NATURAL LANGUAGE PROCESSING

CHATBOT INTERFACE

Group Members:

1. Pranav Kumar
2. Naga Supraja Ramireddy
3. Omkar Pramod Khadamkar
4. Tanmoy Bose
5. Hariom Srivastava

Table of Content

Contents

1. Project Objective

- To undertake a multi-faceted project that demonstrates your understanding and mastery of the key conceptual and technological aspects of Deep Learning.
- To develop an understanding of how challenging human-level problems can be approached and solved using a combination of tools and techniques.
- To understand current scenarios in deep learning, understand the practicalities and the trade-offs that need to be made when solving a problem in real life.

2. Project Milestones

There are 3 deliverables in the project to be covered in 3 milestones. Summary of each milestones can be observed below:

1. Milestone 1:
    a. Input: Context and Dataset
    b. Process:
        i. Import the data.
        ii. Data cleansing.
        iii. Data preprocessing (NLP Preprocessing techniques).
        iv. Data preparation - Cleansed data in .xlsx or .csv file.
        v. Design train and test basic machine learning classifiers.
        vi. Interim report.
    c. Submission:
        i. Interim report
        ii. Jupyter Notebook with all the steps in Milestone-1
2. Milestone 2:
    a. Input: Preprocessed output from Milestone-1
    b. Process:
        i. Design, train and test Neural networks classifiers.
        ii. Design, train and test RNN or LSTM classifiers.
        iii. Choose the best performing classifier and pickle it.
        iv. Final Report.
    c. Submission:
        i. Interim report
        ii. Jupyter Notebook with all the steps in Milestone-2
3. Milestone 3: [Optional]
    a. Process: Design a clickable UI based chatbot interface.
    b. Submission:
        i. Final report.
        ii. Jupyter Notebook with the addition of clickable UI based interface.

Throughout this project, we'll follow a structured approach, encompassing data acquisition, preprocessing, model development, and interface design. Our goal is to deliver a scalable, efficient solution.

Overall, our project underscores the importance of leveraging NLP techniques to enhance workplace safety standards and mitigate safety risks effectively.

Summarizing the Milestone 1

1. Summary of problem statement, data, and findings
2. Summary of the Approach to EDA and Pre-processing
3. Deciding Models and Model Building
4. How to improve your model performance?

3. Solution Approach

Creating an effective ML/DL-based chatbot utility requires a comprehensive approach that includes developing a classification model on top of the dataset. The classification model is

essential as it identifies the top predictor features and variables, which are crucial to the chatbot's performance.

To develop the classification model, we need to leverage NLP data refinement techniques to define features. These techniques help us to preprocess the raw text data and extract meaningful features that can be used as input to the model. Common NLP data refinement techniques include tokenization, stemming, lemmatization, stop word removal, and part-of-speech tagging.

Once we have refined and cleansed the text data, we can develop features for the classification model. Techniques such as bag-of-words, TF-IDF, and word embeddings can be used to transform the text data into numerical features that the model can use as input. We are choosing Word2Vec word embedding as the word embedding library, since text corpus is small. But if we had large corpus then we should have either used 'Cbow' with 'Negative-Sampling' approach with Word2Vec or choose to have Glove library.

The next step is to train and evaluate the classification model. This involves splitting the data into training and testing sets, selecting an appropriate model architecture (such as a neural network or decision tree), and tuning the model hyperparameters to optimize its performance. It's important to use evaluation metrics that are appropriate for the specific task and dataset, such as accuracy, precision, recall, F1 score, or area under the ROC curve.

In addition to the classification model, we may need to develop other components for the chatbot utility, such as a natural language understanding (NLU) module to interpret the user's intent and extract relevant information, and a dialogue management system to generate appropriate responses and maintain the conversation flow.

Overall, designing an ML/DL-based chatbot utility involves a holistic approach that includes several iterative steps, from data preprocessing and feature engineering to model selection and evaluation. By leveraging NLP data refinement techniques and classification models, we can develop a chatbot that can effectively understand and respond to user queries in a natural and conversational manner. Such a chatbot can provide a seamless and enjoyable user experience while reducing the workload of customer service representatives and improving customer satisfaction.

4. Milestone 1: Summary of problem Statement, `data and findings`

Problem Statement

With workplace accidents posing significant challenges globally, there's an urgent need to understand and mitigate these risks effectively. Our capstone project endeavors to harness the power of Natural Language Processing (NLP) to develop a cutting-edge solution: an NLP-based chatbot interface tailored specifically for addressing safety risks in industrial settings.

Data Description

The database comes from one of the biggest industry in Brazil and in the world. It is an urgent need for industries/companies around the globe to understand why employees still suffer some injuries/accidents in plants. Sometimes they also die in such an environment.

Data is downloaded from Kaggle.

Link to download the dataset: Dataset from Kaggle

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

Columns description

- Data: timestamp or time/date information
- Countries: which country the accident occurred (anonymized)
- Local: the city where the manufacturing plant is located (anonymized)
- Industry sector: which sector the plant belongs to
- Accident level: from I to VI, it registers how severe was the accident (I 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.

Data File Format

- Data is downloaded in xlsx format.
- Sheet Name: 'Data Set - industrial_safety_an'
- File-Name: 'Industrial_safety_and_health_database_with_accidents_description.xlsx'

Data Findings Summary

Step 1: Import the data

- We have imported data in pandas dataframe.
- It is having 425 rows and 11 columns of data.
- There are 3 data types
  - Int: for `unnamed:0` column the identifier column
  - dateTime: Incident Reporting time
  - object: For all 9 columns

Step 2: Data Cleaning

- dropping the identifier column from data as it won't be adding any value in analysis.

- Renaming meaningful column headers to enhance data clarity.
    - 'Data'-> 'Date'
    - 'Countries'->'Country'
    - 'Genre'->'Gender'
    - 'Employee or Third Party'->'Employee Type'
- Checking for missing values
    - No missing values found
- Checking for duplicate values
    - 7 duplicate rows found, deleted duplicate entry.
- Converting date field as categorical columns - year, month, year-quarter and year-month. For further analysis.
- Checking unique values for columns.
- Renaming values for Critical Risk
    - "\nNot applicable" -> "Not applicable"
    - "Pressurized Systems / Chemical Substances" -> "Pressurized Systems/Chemical Substances"

Step 3: Observation after data import and data cleaning

- Observations on unique values and describe()
    - Date: We have data starting from Date - 2016-01-01 to 2017-07-09 which covers close to 18 months of data.
    - Country: There are 3 unique countries('Country_01', 'Country_02' and 'Country_03') represented in the dataset, with 'Country_01' being the most frequent, appearing 248 times.
    - Local: There are 12 unique locations represented in the dataset(Local_01 to Local_12), with 'Local_03' being the most frequent, appearing 89 times.
    - Industry Sector: There are 3 unique sectors(Mining, Metals and Others) represented in the dataset, with 'Mining' being the most frequent, appearing 237 times.
    - Accident Level: There are 5 unique accident levels(I,II,III,IV,V) represented in the dataset, with ‘Level I’ being the most frequent, appearing 309 times.
    - Potential Accident Level: There are 6 unique potential accident levels(I,II,III,IV,V,VI) represented in the dataset, with level "IV" being the most frequent, appearing 141 times.
    - Gender: There are 2 unique genders(Male, Female) represented in the dataset, with 'Male' being the most frequent, appearing 396 times.
    - Employee Type: There are 3 unique types of employees(Employee, Third Party, Third Party (Remote)) represented in the dataset, with 'Third Party' being the most frequent, appearing 185 times.
    - Critical Risk: There are 33 unique critical risks(Others, remains of choco, cut, burn, Power Lock, Electrical Installation, ...) represented in the dataset, with 'Others' being the most frequent, appearing 229 times.
    - Description: There are 411 unique descriptions of accidents in the dataset.

5. Milestone 1: Summary of the Approach to EDA and Pre-processing

Summary of the Approach to EDA

We will start with univariate analysis, which includes the values distributions for each categorical columns except for textual columns as the textual columns will be analyzed after next step where we will do the NLP Preprocessing. Followed by Bivariate and multivariate analysis of the categorical columns using crosstabs.

For data visualization we will be using:

- Labelled Bar-plots with percentages (Using custom function on top of sns.countplot())
- Pie-Charts with percentages, where values are very skewed for few values.
- Plotting on pandas crossTabs for textual 2 column relationships.

Based on plots and comparisons draw observations and if needed make suitable changes to dataset.

Step 3: Summary of *Exploratory Data Analysis (Non text columns - except Description Column)*

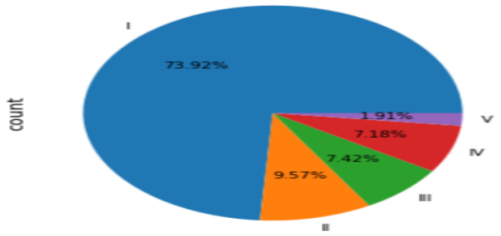
- Univariate Analysis(Plot and Observations for-)
    - Accident Level
    - Potential Accident Level
    - Critical Risk
    - Industry Sector
    - Gender
    - Employee Type
    - Country
    - Locality
    - Extracted date columns
- Bivariate Analysis using crosstab and stacked Charts for categorical columns
    - Country and Local
    - Country and Industry Sector
    - Accident Level and Potential Accident Level
    - Critical Risk and Accident Level
    - Critical Risk and Potential Accident Level
- Multivariate Analysis using crosstab and stacked Charts for categorical columns

- 'Accident Level' and ['Country','Industry Sector','Gender', 'Employee Type', 'year']
- 'Potentail Accident Level' and ['Country','Industry Sector','Gender', 'Employee Type']
- 'Gender' and ['Industry Sector','Employee Type', 'Country', 'Local']
- 'Industry Sector' and ['Gender','Employee Type', 'Country', 'Local']

Sample of few Univariate Analysis

Distribution of incident's Accident Level

```
[ ]  # Visualize a Pie-chart and print percentage
     data['Accident Level'].value_counts().plot(kind='pie',figsize=(6, 4),autopct='%1.2f%%', fontsize=8)
     plt.show()
```

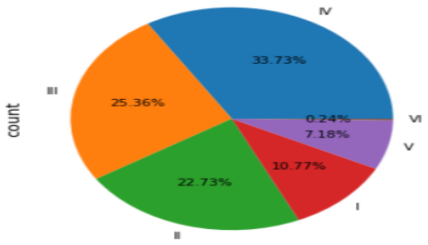


Observations for `Accident Level` values distrubtion

- Most of incidents are from `Accident Level - I (73.9%)` i.e these are least severe incidents.
- Most Severe incidents of `Accident Level - V (1.9%)`
- No data for `Accident Level - VI`
- Data Distribution is very imbalanced with ~74% Type-I and 1.91% Type V

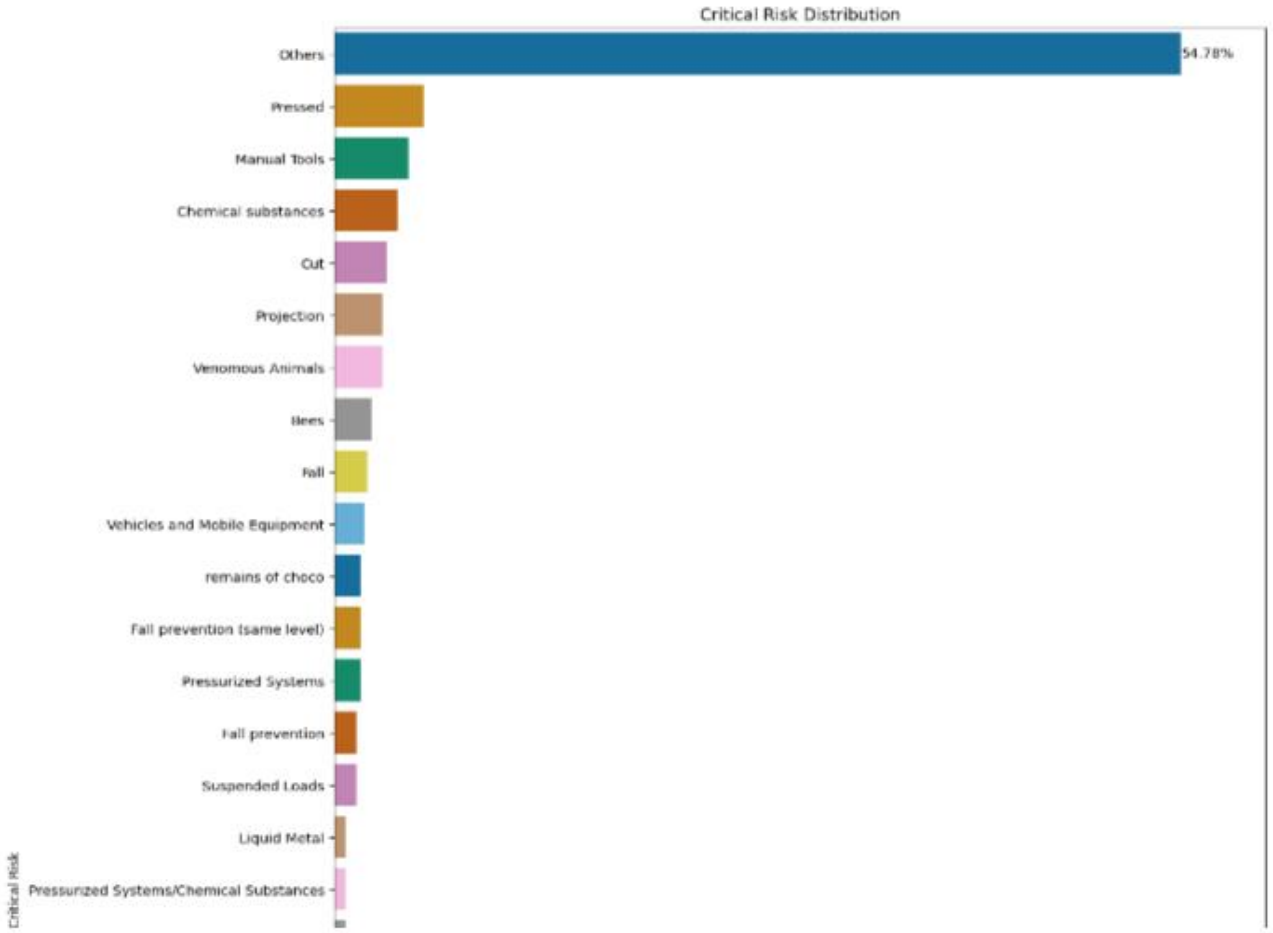
Distribution of incident's Potential Accident Level

```
[ ]  data['Potential Accident Level'].value_counts().plot(kind='pie',figsize=(6, 4),autopct='%1.2f%%', fontsize=8)
     plt.show()
```



Observations for `Potential Accident Level` values distrubtion

- Here the values distribution are better than `Accident Level`.
- `Type VI` and `Type V` values are undersampled, especially `Type-VI with 0.24%` followed by `Type-V 7.18%` and `Type-I(10.77%)`.
- `Type-II, Type-III and Type-IV` are comparatively evenly distributed.

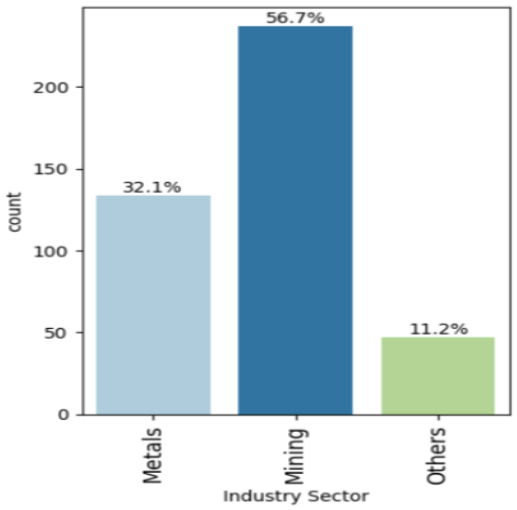


Observations for `Critical Risk` values distrubtion

- Its highly biased towards value - `Others (54.78%)`. This data could have been more explored and further split during data collection or there are huge number of smaller categories coming from other industry sector(excluding Mining and Metals) , may be better to confirm with data collector.
- Specific values like - `Pressed`, `Manual Tools`, `Checmical Substances`, `Cut` have noticebale percentages in range of `5.75` to `4.07`. These might be contributor of Mining and Metals industry.

Distribution of `Industry Sector`

```
[ ]  labeled_barplot(data, "Industry Sector", perc=True)
```



Observations for `Industry Sector` values distrubtion

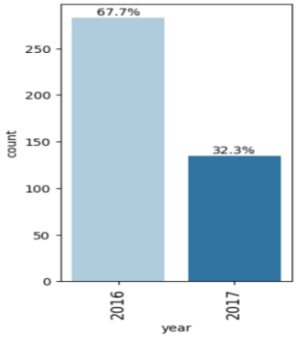
- Majority of the incidents in the data are from
  - `Mining Industry Sector (56.7%)`,
  - followed by `Metals Industry Sector (32.1%)` and
  - rest of are contributed from `Others Industry Sector (11.2%)`

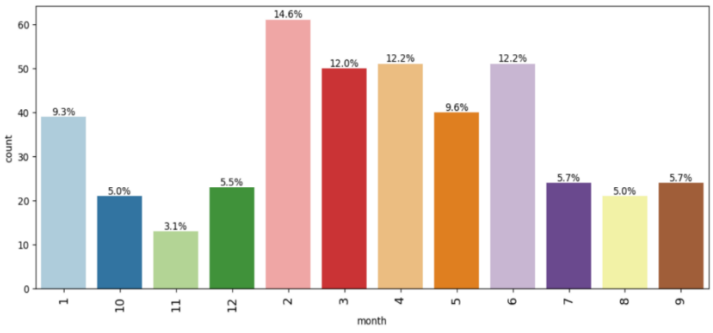
Observations for `Employee Type` values distrubtion

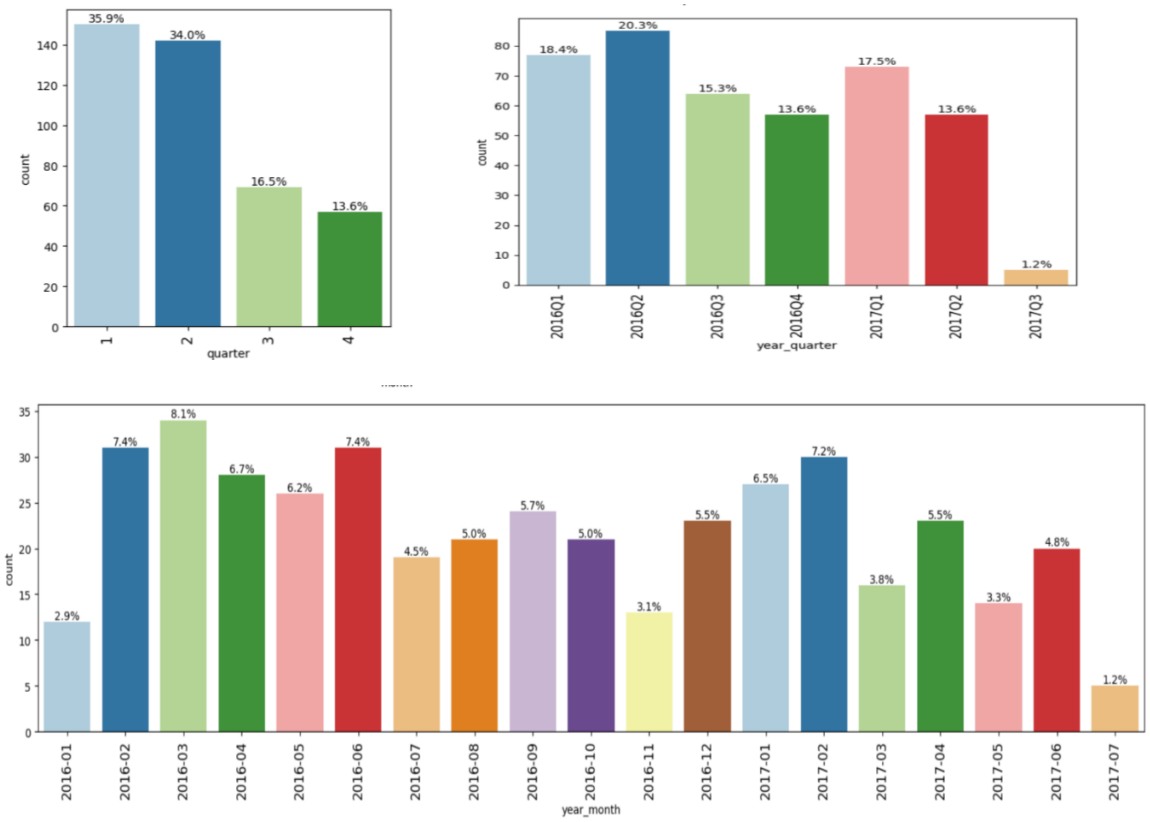
- Third Party employees are more involved in accidents with 44.3%
- Followed by core Employee having 43.6% accident involvement
- Third Party remote employee have 13.2% accident involvement.
- Accidents involvement for Remote Employee looks significant if their overall employment percentage for companies are less compared core Employee and Third Party employee. Same for Third Party employee. This need to be investigated from overall Employee database, which is not provided here.

Distribution of extracted date columns

```
[ ]  labeled_barplot(data, 'year', perc=True)
     labeled_barplot(data, 'month', perc=True)
     labeled_barplot(data, 'year_month', perc=True)
     labeled_barplot(data, 'quarter', perc=True)
     labeled_barplot(data, 'year_quarter', perc=True)
```







Distribution of Date column extracted in Year, Month, Quarter

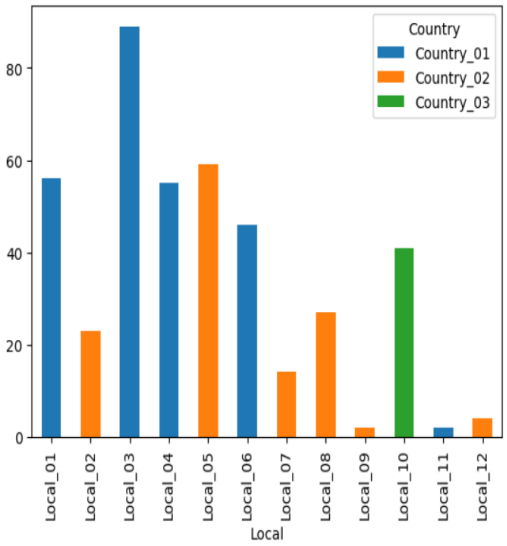
- Year 2016 all months data are available but for year 2017 only 1st 7 months of data is available.
- Except for 1st Month of Year 2017 all subsequent months have reduced incident reporting when compare to months of year 2016
- Year 2016 incidents(67.7%) have reduced to 1/3rd in next year 2017(32.3%). Looks measures have been taken to improve the incidents among employees also have less months of data for year 2017.
- 1st 6 months/(1st 2 quarters) of year have 2/3rd of incidents reported and only 1/3rd are reported in last 2 quarters.
- Incident counts have decreasing trend based on increasing quarters.

Sample of few Bivariate Analysis

Local And Country

```
loc_country = pd.crosstab(data['Local'], data['Country'])
print(loc_country)
loc_country.plot.bar(stacked=True)
plt.show()
```

```
Country   Country_01  Country_02  Country_03
Local
Local_01          56           0           0
Local_02           0          23           0
Local_03          89           0           0
Local_04          55           0           0
Local_05           0          59           0
Local_06          46           0           0
Local_07           0          14           0
Local_08           0          27           0
Local_09           0           2           0
Local_10           0           0          41
Local_11           2           0           0
Local_12           0           4           0
```

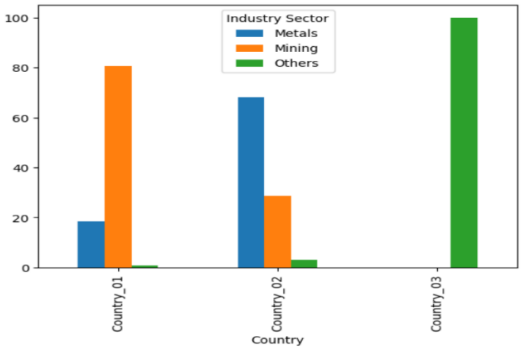


Observations

- Local_01, Local_03, Local_04, Local_06 and Local_11 situated in Country_01
- Local_02, Local_05, Local_07, Local_08, Local_09 and Local_12 situated in Country_02
- Only single location - Local_10 situated in Country_03
- Local_03 have highest incidents around 90.

```
country_industry = pd.crosstab(
    data['Country'], data['Industry Sector'],
    normalize='index').round(4)*100
print(country_industry)
country_industry.plot.bar(stacked=False)
plt.show()
```

```
Industry Sector  Metals  Mining  Others
Country
Country_01        18.55   80.65    0.81
Country_02        68.22   28.68    3.10
Country_03         0.00    0.00  100.00
```



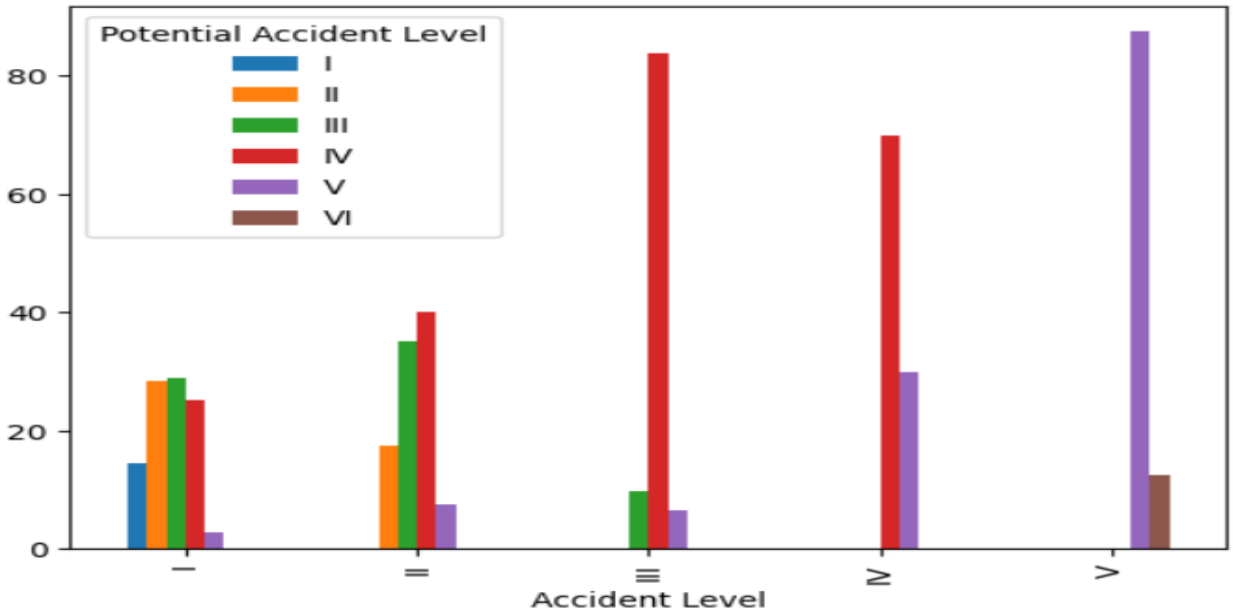
Observations

- `Country_03` has inident reports from `Others` sector only i.e no `Mining(0%)` or `Metals(0%)` sector
- `Country_01(0.81%)` and `Country_02(3.10%)` has minimal inident reports from `Others` sector.
- `Country_01` has majority incident reported from `Mining(80.65%) sector` while for `Country_02` majority incident reported from `Metals(68.2%) sector`
- `Country_3` looks to have no mining and metal sector for the company.

Bivariate analysis for Accident Level and Potential Accident Level

```
[ ]  accLevel_potentialAccLevel = pd.crosstab(
         data['Accident Level'], data['Potential Accident Level'],
         normalize='index').round(4)*100
     print(accLevel_potentialAccLevel)
     accLevel_potentialAccLevel.plot.bar(stacked=False)
     plt.show()
```

Potential Accident Level	I	II	III	IV	V	VI
Accident Level						
I	14.56	28.48	28.80	25.24	2.91	0.0
II	0.00	17.50	35.00	40.00	7.50	0.0
III	0.00	0.00	9.68	83.87	6.45	0.0
IV	0.00	0.00	0.00	70.00	30.00	0.0
V	0.00	0.00	0.00	0.00	87.50	12.5

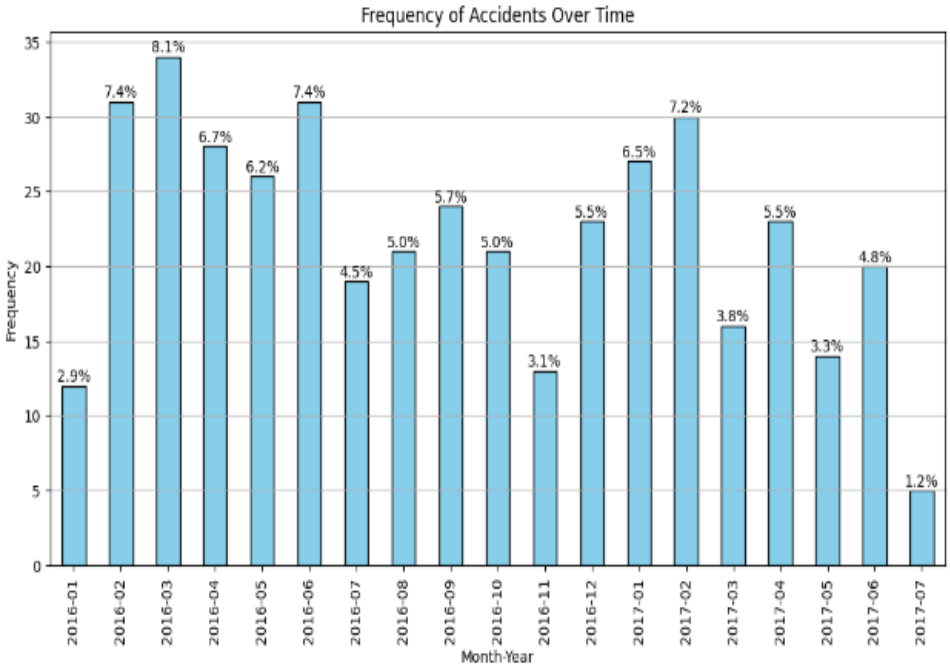


Observations on `Accident Level` w.r.t `Potential Accident Level`

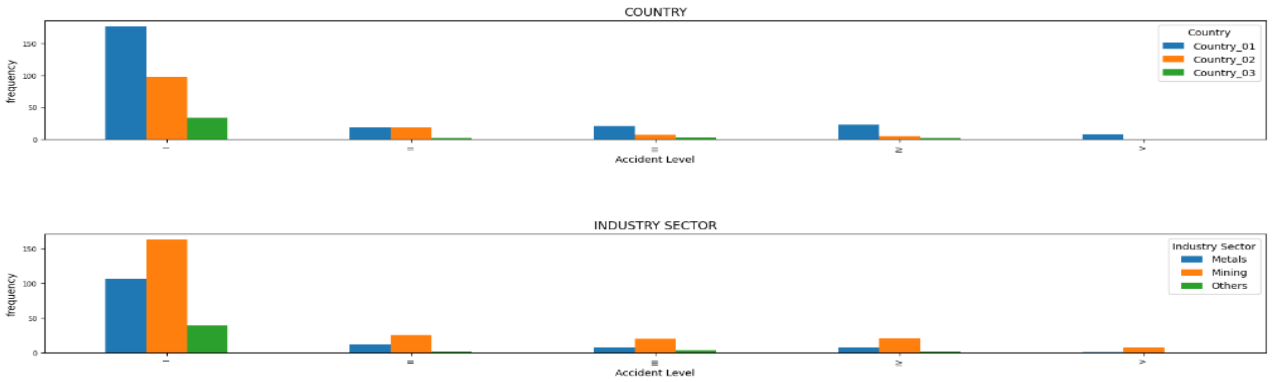
- Only 40 `Type-I Accident Level` also have `Type-I Potential Accident Level` i.e No change in actual and potential.
- About 260+ `Type-I Accident Level` have increased `Potential Accident Level of II to IV` and `very small ~3% to Level-V`
- Most of Critical `Accidents Level of Type V` remain same in `Potential Accident Level 87.5 %` but `12.5% are moved to life threating Type VI of Potential Accident Level`

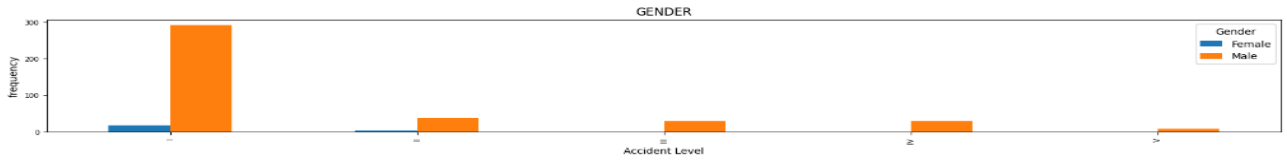
Month-Year Wise accidents analysis

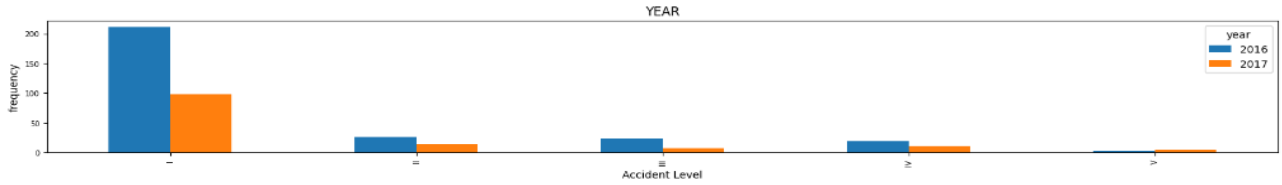
```
date_freq = data['year_month'].value_counts().sort_index()


plt.figure(figsize=(12, 6))
bars = date_freq.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Frequency of Accidents Over Time')
plt.xlabel('Month-Year')
plt.ylabel('Frequency')
plt.grid(axis='y')

for bar in bars.patches:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height + 0.3, f'{height / len(data) * 100:.1f}%', ha='center')

plt.show()
```



Sample of few Multivariate Analysis





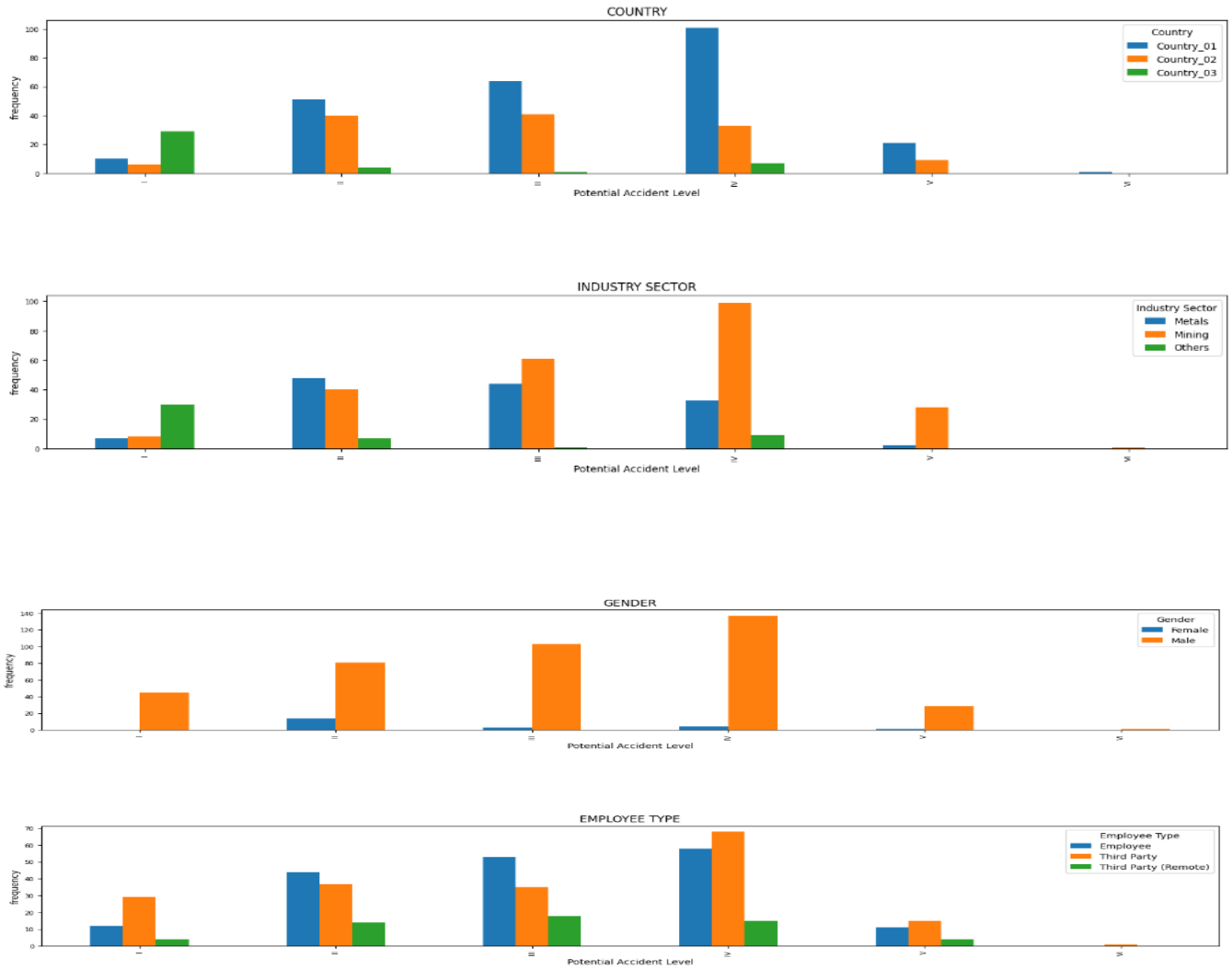


Mutivariate analysis for `Potential Accident Level`

```
[ ] cols = ['Country', 'Industry Sector', 'Gender', 'Employee Type']
    stacked_charts_for_multiCategoricalVar(cols, 'Potential Accident Level', data, 1)
```

```
 Normalized CrossTab for Potential Accident Level and Country
 Country                  Country_01  Country_02  Country_03
Potential Accident Level
I                             22.22       13.33       64.44
II                            53.68       42.11        4.21
III                           60.38       38.68        0.94
IV                            71.63       23.40        4.96
V                             70.00       30.00        0.00
VI                           100.00        0.00        0.00

 Normalized CrossTab for Potential Accident Level and Industry Sector
 Industry Sector          Metals  Mining  Others
Potential Accident Level
I                         15.56   17.78   66.67
II                        50.53   42.11    7.37
III                       41.51   57.55    0.94
IV                        23.40   70.21    6.38
V                          6.67   93.33    0.00
VI                         0.00  100.00    0.00

 Normalized CrossTab for Potential Accident Level and Gender
 Gender                   Female    Male
Potential Accident Level
I                          0.00  100.00
II                        14.74   85.26
III                        2.83   97.17
IV                         2.84   97.16
V                          3.33   96.67
VI                         0.00  100.00

 Normalized CrossTab for Potential Accident Level and Employee Type
 Employee Type            Employee  Third Party  Third Party (Remote)
Potential Accident Level
I                           26.67        64.44                  8.89
II                          46.32        38.95                 14.74
III                         50.00        33.02                 16.98
IV                          41.13        48.23                 10.64
V                           36.67        50.00                 13.33
VI                           0.00       100.00                  0.00
```

Stacked-Charts for Categorical Variables in the dataframe



Summary of the Approach to NLP Preprocessing

Steps to be performed for cleaned text data preparation using custom functions.

- Handling contractions.
- Converting to lower cases.
- Remove special characters, punctuations and numbers using regex. Keep only alphabets and spaces.
- Remove extra whitespaces like – tabs, newlines etc.
- Remove English Stop-Words, based on nltk.corpus module. Also added additional stop-words based on units of numeric values used in text like `a.m`, `p.m`.
- Lemmatize each words based on pos-tagging – NOUN.

Data Visualizations for Textual Column -`Description`

- WordClouds for NLP based cleaned data.
- N-grams words to capture most important frequented words.

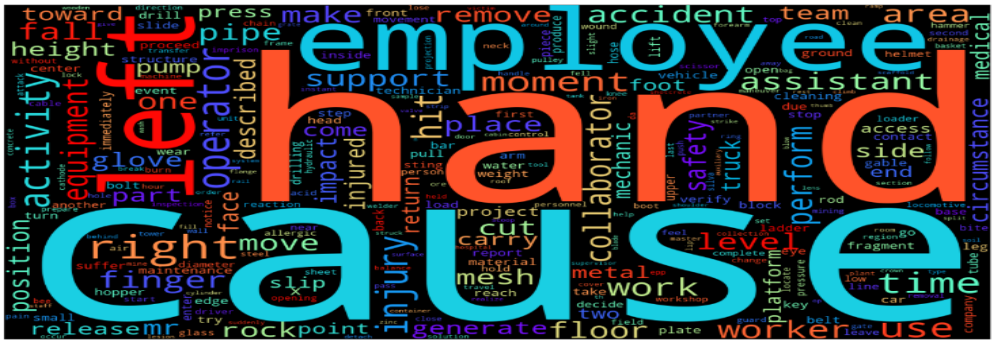
EDA for the cleaned Text(WordClouds and n-grams)

Word clouds

- Overall:

Word cloud- Overall

```
from wordcloud import WordCloud

wordcloud = WordCloud(width = 1500, height = 800, random_state=0, background_color='black', colormap='rainbow', \
                      min_font_size=5, max_words=300, collocations=False).generate(splitedAllWordsAsText)
plt.figure(figsize=(15,10))
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```



- Top Words
  - Employee, hand, cause, left, activity, worker, height, fall, accident, collaborator, activity, equipment, remove, operator are the prominent words.

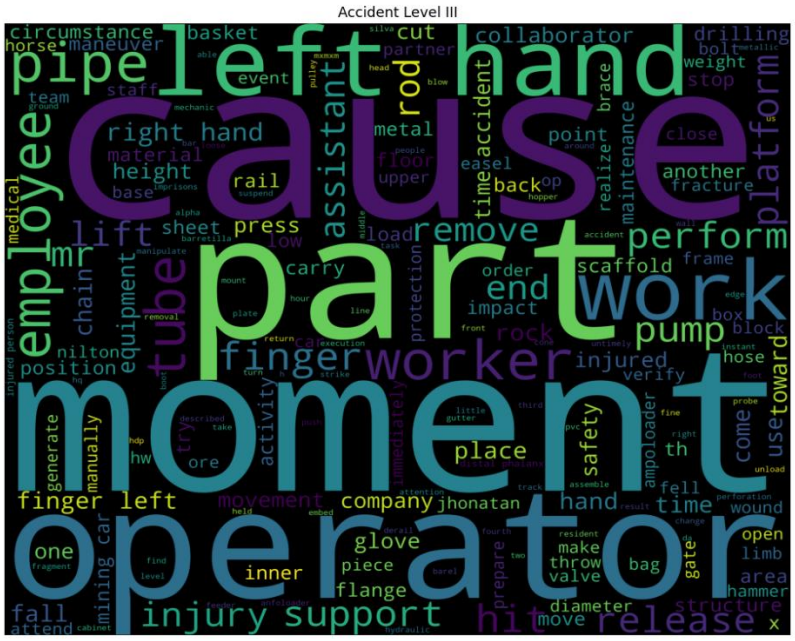
- Accident Level I:

- Most Frequent Words
  - Operator, activity, employee, collaborator, right, hand, floor, time, accident, cut, slip etc. are the prominent words.

- Accident Level II:

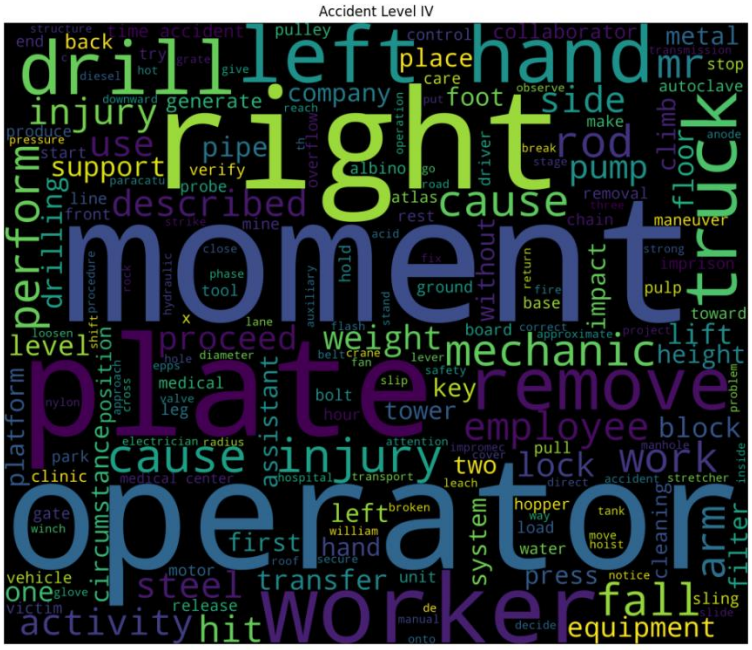
- Most Frequent Words
  - Pipe, employee, hit, pipe, cut, pressure, move, cause, release, finger are the prominent words.

- Accident Level III:



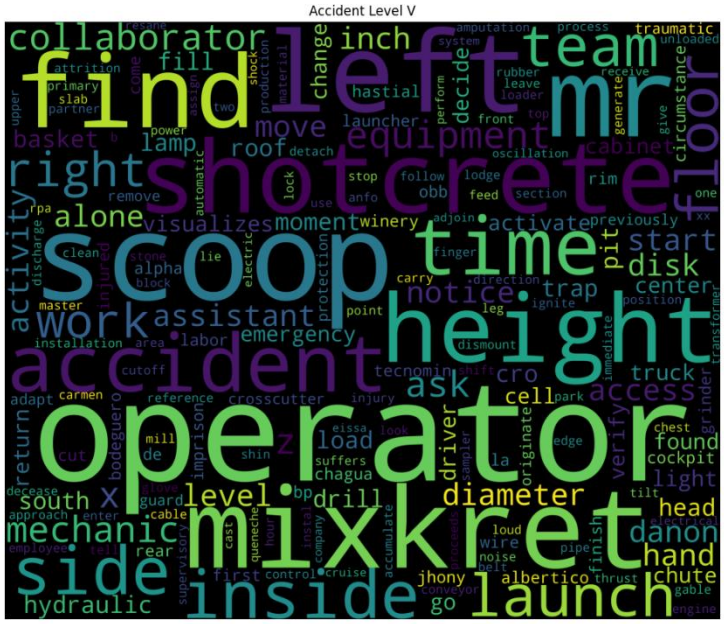
- Most Frequent Words
  - Operator, moment, finger, tube, hand, left, injury, support, left, hand etc. are the prominent words.

- Accident Level IV:



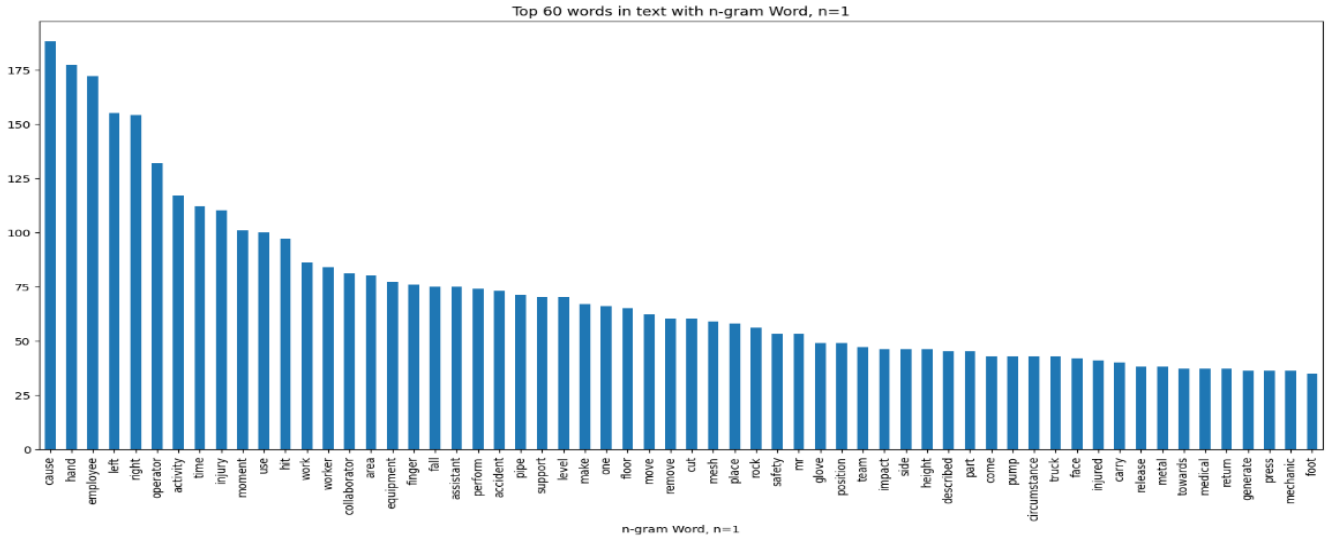
  - Most Frequent Words
    - Operator, plate, moment, steel, truck, right, hit, activity, drill, left, hand, fall, weigh, mechanic etc. are the prominent words.

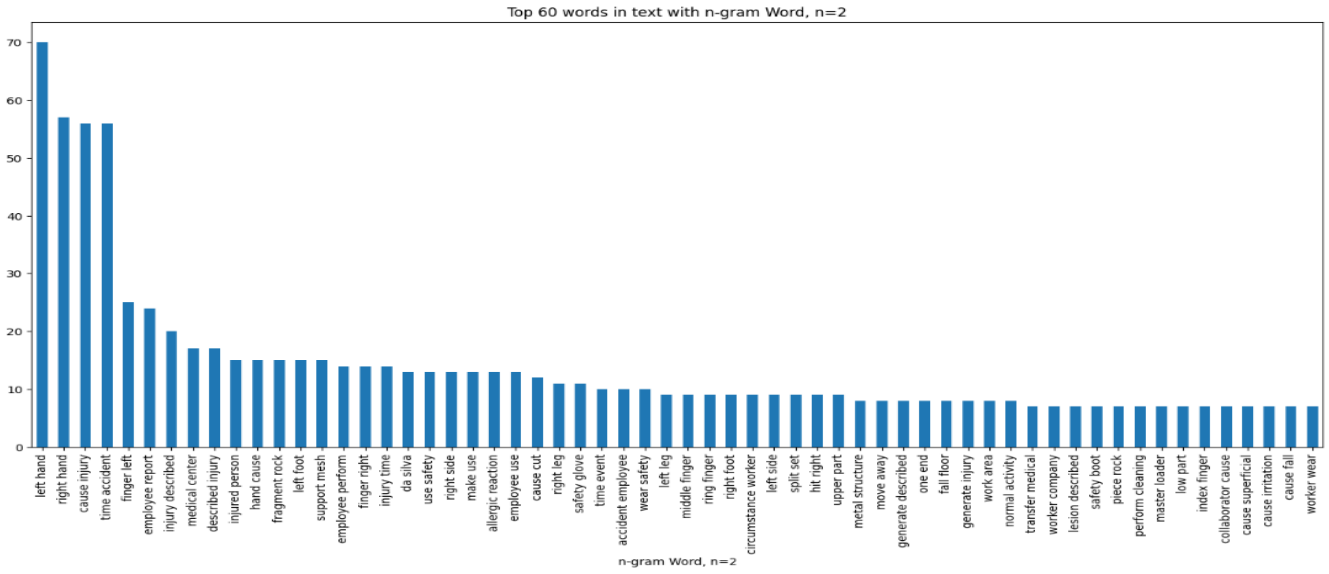
- Accident Level V:

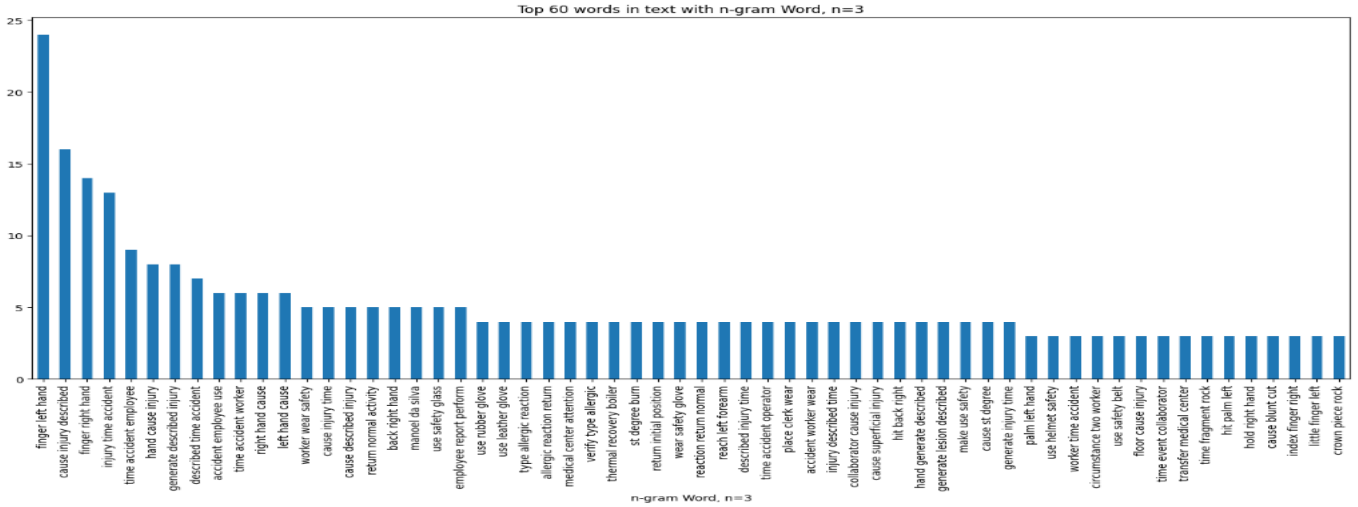


  - Most Frequent Words
    - Shot, create, left, find, mix, accident, scoop, hydraulic, left, equipment etc. are the prominent words.

N-Grams


Top 60 words in text with n-gram Word, n=1
n-gram Word, n=1


Top 60 words in text with n-gram Word, n=2
n-gram Word, n=2


Top 60 words in text with n-gram Word, n=3
n-gram Word, n=3

Observations on n-grams and WordClouds

Accident Descriptions have top words including - body movements specially hand/finger. Also the risk and cause. It also has human involvements

- Hands/Finger/Palm Movement:
  - left, right
- Other Body part Movements:
  - leg, foot, eye
- Equipment:
  - safety gloves, leather glove, safety helmet
- cause/accident:
  - cause, hit, cut, burn, fall, irritation, lose balance
- Human involvement:
  - Employee, party, collaborator, assistant

Summary of the Approach to Feature Engineering and Word-Embeddings

- Categorical Features Encoding
  - Ordinal Columns -> Numbers
  - Nominal Columns -> Dummies/One-hot
- PreProcess the values to have only alphaNumeric and underscore
- Use shortend Prefixes while encoding
- Dropping Date and related fields as - It acts as incident reporting time, no direct relation on when the incident can happen.
- Word2Vec embedding for feature extraction on Description
- Join the extracted features from Description to a single dataframe
- Save the cleansed file with feature extraction in a xlsx file format.

6. Milestone 1: Deciding Models and Model Building

Summary of Steps performed for model building and tuning

- Load the saved dataset
- Choosing target variable as 'Accident Level'.
- Test Train Split 75%-25%
- Normalization and Scaling
- Upscaling of train dataset for target variable minority classes using
    - RandomOverSampler
    - SMOTE
- Training and Testing basic machine learning classifiers
    - LOGISTIC RESGRESSION
    - KNN
    - SVM
    - Decision Tree Classifier
    - RANDOM FOREST
    - Bagging Classifer
    - AdaBoost Classifier
    - GradientBoosting Classifer
- Observation on how to improve performances of base Model
    - Try testing over resampled data
    - Try hyper-parameter tuning with 3-fold validation - GridSearchCV over original Train data
    - Try hyper-parameter tuning with 3-fold validation - GridSearchCV over SMOTE upscaled Train data
- Summary of all models tried

Scores of base models with original Data without any up-samplings

	Name	Train Accuracy	Test Accuracy	Test F1_Level-I	Test F1_Level-II	Test F1_Level-III	Test F1_Level-IV	Test F1_Level-V
1	LogisticRegression	73.16	74.29	85.71	0.00	0.0	0.00	0.0
2	KNeighborsClassifier	76.68	73.33	85.23	18.18	0.0	14.29	0.0
3	SVC	73.48	75.24	85.87	0.00	0.0	0.00	0.0
4	DecisionTreeClassifier	99.68	55.24	73.55	10.00	0.0	0.00	0.0
5	RandomForestClassifier	99.68	75.24	85.87	0.00	0.0	0.00	0.0
6	BaggingClassifier	95.85	71.43	83.80	0.00	0.0	0.00	0.0
7	AdaBoostClassifier	60.38	58.10	74.51	13.33	0.0	25.00	0.0
8	GradientBoostingClassifier	99.68	68.57	82.29	0.00	0.0	0.00	0.0

Observations on base-models with unsampled original training dataset:

- Most of classifiers are able to make test predictions for majority classes-I and performed poorly for non-majority classes.

- None of classifiers able to classify even few datapoints of all classes, at max only up to 3 classes.
- All models have some degree of overfitting, except for LR, KNN and SVC
- Based on only overall test accuracy - Random Forest classifier did best with 75.24% but have 85.87% F1 score only for Level-I and performed poorly on other levels.

Scores of base models with up-samplings of Minority classes over train-set by Random-Over-Sampler

	Name	Train Accuracy	Test Accuracy	Test F1_Level-I	Test F1_Level-II	Test F1_Level-III	Test F1_Level-IV	Test F1_Level-V
1	LogisticRegression	62.35	21.90	33.96	23.53	4.55	8.70	10.0
2	KNeighborsClassifier	93.30	39.05	58.06	17.39	7.14	14.29	0.0
3	SVC	68.00	24.76	42.48	12.50	6.25	0.00	0.0
4	DecisionTreeClassifier	99.48	55.24	72.15	0.00	0.00	10.00	0.0
5	RandomForestClassifier	99.48	75.24	86.34	0.00	0.00	0.00	0.0
6	BaggingClassifier	99.39	65.71	79.53	0.00	18.18	0.00	0.0
7	AdaBoostClassifier	30.96	10.48	2.47	0.00	30.77	18.82	0.0
8	GradientBoostingClassifier	99.48	67.62	81.61	0.00	0.00	0.00	0.0

Observations on base-models with up-sampled ROS training dataset

- Most of classifiers are able to make test predictions for majority classes(I, II) and performed poorly for non-majority classes.
- Best classifier having few correct predictions based on F1-Scores for all levels and not just for Majority class is LogisticRegression with overall accuracy of 23.81% but is able to classify few datapoints of the minority classes as well (I-35.19%, II-11.76%, III-5.41%, IV-21.05%, V-13.79%). Whereas all other models focused on majority classes.
- KNN and SVC also able to make correct predictions based on F1-Scores for all levels except for Level-V
- All models have some degree of overfitting.
- Based on only overall test accuracy - Random Forest classifier did best with 75.24% but have 85.87% F1 score only for Level-I and performed poorly on other levels.

Scores of base models with up-samplings of Minority classes over train-set by Smote

	Name	Train Accuracy	Test Accuracy	Test F1_Level-I	Test F1_Level-II	Test F1_Level-III	Test F1_Level-IV	Test F1_Level-V
1	LogisticRegression	64.17	38.10	56.92	16.67	6.67	0.00	0.0
2	KNeighborsClassifier	90.00	39.05	59.38	9.09	8.33	6.90	0.0
3	SVC	68.70	47.62	65.25	20.00	6.45	15.38	0.0
4	DecisionTreeClassifier	99.91	45.71	62.41	9.09	21.05	8.00	0.0
5	RandomForestClassifier	99.91	61.90	78.79	0.00	0.00	0.00	0.0
6	BaggingClassifier	99.91	55.24	73.89	0.00	0.00	0.00	0.0
7	AdaBoostClassifier	45.39	33.33	50.42	12.50	10.91	11.76	0.0
8	GradientBoostingClassifier	99.91	53.33	71.34	0.00	0.00	0.00	0.0

Observations on base-models with up-sampled ROS training dataset

- Most of classifiers are able to make test predictions for majority classes(I, II)

- Best classifier having few correct predictions based on F1-Scores for all levels and not just for Majority class is DecisionTreeClassifier with overall accuracy of 46.67% but is able to classify few datapoints of the minority classes as well (I-64.66%, II-9.52%, III-9.52%, IV-21.43%, V-28.57%). Whereas all other models focused on majority classes.
- All models have some degree of overfitting.
- Based on only overall test accuracy - Random Forest classifier did best with 63.81 but have 80% F1 score only for Level-I and performed poorly on other levels.

Hyper-Parameter Tuning

Scores of best tuned models with Hyperparameters as given by grids over original Test data

	Name	Train Accuracy	Test Accuracy	Test F1_Level-I	Test F1_Level-II	Test F1_Level-III	Test F1_Level-IV	Test F1_Level-V
1	LogisticClassifier_Tuned	99.68	59.05	76.73	0.0	0.0	0.0	40.0
2	KNeighborsClassifier_Tuned	73.48	74.29	85.25	0.0	0.0	0.0	0.0
3	SVC_Tuned	73.48	75.24	85.87	0.0	0.0	0.0	0.0
4	DecisionTreeClassifier_Tuned	73.48	75.24	85.87	0.0	0.0	0.0	0.0
5	RandomForestClassifier_Tuned	73.48	75.24	85.87	0.0	0.0	0.0	0.0
6	BaggingClassifier_Tuned	99.68	74.29	85.25	0.0	0.0	0.0	0.0
7	AdaBoostClassifier_Tuned	73.48	75.24	85.87	0.0	0.0	0.0	0.0
8	GradientBoostingClassifier_Tuned	73.48	75.24	85.87	0.0	0.0	0.0	0.0

Observations on base-models with up-sampled ROS training dataset:

- With Original training data with tuned models overfitting is reduced for most of classifiers except for LogisticClassifer and Bagging Classifer.
- Because the original test data set is not balanced, most of classifer improved the accuracy over Majority class - Level-I
- Most of them reached the accuracy of
  - 75.24%(SVC_Tuned, DecisionTreeClassifier_Tuned, RandomForestClassifier_Tuned, AdaBoostClassifier_Tuned and GradientBoostingClassifier_Tuned)
  - 74.29%(KNeighborsClassifier_Tuned and BaggingClassifer_Tuned)
  - 59.05% for Logistic Classifer
- But all performed poorly on classifying minority classes

Scores of best tuned models with Hyperparameters as given by grids over SMOTE Test data

	Name	Train Accuracy	Test Accuracy	Test F1_Level-I	Test F1_Level-II	Test F1_Level-III	Test F1_Level-IV	Test F1_Level-V
1	LogisticClassifier_Tuned	99.91	57.14	74.32	11.11	20.00	10.53	40.00
2	KNeighborsClassifier_Tuned	70.96	26.67	38.89	21.62	11.76	8.33	0.00
3	SVC_Tuned	67.22	52.38	70.67	10.00	7.14	0.00	0.00
4	DecisionTreeClassifier_Tuned	47.65	6.67	0.00	17.39	13.33	4.88	9.09
5	RandomForestClassifier_Tuned	45.22	9.52	2.50	12.50	9.30	21.82	0.00
6	BaggingClassifier_Tuned	99.91	60.95	76.36	13.33	0.00	0.00	0.00
7	AdaBoostClassifier_Tuned	35.13	5.71	0.00	0.00	13.33	0.00	7.55
8	GradientBoostingClassifier_Tuned	77.83	28.57	44.64	23.08	6.06	0.00	13.33

Observation for Tuned Models with SMOTE training set

- With balanced training data the performed on classifying minority classes improved.
- Best classifier to classify all classes is 'Logistic Regression' with overall accuracy of 57.14% but is able to classify few datapoints of the minority classes as well (I-74.32%, II-11.11%, III-20.00%, IV-10.53%, V-40.00%).
- Overfitting increased for most of models is continued, test accuracy reduced

Summary for all models

- Best 3 model that can classify all classes from test data are:
  - LogisticRegression
  - (C=100, max_iter=50, multi_class='multinomial',penalty='none', random_state=42, solver='newton-cg')
  - With SMOTE UpSampled train dataset.
  - Overall accuracy 57.14%
  - MutiClassF1-Score: I-74.32%, II-11.11%, III-20.00%, IV-10.53%, V-40.00%
- DecisionTreeClassifier
  - (random_state=42)
  - With SMOTE UpSampled train dataset
  - Overall accuracy 46.67%
  - MutiClassF1-Score: I-64.66%, II-9.52%, III-9.52%, IV-21.43%, V-28.57%
- LogisticRegression
  - (multi_class='multinomial', random_state=42)
  - With ROS UpSampled train dataset
  - Overall accuracy 23.81%
  - MutiClassF1-Score: I-35.19%, II-11.76%, III-5.41%, IV-21.05%, V-13.79%
- We tried Sampling of minority classes using both ROS and SMOTE
- We tried Hyperparameter tuning with Gridsearch having CV=3 for unbalanced original data and balanced train data from SMOTE.
- Next Steps is to try out the ANN and LSTM models to verify the performances on all classes.