Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Industrial Safety and Health Analysis using Machine Learning & Neural Networks

DHANUSH.H-18MIS1015

I. SUMMARY OF PROBLEM STATEMENT, DATA AND FINDINGS

ABSTRACT

Safety is very important aspect for any industry as an accident free work environment boosts the morale of the team members working in any hazardous situations.
Safety means continuing and healthful living without injury. Safety is freedom from harm or the danger of harm. The word safety also refers to the precautions people take to prevent accidents, harm, danger, damage, loss and pollution. Safety also deals with improvement in working conditions for better health. Management is responsible to provide safe working condition and individual's safety.

PROBLEM STATEMENT

The dataset ***Industrial_safety_and_health_database_with_accidents_description.csv*** contains of industrial accidents from 12 different plants from 3 different countries. The database comes from one of the biggest industries in Brazil and in the world. The main Objective/Target of this project is to Predict which model has the highest accuracy so that it can help the professionals to highlight the safety risk as per the incident description.. The dataset given consists of several attributes from which we can understand at what time the accident occurred and from which country it was and from which city it was, accident level's and many more.

The summary of the dataset is as shown below:

- ***Time/Date information:*** The exact time and date of the accident.
- ***Countries:*** From which country the accident occurred. They didn't Mentioned the country names.
- ***Locals:*** In which city the manufacturing plant is located.
- ***Industry Sector:*** This consists of different sectors like mining, metals and others.
- ***Accident Level:*** They've given totally four levels from I to IV. Each record shows how severe was the accident.
- ***Potential Accident Level:*** The severity of the accident is measured by the database taking into account about other factors involved.
- ***Genre:*** If the person is male or female.
- ***Employee or Third party:*** Is the Injured person is an Employee or a Third party.
- ***Critical Risk:*** Given some description of the risk involved in the accident.
- ***Description:*** They've given the detailed elucidation of how the accident happened.

Findings

- The Dataset consists of 425 rows and 11 columns.
- There are no null values.
- "Data" column can be dropped and replaced by five more columns, namely "Weekday", "WeekofYear", "Year", "Month" and "Day". The newly added columns have been extracted from the "Data" column and bring more insight to the data.
- "Unnamed: 0" was dropped due to its redundancy.

II. OVERVIEW OF THE FINAL PROCESS

DATA PREPROCESSING

Pre-Processing is essential for efficient Feature Extraction leading to non-redundant Feature Descriptors. The main steps involved in the Pre-Processing Stage of the Pipeline are:

- Word Tokenization
- Stop words removal
- n-grams
- Lemmatization

A. Word Tokenization

Given a sentence a list of words is generated by considering space as delimiter. As many of the consists of special characters, punctuation normal word tokenizers cannot give good results.

B. Stop words Removal

Stop words refer to the most common words in a language. Although, there is no universal list for the Stop Words, but an exhaustive list has been used to ensure better accuracy and efficient feature extraction. They need to be removed because they most likely do not add up value to the data.

Ex: made, Mr., slide, using etc.

C. N-grams

The list of tokenized words is made into n-grams of size 1 to 6 by joining adjacent words together. As many of descriptions contain multiple words, this step is necessary to extract the key words.

After the pre-processing stage, sufficient amount of redundancy gets removed. The words now need to be converted to equivalent feature descriptors to ensure good classification and low enough to ensure that computation performed on them is tractable.

D. Lemmatization

Lemmatization takes into consideration the morphological analysis of the words. To do so, it is necessary to have detailed dictionaries which the algorithm can look through to link the form back to its lemma.

Features:

Word and POS tags

The token itself (in lowercase) was added as a feature. POS tags were generated using nltk were also added as a feature.

Value Counts

Value counts of *'Potential Accident Level'* and *'Industry Sector'* are taken.

Character n-grams of Token

Character n-grams extracted from token of length 1 to 4 were added

as features. Ex: ***Uni – gram:*** accident, equipment, activity, and collaborator
Bi – gram: time accident, causing injury, employee used, ring finger, upper part

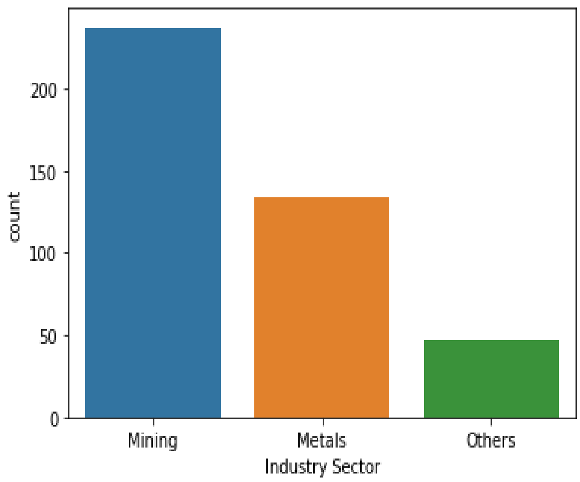
Presence in Dictionary

A binary feature for presence of Chemical element names, Chemical element symbols, Amino acid names, Amino acid codes of length 1 and 3, Systematic names, Trivial names, Family names, Greek letters and Greek symbols.

III. VISUALIZATIONS

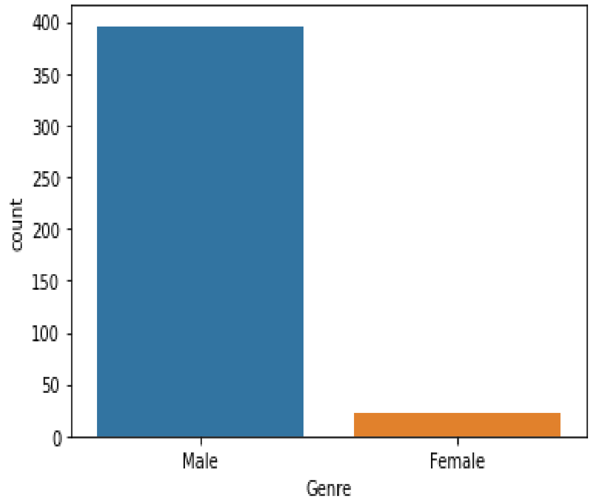
- The below bar plot shows that the highest incidents are occurred in the Mining Industry.

```
sns.countplot(x='Industry Sector',data=ind_data)
     <matplotlib.axes._subplots.AxesSubplot at 0x7f70ed0dae90>
```

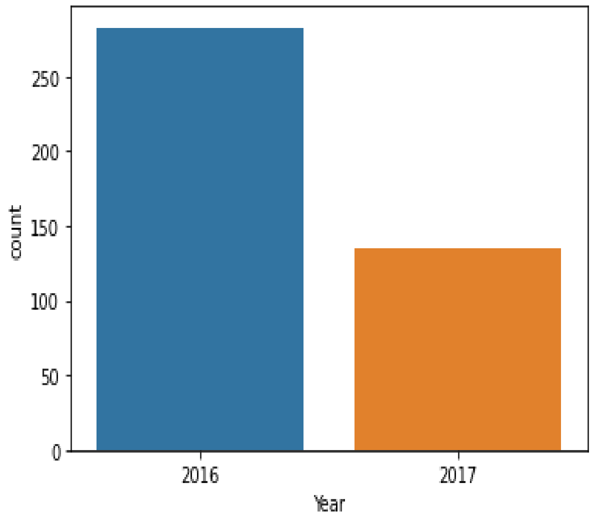



```
sns.countplot(x='Potential Accident Level',data=ind_data)
     <matplotlib.axes._subplots.AxesSubplot at 0x7f70e9074490>
```

- The below plot shows that around 91% of injured persons are Male.

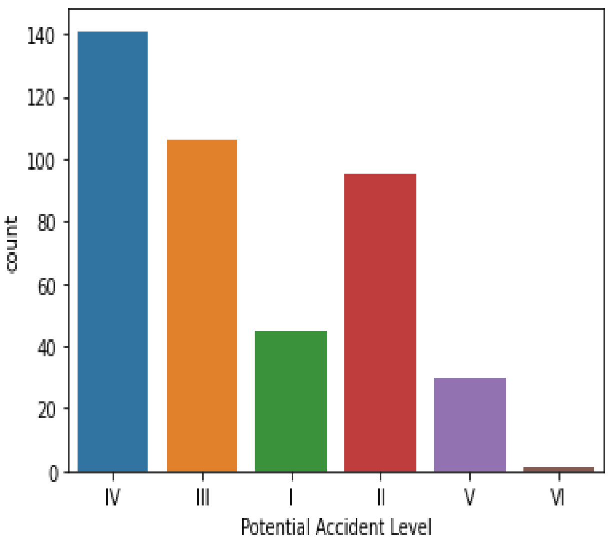


- From the below plot, we can conclude that accidents happened in the year 2016 are around 70%



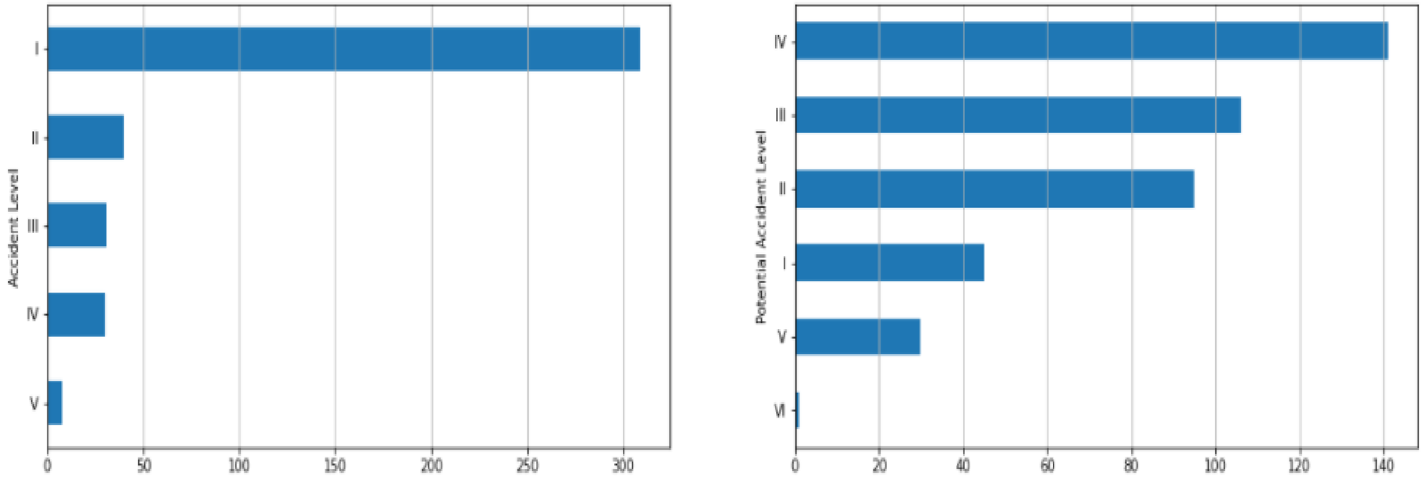
```
ind_data.drop("Time",axis=1,inplace=True)
ind_data.drop_duplicates(inplace=True)
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
sns.countplot(x='Countries',data=ind_data)
<matplotlib.axes._subplots.AxesSubplot at 0x7f70ece86b50>
```

The below plot of Potential Accident Level shows that around 30% of the accidents would have been very severe (Level 4).




```
sns.countplot(x='Potential Accident Level',data=ind_data)
     <matplotlib.axes._subplots.AxesSubplot at 0x7f70e9074490>
```

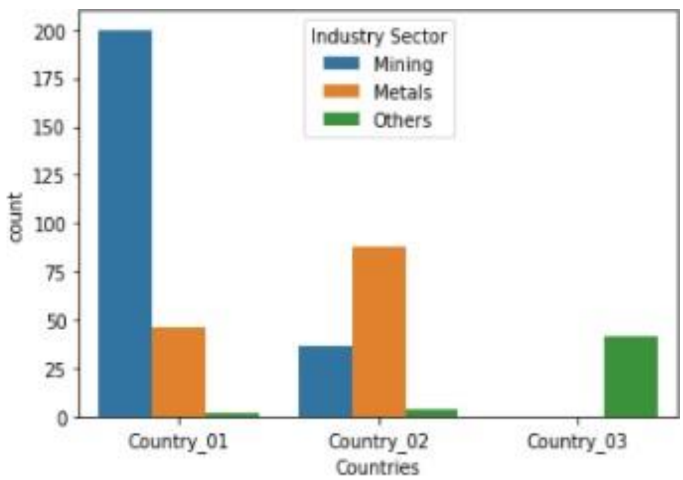
- Graphs for accident level and potential accident level.



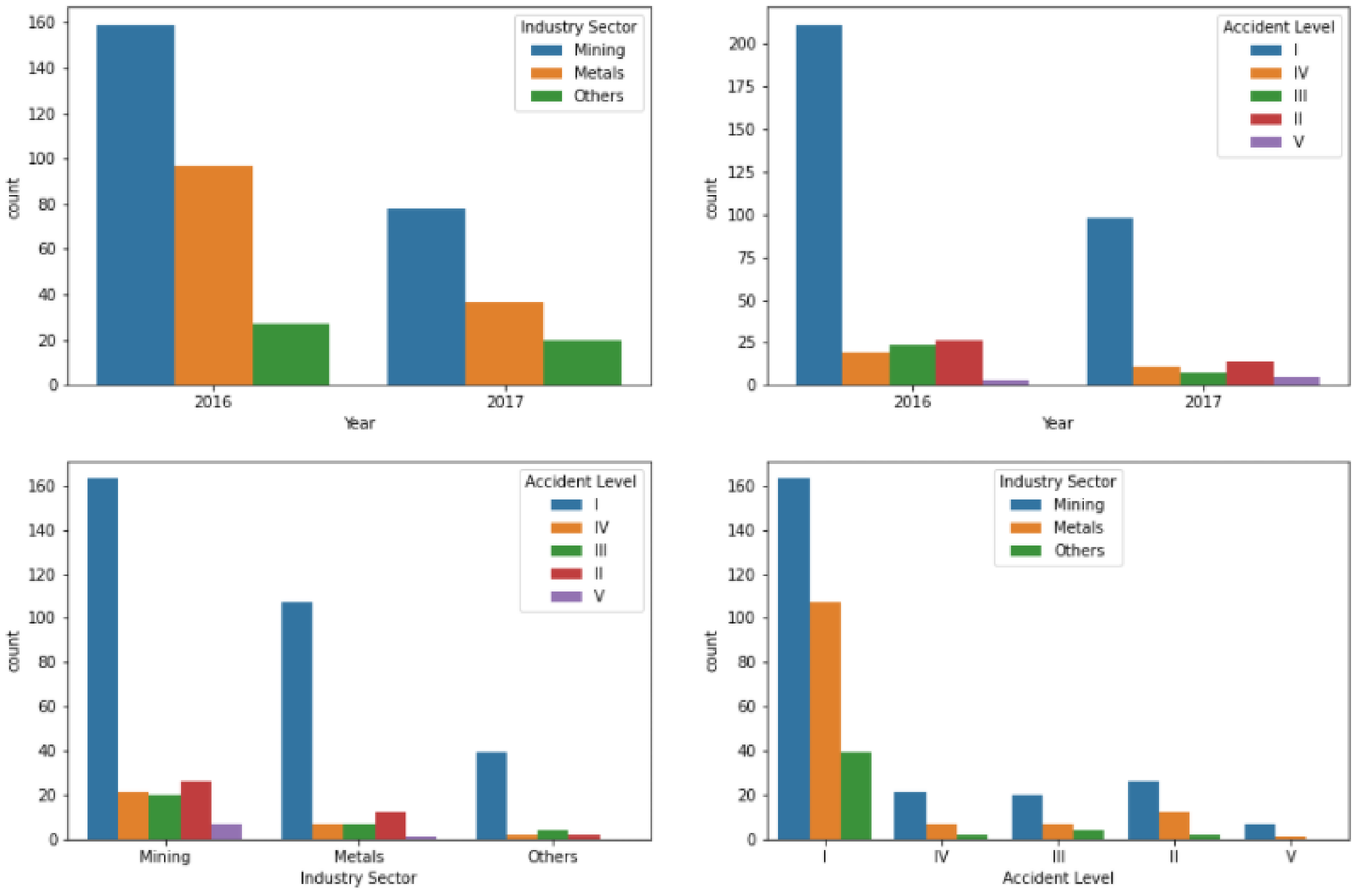

```
fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(20,5))
acc_level = "Accident Level"
ind_data[acc_level].reset_index().groupby(acc_level).count().sort_values(by=
       "index").plot(kind="barh", legend=False,
        ax=ax[0]).grid(axis='x')
pot_acc_level = "Potential Accident Level"
ind_data[pot_acc_level].reset_index().groupby(pot_acc_level).count().sort_values(b
       "index").plot(kind="barh", legend=False,
        ax=ax[1]).grid(axis='x')

plt.show()
```

- The below plot shows incident count of each country with respect to the industries.



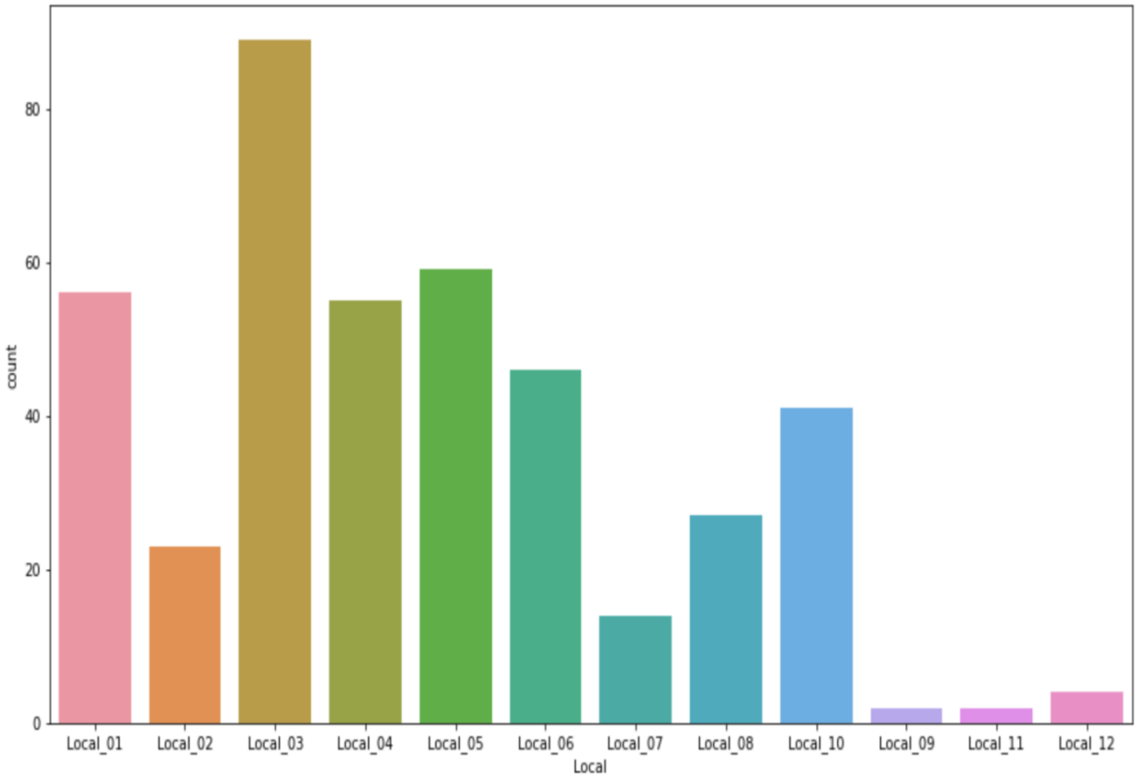
Graphs in respect to industry sector and Accident level.




```
Fig,axs = plt.subplots(nrows=2,ncols=2,figsize=(15,10))
sns.countplot(x=ind_data['year'],hue='Industry Sector',data=ind_data,ax=axs[0][0])
sns.countplot(x=ind_data['year'],hue='Accident Level',data=ind_data,ax=axs[0][1])
sns.countplot(x=ind_data['Industry Sector'],hue='Accident Level',data=ind_data,ax=1
sns.countplot(x=ind_data['Accident Level'],hue='Industry Sector',data=ind_data,ax=1
```

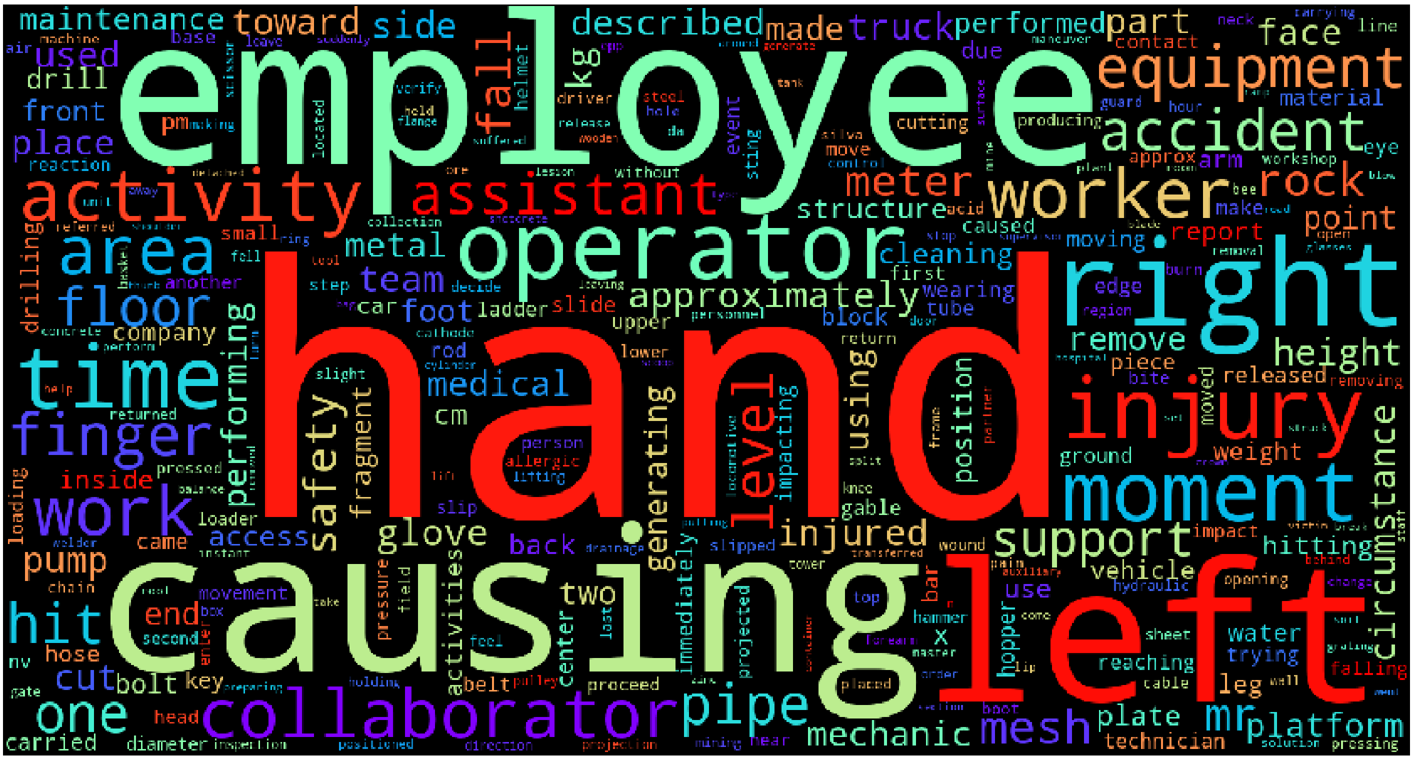
Graphs in respect to industry sector.

```
plt.figure(figsize=(20,5))
descending_order = ind_data['Critical
Risk'].value_counts().sort_values(ascending=
sns.countplot(x=ind_data['Critical Risk'],order=descending_order)
plt.xticks(rotation = 'vertical')
```

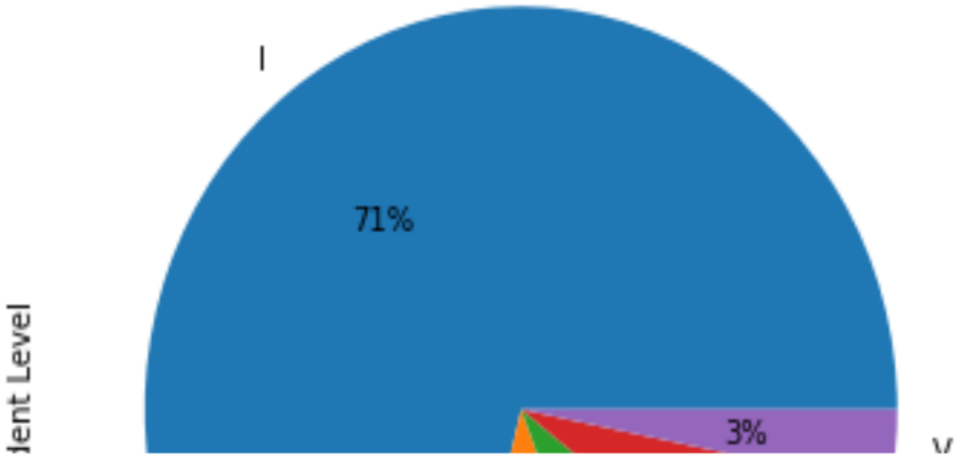



```
import matplotlib.pyplot as plt
plt.figure(figsize=(15,8))
sns.countplot(x='Local',data=ind_data)
     <matplotlib.axes._subplots.AxesSubplot at 0x7f70ee2861d0>
```

- Word Cloud of the words




```
from wordcloud import WordCloud
wordcloud = WordCloud(width = 1500, height = 800, random_state=0, background_color
                      min_font_size=5, max_words=300, collocations=False).generate
plt.figure(figsize=(15,10))
Checking 5 random Descriptions and accident_levels from the data where the
 Description: during withdrawal kelly bar conductive bar 45 kg 15 length 10
 accident_level: 0
 Description: spillway circumstances worker cleaning use absorbent cloth oil
 accident_level: 3
 Description: in welding workshop level 3620 ex 450 tunnel quinoa moments as
 accident_level: 0
 Description: during ore transport works op2 bines filled tenth mining car o
 accident_level: 2
 Description: when performing doosan rb10 equipment hammer repair employee t
 accident_level: 2
 Distributon of accident_level where the length of Description is > 200
n we change potential accident level values
and 2, 3 and 4 and 4 and 5 one level
t(count_value(ind_data["Potential Accident Level"]))
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```




```
Learning rate set to 0.074823
0:      learn: 1.0724255        total: 107ms    remaining: 1m 46s
1:      learn: 1.0485564        total: 147ms    remaining: 1m 13s
2:      learn: 1.0275394        total: 188ms    remaining: 1m 2s
3:      learn: 1.0073333        total: 230ms    remaining: 57.3s
4:      learn: 0.9896080        total: 270ms    remaining: 53.7s
5:      learn: 0.9747871        total: 309ms    remaining: 51.1s
6:      learn: 0.9609752        total: 347ms    remaining: 49.2s
7:      learn: 0.9478226        total: 389ms    remaining: 48.2s
8:      learn: 0.9374837        total: 430ms    remaining: 47.4s
9:      learn: 0.9253568        total: 470ms    remaining: 46.5s
10:     learn: 0.9141556        total: 511ms    remaining: 45.9s
11:     learn: 0.9034016        total: 556ms    remaining: 45.8s
12:     learn: 0.8946438        total: 602ms    remaining: 45.7s
13:     learn: 0.8883174        total: 641ms    remaining: 45.2s
14:     learn: 0.8813167        total: 682ms    remaining: 44.8s
15:     learn: 0.8717840        total: 721ms    remaining: 44.3s
16:     learn: 0.8655703        total: 761ms    remaining: 44s
17:     learn: 0.8593093        total: 800ms    remaining: 43.7s
18:     learn: 0.8521746        total: 853ms    remaining: 44s
19:     learn: 0.8463984        total: 894ms    remaining: 43.8s
20:     learn: 0.8423463        total: 934ms    remaining: 43.6s
21:     learn: 0.8376825        total: 973ms    remaining: 43.3s
22:     learn: 0.8342001        total: 1.01s    remaining: 43.1s
23:     learn: 0.8277750        total: 1.05s    remaining: 42.9s
24:     learn: 0.8230949        total: 1.1s     remaining: 42.8s
25:     learn: 0.8180405        total: 1.15s    remaining: 42.9s
26:     learn: 0.8135982        total: 1.19s    remaining: 42.7s
27:     learn: 0.8106824        total: 1.22s    remaining: 42.5s
28:     learn: 0.8071708        total: 1.27s    remaining: 42.4s
29:     learn: 0.8046025        total: 1.3s     remaining: 42.2s
30:     learn: 0.8014576        total: 1.34s    remaining: 41.9s
31:     learn: 0.7987540        total: 1.38s    remaining: 41.7s
32:     learn: 0.7934596        total: 1.42s    remaining: 41.6s
33:     learn: 0.7901772        total: 1.46s    remaining: 41.4s
34:     learn: 0.7872591        total: 1.5s     remaining: 41.3s
35:     learn: 0.7839707        total: 1.54s    remaining: 41.2s
36:     learn: 0.7788824        total: 1.57s    remaining: 41s
37:     learn: 0.7771179        total: 1.61s    remaining: 40.8s
38:     learn: 0.7752896        total: 1.66s    remaining: 40.8s
39:     learn: 0.7706530        total: 1.7s     remaining: 40.7s
40:     learn: 0.7679547        total: 1.74s    remaining: 40.7s
```

IV. ALGORITHMS

Logistic Regression

Logistic regression is a linear model for classification rather than regression. Logistic regression is named for the function used at the core of the method, the **logistic function**.

The **logistic function,** also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1 / (1 + e^-value)

Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

Gaussian Naive Bayes

It is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data.

Naive Bayes are a group of supervised machine learning classification algorithms based on the **Bayes theorem**. It is a simple classification technique, but has high functionality. They find use when the dimensionality of the inputs is high.

Bayes Theorem can be used to calculate conditional probability. Being a powerful tool in the study of probability, it is also applied in Machine Learning.

The Formula For Bayes' Theorem Is

$$P(A|B) = \frac{P(A \bigcap B)}{P(B)} = \frac{P(A) \cdot P(B|A)}{P(B)}$$

where:

$P(A) =$ The probability of A occurring

$P(B) =$ The probability of B occurring

$P(A|B) =$ The probability of A given B

$P(B|A) =$ The probability of B given A

$P\left(A \bigcap B\right)) =$ The probability of both A and B occurring

When working with continuous data, an assumption often taken is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. The likelihood of the features is assumed to be:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Sometimes assume variance

- is independent of Y (i.e., σi),
- or independent of Xi (i.e., σk)
- or both (i.e., σ)

K – Nearest Neighbors

K Nearest Neighbors (KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In other words, the model structure determined from the dataset. This will be very helpful in practice where most of the real world datasets do not follow mathematical theoretical assumptions. Lazy algorithm means it does not need any training data points for model generation. All training data used in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data. KNN has the following basic steps:

- Calculate distance
- Find closest neighbors
- Vote for labels

Support Vector Machines

Support Vector Machines is one of the most popular and widely used supervised machine learning algorithms. SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees. It is known for its kernel trick to handle nonlinear input spaces.

Generally, Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the dataset into classes.

Bidirectional LSTM

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems. Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.

It involves duplicating the first recurrent layer in the network so that there are now two layers side-by-side, then providing the input sequence as-is as input to the first layer and providing a reversed copy of the input sequence to the second. It also allows you to specify the merge mode, that is how the forward and backward outputs should be combined before being passed on to the next layer.

V. STEP-BY-STEP WALK THROUGH THE SOLUTION

At first, we tried all the machine learning models one by one and saw their accuracies. Out of which Logistic Regression has accuracy 70% whereas Naïve Bayes, XGBoost and Cat Boost has similar accuracy of 66%. KNN, Random Forest same accuracies i.e., 65%. So, after that we tried to build Bidirectional LSTM model which gave accuracy of 74%. The Final model testing accuracy is 74% and training accuracy is 74.83%.

VI. MODEL EVALUATION

Metrics considered for the evaluation of models are:

i. Precision

Out of all the positive predicted, what percentage is truly positive. The precision value lies between 0 and 1.

$$precision = \frac{TP}{TP + FP}$$

ii. Recall

Out of the total positive, what percentage are predicted positive. It is the same as TPR (true positive rate).

$$recall = \frac{TP}{TP + FN}$$

iii. Accuracy

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

iv. F1-Score

The F1 score (also F-score or F-measure) is a measure of a tests accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results, and r is the number of correct positive results divided by the number of positive results that should have been returned. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst at 0.

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

VII.COMPARISON TO BENCHMARK

The table shows the models and their accuracies:

model	accuracy
LogReg	0.702381
Naive Bayes	0.666667
KNN	0.654762
SVM	0.678571
Decision Tree	0.619048
RandomForest	0.654762
Bagging	0.630952
AdaBoost	0.547619
Gradient Boost	0.630952
XGBoost	0.666667
Catboost	0.666667

```
Epoch 6/10
43/43 [==============================] - 4s 86ms/step - loss: 0.9268 - accuracy: 0.7666 - val_loss: 0.9414 - val_accuracy: 0.74
12
Epoch 7/10
43/43 [==============================] - 4s 88ms/step - loss: 1.0116 - accuracy: 0.7346 - val_loss: 0.9328 - val_accuracy: 0.74
12
Epoch 8/10
43/43 [==============================] - 4s 90ms/step - loss: 0.9872 - accuracy: 0.7294 - val_loss: 0.9323 - val_accuracy: 0.74
12
Epoch 9/10
43/43 [==============================] - 4s 86ms/step - loss: 0.9483 - accuracy: 0.7454 - val_loss: 0.9261 - val_accuracy: 0.74
12
Epoch 10/10
43/43 [==============================] - 4s 89ms/step - loss: 0.9497 - accuracy: 0.7483 - val_loss: 0.9248 - val_accuracy: 0.74
12
```

```
class Metrics(tf.keras.callbacks.Callback):
    def __init__(self, validation_data=()):
        super().__init__()
        self.validation_data = validation_data
    def on_train_begin(self, logs={}):
        self.val_f1s = []
        self.val_recalls = []
        self.val_precisions = []
    def on_epoch_end(self, epoch, logs={}):
        xVal, yVal, target_type = self.validation_data
        if target_type == 'multi_class':
          val_predict_classes = model.predict_classes(xVal, verbose=0) # Multiclas
        else:
          val_predict_classes = (np.asarray(self.model.predict(xVal))).round() # M
        val_targ = yVal
        _val_f1 = f1_score(val_targ, val_predict_classes, average='micro')
        _val_recall = recall_score(val_targ, val_predict_classes, average='micro')
        _val_precision = precision_score(val_targ, val_predict_classes, average='m
        self.val_f1s.append(_val_f1)
        self.val_recalls.append(_val_recall)
        self.val_precisions.append(_val_precision)
        #print("— train_f1: %f — train_precision: %f — train_recall %f" % (_val_f1
        return
y_test.shape
     (84, 5)
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix, recall_sco
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=7, min_delta=
rlrp = ReduceLROnPlateau(monitor='val_loss', factor=0.0001, patience=5, min_delta=
target_type = 'multi_label'
metrics = Metrics(validation_data=(X_train, y_train, target_type))
# fit the keras model on the dataset
training_history = model.fit(X_train, y_train, epochs=100, batch_size=8, verbose=1
```

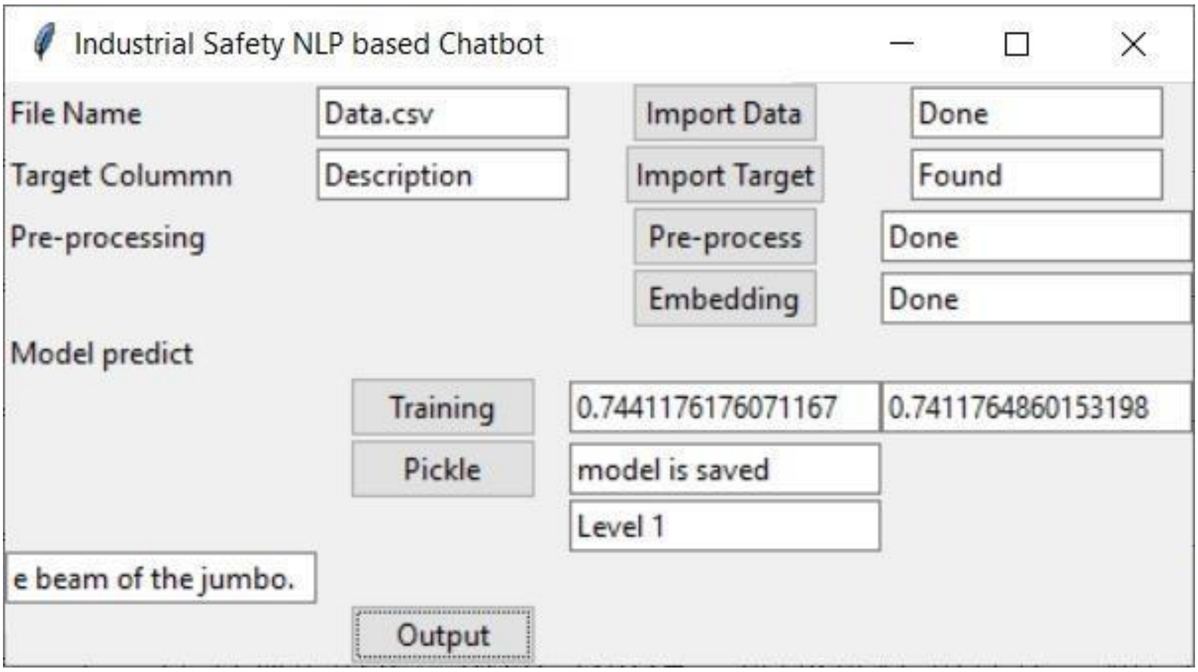
```
import string
PUNCT_TO_REMOVE = string.punctuation
def remove_punctuation(text):
    return text.translate(str.maketrans('','',PUNCT_TO_REMOVE))
text["Description"] = text["Description"].apply(lambda text: remove_punctuation(te
text.head()
```

```
# saving bilstm as model for UI
from keras.models import load_model
model.save('BiLSTM_model.h5')
```

```
# fit the keras model on the dataset
training_history = model.fit(X_train, y_train, epochs=100, batch_size=8, verbose=1
```

	Description
0	while removing the drill rod of the jumbo 08 f...
1	during the activation of a sodium sulphide pum...
2	in the substation milpo located at level 170 w...
3	being 945 am approximately in the nv 1880 cx69...
4	approximately at 1145 am in circumstances that...

GUI of the project:



VIII. IMPLICATIONS

Using this solution, we will able to highlight the risk of the accidents happened in the past and can come up with the precautions to be followed by the employees within the industry. This in turn helps us in prevention of the accidents.

Algorithms used:

1. DATA PREPROCESSING

    - Word Tokenization
    - Stop words Removal
    - N-grams
    - Lemmatization

2. DATA PROCESSING

    - Logistic Regression
    - Gaussian Naïve Bayes
    - Bidirectional LSTM
    - kNN

Tools and techniques used

- The tool we are intended to use is google colab.
- The methodology we use are Machine Learning and neural network.

IX. LIMITATIONS

- The data that we worked has only 500 rows which is not sufficient for deep learning models.
- Only one column is good features of prediction we needed more good features model.
- After building the model, it was unable to give the best results in the production.

X. CLOSING REFLECTIONS

What have you learned from the process?

- How to work on Data Science project to end-to-end.
- How to build different NLP architectures

What you do differently next time?

- Would explore more feature engineering and feature selection techniques.

XI. REFERENCES

- **Ripley, Brian D. (1996) Pattern Recognition and Neural Networks, Cambridge**
- **Bishop, C.M. (1995) Neural Networks for Pattern Recognition, Oxford: Oxford University Press.**
- **Yoshua Bengio, Ian Goodfellow and Aaron Courville wrote a book on deep learning (2016)**
- **Stanford CS224d: Deep Learning for Natural Language Processing (spring 2015) by Richard Socher**
- **A. Gargantini, J. Petke, M. Radavelli, and P. Vavassori, "Validation of Constraints Among Configuration Parameters Using Search-Based Combinatorial Interaction Testing," in Search Based Software Engineering: 8th International Symposium, SSBSE 2016, Raleigh, NC, USA, October 8-10, 2016, Proceedings, F. Sarro and K. Deb, Eds. Cham: Springer International Publishing, 2016, pp. 49–63.**