NLP Capstone Project

Final Report

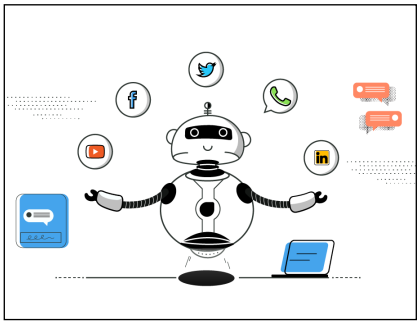
15 January 2023

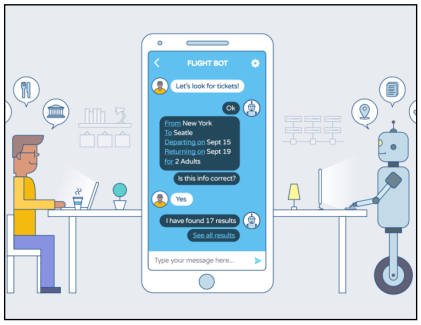
Vandana Kaarthik

Contents

1. Introduction

Chatbots are everywhere, be it a banking website, pizza store, e-commerce platforms or shopping stores. Chatbots provide real-time customer service assistance on a range of pre-defined questions related to the domain it is built on. It adapts natural human language and converses with humans in a human-like manner. Chatbots are the evolution of Question-Answering systems that employ natural language processing (NLP).





Chatbots help organisations increase their operational efficiency by automatically handling requests and queries, which in turn free employees to work on complex & higher value tasks, thereby saving time and effort.

1.1 Problem Statement

The objective of this Capstone project is to design a 'Text Classification model' to be given as an input to an AI based chatbot utility that can help professionals to highlight the safety risk. It is an imperative need for certain industries around the globe to comprehend why employees still suffer injuries/accidents in manufacturing plants.

1.2 Data set

The dataset comes from one of the biggest industries in Brazil - **IHM Stefanini.** The company has world-wide presence with 12 manufacturing plants located in 3 different countries. The dataset records accidents from these manufacturing plants.

The dataset consists of 10 columns that describes the various features of the accident. The description of these columns is as follows:

- ***Data***: The timestamp or time/date information
- ***Countries***: The country in which the accident occurred (anonymised)
- ***Local***: The city where the manufacturing plant is located (anonymised)
- ***Industry sector***: The sector the plant belongs to

- ***Accident level***: Labeled from I to VI, it registers how severe the accident actually was (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***: The gender of the person (male of female)
- ***Employee or Third Party***: If the injured is an employee or is a third party member
- ***Critical Risk***: Some description of the risk involved in the accident
- ***Description***: Detailed description of how the accident happened

1.2 Findings and Implications

The key finding in this dataset after data analysis was the fact that the 'mining industry sector' is the key contributor to the number of accidents. The mining industry has always had a reputation for being a risky business, with health risks that are varied and often quite serious, sometimes even leading to death. It was also noted during data analysis that the most severe accidents happened in the mining sector.

Following that, Text Classification models were built that identifies the potential accident level based on the description of the various accidents. The Final model can be used to predict how severe the accident could be with 91% level of confidence thereby fore-warning industries.

If the respective industries were fore-warned on the type of accidents that are happening frequently in their plants, they could reduce future accidents from happening by introducing strict safety legislation and protocol, as well as by using advanced safety equipment.

2. Data Analysis

The dataset is quite small with only 425 rows containing accident descriptions from the various manufacturing plants.

	Data	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party	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...

- The dataset consists of 425 rows and 10 columns.
- Some of the column names (Data, Local and Genre) in the dataset are misspelt.
- It is worth noting that there are no missing/null values in the dataset.
- However, there are 7 duplicate rows.
- The names of the Countries and the Localities of the various plants have been made anonymous intentionally.
- The Data has been collected from the beginning of 2016 to mid 2017. Since only half of 2017 is included in the dataset, the Date field does not add much value to the analysis

	Data	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party	Critical Risk	Description
count	425	425	425	425	425	425	425	425	425	425
unique	287	3	12	3	5	6	2	3	33	411
top	2017-02-08 00:00:00	Country_01	Local_03	Mining	I	IV	Male	Third Party	Others	During the activity of chuteo of ore in hopper...
freq	6	251	90	241	316	143	403	189	232	3

2.1 Data Cleansing

- The column names have been replaced as follows:

```
{'Data':'Date',
 'Genre':'Gender',
 'Countries': 'Country',
 'Local': 'Locality'},
```

- The duplicate values in the dataset were dropped
- The 'Date' column which was of type 'object' was converted to type 'datetime' for feature engineering purposes

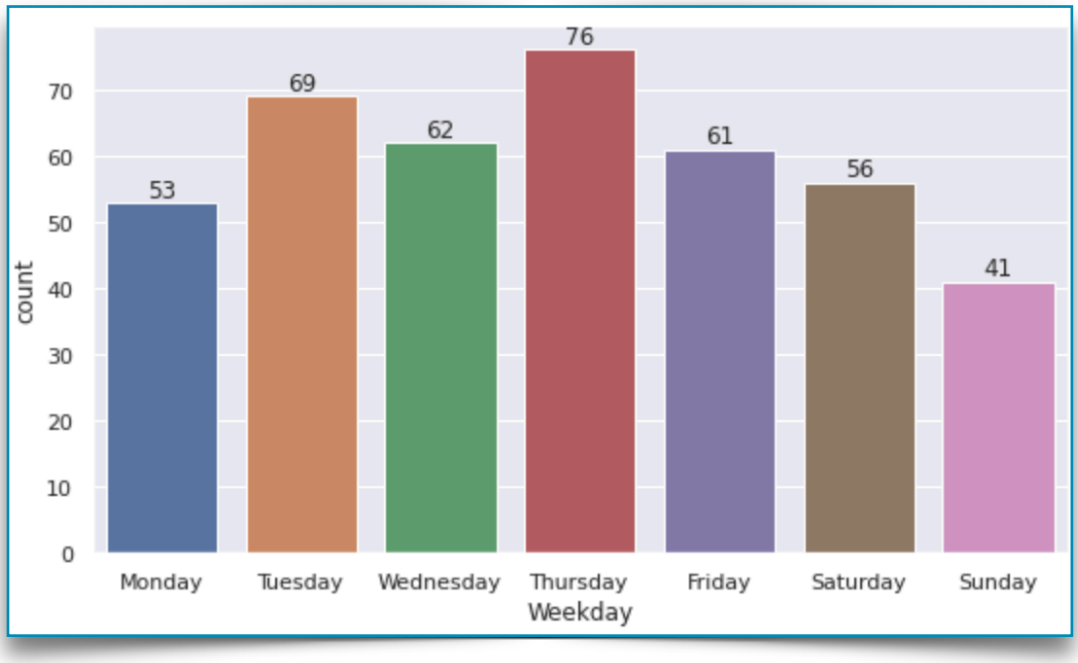
2.2 Exploratory Data Analysis (EDA)

2.2.1 *Time period during which the observations have been taken*

```
Start Date: 2016-01-01 00:00:00
End Date: 2017-07-09 00:00:00
```

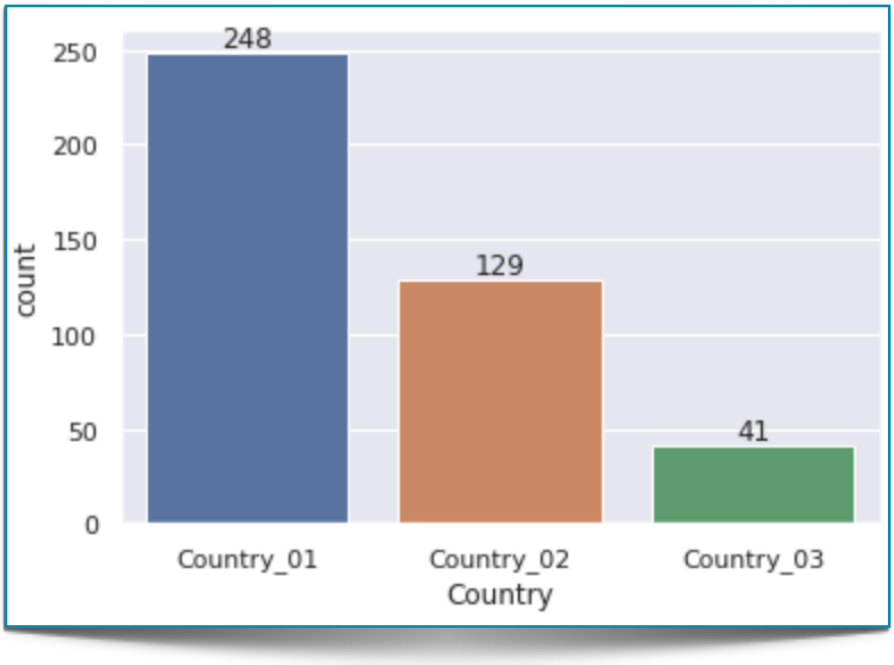
The data observations span from 1st Jan, 2016 to 9th Jul, 2017. Even though, data has been collected for the entire year of 2016, only the 1st half of the observations collected in 2017 has been provided in this dataset. Hence, 'month' or 'year' will not provide any valuable insight for exploratory data analysis. Therefore, only the 'weekday' is considered for further analysis. Hence, a new column 'Weekday' - the day of the week when the accident was observed - was added to the dataset

Distribution of Weekday



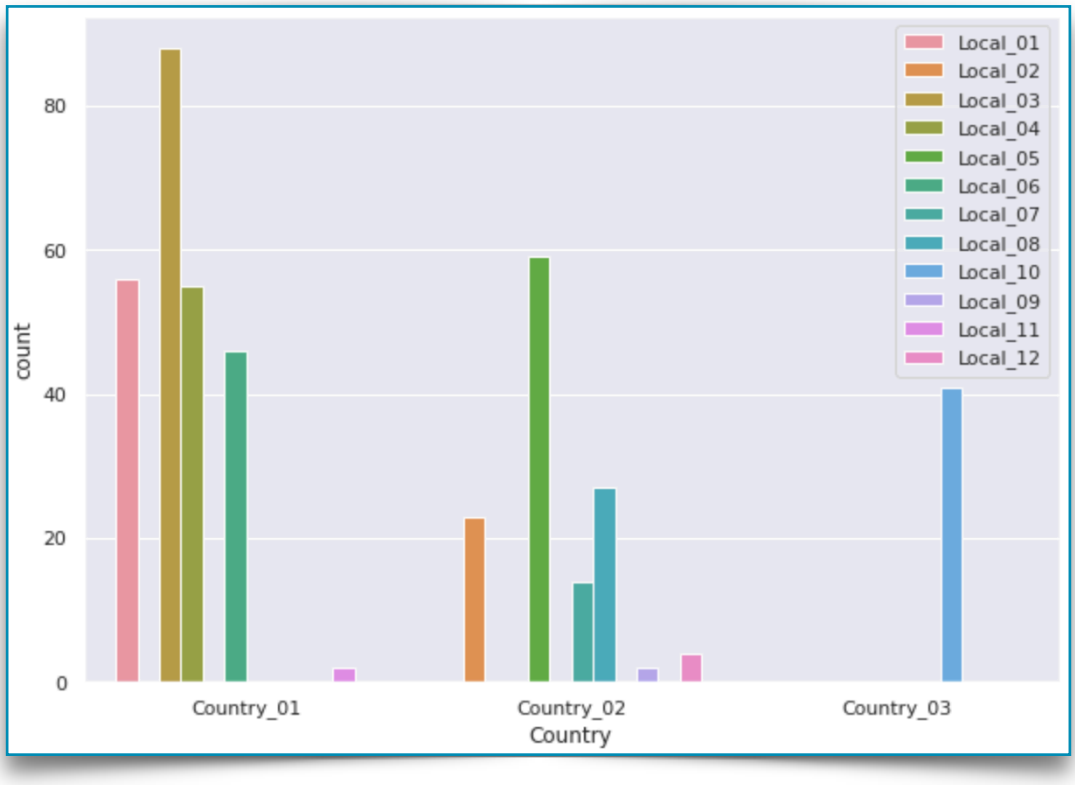
The number of accidents peaks during the mid-week and tapers down as the week-end approaches.

2.2.2 *Distribution of the Country where the accident occurred*



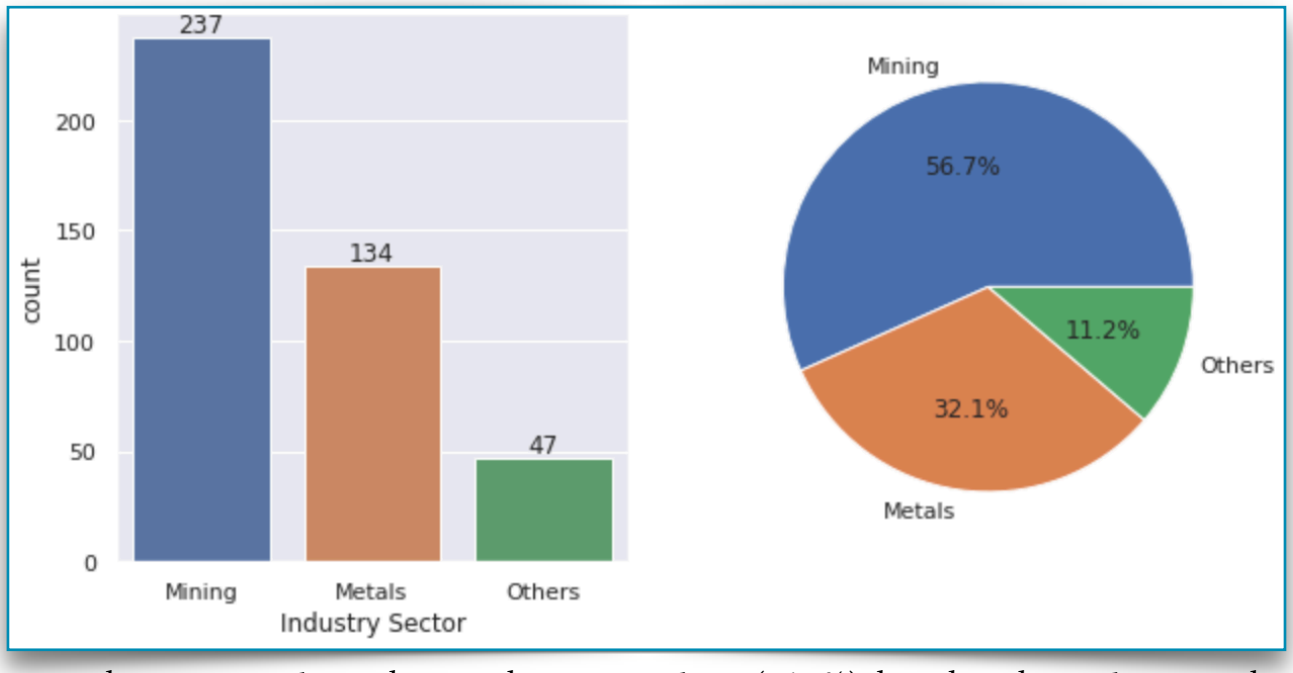
From the above plot, most of the accidents recorded in the dataset happened in Country_01. Further investigation has been done as to why Country_01 has the most number of accidents.

2.2.3 *Checking which Locality belongs to which Country*



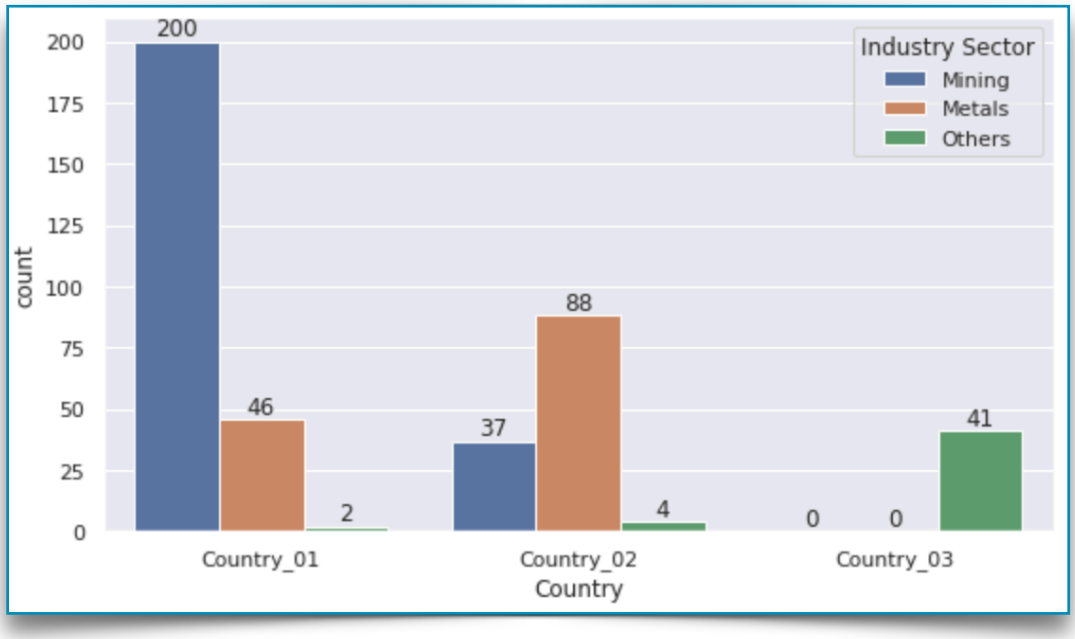
Locality-03 has the maximum number of accidents registered. Since, Locality-03 is present in Country-01, this explains why Country-01 has the maximum number of accidents among the 3 countries.

2.2.4 *Distribution of the 'Industry Sector'*



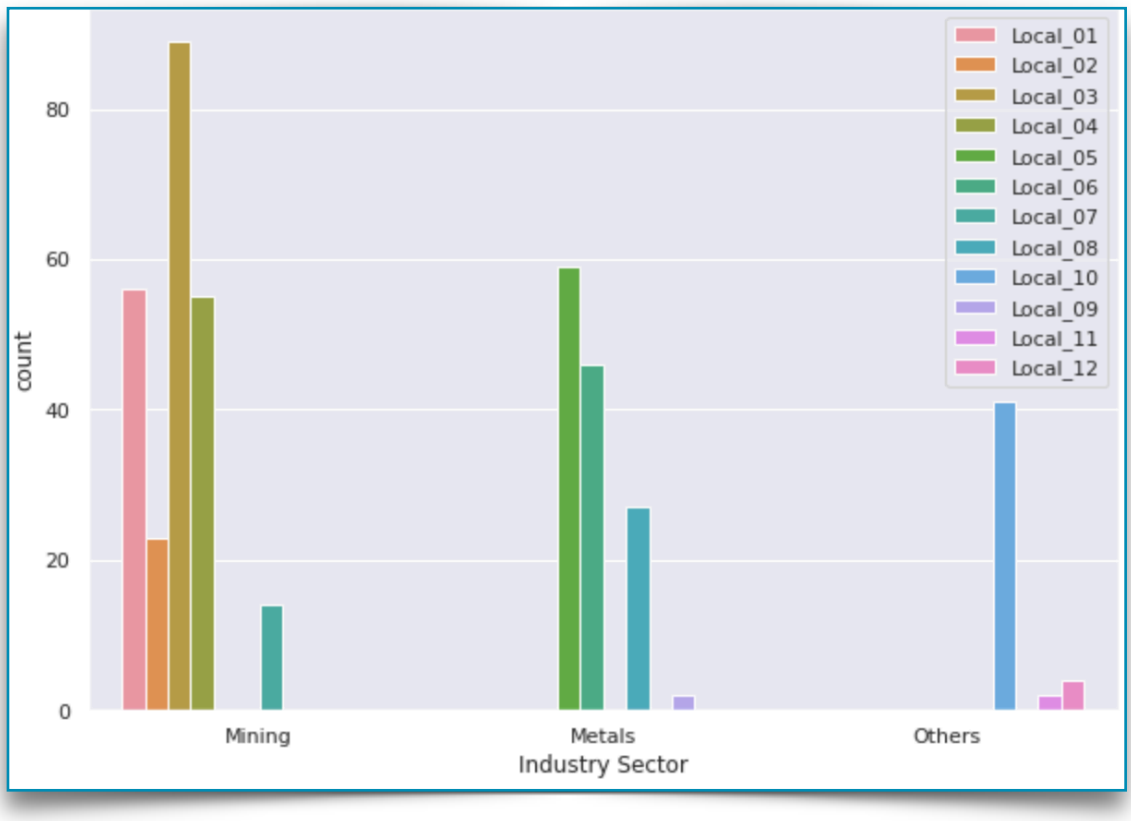
The Mining industry has much more accidents (56.7%) than the other industries. The mining industry has a reputation for being a risky business, with health risks that are varied and often quite serious. By introducing strict safety legislation and protocol, as well as using advanced safety equipment, it is possible to reduce the accidents pertaining to this sector.

2.2.5 *Distribution of the 'Industry Sector' by Country*



Most of the mines are situated in Country_01 and mining, being the riskiest of the industry sectors, has recorded the maximum number of accidents in the dataset. This clearly explains why Country_01 tops the list of the most accident-prone country. Since, Locality-03 is located in Country-01 and has recorded the maximum number of accidents, it is very much possible that Locality-03 is a 'mining' industry.

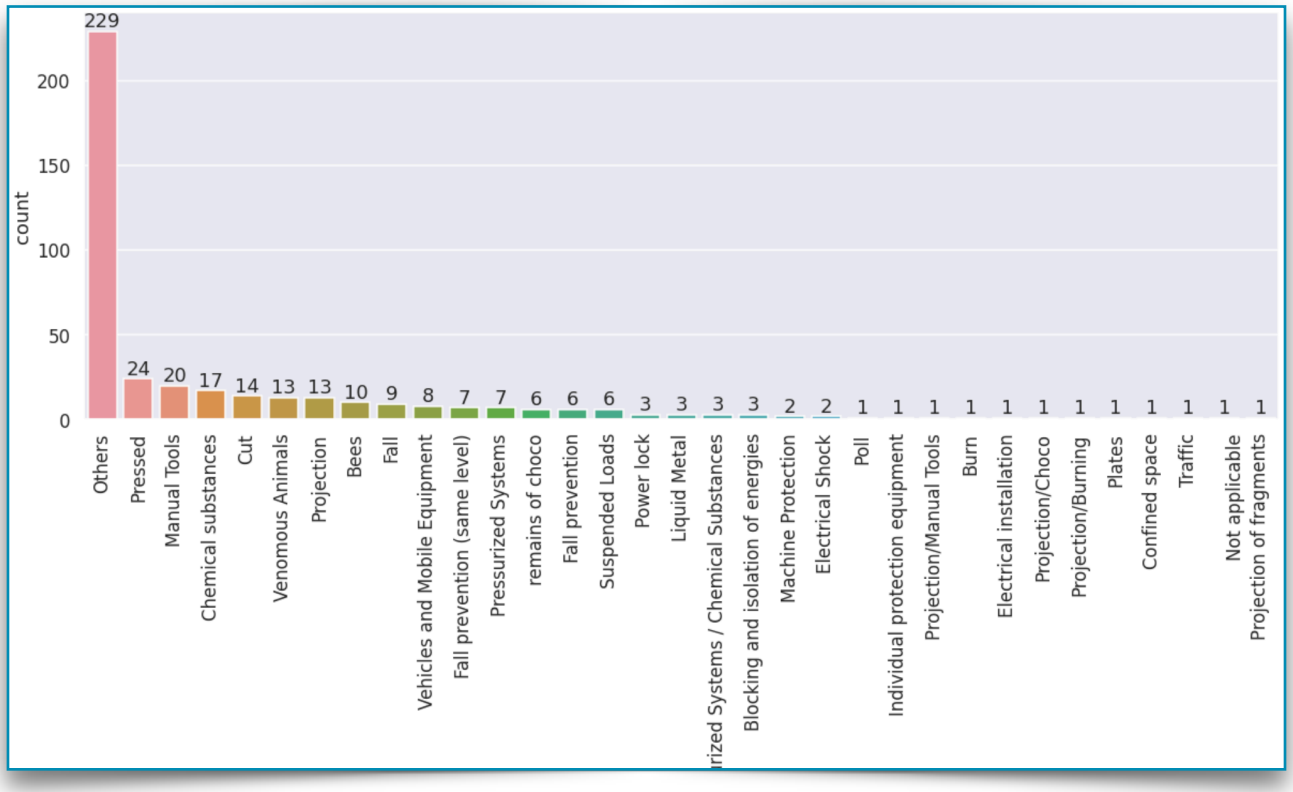
2.2.6 *Which locality belongs to which industry?*



From the above plot, it is clear that

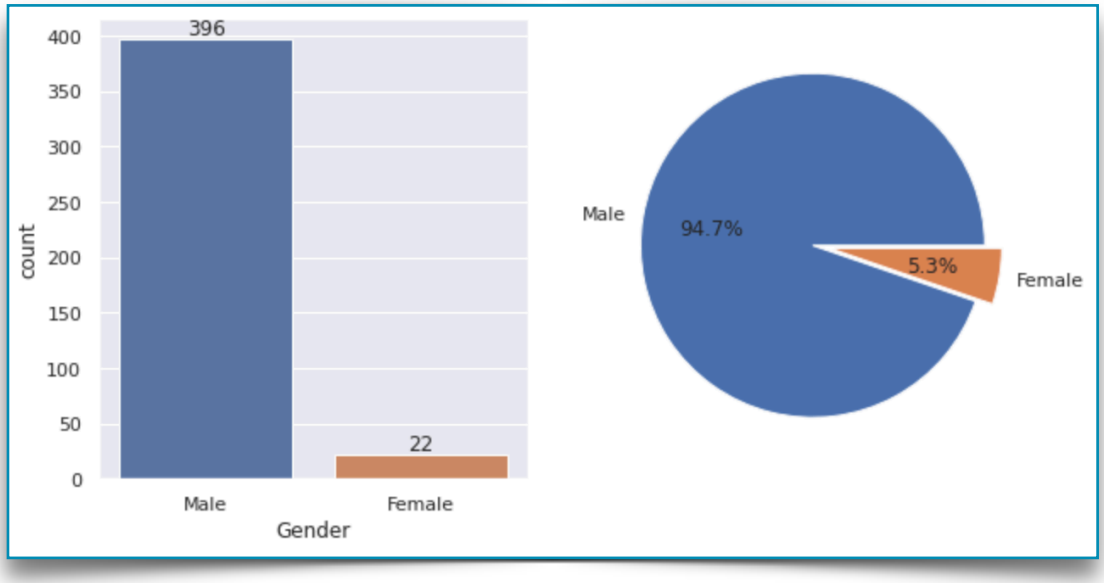
1. Locality-01, Locality-02, Locality-03,
Locality-04 and Locality-07 belongs to the Mining industry.
2. Locality-04, Locality-05, Locality-08 and Locality-9 belong to the Metals industry.
3. Locality-10, Locality-11 and Locality-12 belongs to Others

2.2.7 *Distribution of 'Critical Risk*



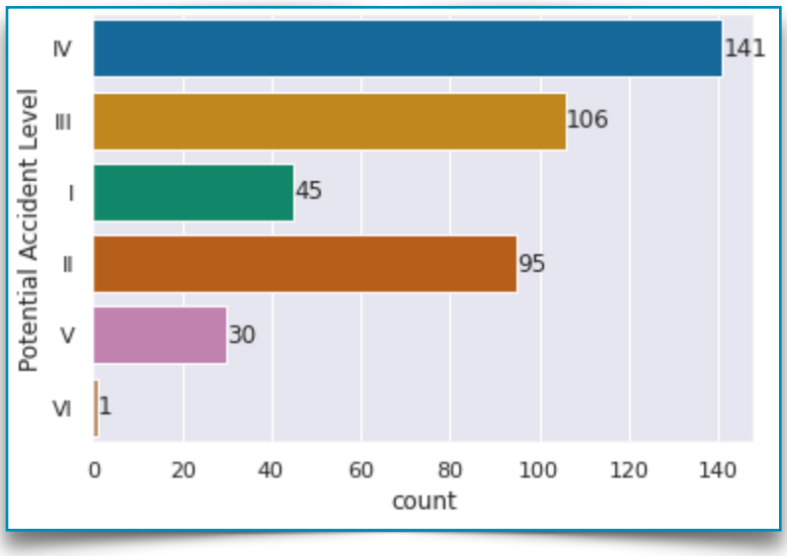
Several factors in 'Critical Risk' have only 1, 2 or 3 incidences of occurrence. The data is heavily imbalanced towards 'Other' factor. It is not clear what this 'Other' factor indicates. It looks like critical risk factors that have unknown causes are categorized as 'Other'.

2.2.8 *Distribution of Gender*



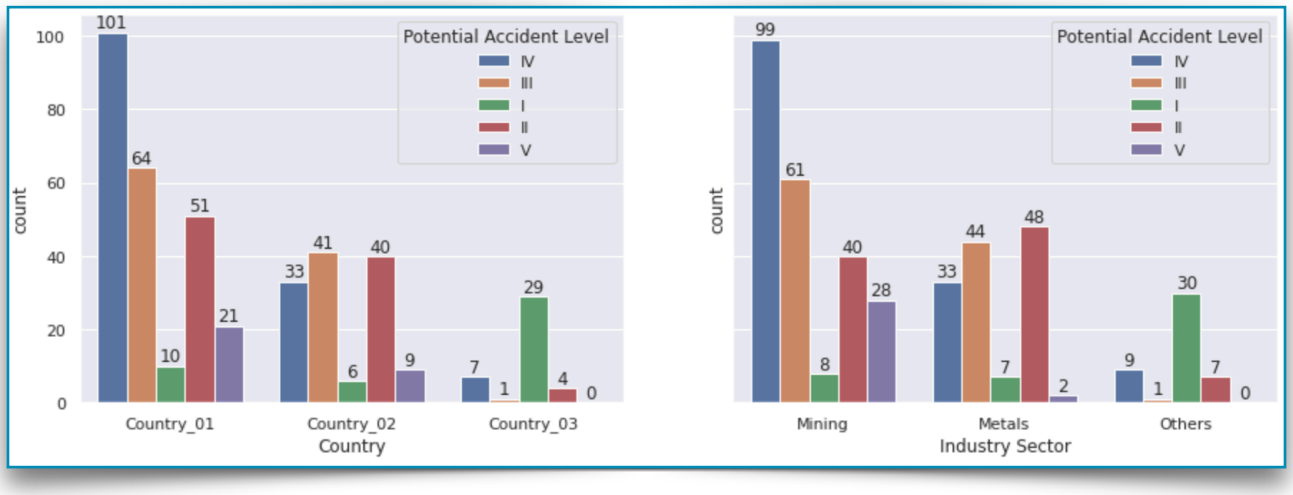
The male seem to be the most affected, accounting for almost 95% of the accidents. This could also be the case where men are more employed than women as mining and metal are primarily male-driven industries.

2.2.9 *Distribution of the target variable - 'Potential Accident Level'*



The distribution of the target classes varies significantly, implying some imbalance in the dataset. Since the Potential Accident Level 'VI' has only one sample, it has been removed, as it is insufficient for training the model.

2.2.10 *Distribution of the 'Potential Accident Level' by Country and Locality*



Severe accidents, of the order of Level IV and Level V, have been recorded in Country_01 and in the Mining industry. Most of the mines are present in Country_01, and mining being a very risky industry has significantly contributed to severe accidents owing to its hazardous working conditions.

2.3 Text Preprocessing (NLP) - with spaCy

SpaCy was chosen over NLTK for text preprocessing tasks in this Capstone project.

Why spaCy?

spaCy is an Industrial strength natural language processing toolkit. It is blazing fast and excels at large-scale information extraction tasks. It is written from the ground up in carefully memory-managed Cython language. spaCy is specifically designed for production use and helps build and train powerful NLP pipelines and package them for easy deployment.

In the five years since its release, spaCy has become an industry standard with a huge ecosystem. spaCy v3.0 uses transformer-based pipelines that bring spaCy's accuracy right up to **current state-of-the-art**.

Some of the text preprocessing tasks performed using spaCy in this project include:

- Tokenization
- Lemmatization
- Removal of Stopwords
- Removal of special characters, numbers and punctuation

spaCy's large English language pipeline 'en_core_web_lg' that is optimised for both CPU and GPU was used for this project.

A sample of the 'unprocessed' and 'cleaned up' text data

```
# Unprocessed text data

indus_safety.Description[4]
```

```
'Approximately at 11:45 a.m. in circumstances that the mechanics Anthony (group leader), Eduardo and Eric Fernández-injur
ed-the three of the Company IMPROMEC, performed the removal of the pulley of the motor of the pump 3015 in the ZAF of M
arcy. 27 cm / Length: 33 cm / Weight: 70 kg), as it was locked proceed to heating the pulley to loosen it, it comes out and fal
ls from a distance of 1.06 meters high and hits the instep of the right foot of the worker, causing the injury described.'
```

```
# Cleaned up text data

indus_safety['Preprocessed Description'][4]
```

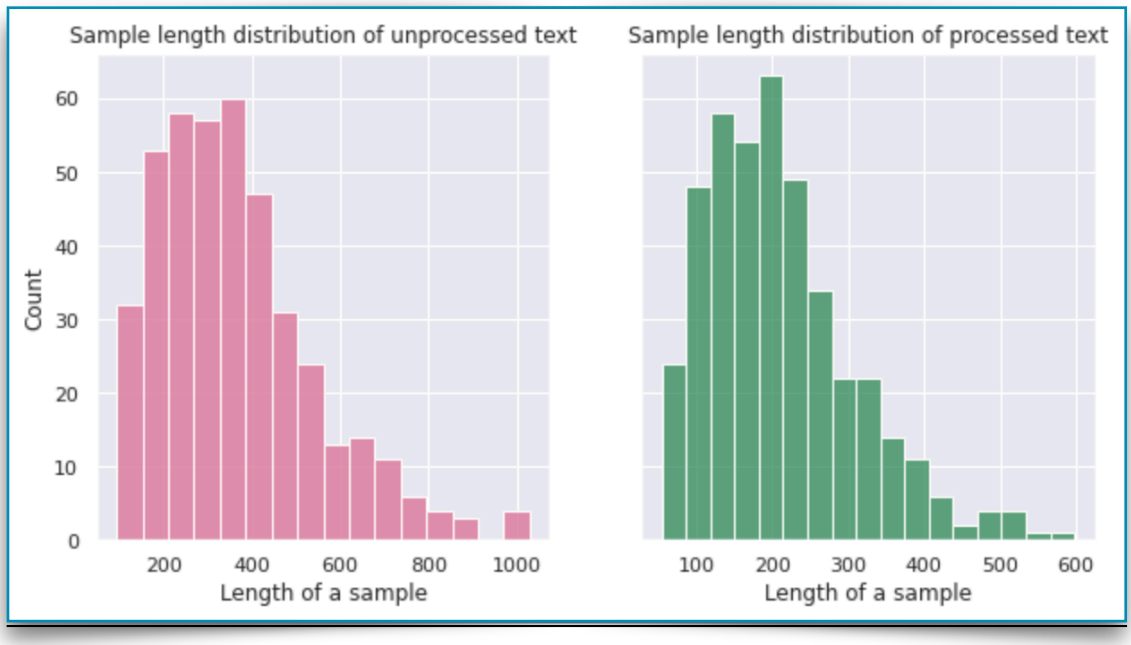
```
'approximately circumstance mechanic anthony group leader eduardo eric fernández injure company impromec perform remo
val pulley motor pump zaf marcy cm length cm weight kg lock proceed heat pulley loosen come fall distance meter high hit in
step right foot worker cause injury describe'
```

2.4 Text Analytics

2.4.1 *Number of words per sample for Industrial safety Accident Description*

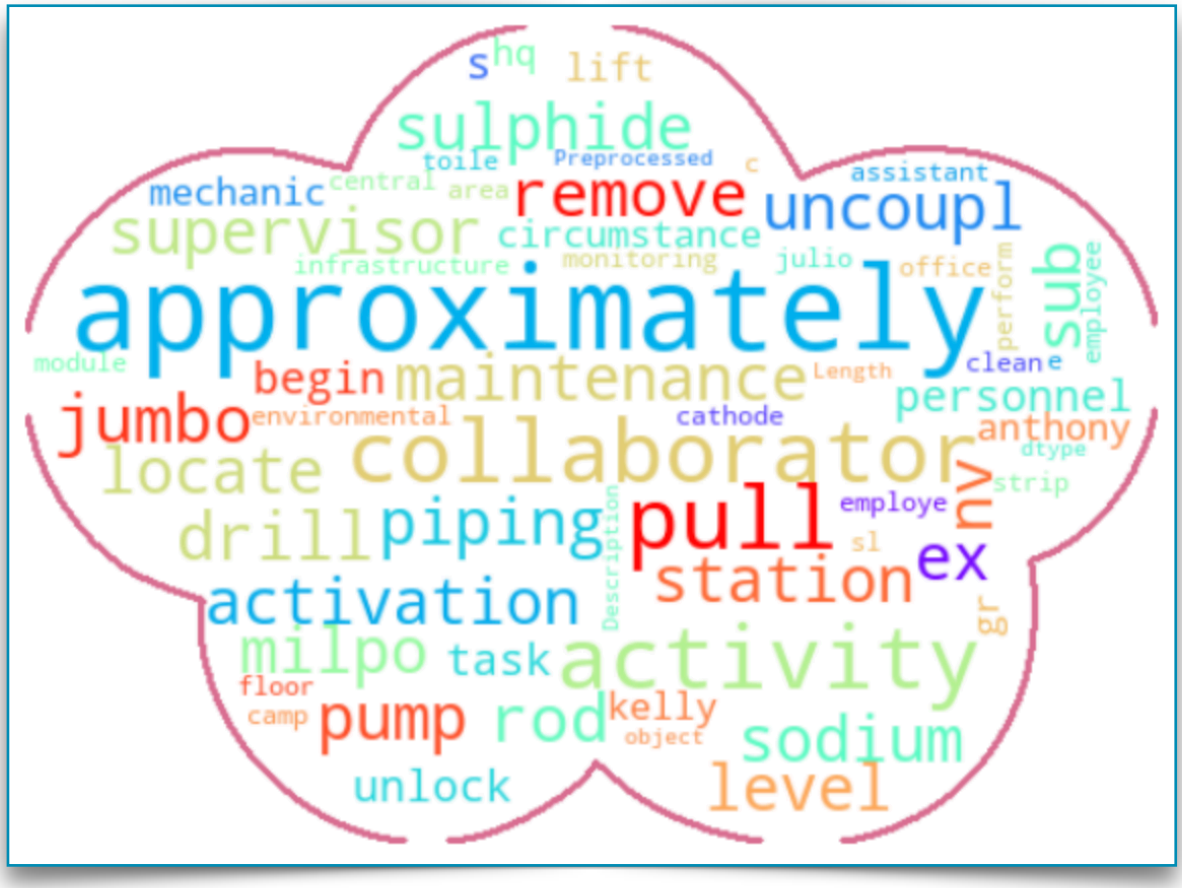
```
Number of words per sample:
Description: 60.0
Preprocessed Description: 28.0
```

2.4.2 *Sample Length Distribution of Industrial safety Accident Description*



The sample length distribution of both unprocessed and processed text are very similar.

2.4.2 *Word Cloud with Image mask*



The word cloud was constructed using the 'preprocessed description' in the industrial safety dataset. A 'cloud image' mask was used to depict the word cloud. This word cloud uses 'bilinear' interpolation to smoothen the image.

3. Data Preparation for ML classifiers

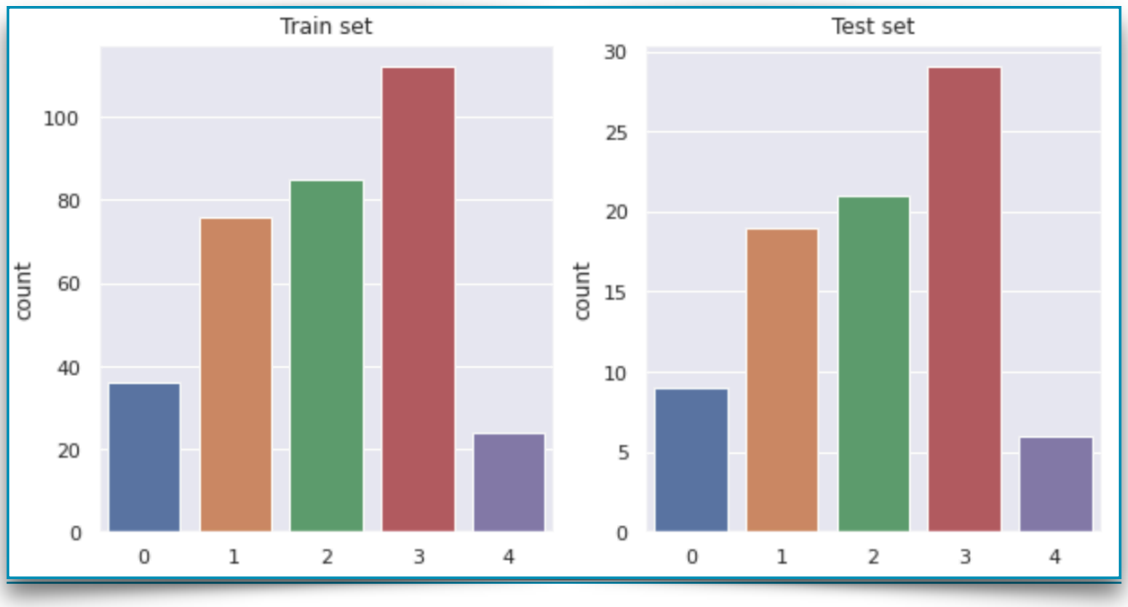
3.1 Target variable

As industrial safety was of prime consideration, the 'Accident Level', the 'Potential Accident Level' and the 'Critical Risk' were considered as target variables. However, for 'Critical Risk', the data was heavily imbalanced towards 'Other' factor and has insufficient data points to train the model. Almost, half the features have only 3 or fewer data points. Also, it is not clear what this 'Other' factor indicates. It looks like critical risk factors that have unknown causes are categorized as 'Other'. Hence, 'Critical Risk' was dropped. On similar criteria, 'Accident Level', was also dropped. Compared to 'Accident Level', 'Potential Accident Level' has somewhat balanced values. Therefore, 'Potential Accident Level' was taken as the target variable for this Capstone project.

Initially, **data augmentation using SMOTE** was considered for this dataset. But, SMOTE does not seem to work for text augmentation. Converting the entire text dataset into numeric data using one hot encoding leads to 'sparsity' and is not an efficient way of solving the problem. Also, the results produced after augmenting the data with SMOTE does not significantly impact performance. Owing to the above disadvantages, SMOTE was eventually given up for text augmentation. Instead, 'synonym replacement' with '**nlpaug'** library was considered for text augmentation as it is the most effective technique and is widely used in the industry.

Label encoding was done using 'cat.codes' instead of 'LabelEncoder()' as it is much faster to use the pandas category datatype. Internally 'cat.codes' uses a hash table, which explains the speed of its execution. On the other hand, LabelEncoder uses a sorted search algorithm, which slows it down for larger datasets.

Distribution of the target variable in the train and test datasets



Upon comparing the above 2 plots, it can be observed that the distribution of the target variable is the same for both the train and test datasets after test-train split

3.2 Vectorization techniques

Raw text data cannot be fed directly to the machine learning algorithms as most of them expect numerical feature vectors with a fixed length rather than the raw text documents with variable length. Hence, Text Feature extraction, also known as, Vectorization is of paramount importance for text classification.

V**ectorization** refers to the general process of turning a collection of text documents into numerical feature vectors.

Word Embeddings, such as, GloVe capture the contextual relationship of words in a sentence and since, ML models are too weak to capture this relationship, Glove vectors are generally used with Sequential models. Hence, two vectorization techniques - BOW and TFIDF - which are well suited for ML algorithms have been considered.

3.2.1 BOW - Count Vectorizer

In Bag-of-Words representation, the vector's values represent the frequency with which the word/words appears in a given text document. In this project, in addition to the unigrams (individual words), bigrams are also considered to preserve some of the ordering information.

```
Length of the vocabulary generated using CountVectorizer: 10138
```

A Few of the unigrams and bigrams generated

```
y(['abb', 'abb furnace', 'abdomen', 'abdomen left', 'able',
'able position', 'able remove', 'abrupt', 'abrupt contact',
'abrupt movement', 'abruptly', 'abruptly drop',
'abruptly imprison', 'abruptly withdraw', 'absorb', 'absorb small',
'absorbent', 'absorbent cloth', 'abutment', 'abutment said'],
```

Top 10 frequently occurring words in BOW

```
cause       157
hand        146
employee    138
right       125
left        115
operator    101
time         96
activity     91
injury       89
moment       82
```

3.2.2 *TF-IDF Vectorizer*

TF-IDF Vectorizer not only focuses on the frequency of words present in the corpus but also provides the importance of the words. We can then remove the words that are less important for analysis, hence making the model building less complex by reducing the input dimensions.

```
Length of the vocabulary generated using tfidf: 2103
```

A Few of the vocabulary generated

```
y(['abb', 'abdomen', 'able', 'abrupt', 'abruptly', 'absorb',
'absorbent', 'abutment', 'acc', 'access', 'accessory', 'accident',
'accidentally', 'accidently', 'accommodate', 'accompany', 'accord',
'accretion', 'accumulate', 'accumulation'], dtype=object)
```

Top 10 words that have the highest TFIDF scores

employee	0.039216
hand	0.035114
cause	0.034552
right	0.031722
left	0.029365
operator	0.027163
activity	0.026310
hit	0.026151
injury	0.023774
time	0.023719

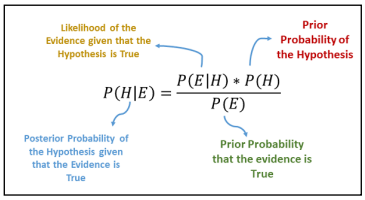
It is worth noting that the dimension of the vectors has reduced significantly in TFIDF when compared to BOW.

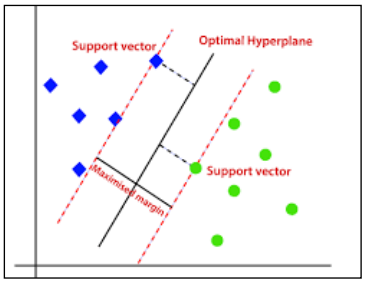
---

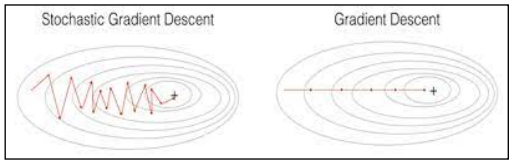
4. Model building - ML models

The most popular supervised machine learning algorithms used in NLP are:

- **Naive Bayes** - are a collection of classification algorithms based on Bayes' Theorem
- **Support Vector Machines** - SVM is one of the most popular Supervised Learning algorithms, which is used for Classification. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category. This best decision boundary is called a hyperplane.
- **Linear SVM with SGD (Stochastic Gradient descent)** - SGD Classifier is a linear classifier optimized by the SGD



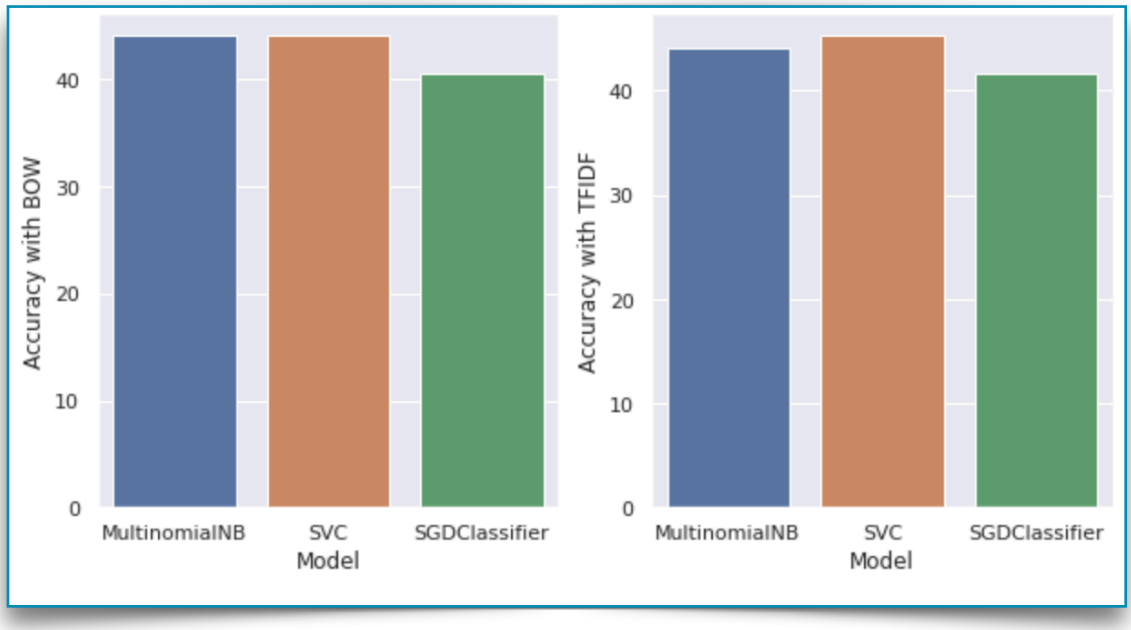




4.1 Performance of ML models

	Model	Accuracy with BOW	Accuracy with TFIDF
0	MultinomialNB	44.05	44.05
1	SVC	44.05	45.24
2	SGDClassifier	41.67	42.86

Performance plots



So far, SVM model has given the best performance of 45.24% when trained with TF-IDF vectors.

4.2 Model Performance improvement

4.2.1 Hyper-parameter tuning using Grid Search cross validation

The following hyper parameters were tuned using Grid Search cross validation:

- **SVM:** {'C':[0.1, 1], 'gamma':[0.0001, 0.001, 0.01, 0.1]}

```
Best params: {'C': 1, 'gamma': 0.0001}
Best score: 0.39023066485753055
Test accuracy using the best hyperparameters: 0.4523809523809524
```

- **SGD:** {'penalty':['l1', 'l2', 'elasticnet'], 'loss':['hinge', 'squared_hinge']}

```
Best params: {'loss': 'hinge', 'penalty': 'l2'}
Best score: 0.4293532338308458
Test accuracy using the best hyperparameters: 0.4166666666666667
```

- **Multinomial Naive Bayes:** {'alpha':[1.0, 0.9, 0.8]}

```
Best params: {'alpha': 0.9}
Best score: 0.38127544097693356
Test accuracy using the best hyperparameters: 0.44047619047619047
```

After tuning the hyper-parameters of the models using Grid Search cross validation, SVM model with 'linear' kernel and with 'C'=10 and 'gamma'=0.001 has given the best performance of 45.24% when trained with TF-IDF vectors.

Since, linear models have been widely regarded as one of the best text classification ML algorithms, opting for a 'linear' kernel has made SVM perform better than the other algorithms.

Performance metrics with BOW vectors

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.56      0.71         9
           1       0.41      0.47      0.44        19
           2       0.25      0.38      0.30        21
           3       0.58      0.48      0.53        29
           4       1.00      0.17      0.29         6

    accuracy                           0.44        84
   macro avg       0.65      0.41      0.45        84
weighted avg       0.53      0.44      0.45        84

Accuracy: 0.4405
```

Performance metrics with TFIDF vectors

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.56	0.71	9
1	0.38	0.32	0.34	19
2	0.33	0.33	0.33	21
3	0.46	0.66	0.54	29
4	1.00	0.17	0.29	6
accuracy			0.45	84
macro avg	0.63	0.41	0.44	84
weighted avg	0.51	0.45	0.45	84

Accuracy: 0.4524

In general, since ML models are quite weak for NLP tasks when compared to the other state-of-the-art DL models, the performance has not been that great. Also, since the dataset is very small, the models could not be trained sufficiently, even after hyper parameter tuning. Hence, 'Target variable balancing' using 'Text Augmentation' was considered for improving the performance of the ML models.

However, among the ML models used for training the given dataset, SVM with a linear kernel, has outperformed the other models.

4.2.2 Balancing the Target Variable with NLPAug - using 'Wordnet'

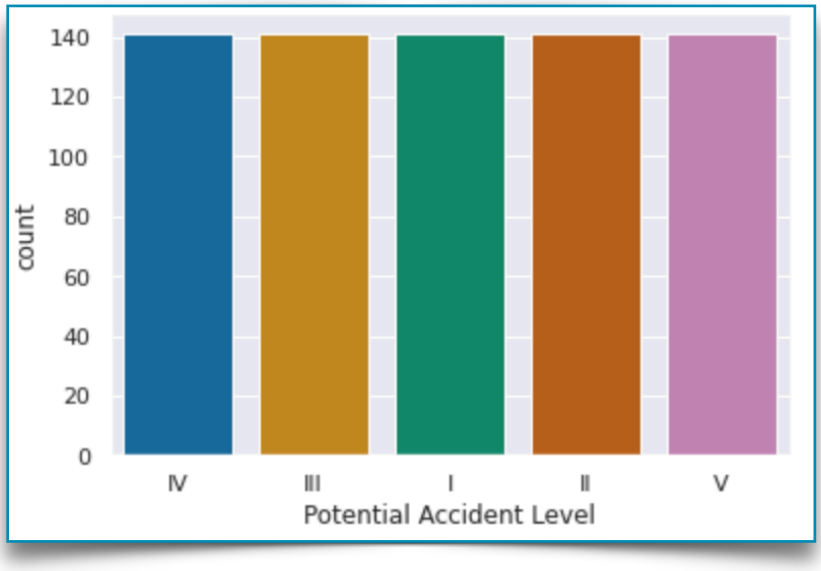
In Natural Language Processing (NLP), it is generally hard to augment text due to the high complexity of the language. However, **'NLPAug'** is a python package that lets us do all the data augmentation easily and and the level of augmentation can be tuned using various arguments.

In general, NLPAug offers three types of augmentation:

- Character level augmentation
- Word level augmentation
- Sentence level augmentation

For this capstone project, 'synonym replacement' at the 'word level' with the '**nlpaug**' library using 'Wordnet' is used for target balancing.

Distribution of the target variable after Target Balancing



Performance of the ML Models with augmented text data

	Model	Accuracy with BOW	Accuracy with TFIDF
0	MultinomialNB	78.01	72.34
1	SVC	74.47	75.89
2	SGDClassifier	71.63	75.18

After balancing the target using text augmentation with 'nlpaug' library, there has been a tremendous improvement in the performance of the ML models.

With BOW vectors, 'Multinomial Naive Bayes' has given the best performance of 78%. This is a significant improvement over 44.05% after hyperparameter tuning. Also, the precision, the recall and the F1 scores are 79%, 78% and 76% respectively. This clearly indicates that the model is well balanced for multi-class classification.

Performance metrics with BOW vectors

Classification Report:

	precision	recall	f1-score	support
0	0.78	1.00	0.88	28
1	0.85	0.39	0.54	28
2	0.76	0.68	0.72	28
3	0.65	0.83	0.73	29
4	0.93	1.00	0.97	28
accuracy			0.78	141
macro avg	0.79	0.78	0.76	141
weighted avg	0.79	0.78	0.76	141

Accuracy: 0.7801

With TFIDF vectors, 'SVM with a linear kernel' has given the best performance of 76%. Again, this is also a significant improvement over 45.24% after hyperparameter tuning. Also, the precision, the recall and the F1 scores are 76%, 76% and 75% respectively, indicating a well balanced model.

Performance metrics with TFIDF vectors

Classification Report:

	precision	recall	f1-score	support
0	0.87	0.96	0.92	28
1	0.67	0.50	0.57	28
2	0.64	0.64	0.64	28
3	0.65	0.76	0.70	29
4	0.96	0.93	0.95	28
accuracy			0.76	141
macro avg	0.76	0.76	0.75	141
weighted avg	0.76	0.76	0.75	141

Accuracy: 0.7589

In general, ML models are quite weak for NLP tasks when compared to the other state-of-the-art DL models. Also, since the dataset was initially very small, the models could not be trained sufficiently. However, after synonym augmentation using the 'nlpaug' library for balancing the target, the performance of the models improved tremendously.

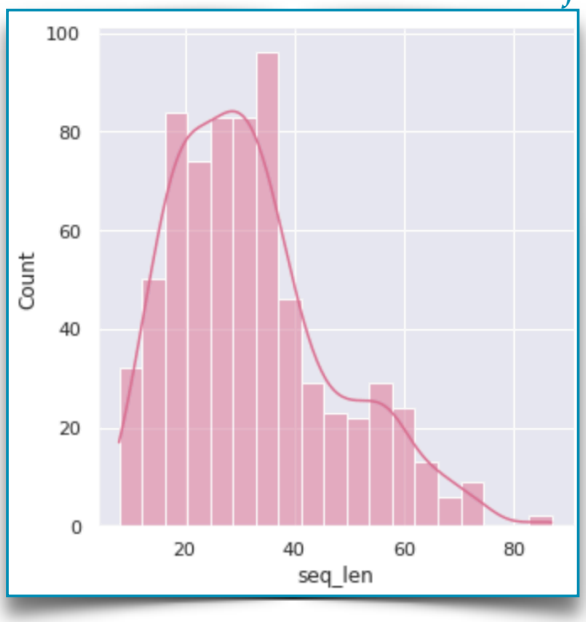
5. Data Preparation for DL classifiers

As specified in the problem statement, the preprocessed and balanced dataset from Milestone-1 was used as the input to the DL classifiers.

5.1 Text preprocessing for Sequential models

For Sequential models, word order is critical and hence, needs to be preserved. Models such as RNNs/LSTMs/CNNs can infer meaning from the order of words in a sample. For these sequential models, we represent the text as a sequence of tokens, thus preserving the order.

Distribution and Statistics of sequence length



count	705.000000
mean	32.295035
std	14.737795
min	8.000000
25%	21.000000
50%	30.000000
75%	40.000000
max	87.000000

The Distribution is slightly skewed on the right and the median sequence length of the sentences seems to be around 30 words.

Tokenization

Text can be represented as either a sequence of characters, or a sequence of words. Word-level representation of text provides better performance than character tokens and is also the general norm that is followed in the industry. Hence, word-level text representation was used for this Capstone project.

The 'Tokenizer' utility class from 'Keras Text Preprocessing' library allows to tokenize a text corpus, by turning each text into a sequence of integers (each integer being the index of a token in a dictionary). By default, all punctuation is removed, turning the texts into space-separated sequences of words.

A text sample turned into sequence of indices

```
'collaborator report street holding left hand volumetric balloon slip place hand ground volumetric balloon end breaking cause s
mall wound left hand'
```

```
array([  17,   91,  972, 1879,    7,    3,  973,  974,   70,   33,    3,
         99,  973,  974,   72, 1880,    2,  100,  164,    7,    3])
```

Sequence Padding

Sequential models expect all sequences to be of the same length. Hence, each sample sequence should be either padded or truncated to bring it to the same length. Hence, the 'pad_sequences' method from 'Keras Preprocessing' was used to make all the sequences have the same length by either adding zeros on the smaller texts or truncating the bigger ones. In this Capstone project, since the median sequence length of the sentences seems to be around 30 words, the 'maximum sequence length' has been truncated/padded to '30'.

A sample sequence padded with zeros

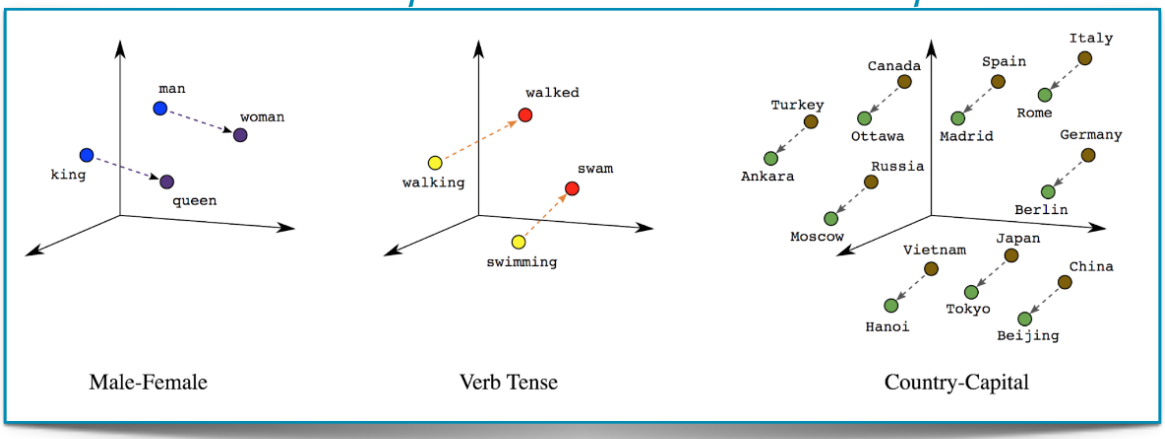
```
array([  17,   91,  972, 1879,    7,    3,  973,  974,   70,   33,    3,
         99,  973,  974,   72, 1880,    2,  100,  164,    7,    3,    0,
          0,    0,    0,    0,    0,    0,    0,    0], dtype=int32)
```

5.2 Vectorization - GloVe Word Embeddings

After the text samples have been converted into sequences of integers, we need to turn these sequences into numerical vectors.

Word Embeddings are generally used with Sequential models, as they capture the semantic relationship of words in a sentence. In word embeddings, sequences are represented using word vectors in a dense vector space, where the location and distance between words indicates how similar they are semantically.

Word Vector representations in n-dimensional space



GloVe is one such unsupervised learning algorithm for obtaining vector representations for words. Word2vec is another. However, word2vec relies only on 'local

context'. In other words, the semantics learnt for a given word, is only affected by the surrounding words. The advantage of GloVe is that, unlike word2vec, GloVe does not rely just on local context, but incorporates global context to obtain word vectors. Hence, for this Capstone project, **'GloVe word embeddings'** was chosen over word2vec.

Sample GloVe vector representations

	0	1	2	3	4	5	6	7	8	9	...
chanty	0.232040	0.025672	-0.70699	-0.045465	0.13989	-0.628070	0.726250	0.341080	0.446140	0.163290	...
kronik	-0.609210	-0.672180	0.23521	-0.111950	-0.46094	-0.007462	0.255780	0.856320	0.055977	-0.237920	...
rolonda	-0.511810	0.058706	1.09130	-0.551630	-0.10249	-0.126500	0.995030	0.079711	-0.162460	0.564880	...
zsombor	-0.758980	-0.474260	0.47370	0.772500	-0.78064	0.232330	0.046114	0.840140	0.243710	0.022978	...
sandberger	0.072617	-0.513930	0.47280	-0.522020	-0.35534	0.346290	0.232110	0.230960	0.266940	0.410280	...

5 rows × 50 columns

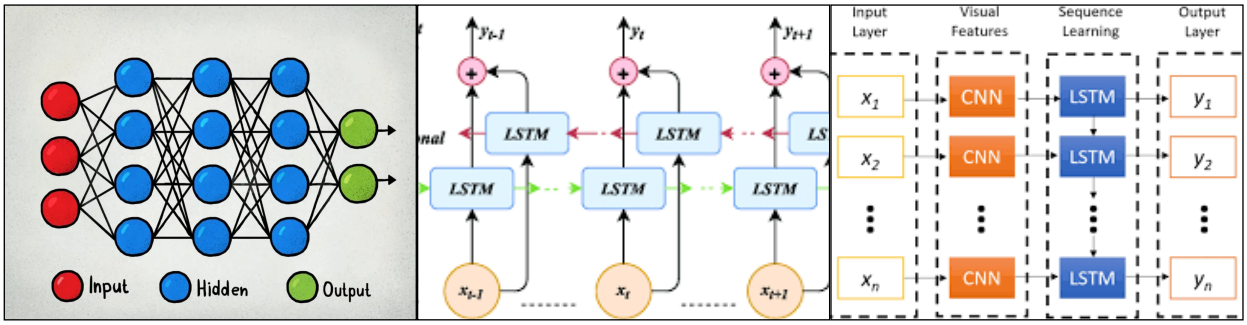
Since the dataset is small, a '50-dimensional GloVe vector' trained on 60 billion wikipedia token corpus has been chosen for this Capstone project. The Glove embeddings were obtained from *https://nlp.stanford.edu/projects/glove/.*

---

6. Model building - DL models

The following DL models were considered for this NLP Capstone project:

- **Neural Network classifier** - Early language models used layers of feedforward Neural networks. However, they were unable to capture the 'contextual relationship' of words.
- **Bidirectional LSTM classifier** - Later, RNNs and LSTMs (a variant of RNN) were then used to capture long-distance contexts. Then, Bidirectional LSTMs (BiLSTM) were used. These were an improvement over LSTMs as they look at word sequences both in the forward and backward directions.
- **CNN-Bidirectional LSTM classifier** - Following that, the CNN layer, which does feature selection on the embedding vector was added. These models achieved better results in many NLP problems when compared to models that use only LSTMs.



Embedding Layer

The 'embedding layer' was used as the first layer. This layer uses pre-trained 50-dimensional 'GloVe' word embeddings. For the NN model, the output from this layer was flattened before being fed into the Dense layers.

Activation Functions

In general, 'ReLU' and 'Leaky ReLU' were used as the activation functions for the hidden layers. However, 'Softmax' was used as the activation function for the Output layer, as it is a multi-class text classification problem.

Regularization

Batch Normalisation, Dropout and Spacial Dropout layers were used for regularising the NN model. For the LSTM and CNN models - dropouts, recurrent dropouts and spatial dropouts were used for regularization.

Optimization

'Adam Optimizer' with a learning rate of 0.01 was used for all the models. The learning rate was varied over the course of training the models, by a factor of 0.1, using the 'ReduceLROnPlateau' callback.

Model Callbacks

'EarlyStopping', 'ReduceLROnPlateau' and 'ModelCheckpoint' are the three callbacks used during the training phase.

- '**EarlyStopping**' is used to stop training when a 'valuation loss' has stopped improving.
- '**ReduceLROnPlateau**' reduces the learning rate by a factor of 0.1 once learning stagnates. This callback monitors 'valuation loss' and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.
- '**ModelCheckpoint**' is used to save the best model in a checkpoint file.

6.1 Performance of DL models

Performance metrics of the Neural Network classifier

Classification Report:

	precision	recall	f1-score	support
0	0.85	0.79	0.81	28
1	0.69	0.39	0.50	28
2	0.44	0.61	0.51	28
3	0.50	0.52	0.51	29
4	0.77	0.82	0.79	28
accuracy			0.62	141
macro avg	0.65	0.62	0.62	141
weighted avg	0.65	0.62	0.62	141

Accuracy: 62.4%

The Neural Network classifier has achieved an accuracy of 62.4% with a 65% precision and a 62% of recall and F1-score, giving a good overall performance. Since the dataset is already target balanced, 'accuracy' can be taken into consideration as a performance parameter.

Performance metrics of the LSTM classifier

Classification Report:

	precision	recall	f1-score	support
0	0.81	0.89	0.85	28
1	0.50	0.32	0.39	28
2	0.56	0.64	0.60	28
3	0.61	0.48	0.54	29
4	0.73	0.96	0.83	28
accuracy			0.66	141
macro avg	0.64	0.66	0.64	141
weighted avg	0.64	0.66	0.64	141

Accuracy: 66.0%

The LSTM classifier has achieved an accuracy of 66%, with a 66% recall and a 64% of precision and F1-score. The accuracy, recall and F1-scores are higher than the NN model. However, the precision has decreased to 64%.

Performance metrics of the CNN-LSTM classifier

Classification Report:

	precision	recall	f1-score	support
0	0.84	0.93	0.88	28
1	0.68	0.61	0.64	28
2	0.54	0.75	0.63	28
3	0.65	0.38	0.48	29
4	0.90	0.93	0.91	28
accuracy			0.72	141
macro avg	0.72	0.72	0.71	141
weighted avg	0.72	0.72	0.71	141

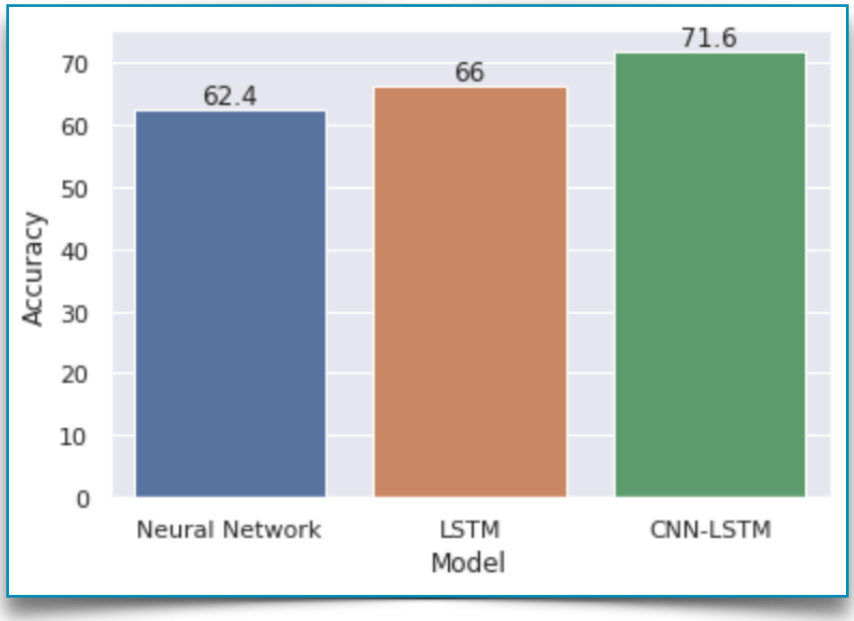
Accuracy: 71.6%

The CNN-LSTM classifier has given the best performance so far among the three models with an accuracy of 71.6%, a precision and recall score of 72% and an F1-score of 71%. The addition of a 'convolution layer' has given a significant increase in performance when compared to the LSTM model.

Performance Summary of DL models

	Model	Accuracy
0	Neural Network	62.4
1	LSTM	66.0
2	CNN-LSTM	71.6

Performance plots of the DL models



6.2 Model Performance improvement

6.2.1 Hyper-parameter tuning with Grid Search cross validation

The following hyper parameters were tuned with Grid Search cross validation using a Keras classifier wrapper:

- **NN:** {'learning_rate': [1e-2, 1e-3], 'alpha_lr': [0.6, 0.9], 'batch_size': [64, 128], 'epochs': [100, 150]}

```
Best params: {'alpha_lr': 0.6, 'batch_size': 128, 'epochs': 150, 'learning_rate': 0.01}
5/5 [==============================] - 0s 4ms/step
Test accuracy using the best hyperparameters: 63.0%
```

- **LSTM:** {'learning_rate': [1e-2, 1e-3], 'lstm_units': [64, 128], 'batch_size': [64, 128], 'epochs': [100, 150]}

```
Best params: {'batch_size': 128, 'epochs': 150, 'learning_rate': 0.01, 'lstm_units': 128}
5/5 [==============================] - 2s 24ms/step
Test accuracy using the best hyperparameters: 0.65
```

- **CNN-LSTM:** {'learning_rate': [1e-2, 1e-3], 'lstm_units': [64, 128], 'batch_size': [64, 128], 'epochs': [100, 150]}

```
Best params: {'batch_size': 128, 'epochs': 150, 'learning_rate': 0.01, 'lstm_units': 128}
5/5 [==============================] - 1s 16ms/step
Test accuracy using the best hyperparameters: 0.67
```

Performance metrics of the best model after hyper-parameter tuning

```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.93      0.88        28
           1       0.67      0.64      0.65        28
           2       0.56      0.54      0.55        28
           3       0.62      0.62      0.62        29
           4       0.96      0.93      0.95        28

    accuracy                           0.73       141
   macro avg       0.73      0.73      0.73       141
weighted avg       0.73      0.73      0.73       141

Accuracy: 73.0%
```

After tuning the hyper-parameters of the models using Grid Search cross validation, CNN-LSTM model with a 'learning rate'=0.01, 'lstm units'=128, 'batch size'=128 and

'epochs'=150 has given the best performance of 73%. The convolution layer does feature selection on the embedding vector and hence, has given a better performance when compared to the LSTM model.

In general, tuning the hyper-parameters has not produced significant improvement in the performance of the models. Also, since the dataset is quite small, the DL models could not be trained sufficiently. Hence, 'Text Augmentation' using BERT contextual embeddings was considered for improving the performance of the models.

However, among the DL models used for training the given dataset, CNN-LSTM combination has outperformed the other models and has emerged as the best model so far after hyperparameter tuning.

6.2.2 Text Augmentation with 'NLPAug' - using BERT contextual embeddings

Since the dataset provided is quite small, it is impossible to achieve further improvement in performance of the existing models without augmenting the dataset.

Hence, 'synonym substitution' technique using BERT contextual embeddings is used for data augmentation as it is the most effective technique and is widely used in the industry. This is achieved with the help of the 'nlpaug' library.

```
Shape of the preprocessed dataset: (705, 2)
Shape of the augmented dataset: (1410, 2)
```

After data augmentation, the dataset has become twice the size of the original dataset.

Vocabulary size of the original dataset

```
vocabulary size: 3003
```

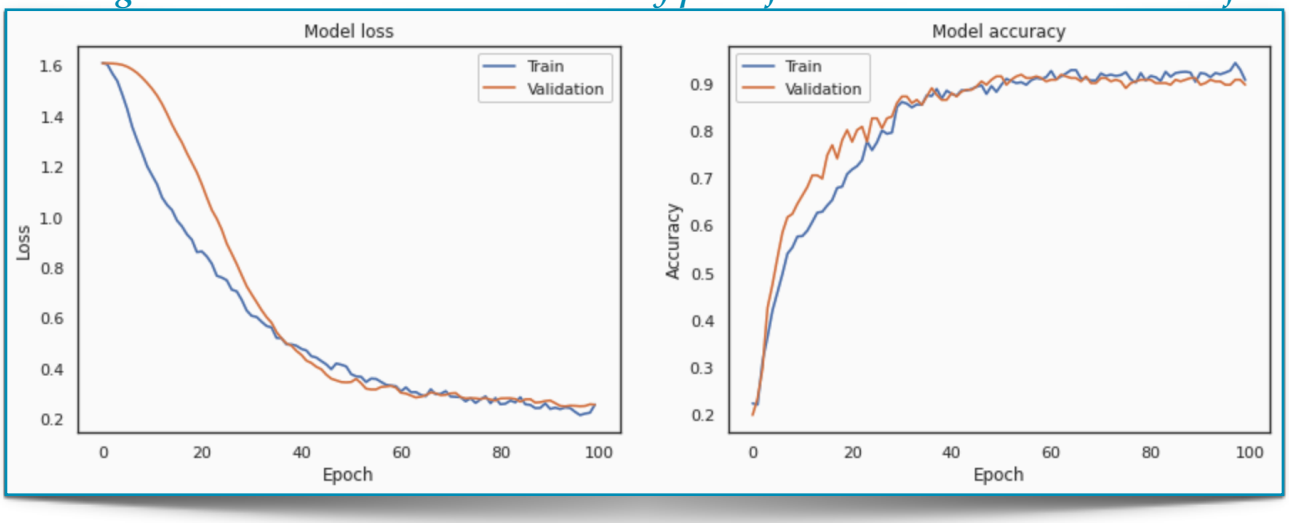
Vocabulary size of the augmented dataset

```
vocabulary size: 3564
```

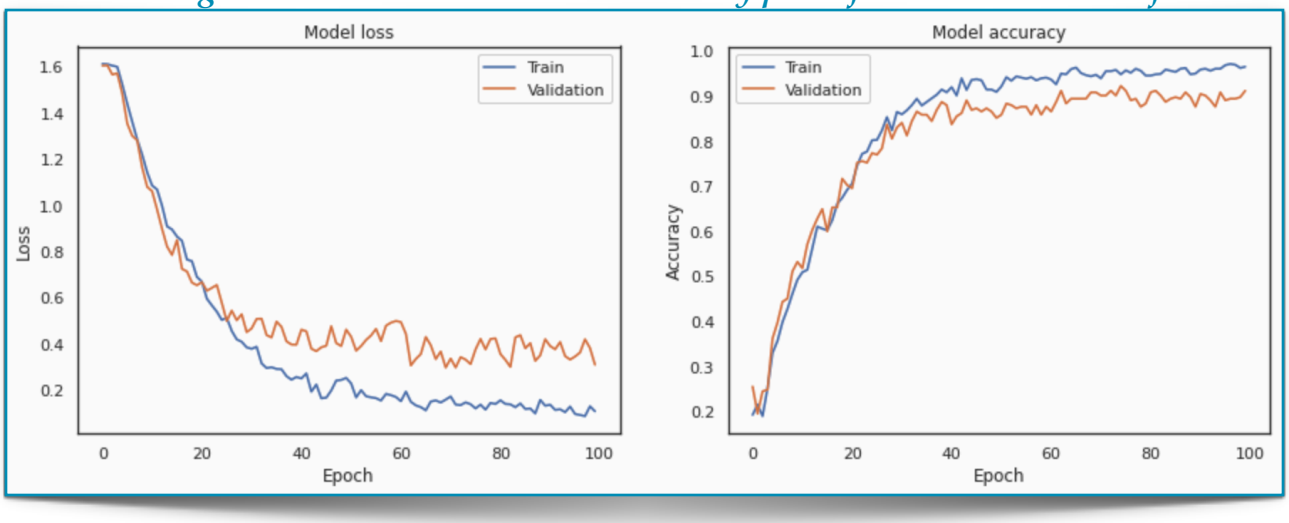
Also, since data augmentation has led to an increase in the number of words in the vocabulary, the 'vocabulary size' of the augmented dataset has increased when compared to the original dataset.

Training the models with the text augmented dataset

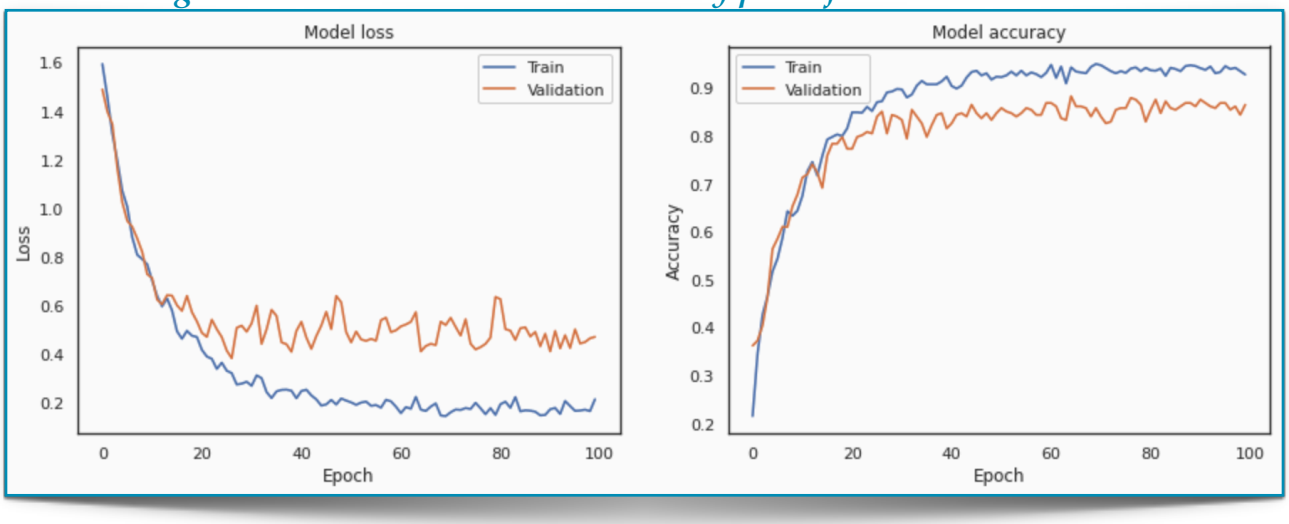
Training and Validation Loss and Accuracy plots for the Neural Network classifier



Training and Validation Loss and Accuracy plots for the LSTM classifier



Training and Validation Loss and Accuracy plots for the CNN-LSTM model



For the NN classifier, the accuracy and loss learning curves seem to stabilize after 80-100 epochs. If the model continues to train beyond this, it might lead to overfitting. Hence, it represents a good fit learning curve.

However, for the LSTM and CNN-LSTM classifiers, there seems to be a small "generalization gap" between the train and validation accuracy and loss learning curves, indicating some amount of overfitting. In spite of using 'dropouts' and 'recurrent dropouts' in the LSTM layers for regularization, this 'generalisation gap' could not be reduced any further without suffering a loss in accuracy.

Performance Evaluation of the models with the test data

Performance Metrics of the Neural Network classifier

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.93	0.95	57
1	0.91	0.89	0.90	56
2	0.85	0.84	0.85	56
3	0.77	0.82	0.80	57
4	1.00	1.00	1.00	56
accuracy			0.90	282
macro avg	0.90	0.90	0.90	282
weighted avg	0.90	0.90	0.90	282

Accuracy: 89.7%

Performance Metrics of the LSTM classifier

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.95	0.97	57
1	0.93	0.91	0.92	56
2	0.92	0.82	0.87	56
3	0.79	0.91	0.85	57
4	0.95	0.96	0.96	56
accuracy			0.91	282
macro avg	0.92	0.91	0.91	282
weighted avg	0.92	0.91	0.91	282

Accuracy: 91.10000000000001%

Performance metrics of the CNN-LSTM classifier

Classification Report:

	precision	recall	f1-score	support
0	0.84	0.95	0.89	57
1	0.89	0.88	0.88	56
2	0.90	0.80	0.85	56
3	0.85	0.79	0.82	57
4	0.93	1.00	0.97	56
accuracy			0.88	282
macro avg	0.88	0.88	0.88	282
weighted avg	0.88	0.88	0.88	282

Accuracy: 88.3%

Performance of the DL Models after text augmentation

	Model	Accuracy after Text Augmentation
0	Neural Network classifier	89.72
1	LSTM classifier	91.13
2	CCN-LSTM Classifier	88.3

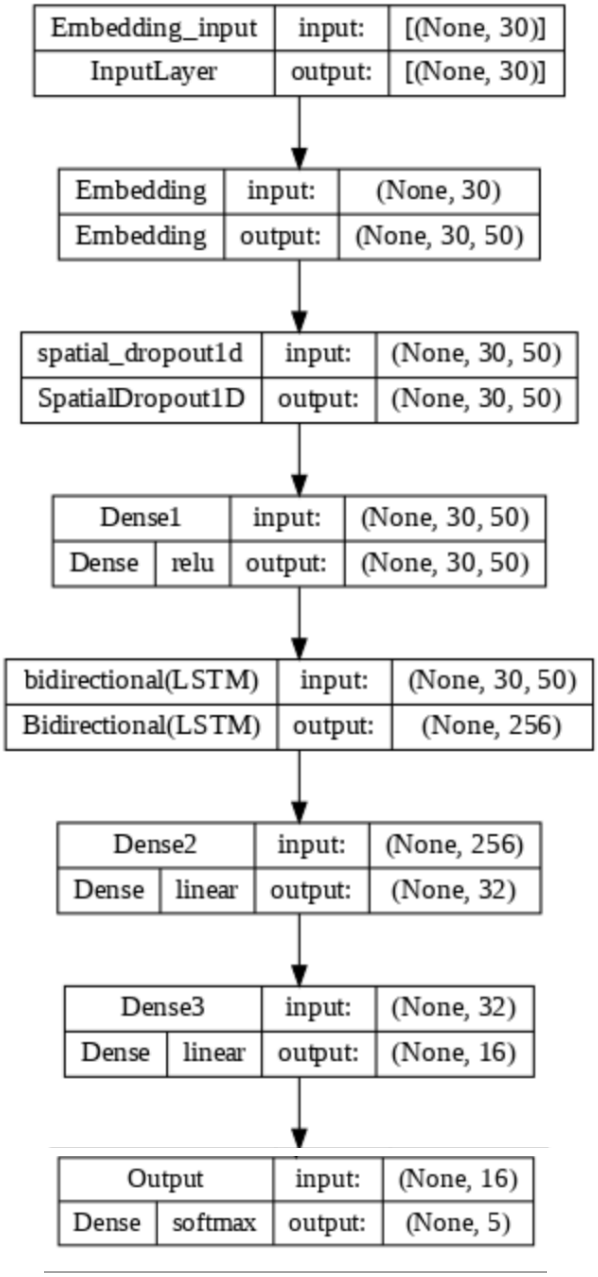
After text augmentation with 'nlpaug' library using BERT contextual embeddings, there has been a significant improvement in the performance of the DL models.

The 'Bidirectional LSTM model' has given the best performance of 91.1%. This is a significant improvement over 66% after hyperparameter tuning. Also, the precision, the recall and the F1 scores are 92%, 91% and 91% respectively, indicating a well-balanced model.

Initially, since the dataset was very small, the DL models could not be trained sufficiently. However, after 'synonym substitution' using BERT contextual embeddings, the performance of the models improved significantly.

7. Final Model

7.1 Model Architecture and Training



Embedding Layer

The 'embedding layer' was used as the first layer. This layer uses pre-trained 50-dimensional 'GloVe' word embeddings. Hence, the 'trainable' parameter has been made 'False'.

Hidden Layers

There is a Dense layer having the same number of neurons as the embedding dimension as it connects the Embedding layer to the LSTM layer.

The hidden layers, primarily, consists of a 'bidirectional LSTM' with 128 LSTM units, a 'uniform' kernel initialiser, a dropout rate and a recurrent dropout rate of 0.4. The 'return_sequences' parameter has been made 'False' as it is connected to a Dense layer below.

Following the LSTM layer are two fully connected Dense layers of 32 and 16 neurons each.

Output Layer

The output layer consists of a Dense layer with 5 neurons, as there are 5 different classes in the target variable. It uses 'softmax' activation as it is a multi-class classification problem and has a 'uniform' kernel initializer.

Regularization

As LSTMs are prone to overfitting, dropouts, recurrent dropouts and spatial dropouts were used for regularization purposes.

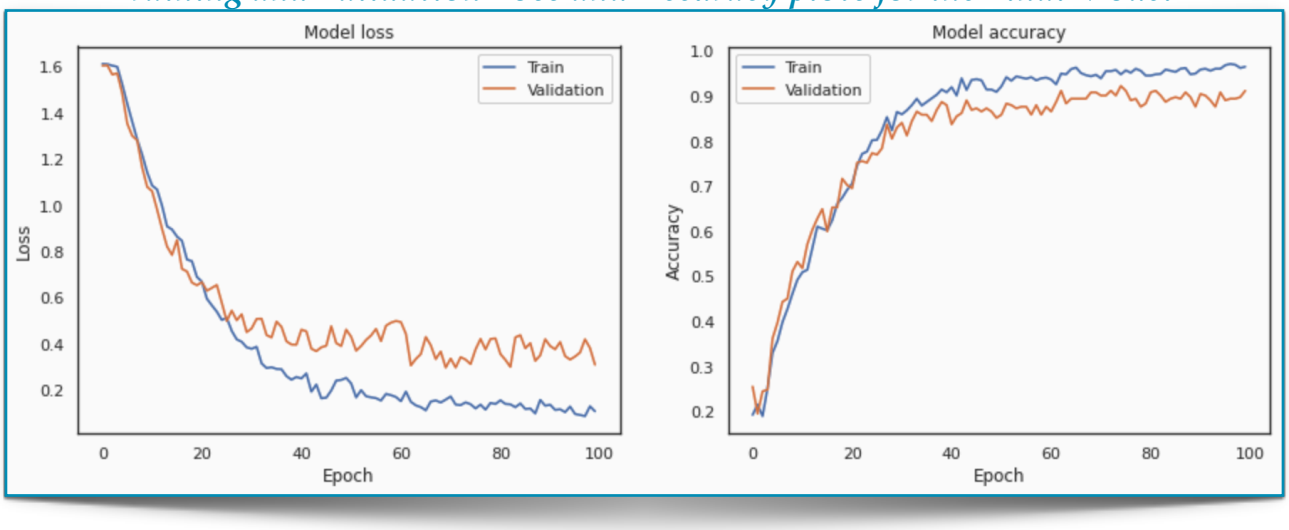
Model Compilation

- **Optimizer** - 'Adam Optimizer' with a learning rate of 0.01 was used. The learning rate was varied over the course of training the model, by a factor of 0.1, using the 'ReduceLROnPlateau' callback.
- **Loss Function** - As 'Label Encoding' was used for the target variable, 'sparse_categorical_crossentropy' was used as the loss function for this multi-class classification problem.

Model Training

The model was trained for 100 epochs with a batch size of 128. Both training and validation 'loss and accuracy' were the parameters that were monitored during the training phase.

Training and Validation Loss and Accuracy plots for the Final Model



There seems to be a small "generalization gap" between the train and validation accuracy and loss learning curves, indicating some amount of overfitting. In spite of using 'dropouts' and 'recurrent dropouts' in the LSTM layer for regularization, this 'generalisation gap' could not be reduced any further without suffering a loss in accuracy.

7.2 Model Performance Evaluation

Performance Metrics of the Final Model

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.95	0.97	57
1	0.93	0.91	0.92	56
2	0.92	0.82	0.87	56
3	0.79	0.91	0.85	57
4	0.95	0.96	0.96	56
accuracy			0.91	282
macro avg	0.92	0.91	0.91	282
weighted avg	0.92	0.91	0.91	282

Accuracy: 91.10000000000001%

The Final model has given the best performance of 91.1% after 'text augmentation' using BERT contextual embeddings. Also, the precision, the recall and the F1 scores are 92%, 91% and 91% respectively. This clearly indicates that the model is well balanced for multi-class classification.

Comparison to benchmark

The performance of the base models was the 'benchmark' that was laid out. The LSTM base model had an accuracy and recall of 66% and a precision and F1-score of 64%.

However, the Final model has given the best performance of 91% accuracy, recall and FI-scores and a 92% precision. This is a significant improvement over the base model.

'Text augmentation' using BERT contextual embeddings has been primarily responsible for this tremendous improvement in performance of the Final model over the base model.

8. Conclusion

8.1 Business Implications

The objective of this Capstone project was to help professionals to highlight the safety risk in industrial plants. It is an imperative need for certain industries around the globe to comprehend why employees still suffer injuries/accidents in manufacturing plants.

The dataset that was provided for this capstone project records accidents from various manufacturing plants around the globe.

The key finding in this dataset after exploratory data analysis was the fact that the 'mining industry sector' is the key contributor to the number of accidents. The mining industry has always had a reputation for being a risky business, with health risks that are varied and often quite serious, sometimes even leading to death. It was also noted during data analysis that the most severe accidents happened in the mining sector.

Following that, Text Classification models were built that identifies the potential accident level based on the description of the various accidents. The Final model can be used to predict how severe the accident could be with 91% level of confidence thereby fore-warning industries.

If the respective industries were fore-warned on the type of accidents that are happening frequently in their plants, they could reduce future accidents from happening by introducing strict safety legislation and protocol, as well as by using advanced safety equipment.

8.2 Limitations and Enhancements

Limitations

There was only one incident that was recorded for the most severe accident - 'Potential Accident Level VI'. Since, it was impossible to train the models sufficiently with only one happening, it was removed from the dataset. Hence, the Final model does not have the capability to predict 'Potential Accident Level VI'.

Also, if the real world accidents are different from the accident descriptions that were used during the training phase, then the Final model will not be able to predict the severity of the accident accurately.

Enhancements

- **Model Architecture** - 'LSTMs with Attention Mechanisms ' can be explored to enhance the performance of the model.

- **Word Embeddings** - Latest variations of BERT and ELMo can be explored and used in place of GloVe embeddings to enhance the performance.
- **Text Augmentation technique** - 'Synonym substitution' using BERT contextual embeddings was the text augmentation technique that was used in this capstone project. Other text augmentation techniques, such as, 'back translation' can be explored further .

8.3 Closing Reflections

Significant Learnings

- **NLPAug** - The most significant learning in this Capstone project has been the use of 'NLPAug' library for text augmentation using BERT contextual embeddings. Post text augmentation, there has been a significant improvement in the performance of the models.
- **spaCy** - The use of 'spaCy' for text preprocessing in place of NLTK library has been another significant learning. SpaCy is blazing fast and excels at large-scale information extraction tasks. Since spaCy was specifically designed for production use, it helps build and train powerful NLP pipelines and package them for easy deployment. SpaCy v3.0 uses transformer-based pipelines that bring spaCy's accuracy right up to current state-of-the-art. SpaCy's large English language pipeline 'en_core_web_lg' that is optimised for both CPU and GPU was used for this project.

Future Work

- **Attention Mechanisms** - LSTM models would be enhanced with Attention mechanisms to improve performance.
- **Transformers** - Pre-trained models based on the transformer architecture would be used in place of LSTMs.

---

9. References

- *https://nlpaug.readthedocs.io/en/latest/augmenter/word/context_word_embs.html*
- *https://spacy.io/usage*
- *https://nlp.stanford.edu/projects/glove/*