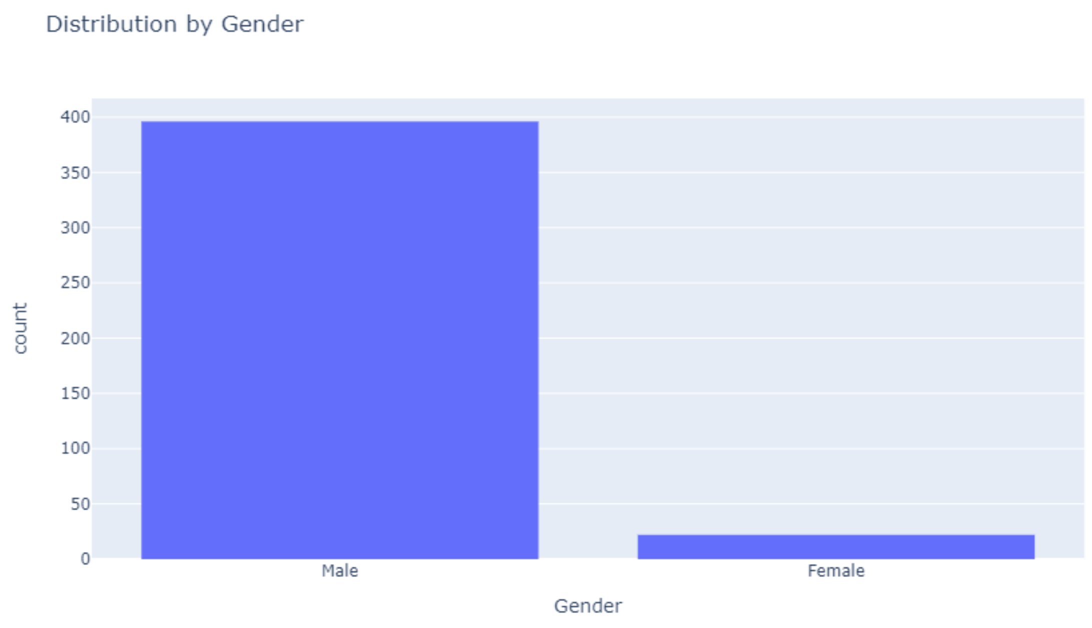
From the graph it is very clear that accidents have happened in almost equal proportions among permanent employees or third-party contractors, with thrid party contractors a bit on the higher side.

Distribution of accidents as per industry sector.



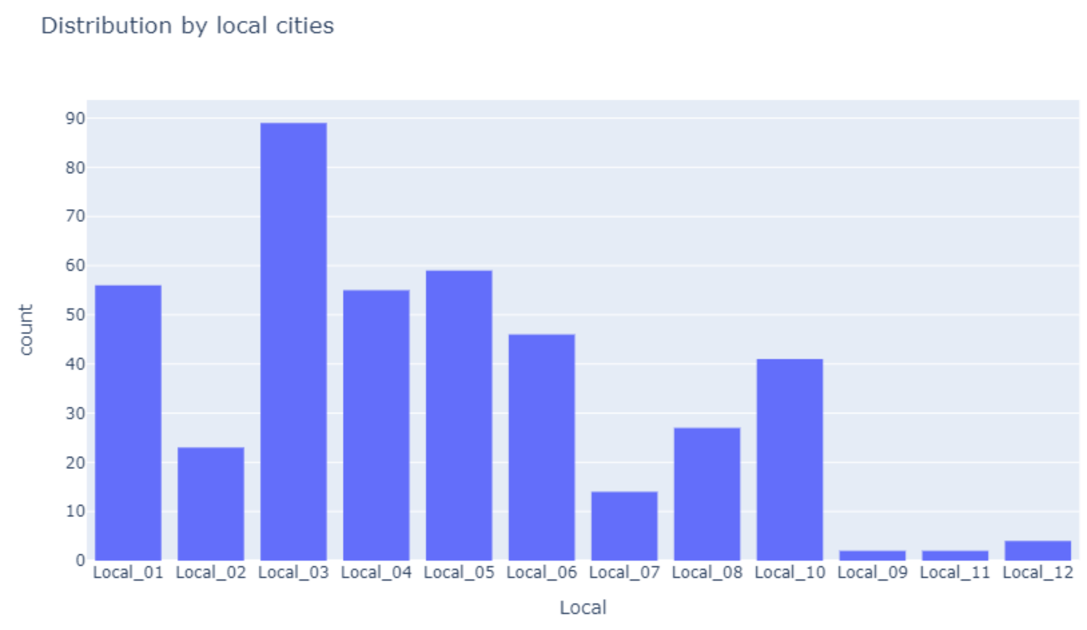
Majority of the accidents have happened in the mining sector, followed by metal industry and other type of industries.

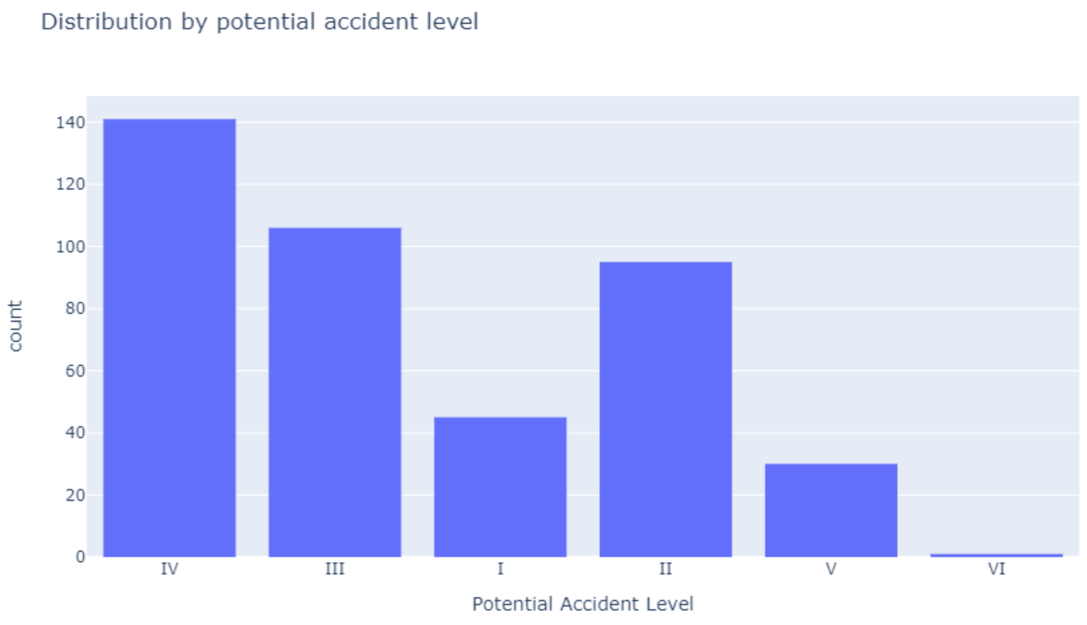
Distribution of accidents as per Gender



The distribution of accidents is imbalanced when checked by "Genre". The count of accidents in males is way higher than that in females.

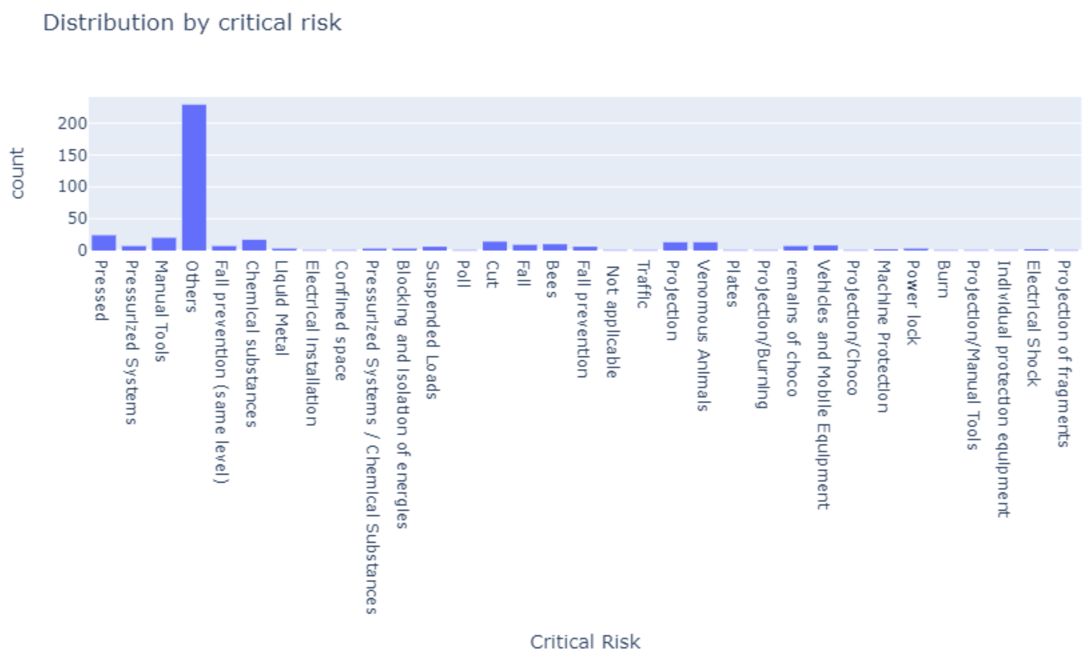
Distribution by Locals





Observation from the above graph

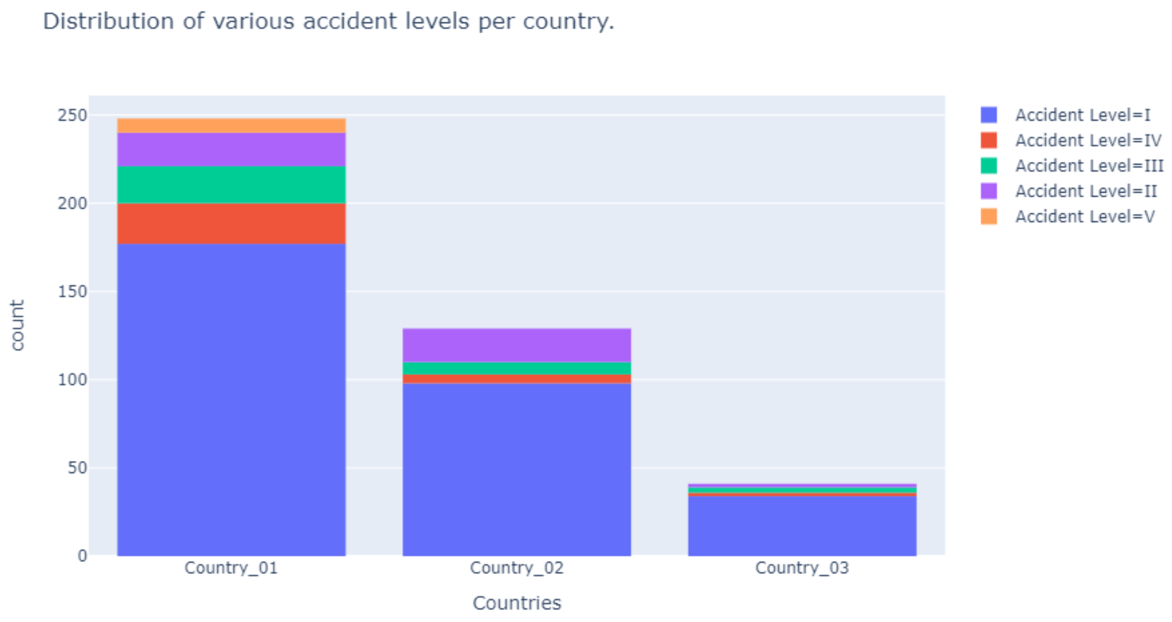
1. Local_03 has the highest number of accident cases.
2. Potential accident Level IV has the highest count of accidents



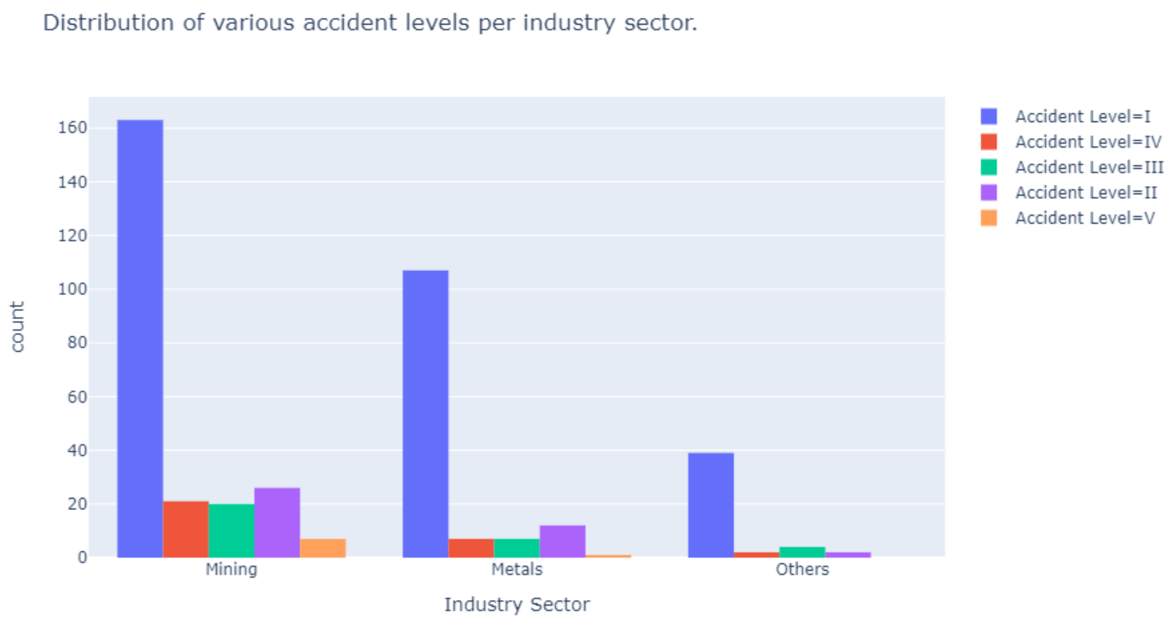
We can see from the graph that the Critical risk category "Others" have the most number of accidents. This means we are not clear about the exact risk factor associated with accidents in this dataset.

Bivariate Analysis

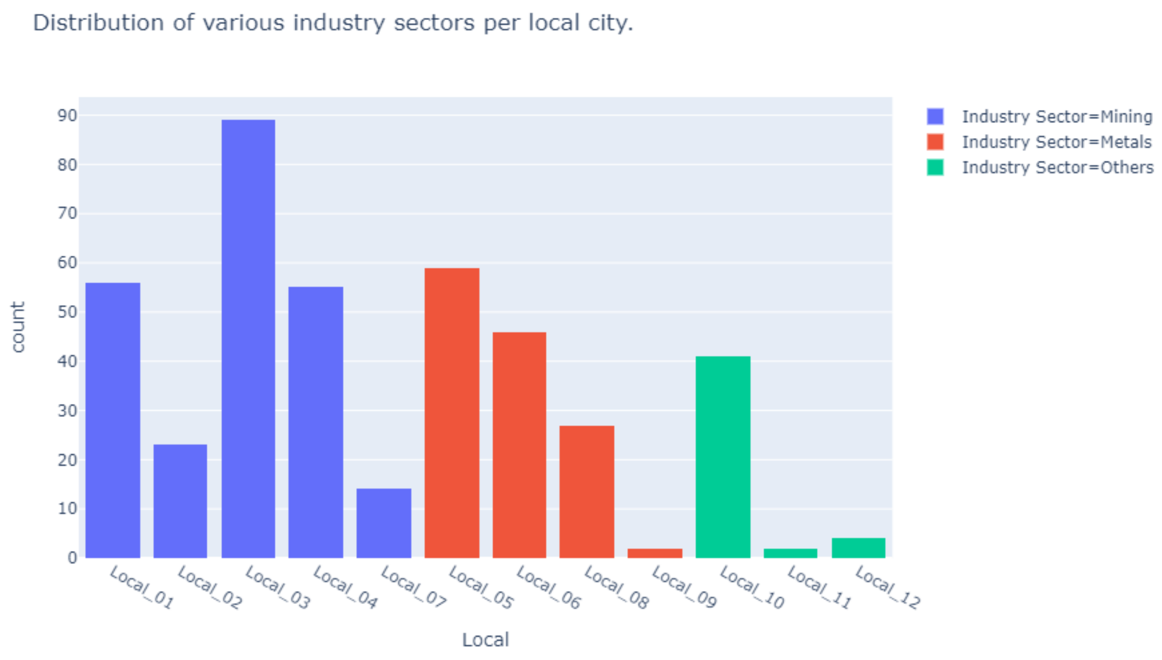
Distribution of different accident levels occurred per country

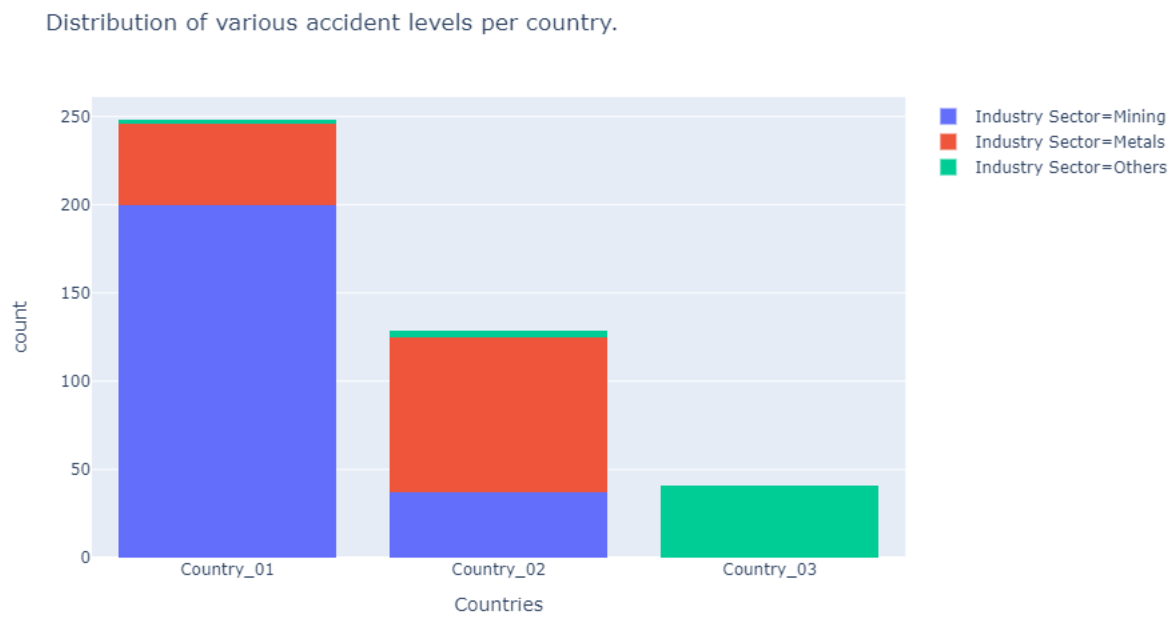


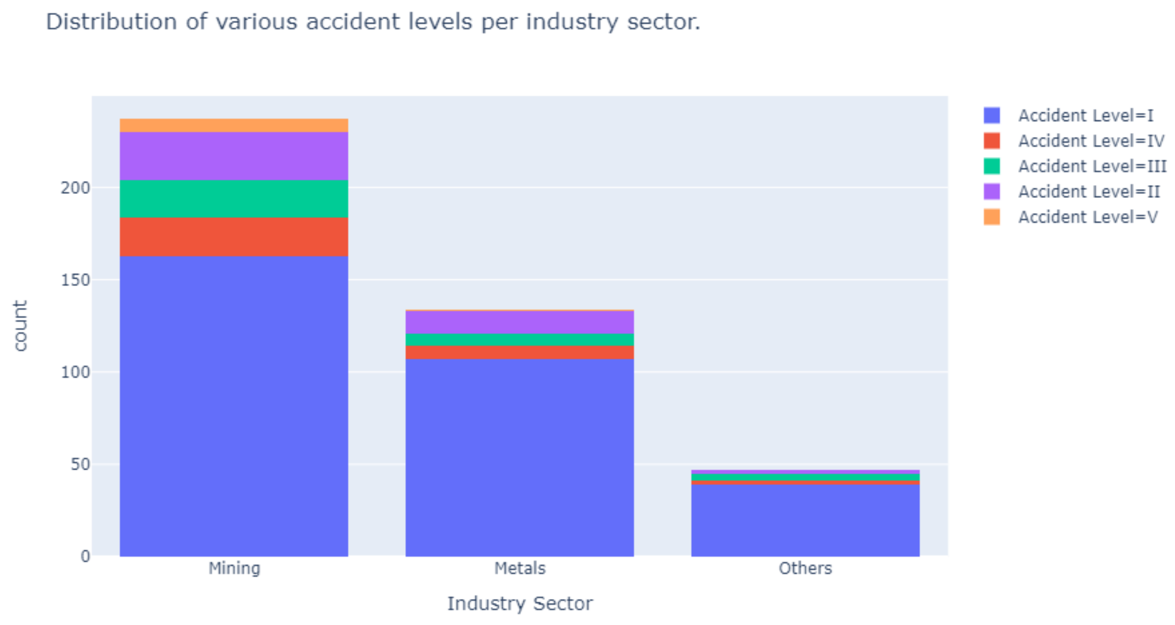
1. Majority of the accident Level V accidents has occured only in Country I.
2. Maximum number of accidents in all countries are mainly of type Accident Level I.
3. Country_01 has had accidents of all Accident types, making it the riskiest place as per the dataset.



The greatest number of accidents have occurred in the Mining Industry in Country 1 so far, followed by the metal industry, also in Country 1.

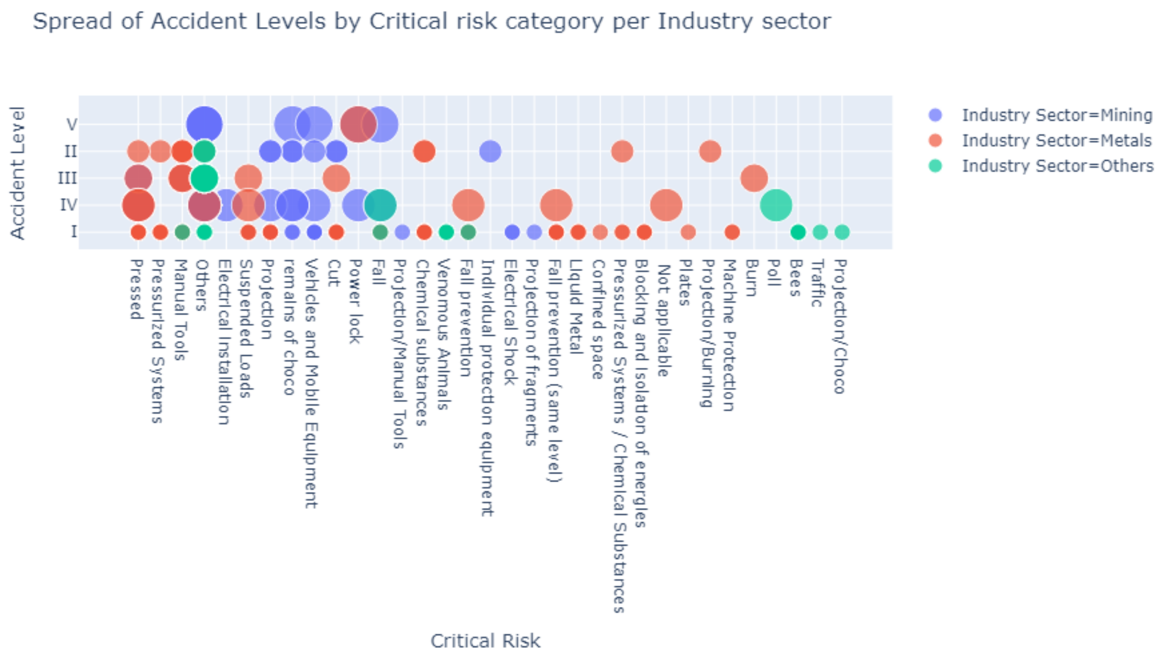






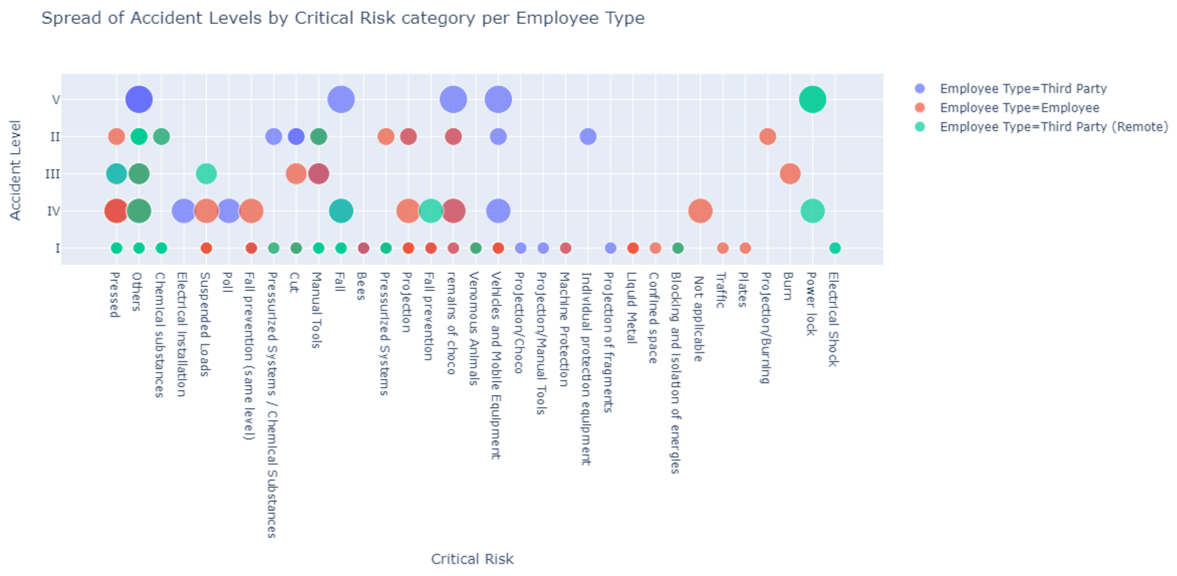
Observations from the above graphs:

1. Local 01,Local 02,Local 03,Local 04,Local 07 all have plants belonging to the Mining Sector and they have had the most number of accidents.
2. Other industry sectors have had the least number of accidents.
3. Local 09 and Local_11 seems to be the safest cities, with only 2 accidents, even though it has plants belonging to the Metal sector.



Observations from the above graph

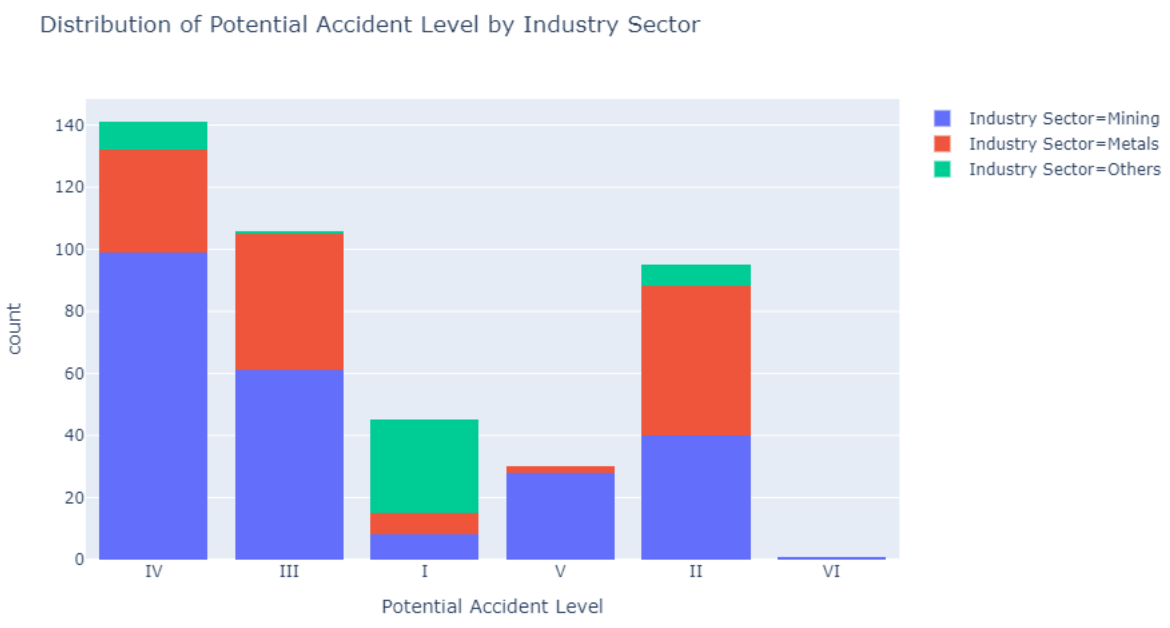
1. There are numerous risks involved in the Metals sector, followed by the ones in the Mining sector.
2. Comparitively very low risks are there in the "Other" industry sector.



Observations from the above graph:

1. Mostly third party contractors(both on site and remote) have had accidents of notably all Accident Levels in the "Others" risk category.
2. "Pressed" risks are the second most dangerous ones where employees and contractors both have had accidents.
3. Here it is clearly visible that in the mining industry, third party employees have met with the maximum number of accidents as compared to the metal industry where their

employees have met with the highest number of accidents.



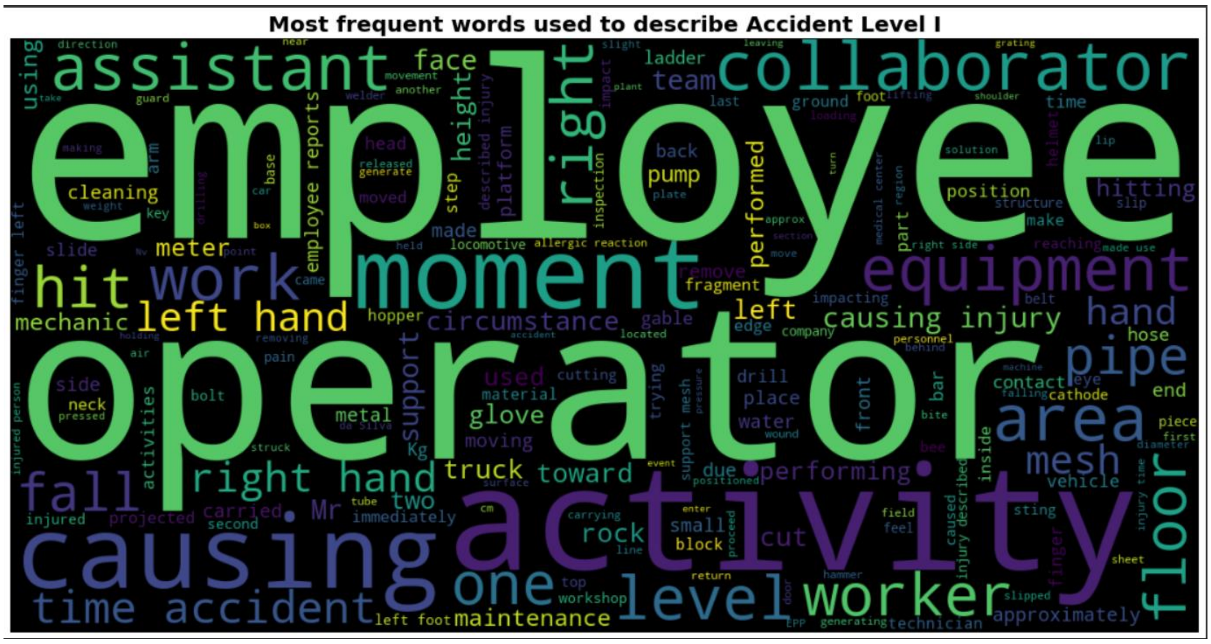
Observations from the above graph:

1. Major number of accidents have occured in the Potential Accident Level 3 category.
2. Potential Accident Level 5 is least in the mining industry.

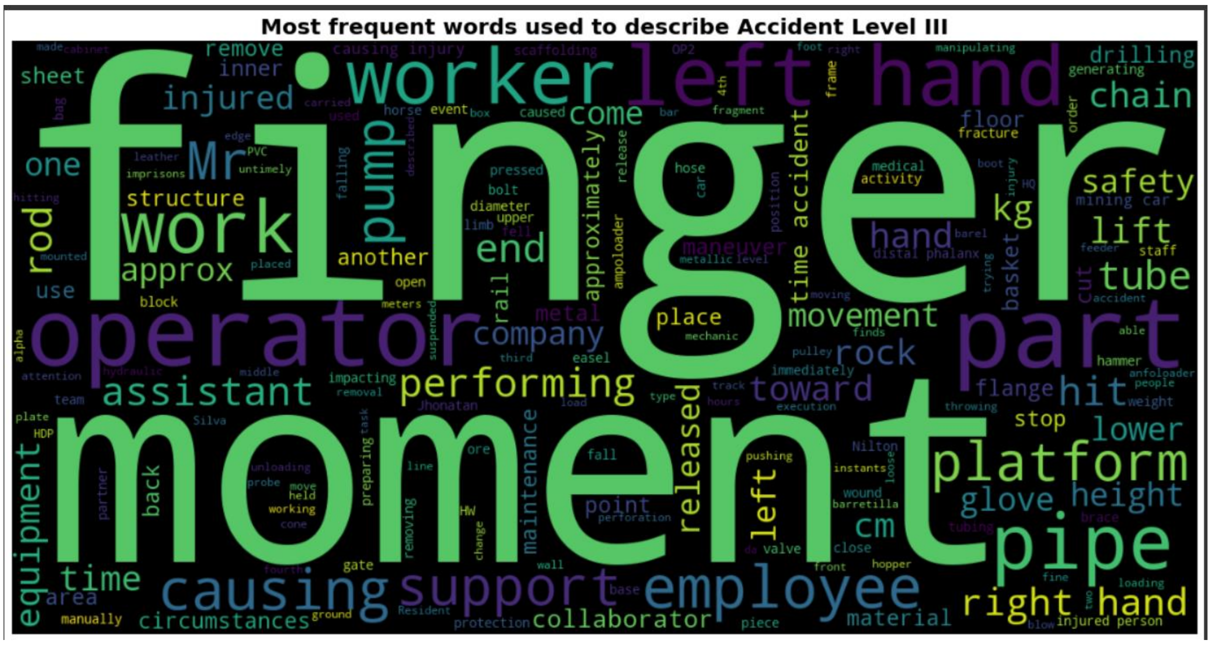
NLP Analysis:

The most frequent words used for each accident level.

1. **Accident Level 1**

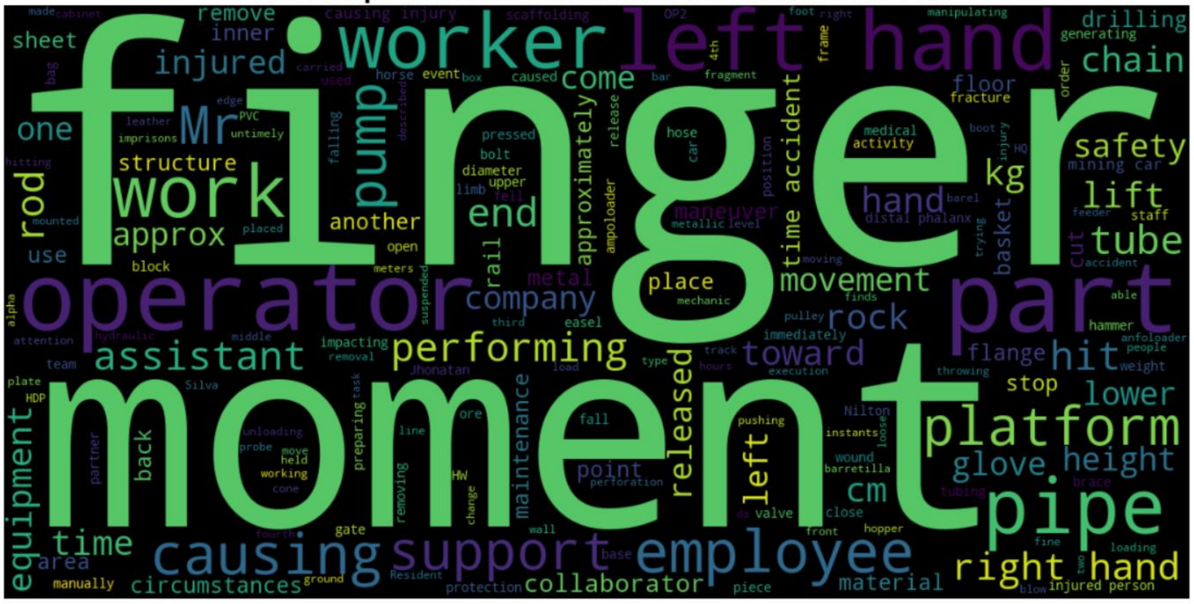


2. **Accident Level II**



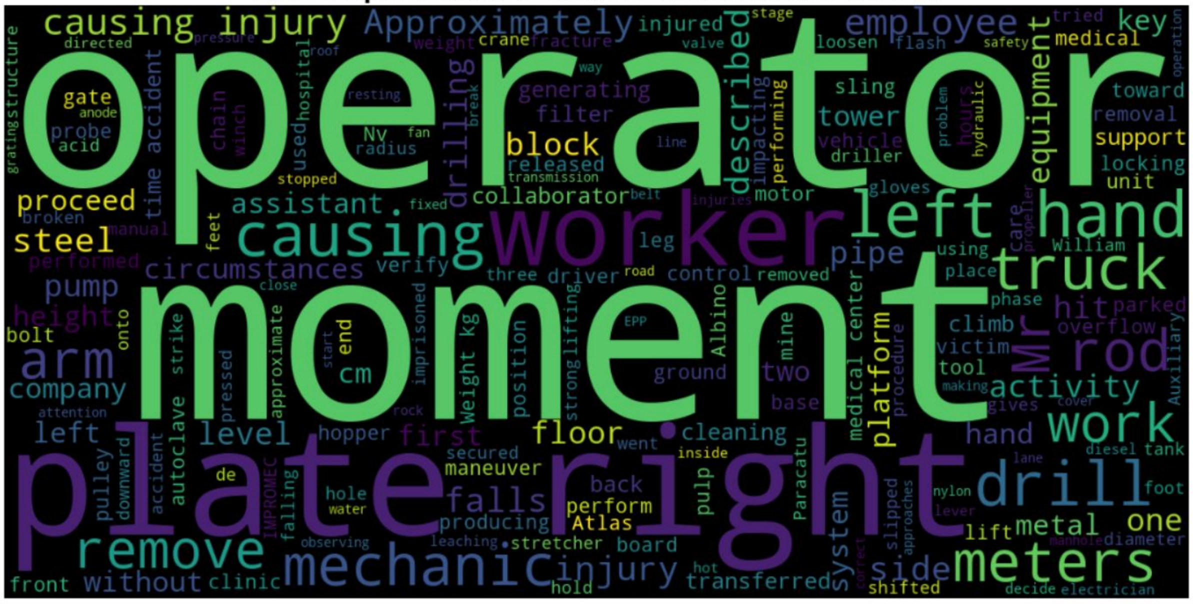
3. **Accident Level III**

Most frequent words used to describe Accident Level III

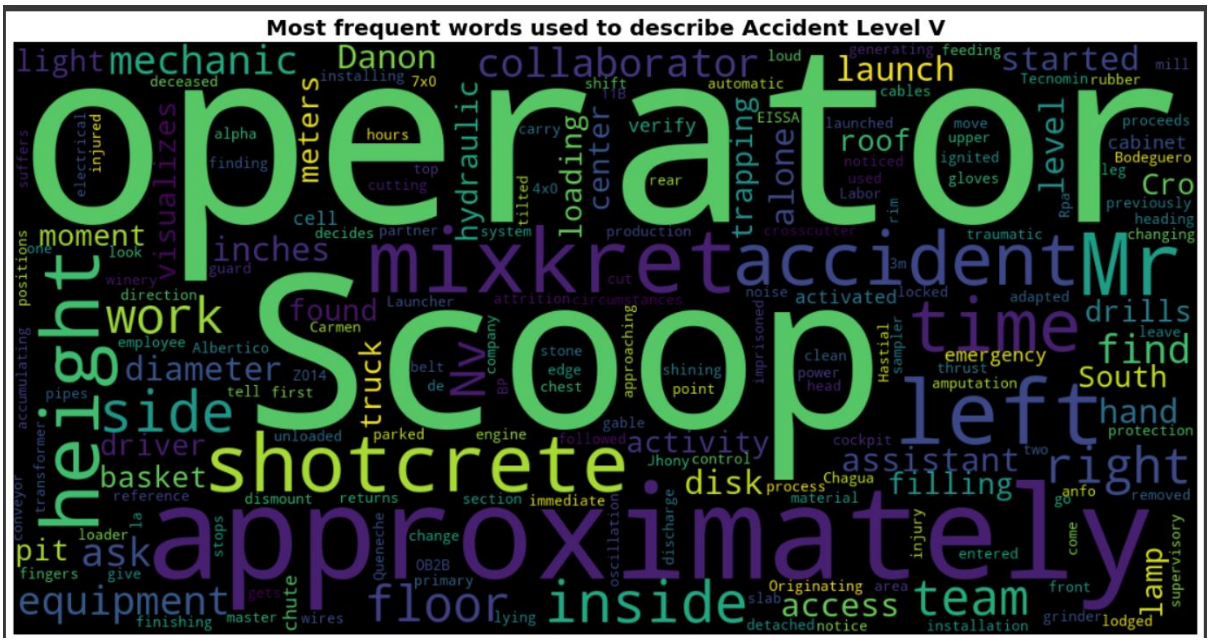


4. Accident Level IV

Most frequent words used to describe Accident Level IV

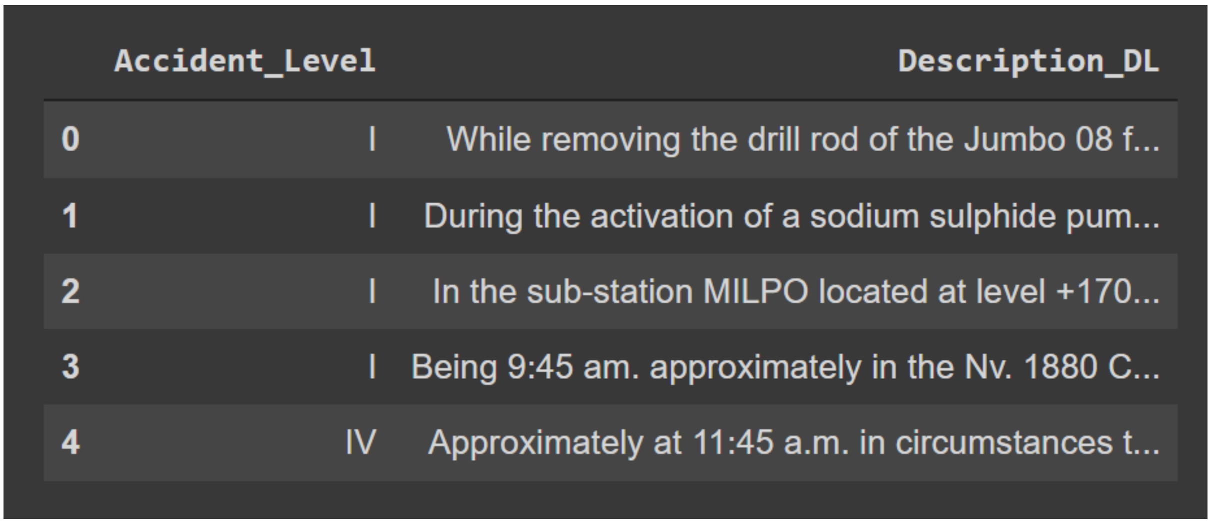


5. **Accident Level V**

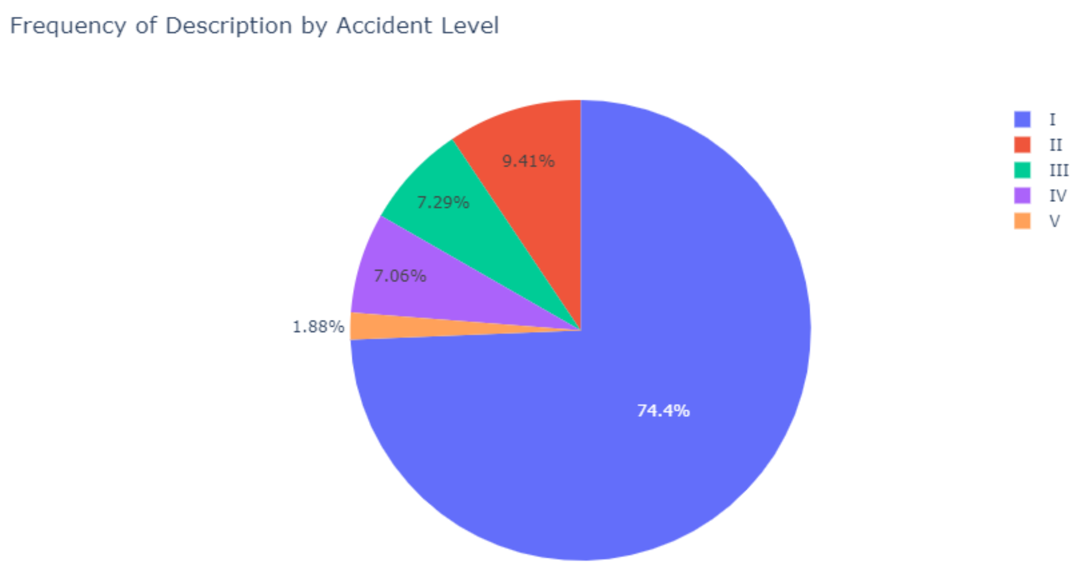


Data Augmentation

A dataset is created using only the class variable "Accident Level" and Description column.

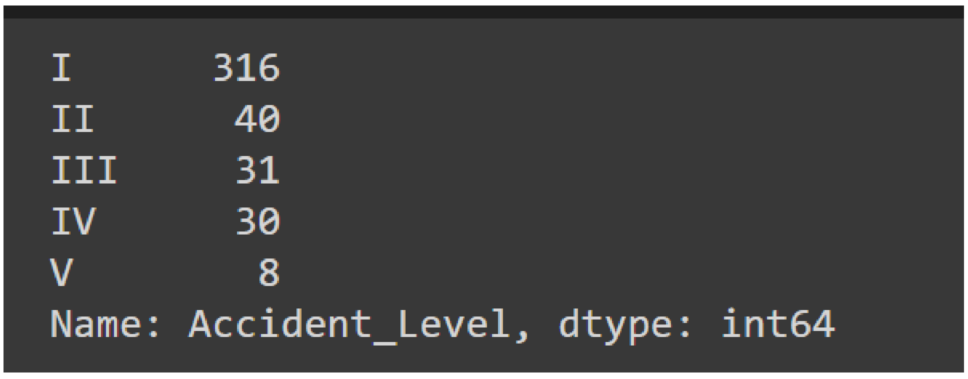


	Accident_Level	Description_DL
0	I	While removing the drill rod of the Jumbo 08 f...
1	I	During the activation of a sodium sulphide pum...
2	I	In the sub-station MILPO located at level +170...
3	I	Being 9:45 am. approximately in the Nv. 1880 C...
4	IV	Approximately at 11:45 a.m. in circumstances t...



Description column is imbalanced in the dataset. Most of the description is present only for Accident Level I(0)

Checking the exact counts of Descriptions per Accident level.



```
I       316
II       40
III      31
IV       30
V         8
Name: Accident_Level, dtype: int64
```

Trying different data augmentation techniques so that the data is balanced properly before it is passed into the dataset.

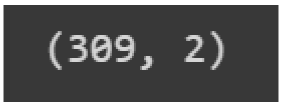
Simple up-sampling

Using EDA let us perform data augmentation

Data Augmentation for the accident **level 'II'**

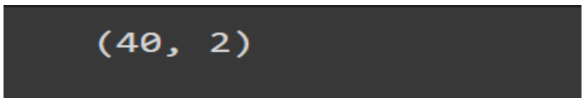
Let us divide data of each Accident Level in different data frames

df_0 of accident level I



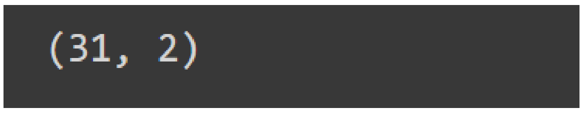
```
(309, 2)
```

df_1 of accident level **'II'’**



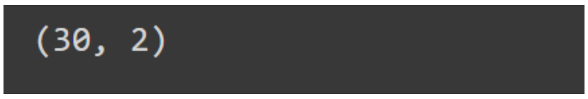
```
(40, 2)
```

df_2 of accident level **'III'**



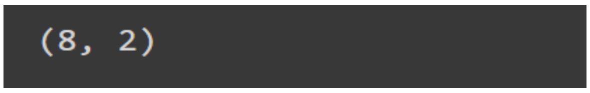
```
(31, 2)
```

df_3 of accident level **'IV**



```
(30, 2)
```

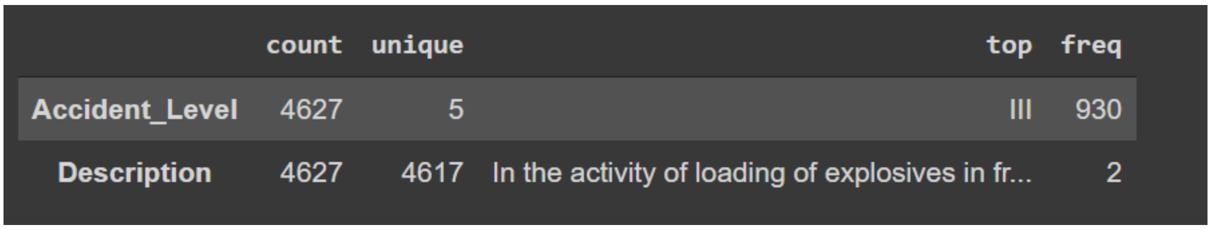
df_4 of accident level **'V**



```
(8, 2)
```

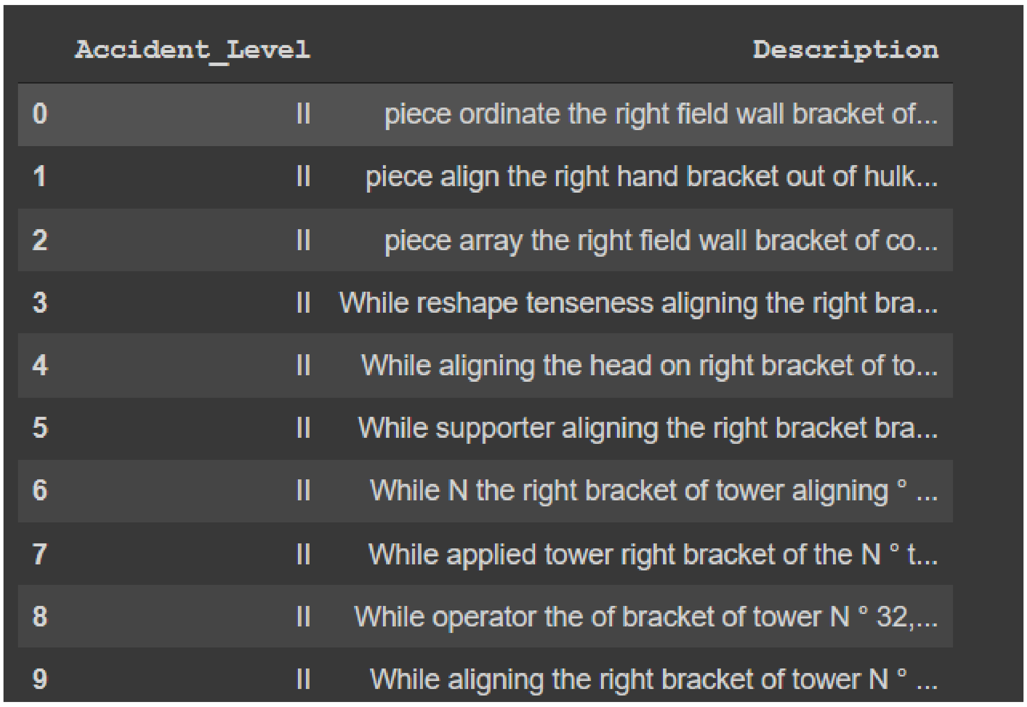
Now, we will augment each dataset separately. Here the gen_eda function from data_augmentation.py takes in the below parameters:

dataset - dataframe name alpha_sr - percentage of words in the dataset we want to replace with synonyms.
alpha_ri - percentage of words in the dataset we want to randomly insert.
alpha_rs - percentage of words in the dataset we want to randomly swap.
alpha_rd - percentage of words in the dataset we want to randomly delete.
num_aug - total number of augmented sentences we want per sentence in the dataset.
Accident_safety_data_upsampled is as below after concatenating



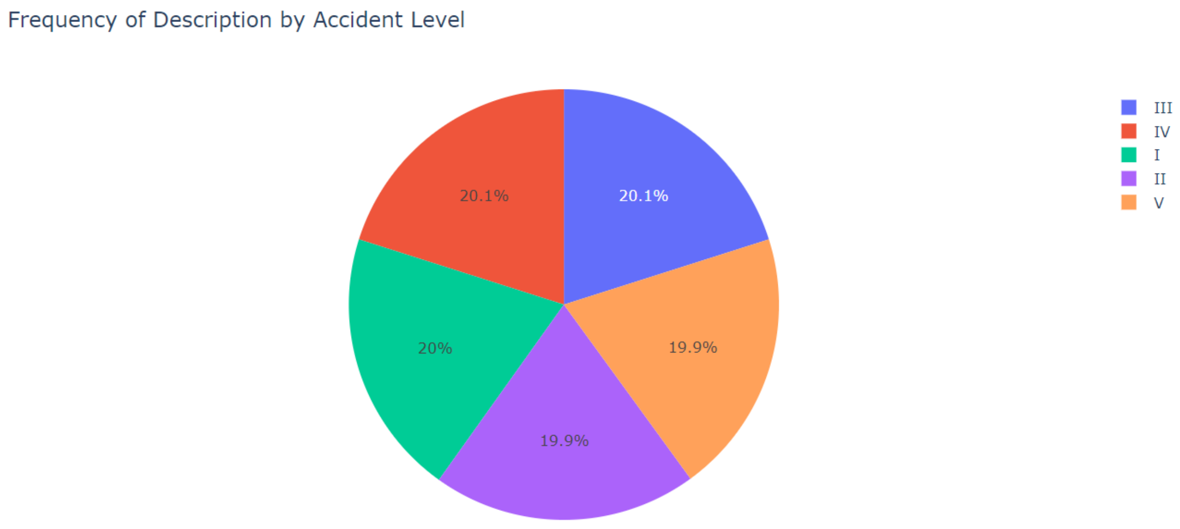
	count	unique	top	freq
Accident_Level	4627	5	III	930
Description	4627	4617	In the activity of loading of explosives in fr...	2

Up sampled data is as below

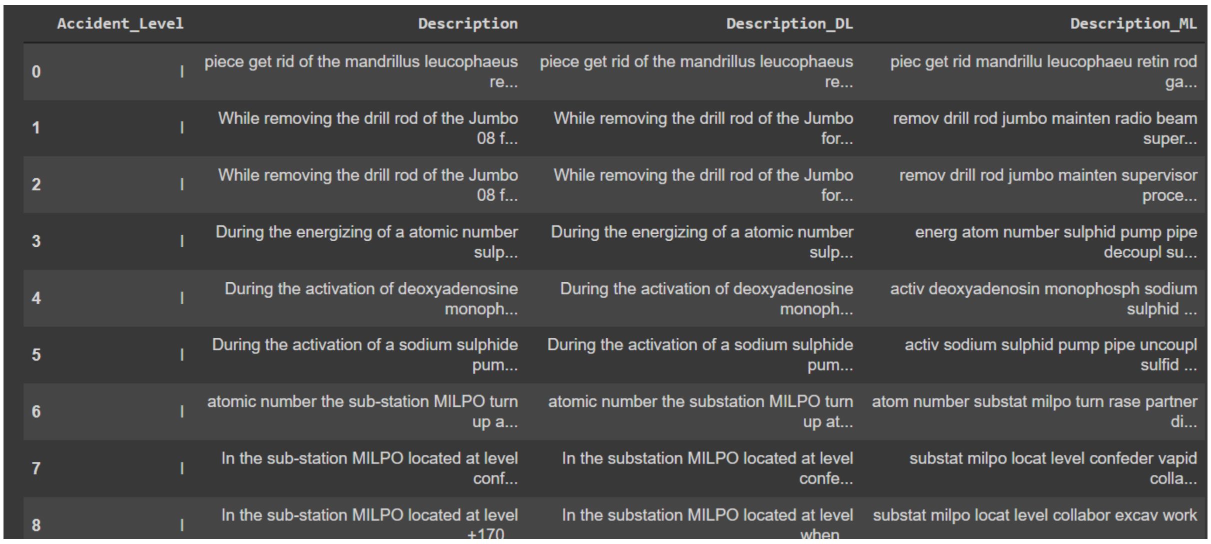


	Accident_Level	Description
0	II	piece ordinate the right field wall bracket of...
1	II	piece align the right hand bracket out of hulk...
2	II	piece array the right field wall bracket of co...
3	II	While reshape tenseness aligning the right bra...
4	II	While aligning the head on right bracket of to...
5	II	While supporter aligning the right bracket bra...
6	II	While N the right bracket of tower aligning ° ...
7	II	While applied tower right bracket of the N ° t...
8	II	While operator the of bracket of tower N ° 32,...
9	II	While aligning the right bracket of tower N ° ...

Frequency of distribution – Accident Level after Up Sampling



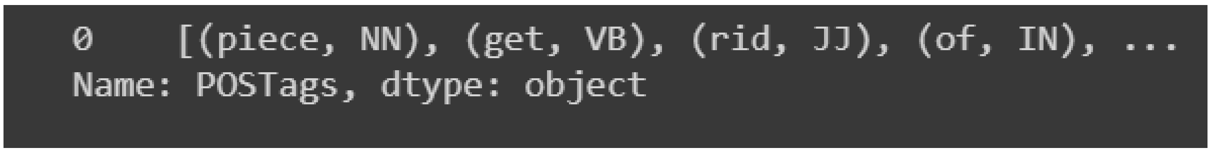
Cleaned Up Sampled Data:



	Accident_Level	Description	Description_DL	Description_ML
0	I	piece get rid of the mandrillus leucophaeus re...	piece get rid of the mandrillus leucophaeus re...	piec get rid mandrillu leucophaeu retin rod ga...
1	I	While removing the drill rod of the Jumbo 08 f...	While removing the drill rod of the Jumbo for...	remov drill rod jumbo mainten radio beam super...
2	I	While removing the drill rod of the Jumbo 08 f...	While removing the drill rod of the Jumbo for...	remov drill rod jumbo mainten supervisor proce...
3	I	During the energizing of a atomic number sulp...	During the energizing of a atomic number sulp...	energ atom number sulphid pump pipe decoupl su...
4	I	During the activation of deoxyadenosine monoph...	During the activation of deoxyadenosine monoph...	activ deoxyadenosin monophosph sodium sulphid ...
5	I	During the activation of a sodium sulphide pum...	During the activation of a sodium sulphide pum...	activ sodium sulphid pump pipe uncoupl sulfid ...
6	I	atomic number the sub-station MILPO turn up a...	atomic number the substation MILPO turn up at...	atom number substat milpo turn rase partner di...
7	I	In the sub-station MILPO located at level conf...	In the substation MILPO located at level confe...	substat milpo locat level confeder vapid colla...
8	I	In the sub-station MILPO located at level +170	In the substation MILPO located at level when	substat milpo locat level collabor excav work

Named Entity Recognition:

POS Taggings:



```
0    [(piece, NN), (get, VB), (rid, JJ), (of, IN), ...
Name: POSTags, dtype: object
```

Feature extraction

We will try the below vectorizers

1. Count Vectorizer
2. TF IDF vectorizer
3. Glove

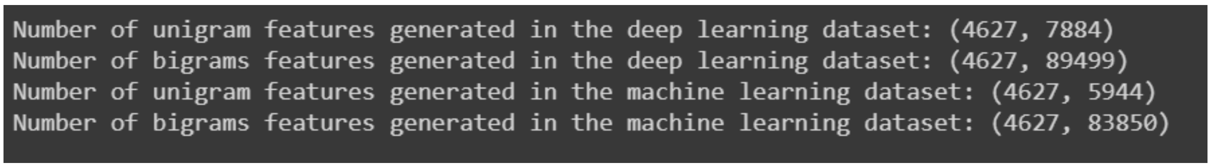
We will first work with data cleaned for machine learning and then data cleaned for deep learning

Using Count Vectorizer, Unigrams, Bigrams, Trigrams features are created

Shape of the features as shown below

Machine learning features

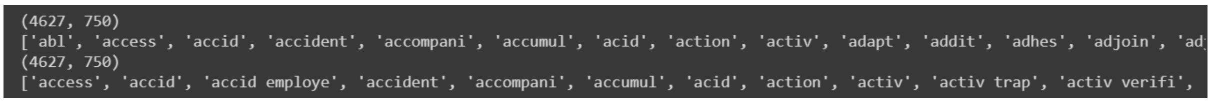
We have tried vectorizing the data with both count vectorizer and TF IDF vectorizers and found that TF_IDF with unigrams-bigrams give the best results.



```
Number of unigram features generated in the deep learning dataset: (4627, 7884)
Number of bigrams features generated in the deep learning dataset: (4627, 89499)
Number of unigram features generated in the machine learning dataset: (4627, 5944)
Number of bigrams features generated in the machine learning dataset: (4627, 83850)
```

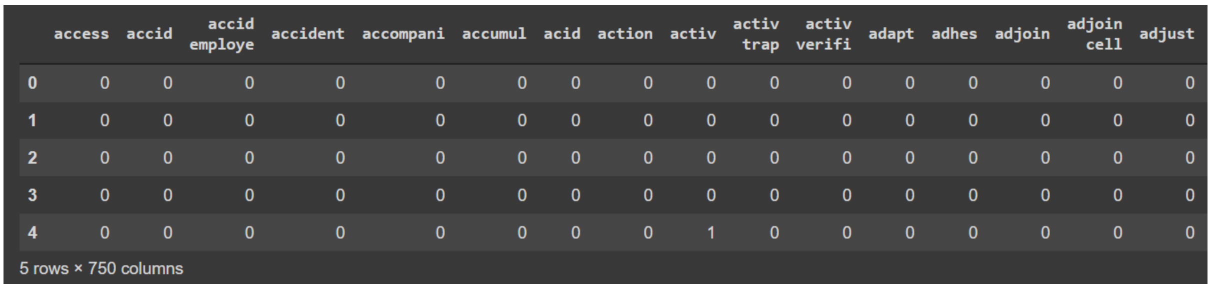
The number of features generated are very large in number hence reducing the number of features to 750

Machine learning features



```
(4627, 750)
['abl', 'access', 'accid', 'accident', 'accompani', 'accumul', 'acid', 'action', 'activ', 'adapt', 'addit', 'adhes', 'adjoin', 'ad
(4627, 750)
['access', 'accid', 'accid employe', 'accident', 'accompani', 'accumul', 'acid', 'action', 'activ', 'activ trap', 'activ verifi',
```

Vectorized data



	access	accid	accid employe	accident	accompani	accumul	acid	action	activ	activ trap	activ verifi	adapt	adhes	adjoin	adjoin cell	adjust
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

5 rows × 750 columns

We can now use this data to train our machine learning models.

Dataset to be used for deep learning

TF IDF Vectorized data

- Bigrams : x_DL_tfidf_2 , y_DL

Let us now input this data into machine learning models

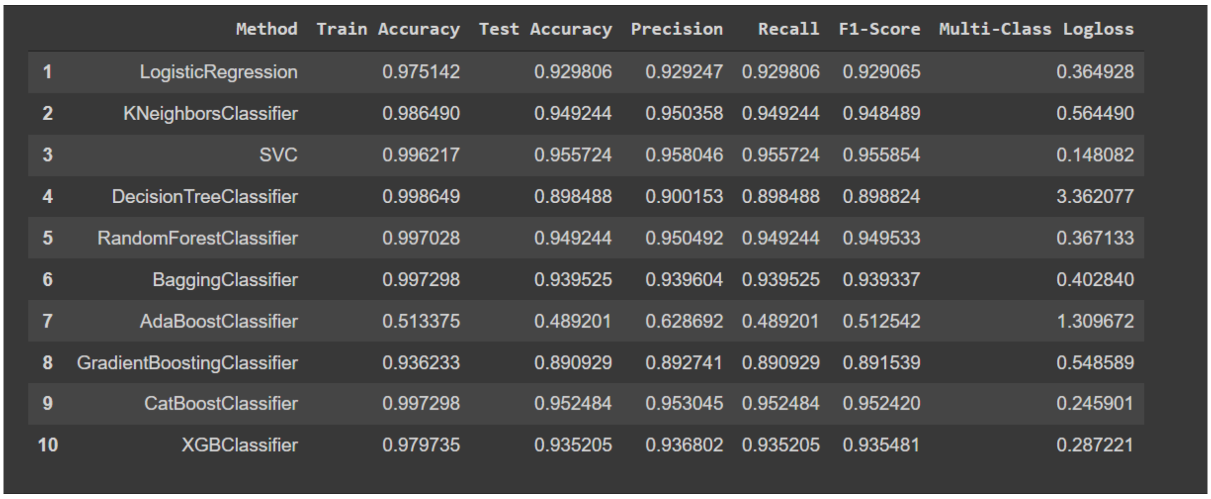
1. Split the data into 80 and 20
2. Using TF IDF vectorized data
3. Unigrams

Machine Learning Model Training:

Training data is split into 80 and 20 ratio. We have used stratify =True so that the classes are equally balanced while splitting.

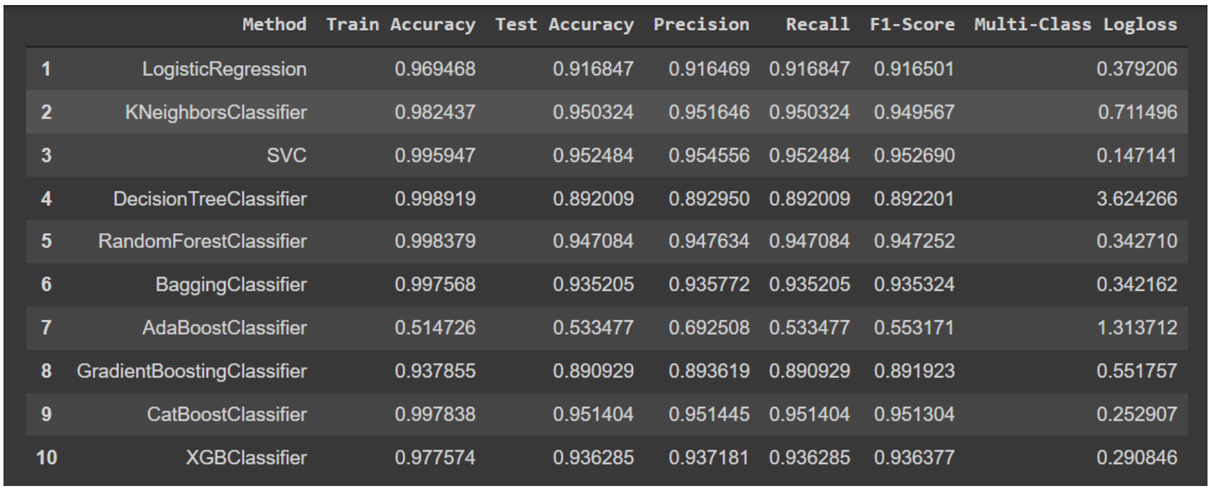
```
X_train, X_test, y_train, y_test = train_test_split(X,  Y, test_size = 
0.20, random_state = 1, stratify = y_ML)
```

1. **Unigrams**



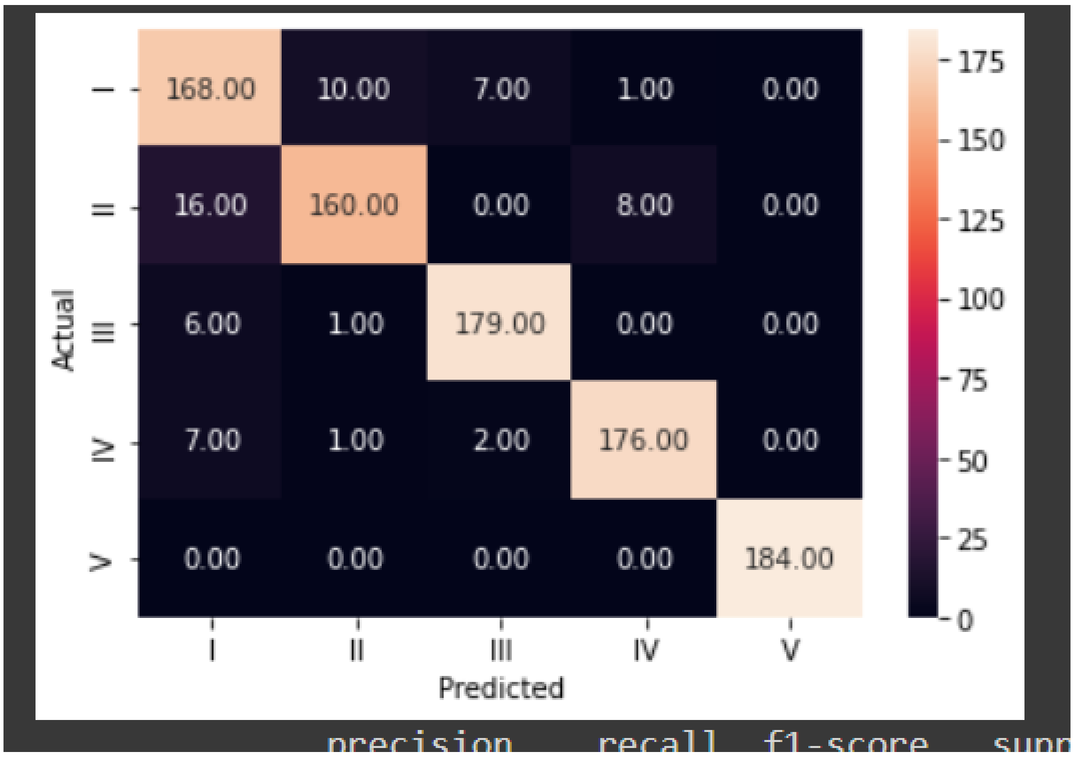
	Method	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Multi-Class Logloss
1	LogisticRegression	0.975142	0.929806	0.929247	0.929806	0.929065	0.364928
2	KNeighborsClassifier	0.986490	0.949244	0.950358	0.949244	0.948489	0.564490
3	SVC	0.996217	0.955724	0.958046	0.955724	0.955854	0.148082
4	DecisionTreeClassifier	0.998649	0.898488	0.900153	0.898488	0.898824	3.362077
5	RandomForestClassifier	0.997028	0.949244	0.950492	0.949244	0.949533	0.367133
6	BaggingClassifier	0.997298	0.939525	0.939604	0.939525	0.939337	0.402840
7	AdaBoostClassifier	0.513375	0.489201	0.628692	0.489201	0.512542	1.309672
8	GradientBoostingClassifier	0.936233	0.890929	0.892741	0.890929	0.891539	0.548589
9	CatBoostClassifier	0.997298	0.952484	0.953045	0.952484	0.952420	0.245901
10	XGBClassifier	0.979735	0.935205	0.936802	0.935205	0.935481	0.287221

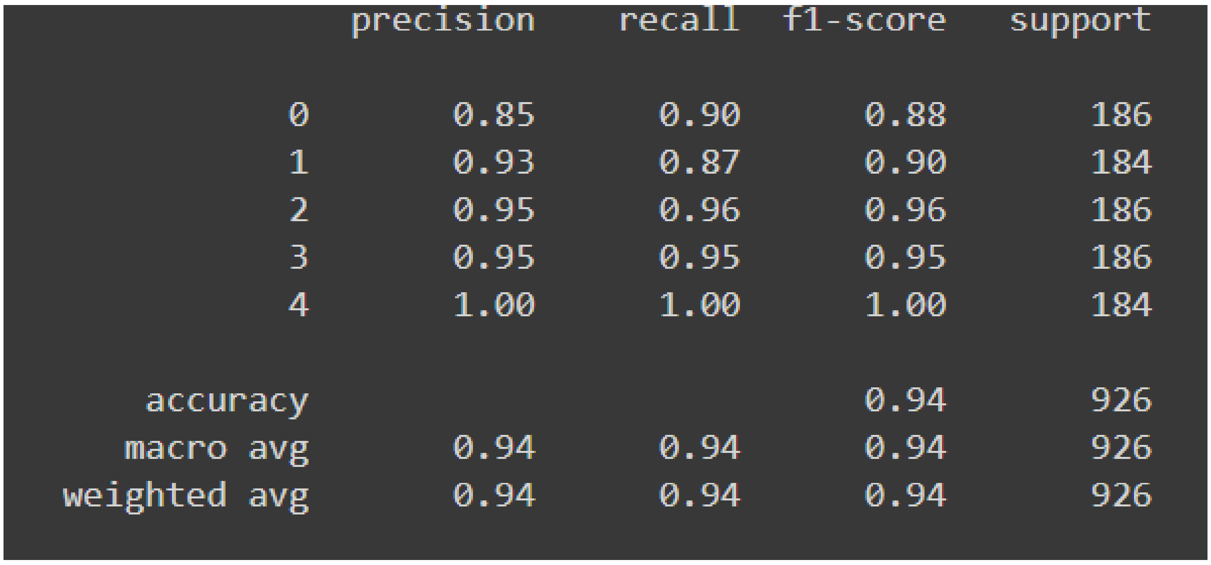
2. **Bi grams**



	Method	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Multi-Class Logloss
1	LogisticRegression	0.969468	0.916847	0.916469	0.916847	0.916501	0.379206
2	KNeighborsClassifier	0.982437	0.950324	0.951646	0.950324	0.949567	0.711496
3	SVC	0.995947	0.952484	0.954556	0.952484	0.952690	0.147141
4	DecisionTreeClassifier	0.998919	0.892009	0.892950	0.892009	0.892201	3.624266
5	RandomForestClassifier	0.998379	0.947084	0.947634	0.947084	0.947252	0.342710
6	BaggingClassifier	0.997568	0.935205	0.935772	0.935205	0.935324	0.342162
7	AdaBoostClassifier	0.514726	0.533477	0.692508	0.533477	0.553171	1.313712
8	GradientBoostingClassifier	0.937855	0.890929	0.893619	0.890929	0.891923	0.551757
9	CatBoostClassifier	0.997838	0.951404	0.951445	0.951404	0.951304	0.252907
10	XGBClassifier	0.977574	0.936285	0.937181	0.936285	0.936377	0.290846

Confusion Matrix





```
              precision    recall  f1-score   support

           0       0.85      0.90      0.88       186
           1       0.93      0.87      0.90       184
           2       0.95      0.96      0.96       186
           3       0.95      0.95      0.95       186
           4       1.00      1.00      1.00       184

    accuracy                           0.94       926
   macro avg       0.94      0.94      0.94       926
weighted avg       0.94      0.94      0.94       926
```

There is not much of a difference between the results of count vectorized data and tf idf vectorized data. In both cases, SVC performs the best followed by the catboost classifier. One more observation is that the data performs the best using tf idf vectorizer (bi-grams) with a training accuracy of 99% and a test accuracy of 95.2%. The precision, recall and F1 scores are also very good, making it the best performed model.

Deep Learning Model Training

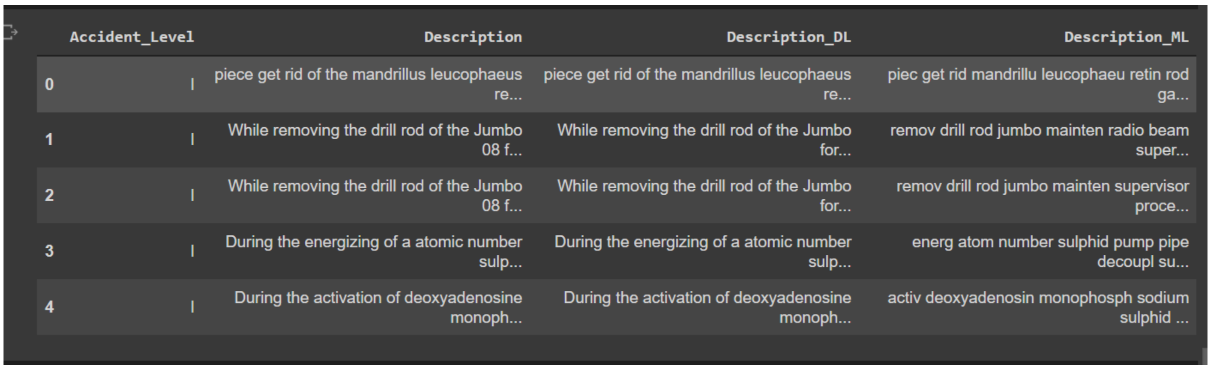
We will now work on the deep learning data. Will pass the deep learning data to the below models:

1. Simple Neural Network
2. Bi-directional LSTM
3. LSTM

We will embed our deep learning data using Glove embeddings

First, we will embed our deep learning data using Glove embeddings

Up sampled data for DL



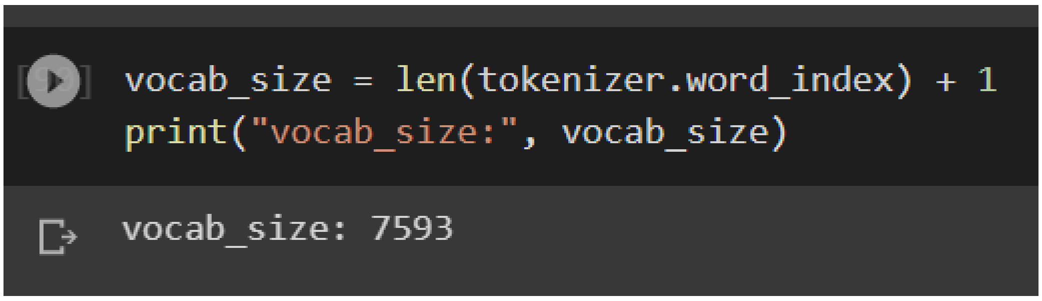
	Accident_Level	Description	Description_DL	Description_ML
0	I	piece get rid of the mandrillus leucophaeus re...	piece get rid of the mandrillus leucophaeus re...	piec get rid mandrillu leucophaeu retin rod ga...
1	I	While removing the drill rod of the Jumbo 08 f...	While removing the drill rod of the Jumbo for...	remov drill rod jumbo mainten radio beam super...
2	I	While removing the drill rod of the Jumbo 08 f...	While removing the drill rod of the Jumbo for...	remov drill rod jumbo mainten supervisor proce...
3	I	During the energizing of a atomic number sulp...	During the energizing of a atomic number sulp...	energ atom number sulphid pump pipe decoupl su...
4	I	During the activation of deoxyadenosine monoph...	During the activation of deoxyadenosine monoph...	activ deoxyadenosin monophosph sodium sulphid ...

Train and Test Splits

Since we will be passing this data to a deep learning model, we will have to one hot encode the Y variable.

Step 1 : convert the words into thier corresponding numeric indexes.

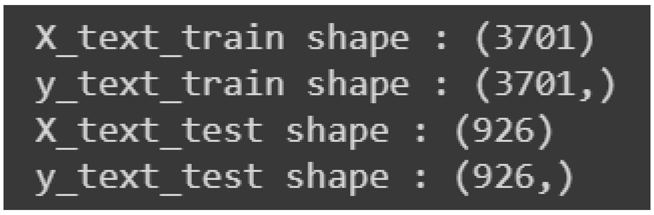
Step 2: Since the length of the sentences returned by the tokenizer are of varying lengths, we will need to pad the sequences



```
vocab_size = len(tokenizer.word_index) + 1
print("vocab_size:", vocab_size)
```

```
vocab_size: 7593
```

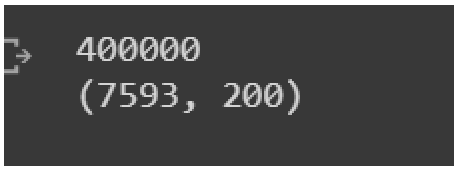
X_train,X_test, y_train, y_test shape



```
X_text_train shape : (3701)
y_text_train shape : (3701,)
X_text_test shape : (926)
y_text_test shape : (926,)
```

Let us make a weight matrix of all words in corpus using pre-trained glove embeddings

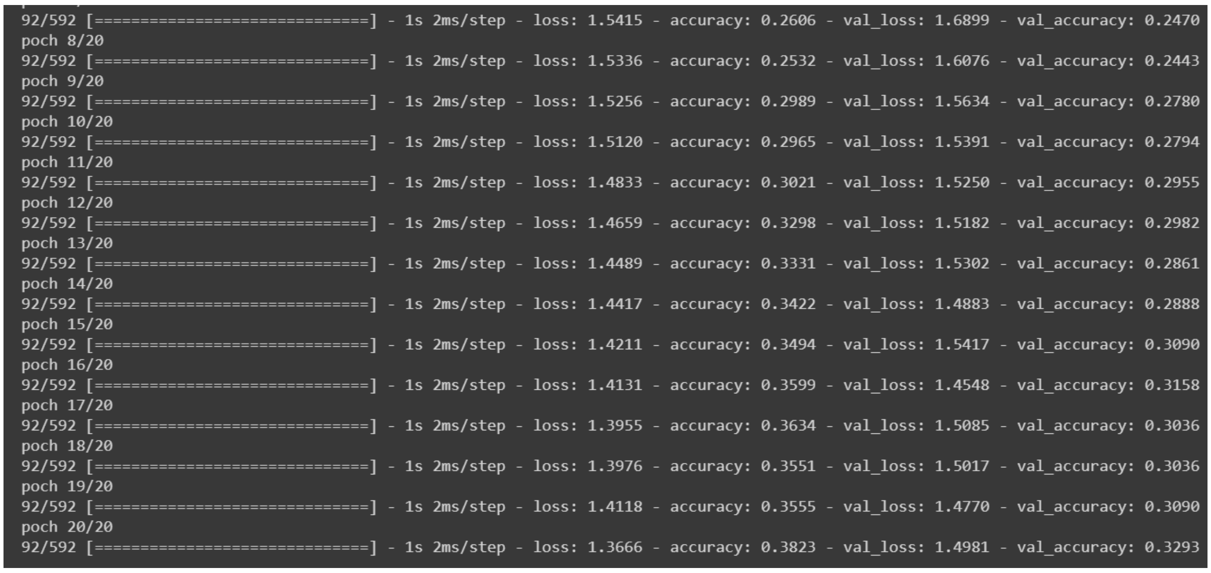
Shape of the embedding matrix is as below



```
400000
(7593, 200)
```

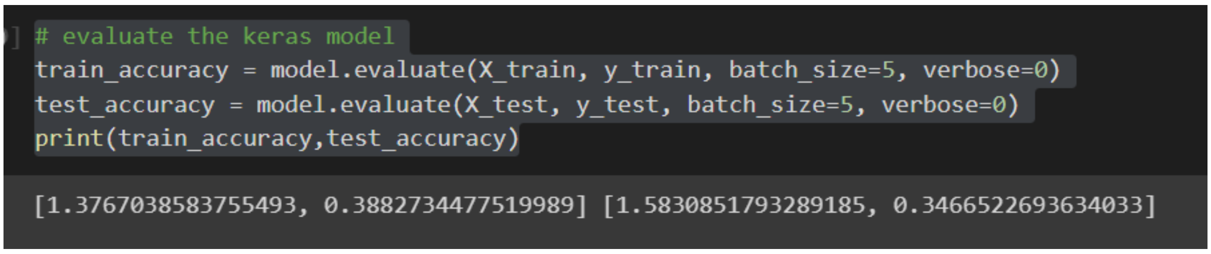
Simple NN Model Building

A simple Neural Network is used to train the model



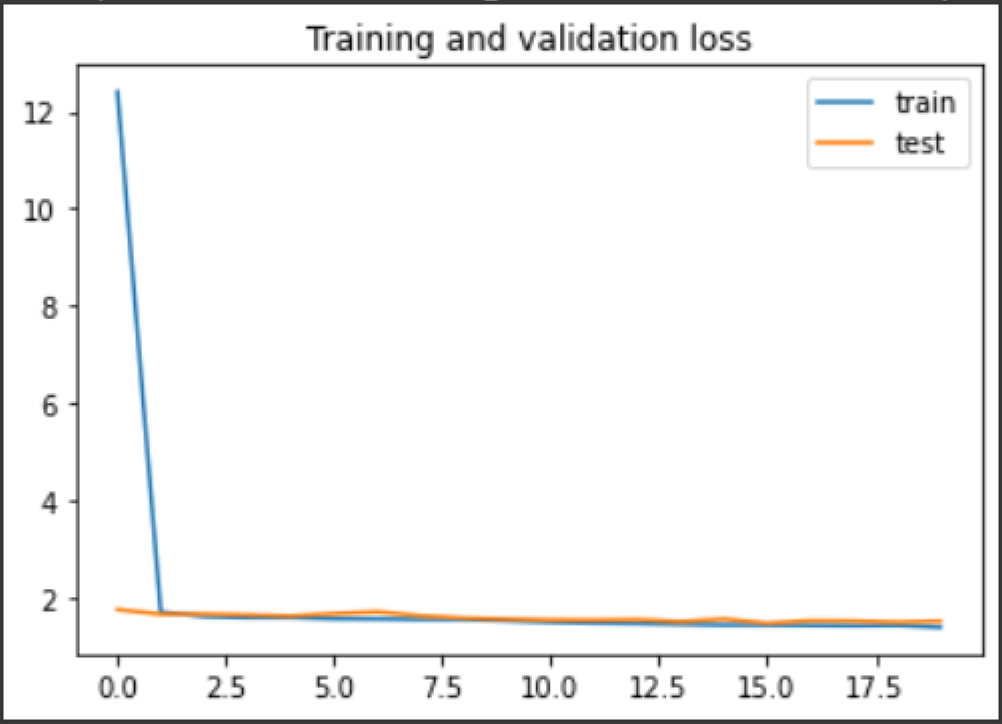
```
92/592 [==============================] - 1s 2ms/step - loss: 1.5415 - accuracy: 0.2606 - val_loss: 1.6899 - val_accuracy: 0.2470
poch 8/20
92/592 [==============================] - 1s 2ms/step - loss: 1.5336 - accuracy: 0.2532 - val_loss: 1.6076 - val_accuracy: 0.2443
poch 9/20
92/592 [==============================] - 1s 2ms/step - loss: 1.5256 - accuracy: 0.2989 - val_loss: 1.5634 - val_accuracy: 0.2780
poch 10/20
92/592 [==============================] - 1s 2ms/step - loss: 1.5120 - accuracy: 0.2965 - val_loss: 1.5391 - val_accuracy: 0.2794
poch 11/20
92/592 [==============================] - 1s 2ms/step - loss: 1.4833 - accuracy: 0.3021 - val_loss: 1.5250 - val_accuracy: 0.2955
poch 12/20
92/592 [==============================] - 1s 2ms/step - loss: 1.4659 - accuracy: 0.3298 - val_loss: 1.5182 - val_accuracy: 0.2982
poch 13/20
92/592 [==============================] - 1s 2ms/step - loss: 1.4489 - accuracy: 0.3331 - val_loss: 1.5302 - val_accuracy: 0.2861
poch 14/20
92/592 [==============================] - 1s 2ms/step - loss: 1.4417 - accuracy: 0.3422 - val_loss: 1.4883 - val_accuracy: 0.2888
poch 15/20
92/592 [==============================] - 1s 2ms/step - loss: 1.4211 - accuracy: 0.3494 - val_loss: 1.5417 - val_accuracy: 0.3090
poch 16/20
92/592 [==============================] - 1s 2ms/step - loss: 1.4131 - accuracy: 0.3599 - val_loss: 1.4548 - val_accuracy: 0.3158
poch 17/20
92/592 [==============================] - 1s 2ms/step - loss: 1.3955 - accuracy: 0.3634 - val_loss: 1.5085 - val_accuracy: 0.3036
poch 18/20
92/592 [==============================] - 1s 2ms/step - loss: 1.3976 - accuracy: 0.3551 - val_loss: 1.5017 - val_accuracy: 0.3036
poch 19/20
92/592 [==============================] - 1s 2ms/step - loss: 1.4118 - accuracy: 0.3555 - val_loss: 1.4770 - val_accuracy: 0.3090
poch 20/20
92/592 [==============================] - 1s 2ms/step - loss: 1.3666 - accuracy: 0.3823 - val_loss: 1.4981 - val_accuracy: 0.3293
```

Accuracy:



```
# evaluate the keras model
train_accuracy = model.evaluate(X_train, y_train, batch_size=5, verbose=0)
test_accuracy = model.evaluate(X_test, y_test, batch_size=5, verbose=0)
print(train_accuracy,test_accuracy)

[1.3767038583755493, 0.3882734477519989] [1.5830851793289185, 0.3466522693634033]
```

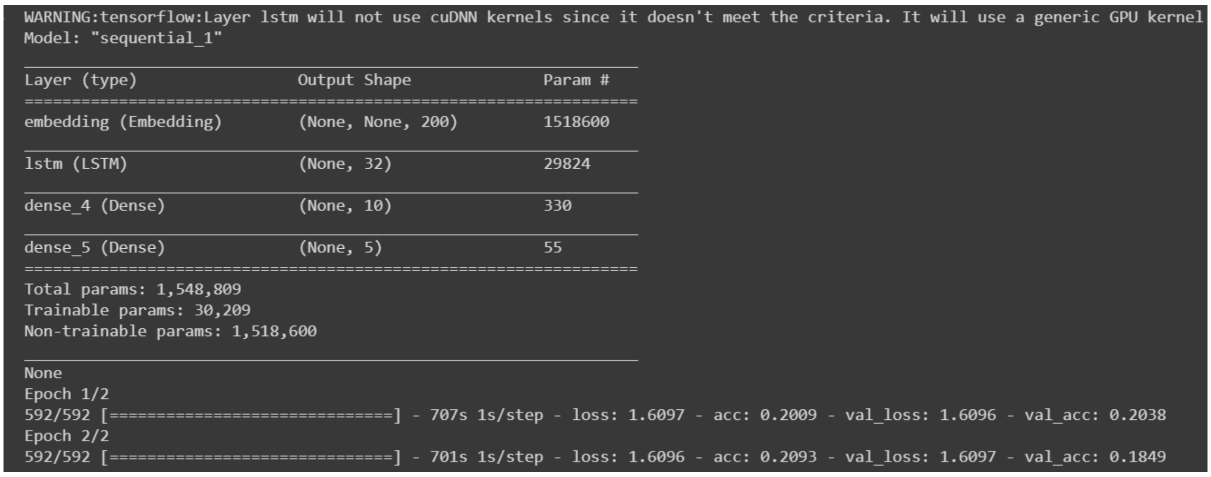
```
Text(0.5, 1.0, 'Training and validation loss')
```



We can see that a normal Neural network model does not perform well on the data. The accuracy and f1 scores are very low. Let us try LSTM model

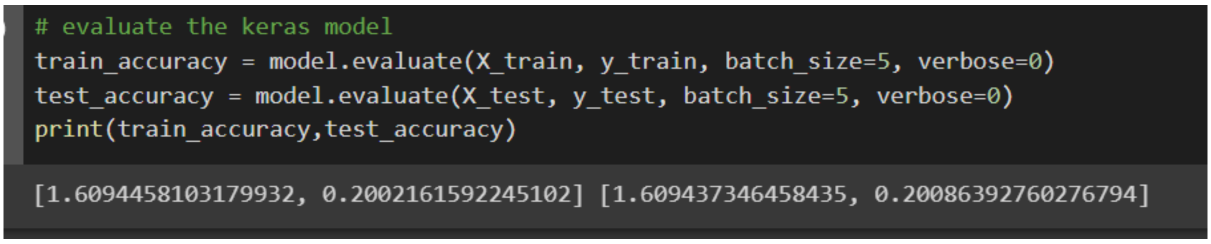
LSTM Model Building

Trying to train LSTM model



```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 200)         1518600
_________________________________________________________________
lstm (LSTM)                  (None, 32)                29824
_________________________________________________________________
dense_4 (Dense)              (None, 10)                330
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 55
=================================================================
Total params: 1,548,809
Trainable params: 30,209
Non-trainable params: 1,518,600
_________________________________________________________________
None
Epoch 1/2
592/592 [==============================] - 707s 1s/step - loss: 1.6097 - acc: 0.2009 - val_loss: 1.6096 - val_acc: 0.2038
Epoch 2/2
592/592 [==============================] - 701s 1s/step - loss: 1.6096 - acc: 0.2093 - val_loss: 1.6097 - val_acc: 0.1849
```

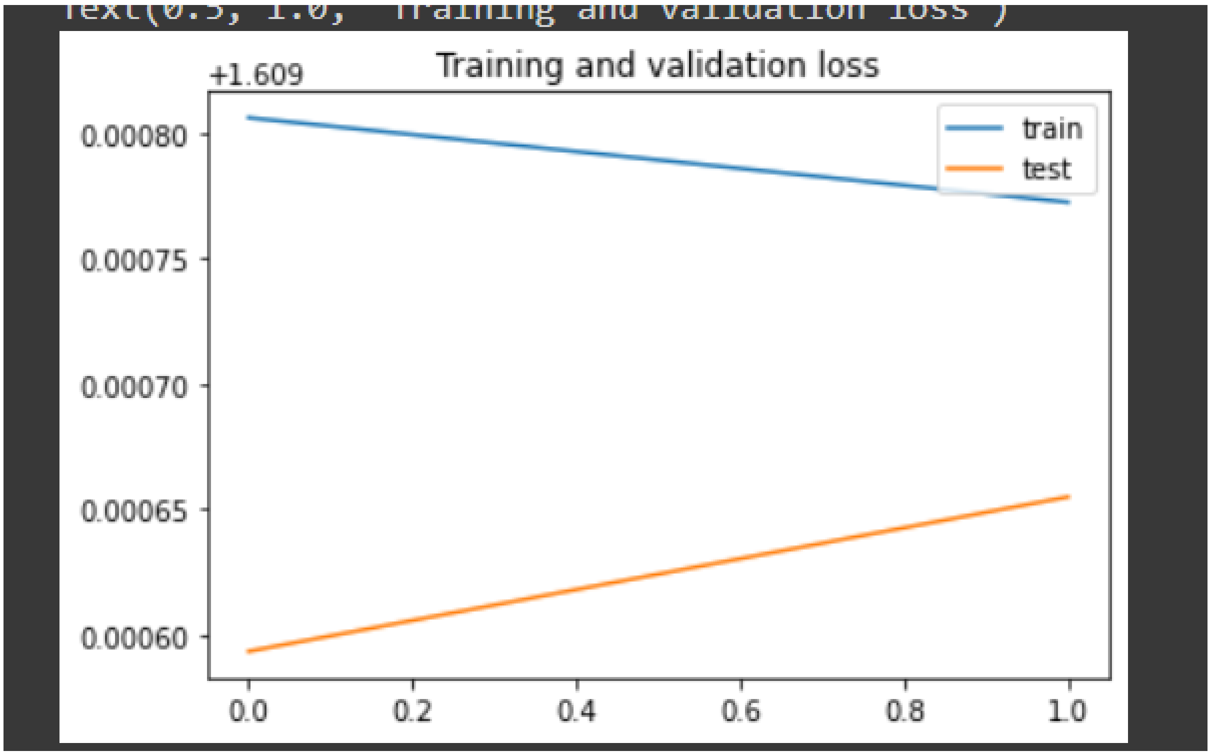
Accuracy:



```
# evaluate the keras model
train_accuracy = model.evaluate(X_train, y_train, batch_size=5, verbose=0)
test_accuracy = model.evaluate(X_test, y_test, batch_size=5, verbose=0)
print(train_accuracy,test_accuracy)
```

```
[1.6094458103179932, 0.2002161592245102] [1.609437346458435, 0.20086392760276794]
```

Train & Test results:

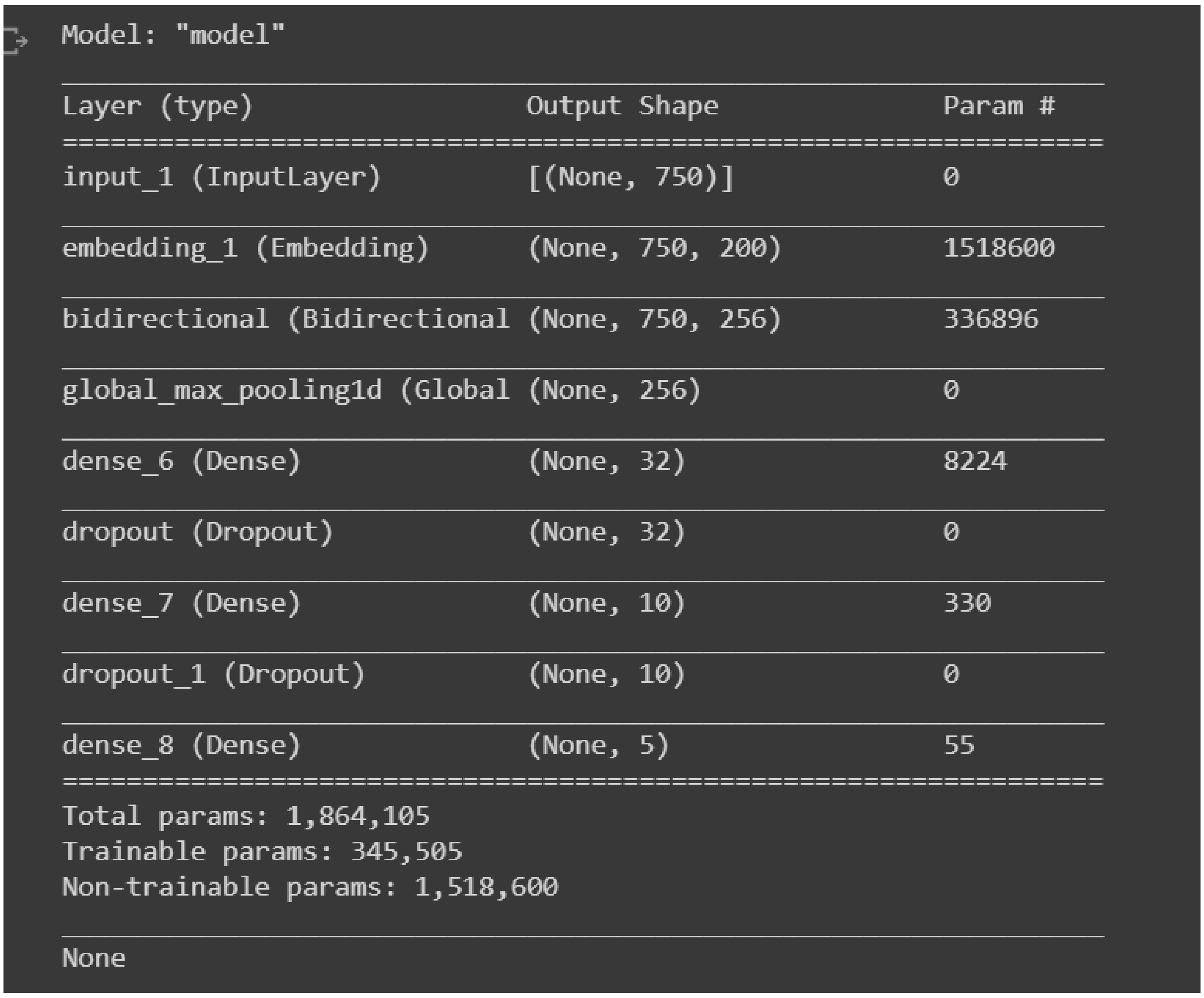


```
Text(0.5, 1.0, 'Training and validation loss')
```

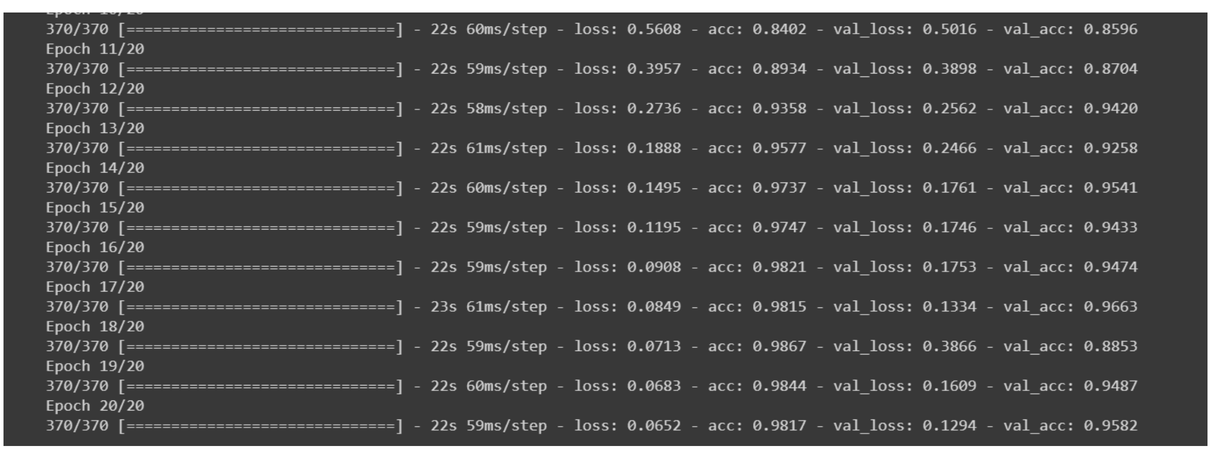
We can see that LSTM performs worse than a simple NN model. We will try using the Bi directional LSTM model.

Bi-directional LSTM Neural Network

Data is trained using Bi-directional LSTM Neural Network

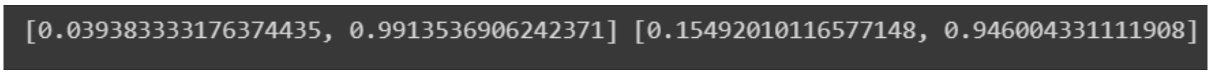


```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 750)]             0
_________________________________________________________________
embedding_1 (Embedding)      (None, 750, 200)          1518600
_________________________________________________________________
bidirectional (Bidirectional (None, 750, 256)          336896
_________________________________________________________________
global_max_pooling1d (Global (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 32)                8224
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                330
_________________________________________________________________
dropout_1 (Dropout)          (None, 10)                0
_________________________________________________________________
dense_8 (Dense)              (None, 5)                 55
=================================================================
Total params: 1,864,105
Trainable params: 345,505
Non-trainable params: 1,518,600
_________________________________________________________________
None
```



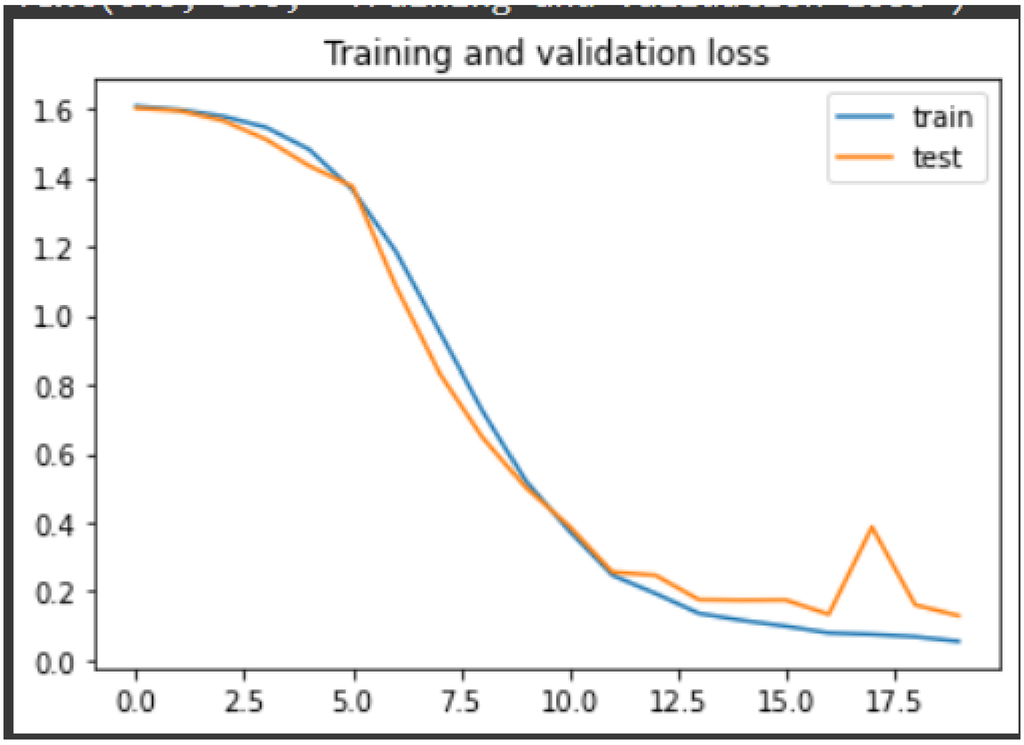
```
370/370 [============================] - 22s 60ms/step - loss: 0.5608 - acc: 0.8402 - val_loss: 0.5016 - val_acc: 0.8596
Epoch 11/20
370/370 [============================] - 22s 59ms/step - loss: 0.3957 - acc: 0.8934 - val_loss: 0.3898 - val_acc: 0.8704
Epoch 12/20
370/370 [============================] - 22s 58ms/step - loss: 0.2736 - acc: 0.9358 - val_loss: 0.2562 - val_acc: 0.9420
Epoch 13/20
370/370 [============================] - 22s 61ms/step - loss: 0.1888 - acc: 0.9577 - val_loss: 0.2466 - val_acc: 0.9258
Epoch 14/20
370/370 [============================] - 22s 60ms/step - loss: 0.1495 - acc: 0.9737 - val_loss: 0.1761 - val_acc: 0.9541
Epoch 15/20
370/370 [============================] - 22s 59ms/step - loss: 0.1195 - acc: 0.9747 - val_loss: 0.1746 - val_acc: 0.9433
Epoch 16/20
370/370 [============================] - 22s 59ms/step - loss: 0.0908 - acc: 0.9821 - val_loss: 0.1753 - val_acc: 0.9474
Epoch 17/20
370/370 [============================] - 23s 61ms/step - loss: 0.0849 - acc: 0.9815 - val_loss: 0.1334 - val_acc: 0.9663
Epoch 18/20
370/370 [============================] - 22s 59ms/step - loss: 0.0713 - acc: 0.9867 - val_loss: 0.3866 - val_acc: 0.8853
Epoch 19/20
370/370 [============================] - 22s 60ms/step - loss: 0.0683 - acc: 0.9844 - val_loss: 0.1609 - val_acc: 0.9487
Epoch 20/20
370/370 [============================] - 22s 59ms/step - loss: 0.0652 - acc: 0.9817 - val_loss: 0.1294 - val_acc: 0.9582
```

Accuracy

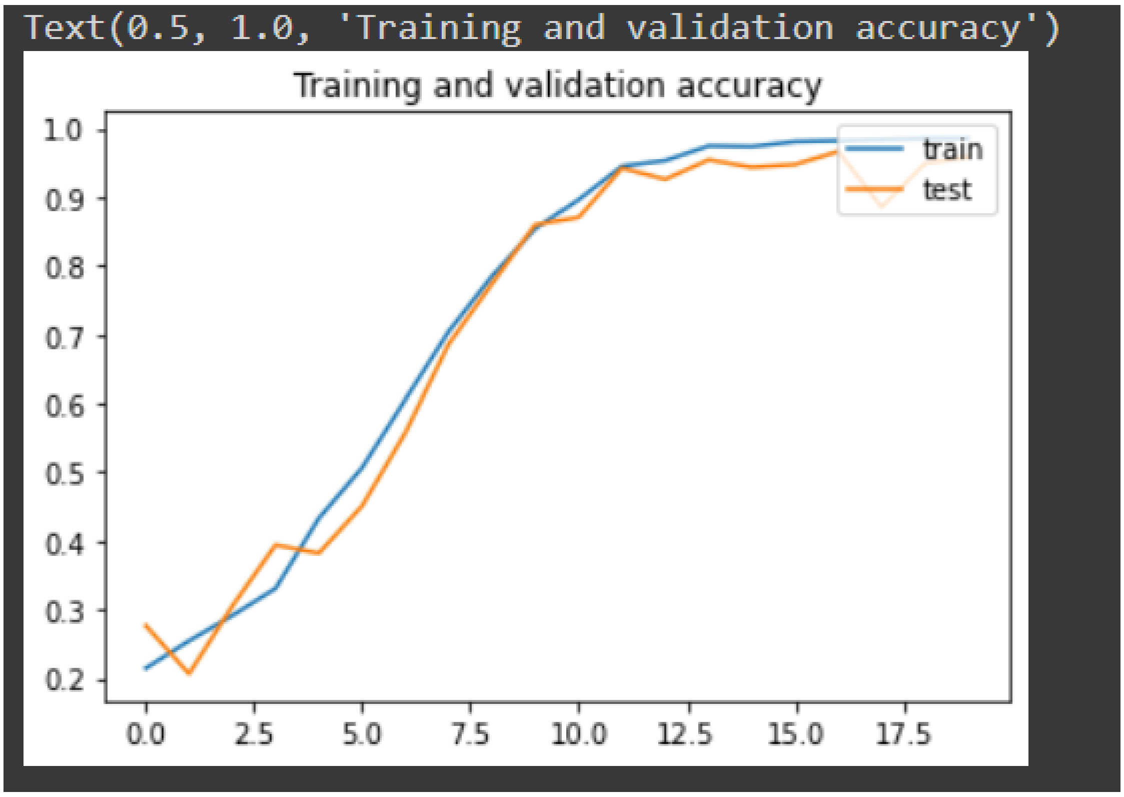


```
[0.039383333176374435, 0.9913536906242371] [0.15492010116577148, 0.946004331111908]
```

Train & Validation Loss:



Training and validation accuracy

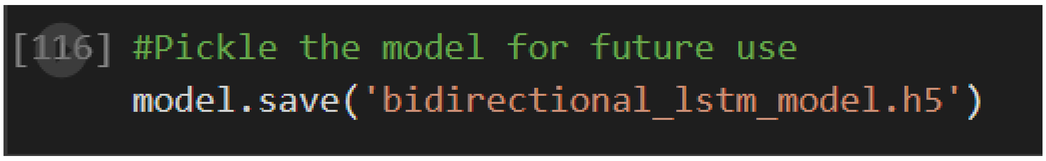


```
Text(0.5, 1.0, 'Training and validation accuracy')
```

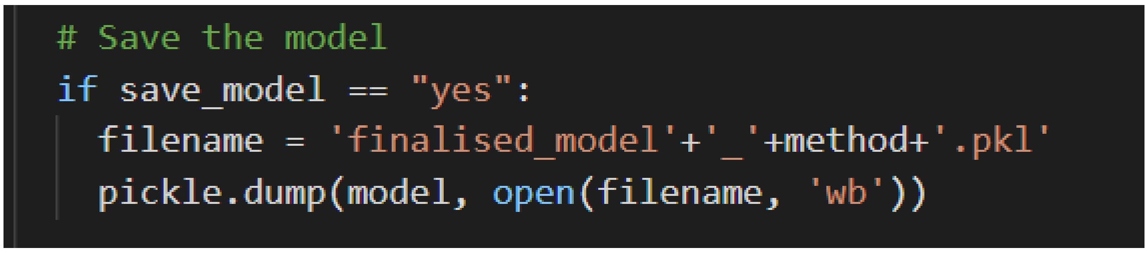
Conclusion:

We can see from the above scores that the bi directional LSTM model has performed the best out of all machine learning and deep learning models. The accuracy is very high and the loss is also very low.

Since the bidirectional LSTM performed the best we will be working with the chatbot using this model . Let us first pickle the best machine learning model(SVC) and deep learning model(Bi Direcional LSTM).



```
[116] #Pickle the model for future use
      model.save('bidirectional_lstm_model.h5')
```



```
# Save the model
if save_model == "yes":
    filename = 'finalised_model'+'_'+method+'.pkl'
    pickle.dump(model, open(filename, 'wb'))
```

Here methodname=”SVC”

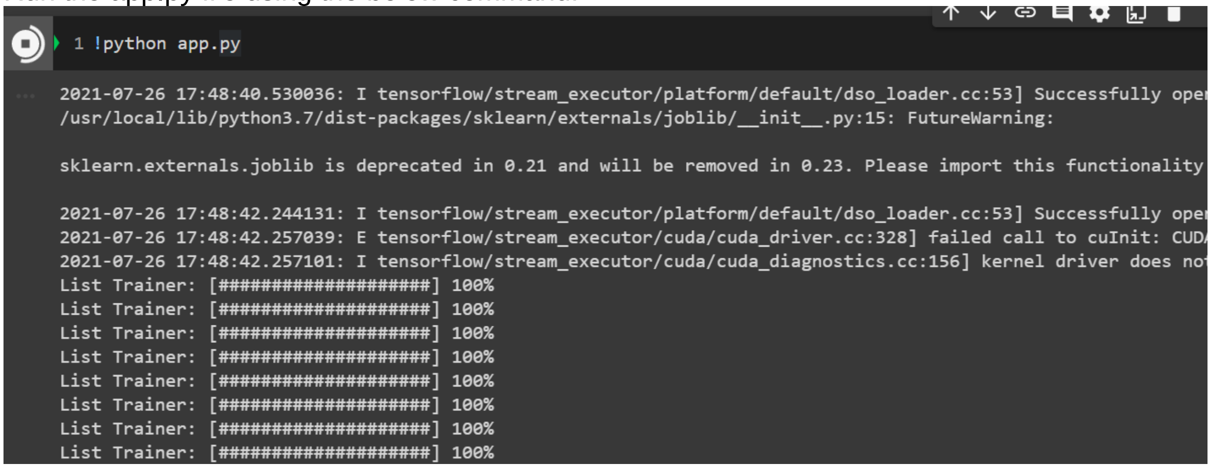
V. Clickable UI for Machine learning models

Created an UI where user will be able to perform the following tasks:

1. Upload a csv file to read data.
2. Augment the dataset by hyperparameter tuning
3. Clean data for machine learning and deep learning models.
4. Show the model scores. A total of 9 machine learning models were used totally.

Prerequisities

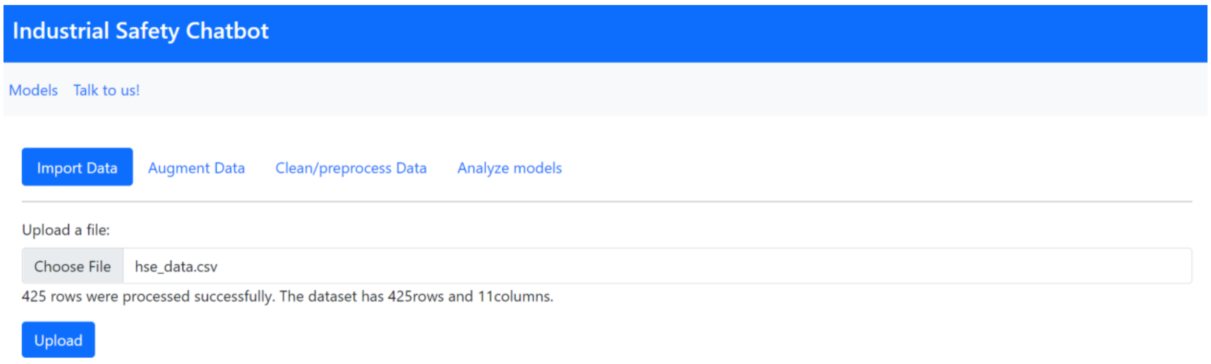
1. Create a templates folder and add Index.html in there.
2. Download the chatterbot English corpus and place it in a folder "English".
3. Run the app.py file using the below command:



```
!python app.py
```

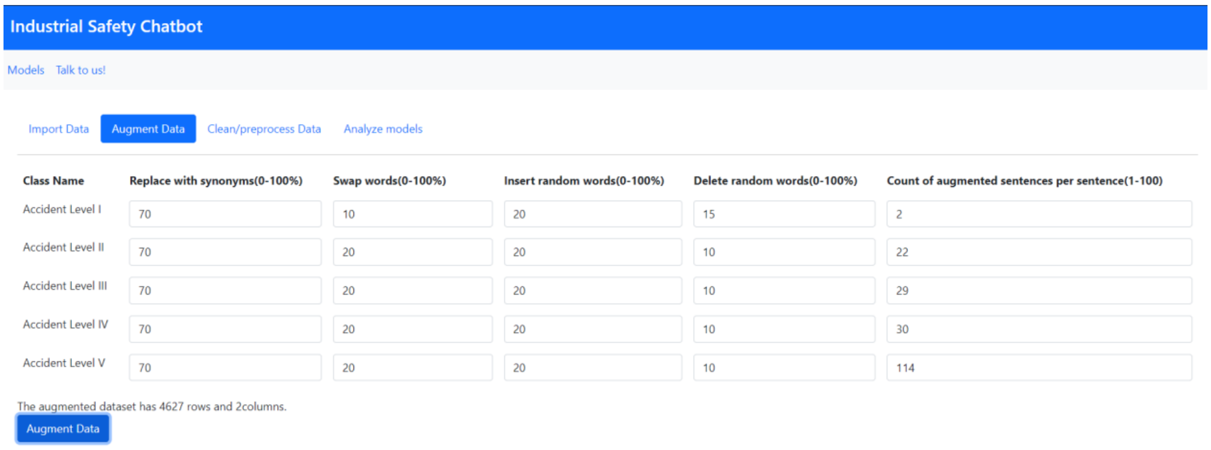
```
2021-07-26 17:48:40.530036: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully ope
/usr/local/lib/python3.7/dist-packages/sklearn/externals/joblib/__init__.py:15: FutureWarning:

sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this functionality

2021-07-26 17:48:42.244131: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully ope
2021-07-26 17:48:42.257039: E tensorflow/stream_executor/cuda/cuda_driver.cc:328] failed call to cuInit: CUD
2021-07-26 17:48:42.257101: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does no
List Trainer: [####################] 100%
List Trainer: [####################] 100%
List Trainer: [####################] 100%
List Trainer: [####################] 100%
List Trainer: [####################] 100%
List Trainer: [####################] 100%
List Trainer: [####################] 100%
List Trainer: [####################] 100%
```

Upload csv file



425 rows were processed successfully. The dataset has 425rows and 11columns.

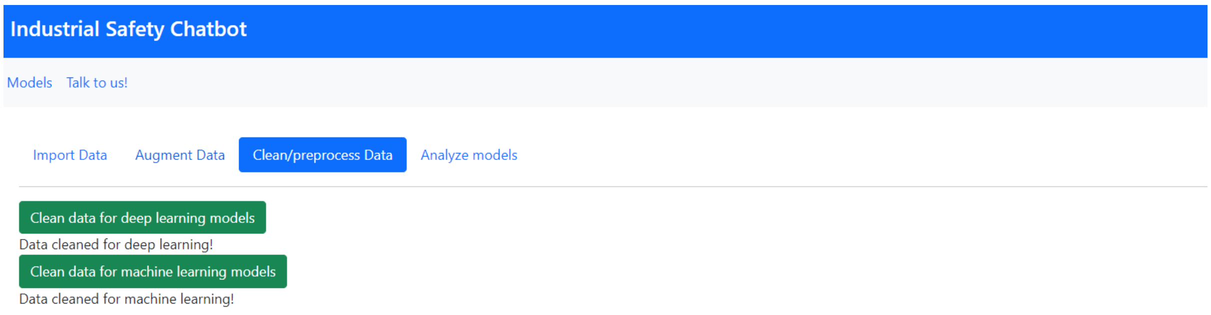
Augment the dataset by hyperparameter tuning



Class Name	Replace with synonyms(0-100%)	Swap words(0-100%)	Insert random words(0-100%)	Delete random words(0-100%)	Count of augmented sentences per sentence(1-100)
Accident Level I	70	10	20	15	2
Accident Level II	70	20	20	10	22
Accident Level III	70	20	20	10	29
Accident Level IV	70	20	20	10	30
Accident Level V	70	20	20	10	114

The augmented dataset has 4627 rows and 2columns.

Clean the machine learning and deep learning datasets.

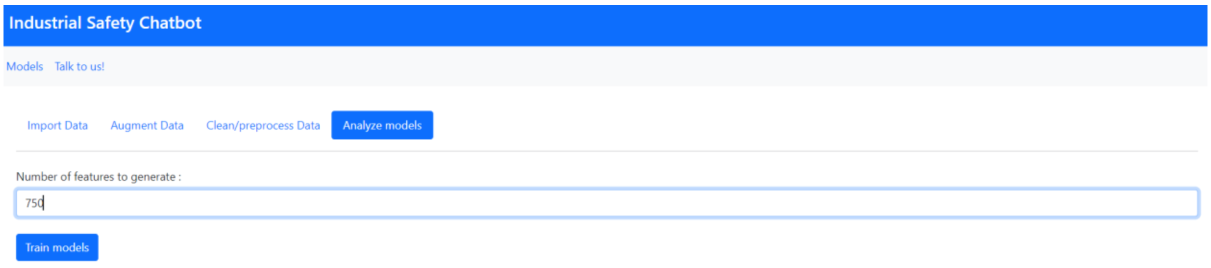

Industrial Safety Chatbot
Models
Talk to us!
Import Data
Augment Data
Clean/preprocess Data
Analyze models
Clean data for deep learning models

Data cleaned for deep learning!

Clean data for machine learning models

Data cleaned for machine learning!

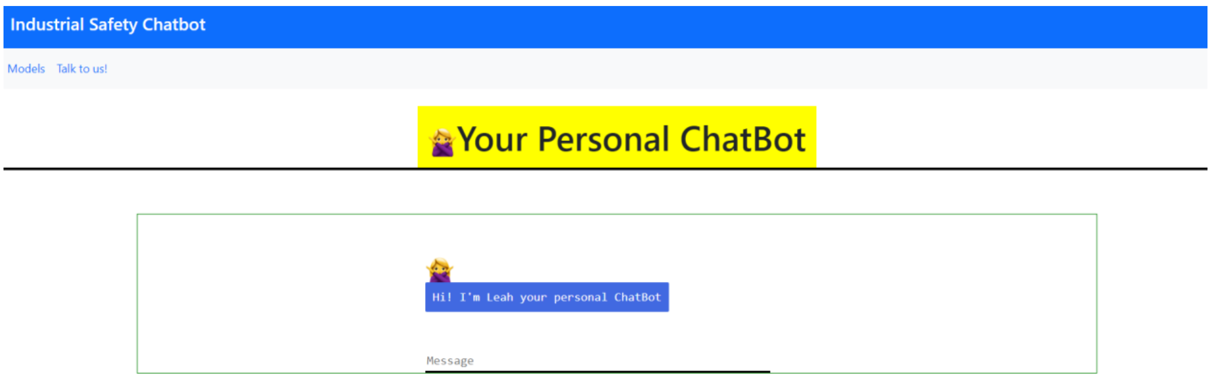
Enter the number of features to be generated and create a table of machine learning scores.



Number of features to generate : 750

I am not able to put up the screenshot of the trained models scores because I could not get the GPU to run in colab.

VI. NLP Chatbot



Hi! I'm Leah your personal ChatBot