CAPSTONE Project

Industrial Safety and Health Analytics Database

Final Report- Milestone Based

Table of Contents

Overview of the Project

Goal of the project

Machine learning chatbots work using artificial intelligence. Users need not be extremely specific while talking with a bot because it can understand the natural language, not only commands. This kind of bots get continuously better or smarter as it learns from past conversations it had with people.

Problem Statement

In this project we aim to build NLP based chatbot for Industrial Safety and Health Analytics database. In this dataset, the information about accidents in 12 manufacturing plants in 3 countries are given by a Brazilian company, IHM Stefanini. We will use this dataset to understand why accidents occur and discover clues to reduce tragic accidents. Below are the Data Set Columns which are defined as per the XLS sheet

Dataset columns are below

Data Set	Description
Date	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

- **Usage of Pre-processing**

1. Step 1: Import the data
2. Step 2: Data cleansing
3. Step 3: Data pre-processing
4. Step 4: Data preparation to be used for AIML model learning.

Step by Step walk through of the solution

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.
When creating a machine learning project, it is not always a case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. for this, we use data preprocessing tasks.
Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

It involves below steps:
- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

The parts and features of the csv file

- **CSV File Header**: The header in a CSV file is used in automatically assigning names or labels to each column of the dataset. If the file does not have a header, attributes have to be manually named.
- **Comments**: one can identify comments in a CSV file when a line starts with a hash sign (#). Depending on the method chosen to load the machine learning data, one will have to determine if these comments show up, and how to identify them.
- **Delimiter**: A delimiter separates multiple values in a field and is indicated by the comma (,). The tab (\t) is another delimiter that one can use but has to be specified clearly.
- **Quotes**: If field values in the file contain spaces, these values are often quoted and the symbol that denotes this is a double quotation mark. If chosen to use other characters, one needs to specify this in the file.

Step 1: Import the data

When running python programs, we need to use datasets for data analysis. Python has various modules which help us in importing the external data in various file formats to a python program. Below, we will see how to import data of various formats to a python program.

```
import plotly
print(plotly.__version__)
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from sklearn.impute import SimpleImputer
```

Import csv file

The csv module enables us to read each of the row in the file using a comma as a delimiter.

```
data = pd.read_csv("D:/DataScience/GL_Projects/CapstoneProject/IHMStefanini_industrial_safety_and_health_database_with_accidents_description.csv")
data.head(5)
```

Out[2]:

	Unnamed: 0	Data	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party	Critical Risk	Description
0	0	2016-01-01 00:00:00	Country_01	Local_01	Mining	I	IV	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...
1	1	2016-01-02 00:00:00	Country_02	Local_02	Mining	I	IV	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...
2	2	2016-01-06 00:00:00	Country_01	Local_03	Mining	I	III	Male	Third Party (Remote)	Manual Tools	In the sub-station MILPO located at level +170...
3	3	2016-01-08 00:00:00	Country_01	Local_04	Mining	I	I	Male	Third Party	Others	Being 9:45 am. approximately in the Nv. 1880 C...
4	4	2016-01-10 00:00:00	Country_01	Local_04	Mining	IV	IV	Male	Third Party	Others	Approximately at 11:45 a.m. in circumstances t...

Above is the code snippet and output which shows the importing of different packages to load the data. The major packages present here are pandas and numpy which help in importing the data where any numeric values in the file are read by numpy.

Step 2 and 3: Data cleansing and pre-processing

Data cleansing or **data cleaning** is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

In any Machine Learning process, Data Preprocessing is that step in which the data gets transformed, or *Encoded*, to bring it to such a state that now the machine can easily parse it. In other words, the *features* of the data can now be easily interpreted by the algorithm.

Out[4]:

	Date	Country	Local	Industry Sector	Accident Level	Potential Accident Level	Gender	Employee type	Critical Risk	Description
0	2016-01-01 00:00:00	Country_01	Local_01	Mining	I	IV	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...
1	2016-01-02 00:00:00	Country_02	Local_02	Mining	I	IV	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...
2	2016-01-06 00:00:00	Country_01	Local_03	Mining	I	III	Male	Third Party (Remote)	Manual Tools	In the sub-station MILPO located at level +170...
3	2016-01-08 00:00:00	Country_01	Local_04	Mining	I	I	Male	Third Party	Others	Being 9:45 am. approximately in the Nv. 1880 C...
4	2016-01-10 00:00:00	Country_01	Local_04	Mining	IV	IV	Male	Third Party	Others	Approximately at 11:45 a.m. in circumstances t...

In the above output we rename column names to their appropriate ones.

Once the column names are renamed, we look for data types

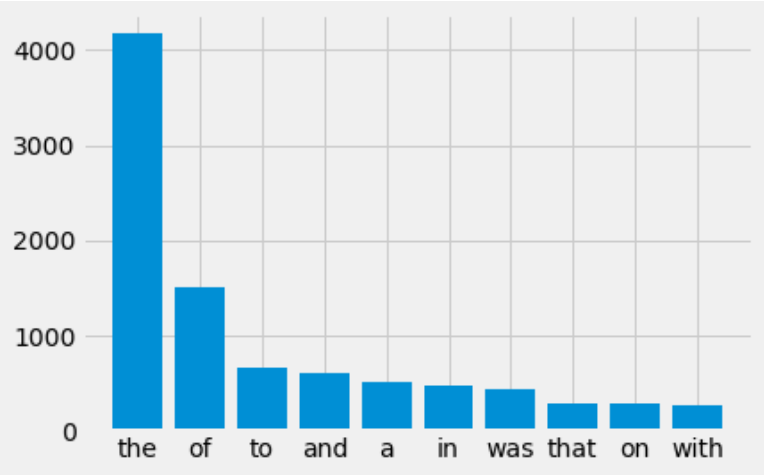
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 425 entries, 0 to 424
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Date                      425 non-null    object
 1   Country                   425 non-null    object
 2   Local                     425 non-null    object
 3   Industry Sector           425 non-null    object
 4   Accident Level            425 non-null    object
 5   Potential Accident Level  425 non-null    object
 6   Gender                    425 non-null    object
 7   Employee type             425 non-null    object
 8   Critical Risk             425 non-null    object
 9   Description               425 non-null    object
dtypes: object(10)
memory usage: 33.3+ KB
```

Out[8]:

	Date	Country	Local	Industry Sector	Accident Level	Potential Accident Level	Gender	Employee type	Critical Risk	Description
0	2016-01-01 00:00:00	Country_01	1	Mining	0	4	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...
1	2016-01-02 00:00:00	Country_02	2	Mining	0	4	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...
2	2016-01-06 00:00:00	Country_01	3	Mining	0	3	Male	Third Party (Remote)	Manual Tools	In the sub-station MILPO located at level +170...
3	2016-01-08 00:00:00	Country_01	4	Mining	0	0	Male	Third Party	Others	Being 9:45 am. approximately in the Nv. 1880 C...
4	2016-01-10 00:00:00	Country_01	4	Mining	4	4	Male	Third Party	Others	Approximately at 11:45 a.m. in circumstances t...

Analysis Using Stop Words in Text Mining

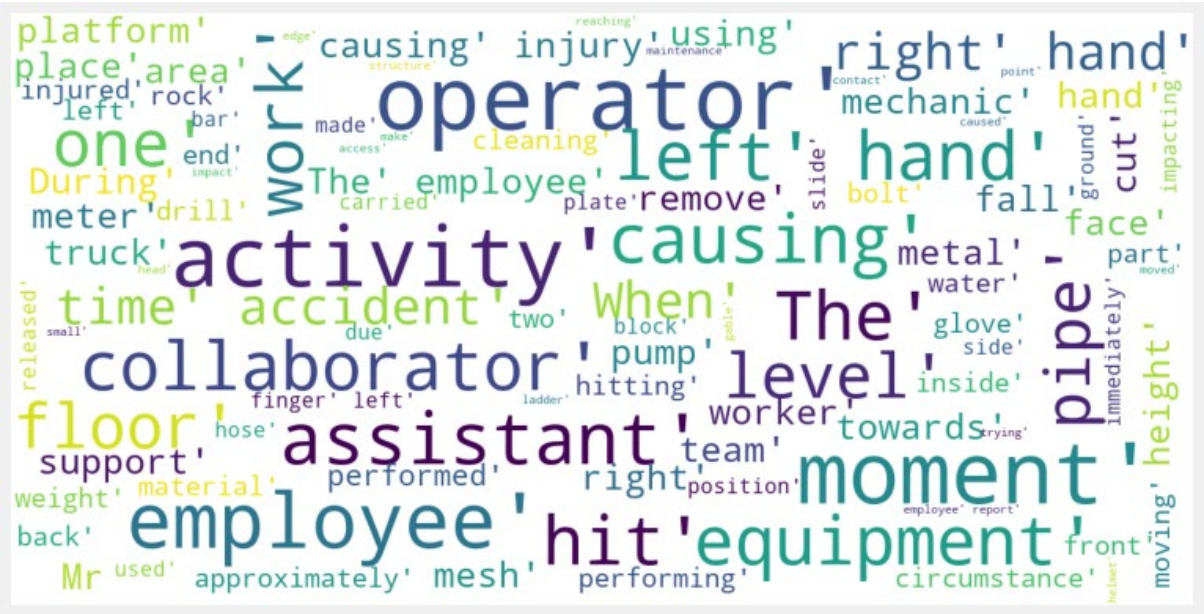
Stop words are basically a set of commonly used words in any language, not just English. The reason why stop words are critical to many applications is that, if we remove the words that are very commonly used in each language, we can focus on the important words instead.



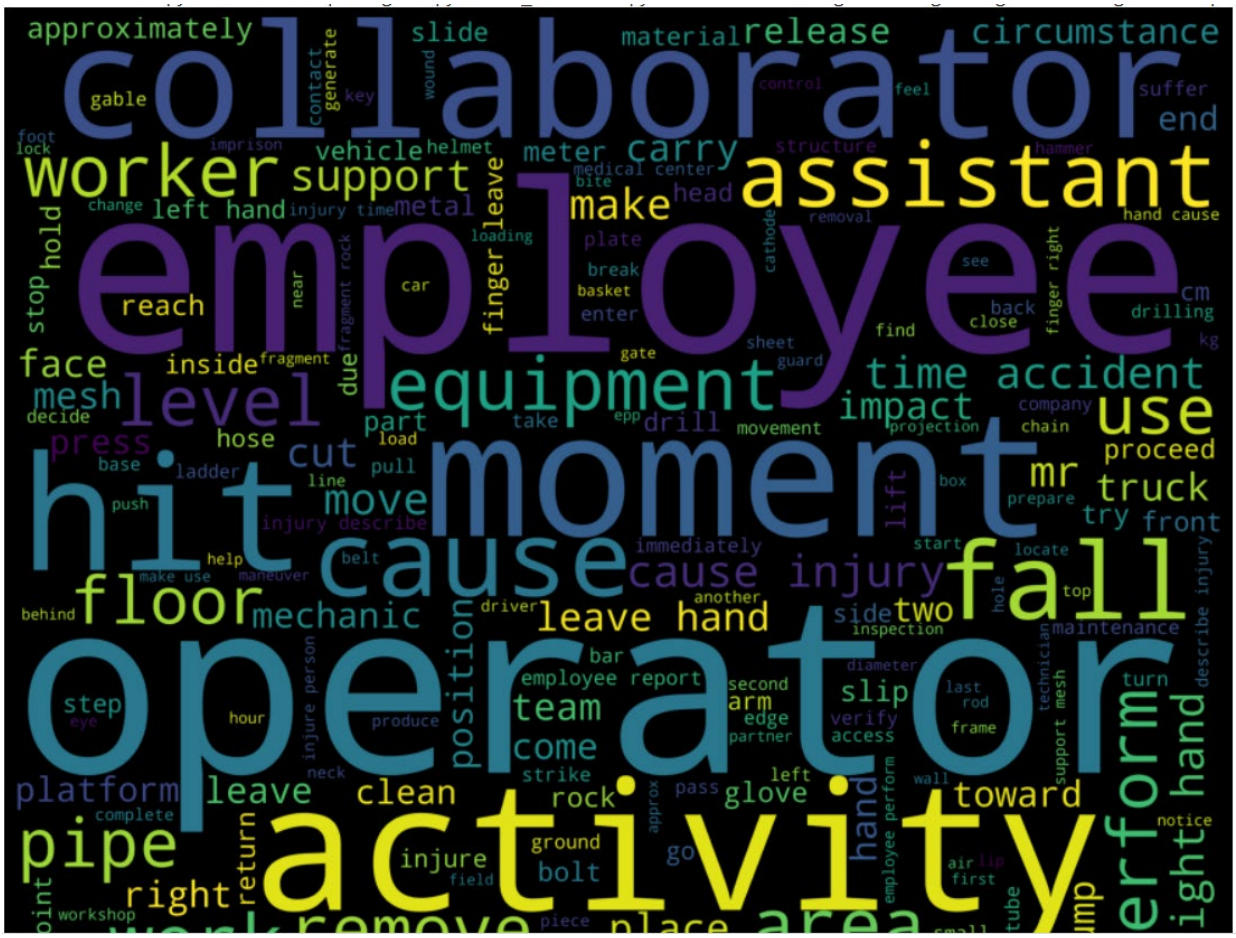
Analysing the N-grams Using Word Cloud

When the data is pre-processed, we then make usage of word cloud. The word cloud does this task according to the frequency of words in which text size tells the relative importance of words of our entire dataset very quickly. This can be used where we need to quickly show how people feel about our product in presentation and grabbing attention to the important keywords that we want to represent. Below diagram represents the same.

Before usage of Word Cloud:-



After the Usage of Wordcloud :-



The above output shows the scaling of the data from the word cloud. Here we randomly generate words to fit into a scale with words no greater than hundred in number. This reduces the data per frame and makes it accessible to pre-process. As shown above, randomly generated words in a frame ready for analysis.

Step 4: Data preparation to be used for AIML model learning.

Data preparation may be one of the most difficult steps in any machine learning project. The reason is that each dataset is different and highly specific to the project. Nevertheless, there are enough commonalities across predictive modelling projects that we can define a loose sequence of steps and subtasks that you are likely to perform.

This process provides a context in which we can consider the data preparation required for the project, informed both by the definition of the project performed before data preparation and the evaluation of machine learning algorithms performed after.

Out[11]:

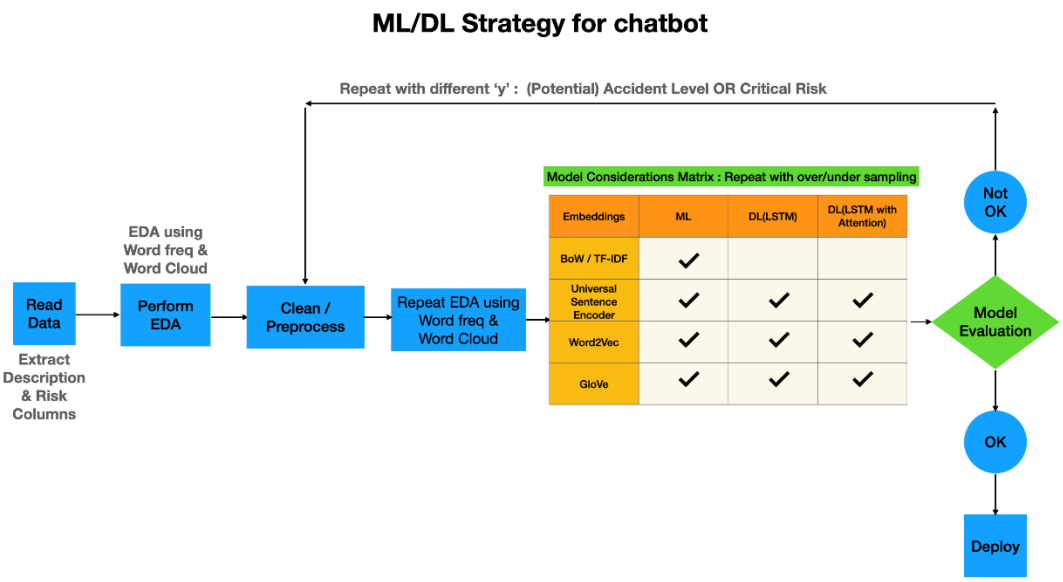
	Date	Country	Local	Industry Sector	Accident Level	Potential Accident Level	Gender	Employee type	Critical Risk	Description	Year	Month	Day	Weekday	WeekofYear
0	2016-01-01	Country_01	1	Mining	0	4	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...	2016	1	1	Friday	53
1	2016-01-02	Country_02	2	Mining	0	4	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...	2016	1	2	Saturday	53
2	2016-01-06	Country_01	3	Mining	0	3	Male	Third Party (Remote)	Manual Tools	In the sub-station MILPO located at level +170...	2016	1	6	Wednesday	1
3	2016-01-08	Country_01	4	Mining	0	0	Male	Third Party	Others	Being 9:45 am. approximately in the Nv. 1880 C...	2016	1	8	Friday	1
4	2016-01-10	Country_01	4	Mining	4	4	Male	Third Party	Others	Approximately at 11:45 a.m. in circumstances t...	2016	1	10	Sunday	1

Out[12]:

	Date	Country	Local	Industry Sector	Accident Level	Potential Accident Level	Gender	Employee type	Critical Risk	Description	Year	Month	Day	Weekday	WeekofYear	Qu
0	2016-01-01	Country_01	1	Mining	0	4	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...	2016	1	1	Friday	53	Fi
1	2016-01-02	Country_02	2	Mining	0	4	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...	2016	1	2	Saturday	53	Fi
2	2016-01-06	Country_01	3	Mining	0	3	Male	Third Party (Remote)	Manual Tools	In the sub-station MILPO located at level +170...	2016	1	6	Wednesday	1	Fi
3	2016-01-08	Country_01	4	Mining	0	0	Male	Third Party	Others	Being 9:45 am. approximately in the Nv. 1880 C...	2016	1	8	Friday	1	Fi
4	2016-01-10	Country_01	4	Mining	4	4	Male	Third Party	Others	Approximately at 11:45 a.m. in circumstances t...	2016	1	10	Sunday	1	Fi

In the above output tables, we can see that the data is being prepared for various analyses as per requirement for AIML model learning. The data is rearranged accordingly where the null values are eliminated during the cleaning phase. We use an elif statement loop to achieve this task.

Model



The above flowchart shows the various steps involved in building a strategy for chatbot. In the first few steps we perform data preprocessing and loading. We then repeat EDA with over and under sampling. In the final phase we evaluate the model. If the model runs well, we deploy it else the process is repeated.

Summary of Milestone 1

Data preprocessing is an important step when it comes to machine learning. Any dataset given, generally has many null entries and the data is scrambled. Without proper preprocessing analysis on the dataset is impossible. The techniques mentioned above help in sorting and scrumming data according to the needs of the user making it possible for performing analysis and different operations as per the problem statement of the project. We hereby conclude that all the data pre-processing techniques for the given problem statement have been applied and successfully executed.

Model Evaluation- Milestone 2

The approach of the modelling part of this problem in different ways. We could take it as a regression problem and predict the number of fatalities based on the attributes of the accident dataset. We can also approach it as a classification problem and predict the severity of the accident based on the accident dataset. In this project the approach it as a classification model approach. The attributes we have are enough to build a baseline, and we can always revisit this and keep improving our model accuracy. We first need to convert the categorical features into numerical values.

Objectives of this Model:

- Presumption of cause of accidents
- Surveying a factor that increases severity of accidents.

Building the model which classify the severity of accidents, we can understand the factor related to the causality of accidents.

Step 1: NLP Pre-Processing – PCA

Description about accidents is important to understand the cause of accidents, so we need to discover characteristically words or phrases indicating situation when accidents occurred.

Principal Component Analysis is basically a statistical procedure to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables.

Each of the principal components is chosen in such a way so that it would describe most of the still available variance and all these principal components are orthogonal to each other. In all principal components the first principal component has maximum variance.

Steps involved in PCA:

1.Import the dataset and distribute the dataset into X and y components for data analysis.

2.Split the dataset into the Training set and Test set.

3.Do the pre-processing part on a training and testing set such as fitting the Standard scale.

4.Apply the PCA function into training and testing sets for analysis.

5.Fit Regression model to the train set.

6.Visualise the results.

While pre-processing on training data we have used Helper function for text cleaning. The Helper function consists of remove special character which helps in removing the special characters and stop words function which is commonly used words like ((such as “the”, “a”, “an”, “in”). The final step after the text cleaning is shown below along with the target converting into category. Extraction of the unique words and padding is also done as a part of pre-processing techniques.

decontracted() functions takes text and convert it to natural form

```
import re
def decontracted(phrase):
  """decontracted takes text and convert contractions into natural form.
     ref: https://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python/47091490#47091

  # specific
  phrase = re.sub(r"won\'t", "will not", phrase)
  phrase = re.sub(r"can\'t", "can not", phrase)
  phrase = re.sub(r"won\’t", "will not", phrase)
  phrase = re.sub(r"can\’t", "can not", phrase)

  # general
  phrase = re.sub(r"n\'t", " not", phrase)
  phrase = re.sub(r"\'re", " are", phrase)
  phrase = re.sub(r"\'s", " is", phrase)
  phrase = re.sub(r"\'d", " would", phrase)
  phrase = re.sub(r"\'ll", " will", phrase)
  phrase = re.sub(r"\'t", " not", phrase)
  phrase = re.sub(r"\'ve", " have", phrase)
  phrase = re.sub(r"\'m", " am", phrase)

  phrase = re.sub(r"n\’t", " not", phrase)
  phrase = re.sub(r"\’re", " are", phrase)
  phrase = re.sub(r"\’s", " is", phrase)
  phrase = re.sub(r"\’d", " would", phrase)
  phrase = re.sub(r"\’ll", " will", phrase)
  phrase = re.sub(r"\’t", " not", phrase)
  phrase = re.sub(r"\’ve", " have", phrase)
  phrase = re.sub(r"\’m", " am", phrase)

  return phrase
```

remove_special_character() it removes special characters from text

```
def remove_special_character(phrase, remove_number=False):
  """remove_special_character takes text and removes special charcters.
     ref: https://stackoverflow.com/a/18082370/4084039"""

  phrase = re.sub("\S*\d\S*", "", phrase).strip()
  if remove_number:
    phrase = re.sub('[^A-Za-z]+', ' ', phrase)
  else:
    phrase = re.sub('[^A-Za-z0-9]+', ' ', phrase)
  return phrase
```

remove_stop_words() it removes stopwords

```
def remove_stop_words(text):
    stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', \
                "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', \
                'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", \
                'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', \
                'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', \
                'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', \
                'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
                'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', \
                'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', \
                'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', \
                'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'only', \
                'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", \
                'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", \
                'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', \
                "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', \
                "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", \
                'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"])
    return ' '.join(e.lower() for e in text.split() if e.lower() not in stopwords)
```

In **lemmatization**, the algorithms refer a dictionary to understand the meaning of the word before reducing it to its root word, or lemma.

```
def lemmatize_text(text_data):
  """lem_text takes text and lemmatize it using WordNetLemmatizer.
     ref: https://stackoverflow.com/a/25535348"""
  lem = WordNetLemmatizer()
  n_text = []
  for word in text_data.split(' '):
    n_word = lem.lemmatize(word, pos='a')
    n_word = lem.lemmatize(n_word, pos='v')
    n_text.append(n_word)

  return ' '.join(n_text)
```

stem_and_stopwords() it stems the words which are not in stopwords

In **stemming**, a part of the word is just chopped off at the tail end to arrive at the stem of the word. There are different algorithms used to find out how many characters have to be chopped off, but the algorithms don't actually know the meaning of the word in the language it belongs to.

	Unnamed: 0	Data	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party	Critical Risk	Description
0	0	01-01-2016 00:00	Country_01	Local_01	Mining	I	IV	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...
1	1	02-01-2016 00:00	Country_02	Local_02	Mining	I	IV	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...

Convert target to Category results as shown below.

```
0       0
1       0
2       0
3       0
4       3
       ..
420     0
421     0
422     0
423     0
424     0
Name: target, Length: 425, dtype: category
Categories (5, int64): [0, 1, 2, 3, 4]
```

Split the dataset into the Training set and Test set.

Embedding Matrix is created. Embedding is required as a approach for representing words and documents. Word Embedding or Word Vector is a numeric vector input that represents a word in a lower-dimensional space. It allows words with similar meaning to have a similar representation.

Feature engineering includes creating new features from accident description, the accident description contains lot of information but it can't be used directly for the model because it is a text data. So, we need to find the way to convert that unstructured data to structured data. So, we have lot of methods to complete this task.

Word embedding using word2vec algorithm is one of those methods. So this word2vec method will convert the text data to numeric format and that numeric format we call as vectors.

Word2vec algorithm give vectors keeping the context intact

Create Word2Vec from genism

Gensim = "Generate Similar" is a popular open source natural language processing (NLP) library used for unsupervised topic modeling. It uses top academic models and modern statistical machine learning to perform various complex tasks such as – Building document or word vectors.

Word2vec is a technique/model to produce word embedding for better word representation. It is a natural language processing method that captures a large number of precise syntactic and semantic word relationships. It is a shallow two-layered neural network that can detect synonymous words and suggest additional words for partial

sentences once it is trained the results showcased below are for usage of Word2Vec technique in our project

Vocabulary Size output- `Vocabulary size: 2372`

Most Similar type:-

```
[('fragment', 0.9874339699745178),
 ('use', 0.987318217754364),
 ('make', 0.986915111541748),
 ('operator', 0.9858063459396362),
 ('work', 0.9849867820739746),
 ('hand', 0.9846134185791016),
 ('area', 0.98447585105896),
 ('hit', 0.9843685626983643),
 ('right', 0.9843416213989258),
 ('cause', 0.9841146469116211)]
```

Usage of **Word2Vec** vectorize the text samples into a 2D integer tensor output:-

```
Found 2372 unique tokens.
Shape of Desc tensor: (425, 622
```

Train and Val split:: (90-10)

For any model we are going to train, it's important to check if it's as per our expectations. This called evaluating the model.

```
########################################
Number of rows in training dataset: 382
Number of columns in training dataset: 622
Number of unique words in training dataset: 2258
########################################
Number of rows in test dataset: 43
Number of columns in test dataset: 622
Number of unique words in test dataset: 677
```

Step 2: Design, Test and Train Machine learning classifiers

Evaluating the model's accuracy

The sequential model is a linear stack of layers. You create a sequential model by calling the keras_model_sequential () function then a series of layer functions: Note that Keras objects are modified in place which is why it's not necessary for model to be assigned back to after the layers are added. The output here is shown below

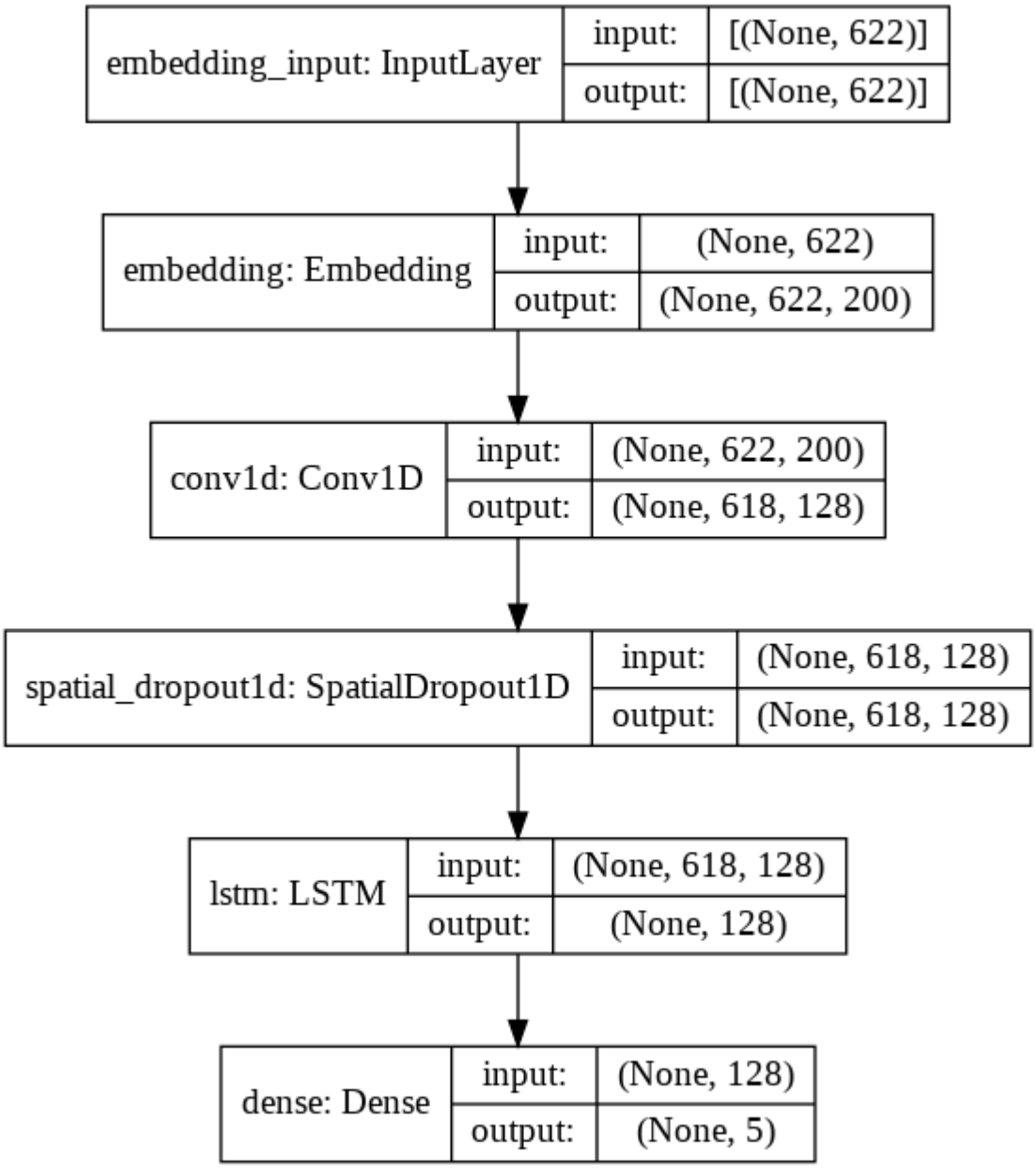
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 622, 200)          475600
_________________________________________________________________
conv1d (Conv1D)              (None, 618, 128)          128128
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 618, 128)          0
_________________________________________________________________
lstm (LSTM)                  (None, 128)               131584
_________________________________________________________________
dense (Dense)                (None, 5)                 645
=================================================================
Total params: 735,957
Trainable params: 260,357
Non-trainable params: 475,600
_________________________________________________________________
None
```

Next step is using Call back function.

A callback is a function that is passed as an argument to other function. This other function is expected to call this callback function in its definition. The point at which other function calls our callback function depends on the requirement and nature of other function.

Callback Functions are generally used with asynchronous functions.

Below results are Callback Function: Callback Function can be passed to a function to print out the size of file after the function reads given text file.



Step 3: Design, train and test using Neural Networks classifiers

Model done using imblearn :

Imbalanced-Learn is a Python module that helps in balancing the datasets which are highly skewed or biased towards some classes. Thus, it helps in resampling the classes which are otherwise oversampled or undersampled. If there is a greater imbalance ratio, the output is biased to the class which has a higher number of examples. The following dependencies need to be installed to use imbalanced-learn:

While using Imblearn one of the most used function is over sampling. Here we have used oversampling function. It is a naive method where classes that have low examples are generated and randomly resampled using train set data.

Next step is to work on Model fit the train dataset.

Imbalanced classification involves developing predictive models on classification datasets that have a severe class imbalance.

The challenge of working with imbalanced datasets is that most machine learning techniques will ignore, and in turn have poor performance on, the minority class, although typically it is performance on the minority class that is most important.

One approach to addressing imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short. The below shows the sampling results

```
Before Sampling: X_train: (382, 622), y_train: (382, 5)

After Sampling: X: (1395, 622), y: (1395, 5)
```

Models are trained by NumPy arrays using fit (). The main purpose of this fit function is used to evaluate your model on training. The results are shown below

```
Epoch 1/5
8/8 - 33s - loss: 1.0417 - accuracy: 0.5912 - val_loss: 1.0422 - val_accuracy: 0.5581

Epoch 00001: val_loss improved from 1.48348 to 1.04220, saving model to risk_classifier.h5
Epoch 2/5
8/8 - 31s - loss: 0.9609 - accuracy: 0.6445 - val_loss: 1.3789 - val_accuracy: 0.5814

Epoch 00002: val_loss did not improve from 1.04220
Epoch 3/5
8/8 - 30s - loss: 0.8577 - accuracy: 0.6824 - val_loss: 1.7377 - val_accuracy: 0.4419

Epoch 00003: val_loss did not improve from 1.04220
Epoch 4/5
8/8 - 30s - loss: 0.7544 - accuracy: 0.7346 - val_loss: 1.8792 - val_accuracy: 0.3953

Epoch 00004: val_loss did not improve from 1.04220
Epoch 5/5
8/8 - 31s - loss: 0.6469 - accuracy: 0.7838 - val_loss: 1.3569 - val_accuracy: 0.6977

Epoch 00005: val_loss did not improve from 1.04220
```

Step 4: Design, train, and test LSTM Classifier – Evaluation of the Model

Long short-term memory is an artificial recurrent neural network architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process single data points as well as entire sequences of data.

It contains three layers:

1. Embedding layer
2. SpatialDropout layer
3. LSTM layer
4. Dense layer

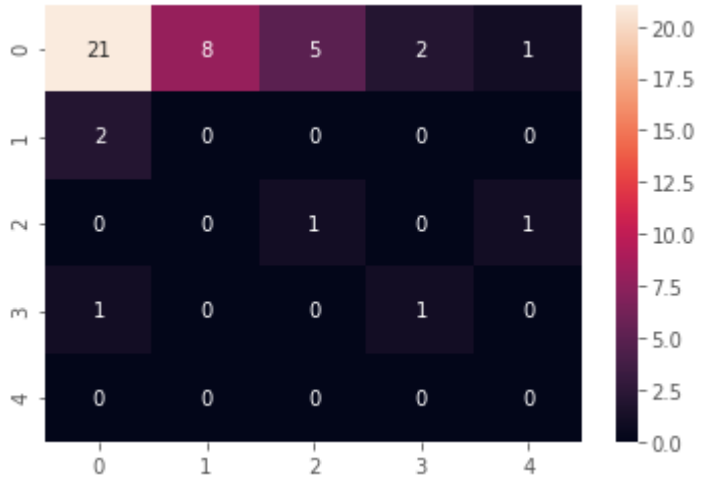
During the evaluation of the Model we will using Loss function (Val_loss). At the most basic level, a loss function quantifies how “good” or “bad” a given predictor is at classifying the input data points in a dataset. The smaller the loss, the better a job the classifier is at modeling the relationship between the input data and the output targets. There is a point where we can overfit our model — by modeling the training data too closely, our model loses the ability to generalize for the train data.

When we are training the model in keras, accuracy and loss in keras model for validation data could be variating with different cases. Usually with every epoch increasing, loss should be going lower and accuracy should be going higher. But with val_loss(keras validation loss) and val_acc(keras validation accuracy), many cases can be possible like below:

This result is achieved by using Train and Test data. Below is the results:- Shown as below

Version 1 Accident Accuracy 54%

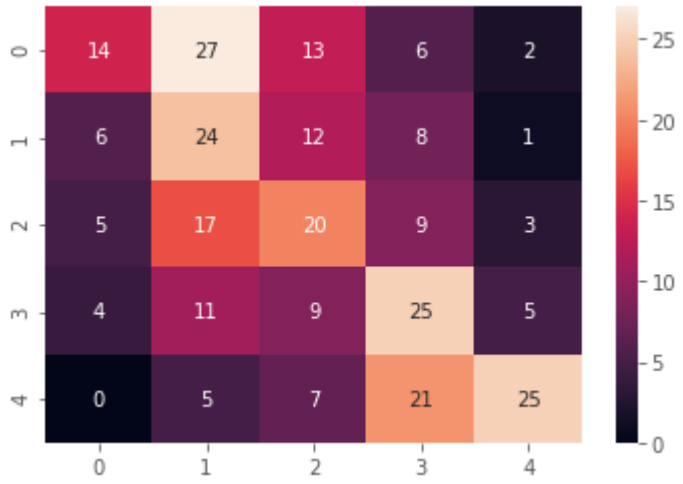
Version1: (preprocess_train_v1.py)					
Embedding	**Model type**	**Target feature**	**Accuracy**	**imblearn strategy**	**Observations**
Word2Vec	LSTM	Accident Level	54%	1. Do 90-10 split of train & validation samples 2. Use this 90% data to resample using imblearn 3. Split resampled data to 80-20 split. 4. Use this 80% to fit and 20% as validation during DL fit. 4. Use the saved 10% samples for evaluation	1. Since # of test samples is small, the exact accuracy is difficult to estimate. 2. DL model's accuracy increases & loss decreases over the epochs.



Version 1 Accident Accuracy 39%

Version2 (preprocess_train_v2.py):					
Embedding	**Model type**	**Target feature**	**Accuracy**	**imblearn strategy**	**Observations**
Word2Vec	LSTM	Accident Level	39%	1. Do 90-10 split of train & validation samples 2. Use this 90% data to resample using imblearn 3. Split resampled data to 80-20 split. 4. Use this 80% to fit and 10% as validation (from step#1) during DL fit. 4. Use the saved 20% samples for evaluation (created in step#3)	1. Since # of test samples is higher, the accuracy and model performance can be evaluated. 2. DL model's accuracy increases & loss decreases over the epochs.

Version snapshot 2



When we do the model evaluation the accuracy results are mentioned below

goal is to identify which class/category the new data will fall into

Classifier – It is an algorithm that is used to map the input data to a specific category.

Classification Model – The model predicts or draws a conclusion to the input data given for training, it will predict the class or category for the data.

Feature – A feature is an individual measurable property of the phenomenon being observed.

A classification report will give the following results, it is a sample classification report using Potential Accident Level data set

```
4/4 [==============================] - 2s 514ms/step
              precision    recall  f1-score   support

           0       0.58      0.70      0.63        84
           1       0.44      0.32      0.37        78
           2       0.43      0.31      0.36        84
           3       0.35      0.27      0.30        89
           4       0.43      0.67      0.52        84

    accuracy                           0.45       419
   macro avg       0.45      0.45      0.44       419
weighted avg       0.44      0.45      0.44       419
```

Usage of Confusion Matrix

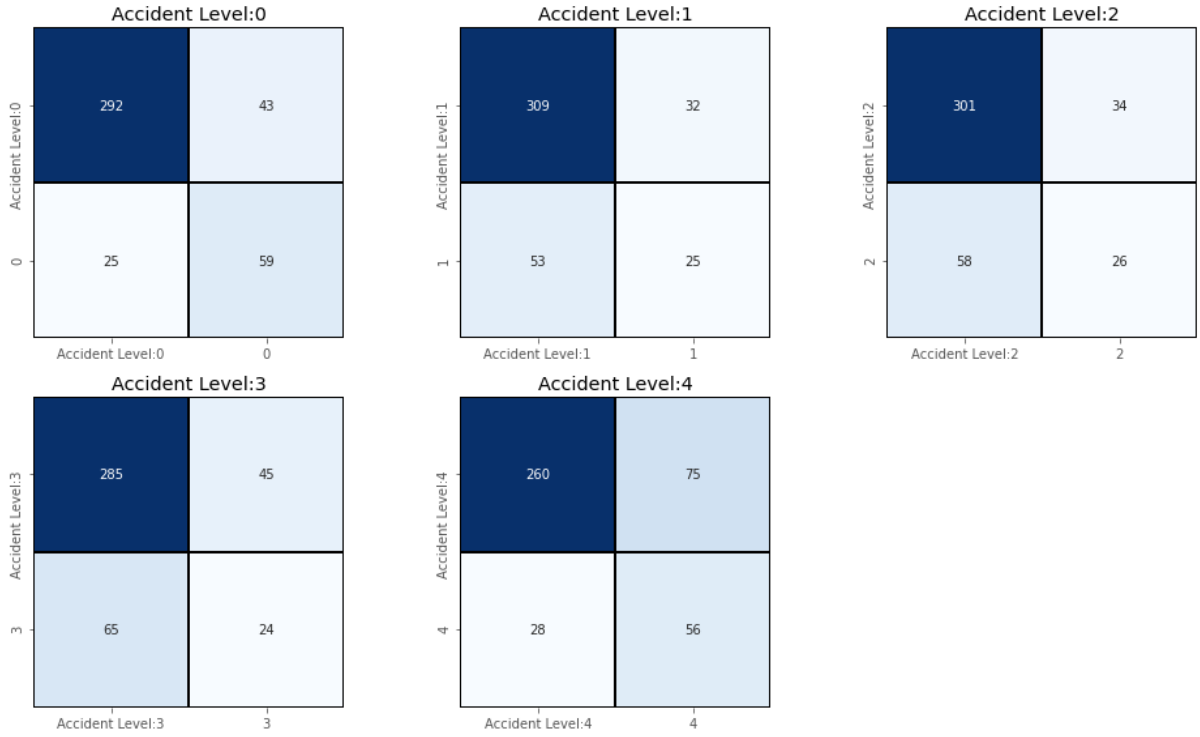
A much better way to evaluate the performance of a classifier is to look at the confusion matrix. The general idea is to count the number of times instances of class A are classified as class B. For example, to know the number of times the classifier confused images of 5s with 3s, you would look in the 5th row and 3rd column of the confusion matrix.

Each row in a confusion matrix represents an actual class, while each column represents a predicted class. Below is the output of the confusion matrix

```
array([[59,  7,  8,  6,  4],
       [12, 25, 14, 11, 16],
       [14, 11, 26, 10, 23],
       [14, 10,  9, 24, 32],
       [ 3,  4,  3, 18, 56]])
```

Usage of Multilevel Confusion Matrix.

It is used when there are two or more classes and the data we want to classify may belong to none of the classes or all of them at the same time, e.g. to classify on accident level are contained on an image.



Summary of Milestone 2:

In milestone 2 we used output of milestone 1 as its input. Then next step was preprocessing

Step 1: NLP pre-processing

Imported important libraries, defined some functions for cleaning the text. Then we split the 'Description' input column and 'Potential Accident Level' output column in training and testing data. Then changed the categorical values of 'Potential accident column' to numeric.

Step 2: Design ,Train and Test ML classifiers:

We used TF-IDF where, tfidfvectorizer transformed the text of description column to feature vectors which further was used as input to estimator. Then converted the test_vectors and train_vectors to dataframe, where columns will be features, those we got from feature vectors after applying tfidfvectorizer.

Then used H2oAutoML for automating ML workflow. Through it we got evaluation metrics for different algorithms and evaluation metrics separately for highest performing algorithm, which was GBM.

Then applied XGBoost algorithm, where got 33% accuracy. Further to improve accuracy, used weights and then got 42% accuracy.

Step 3: Feature Engineering

Created new features from accident description features using word2vec algorithm.

Word2vec algorithm converted the text data to numeric format(vectors).

Then used Imblearn technique, which is used to treat imbalanced dataset.

Step 4: Design, Train and Test LSTM Classifier

Here, used Long Short Term Memory classifier which has feedback connections to process single data points as well as entire sequences of data.

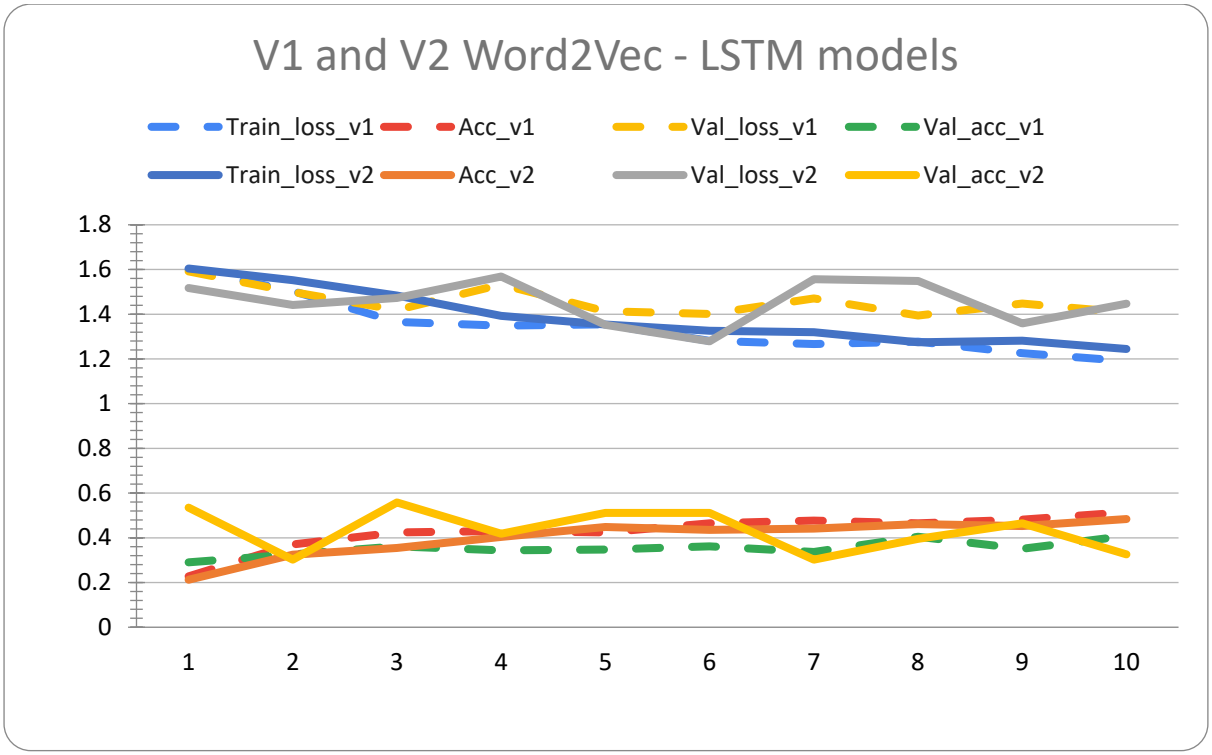
Step 5: Evaluation:

For training data, model loss decreased continuously, and model accuracy increased continuously. For testing data, model loss first decreased then increased and model accuracy first increased then decreased.

Model accuracy 39%. Using Glove and Work2Vec as shown below

Final Comparison of both Version 1 & 2 as mentioned below

	V1 Version					V2 Version				
Epoch	Timing_v1(sec)	Train_loss_v1	Acc_v1	Val_loss_v1	Val_acc_v1	Timing_v2(sec)	Train_loss_v2	Acc_v2	Val_loss_v2	Val_acc_v2
1	15	1.601	0.228	1.592	0.29	10	1.6045	0.2124	1.5176	0.5349
2	11	1.503	0.37	1.499	0.326	7	1.5526	0.3244	1.441	0.3023
3	11	1.365	0.424	1.419	0.362	7	1.4836	0.3548	1.4725	0.5581
4	11	1.349	0.43	1.533	0.344	7	1.3927	0.4041	1.5686	0.4186
5	11	1.355	0.424	1.414	0.348	7	1.3539	0.448	1.3524	0.5116
6	11	1.281	0.465	1.401	0.362	7	1.3263	0.4355	1.2789	0.5116
7	11	1.267	0.477	1.471	0.337	7	1.3198	0.4409	1.5566	0.3023
8	11	1.277	0.465	1.394	0.405	7	1.2744	0.4615	1.5485	0.3953
9	11	1.226	0.481	1.448	0.351	7	1.2816	0.4525	1.3591	0.4651
10	11	1.191	0.514	1.407	0.409	7	1.2446	0.4839	1.4468	0.3256



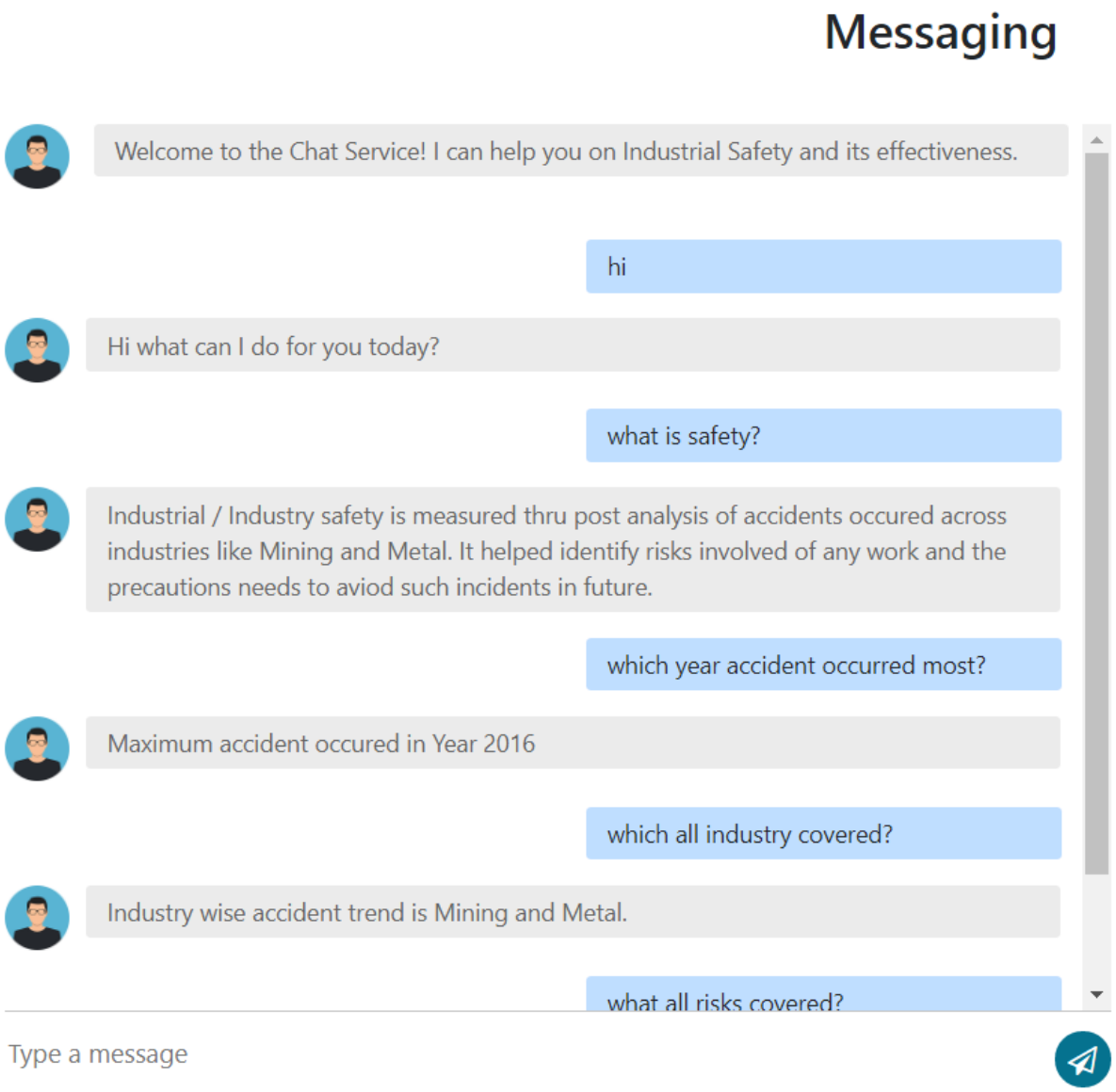
Visualizations- Milestone 3

UI Interactive Board chatbot

In the industrial design field of human–computer interaction, a user interface (UI) is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision-making process.

Step 3: Design a clickable UI based chatbot interface

Here we have created the flask application, HTML webpage, and deployed the model Webfrom.html It is the interface for Milestone 1&2. Here, we upload the data, apply pre-processing on it, so that we can train different models.



Index.html is the landing page of application, when we start the application, it is the first page that appears in front of user. It contain text box where we can write any accident description, then click on predict accident severity, and it will show the result of webpage coming from backend.

Using Flask , we created backend service (app.py)which will be responsible for all the services coming from webform.html. In backend it will listen to incoming request and then it will call proper API.

In app.py file we loaded the trained model and it contain multiple APIs:

- Home API will show home or initial webpage.

- after loading input data, predict API will be called.

- result API will serialize the result and result will displayed on webpage.

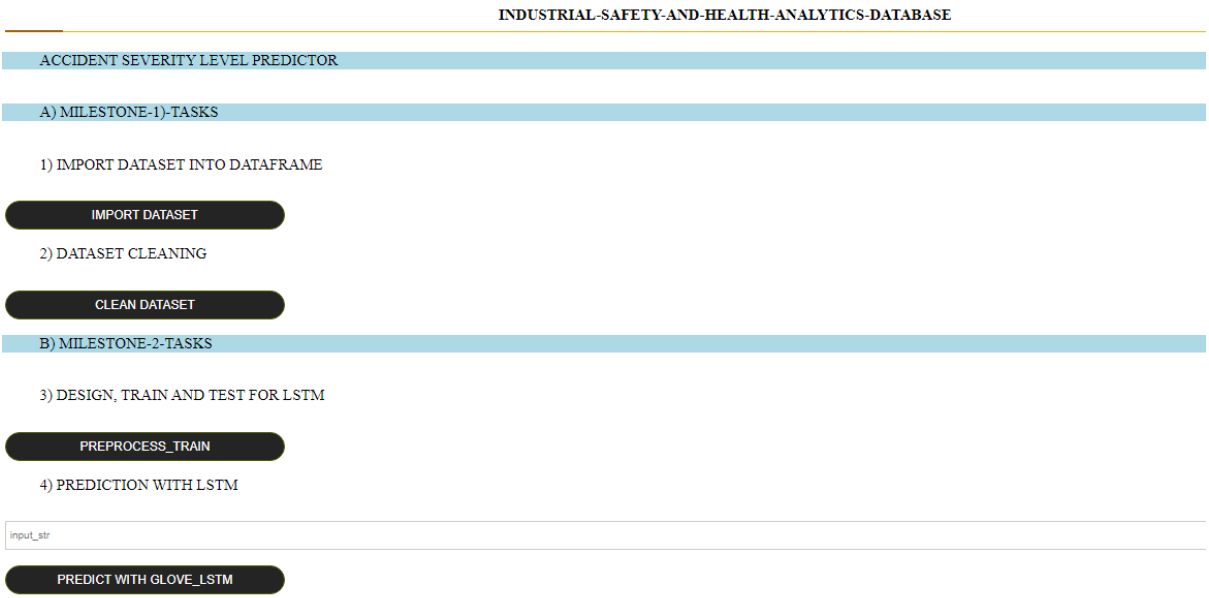
Using Flask , we created backend service (app.py)which will be responsible for all the services coming from webform.html. In backend it will listen to incoming request and then it will call proper API.

In app.py file we loaded the trained model and it contain multiple APIs:

- Home API will show home or initial webpage.

- after loading input data, predict API will be called.

- result API will serialize the result and result will displayed on webpage.



Summary of Milestone 3:

Milestone 3 is responsible for model deployment using flask and it is a interactive screen for the users to interact with the application directly from webpage

Limitations

With more detailed information such as machining data(ex. CNC, Current, Voltage) in plants, weather information, employee's personal data(ex. age, experience in the industry sector, work performance), we can clarify the cause of accidents more correctly.

Closing Reflections

In this project, we discovered that the main causes of accidents are mistakes in hand-operation and time-related factor. To reduce the occurrences of accidents, more stringent safety standards in hand-operation will be needed in period when many accidents occur.