AIML Capstone Project Report

CHATBOT INTERFACE

Group 3

Members:

Abhishek Base

Manjunath R

Shipra Jain

Aditi Gangar

Tharoh Krishnamurti

Table of Contents

Introduction (Industry and Problem Statement)

The project focuses on industrial safety, specifically developing a chatbot utilizing Natural Language Processing (NLP). It addresses the critical issue of understanding and preventing accidents and injuries in industrial environments. The project's relevance is underscored by the ongoing occurrences of such incidents, sometimes leading to fatalities, in various industries globally. This NLP-based solution aims to enhance safety protocols and awareness in industrial settings.

Objective of the Project

The primary objective is to design a machine learning (ML) and deep learning (DL) based chatbot utility. This tool is intended to assist professionals in identifying and highlighting safety risks based on incident descriptions. By analyzing accident reports and related data, the chatbot will provide insights into potential hazards and preventive measures, thereby contributing to improved safety standards in industrial workplaces.

Dataset Description (Including EDA)

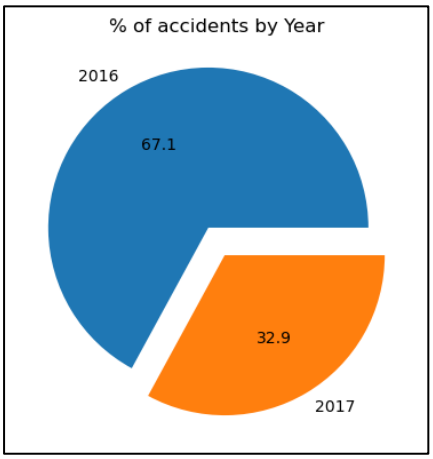
The dataset comprises accident records from 12 different plants across three countries. Key features include timestamps of accidents, anonymized country and city information, industry sectors, accident severity levels, potential accident levels, gender of the individuals involved, whether they are employees or third parties, critical risks involved, and detailed accident descriptions. This comprehensive dataset forms the basis for exploratory data analysis (EDA) to understand patterns and insights related to industrial accidents.

Columns description:

- Data: timestamp or time/date information
- Countries: which country the accident occurred (anonymised)
- Local: the city where the manufacturing plant is located (anonymised)
- 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.

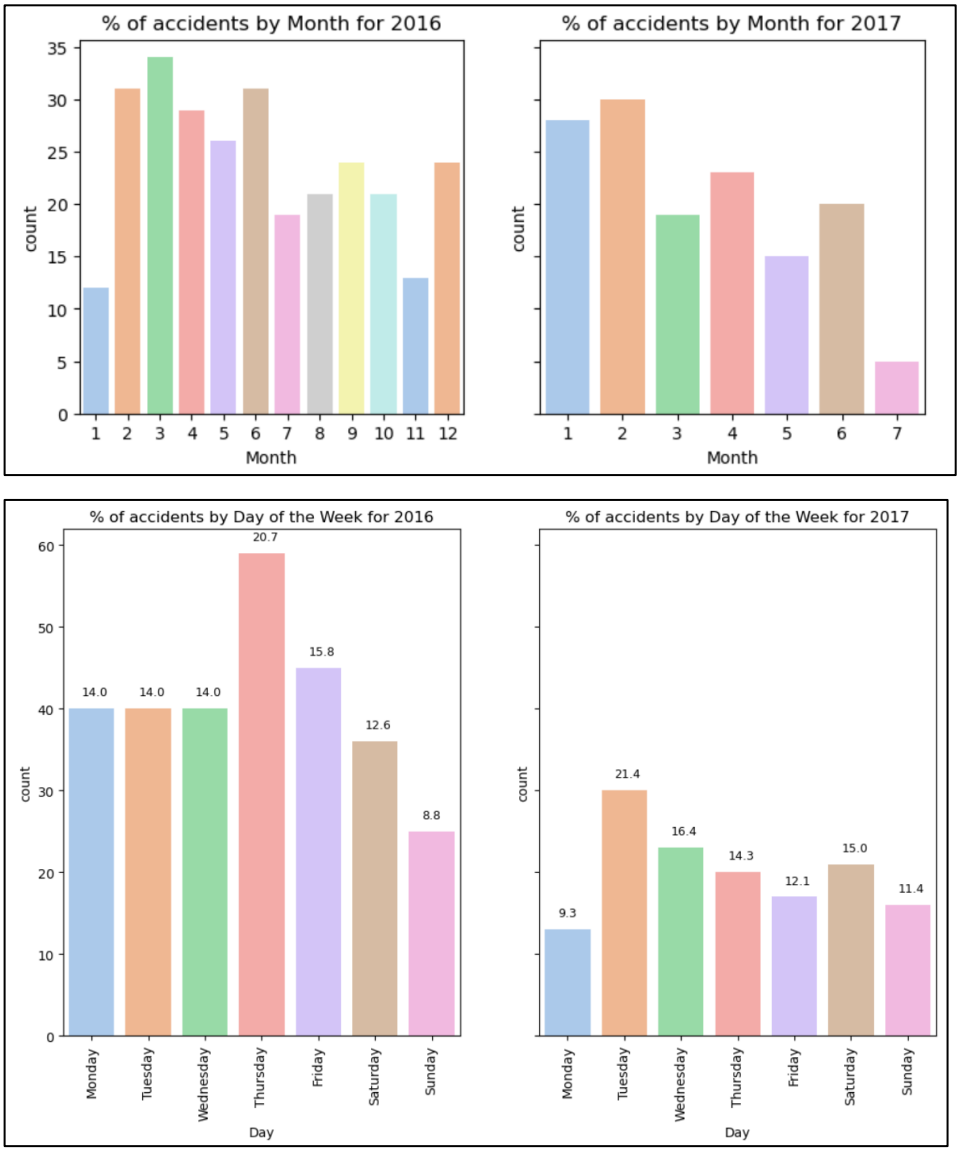
Univariate Analysis

1. **Analysis by Year:**



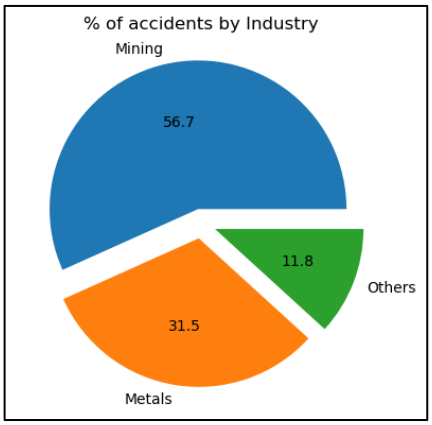
We utilized a pie chart to represent the percentage of accidents by year. It is important to note that the data is only available until 9th July 2017, which may not provide conclusive evidence to determine if accidents increased or decreased in 2017. A comparison of the first half of both years shows a reduction of about 20% in accidents in 2017.

2. **Analysis by Month and Day:**



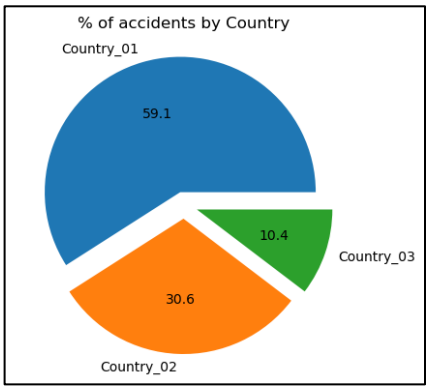
Count plots for 2016 and 2017 display the distribution of accidents by month and day. The data suggests that for each month from January to June, the number of accidents reduced in 2017. Approximately 70-80% of accidents occurred on weekdays, possibly influenced by to work-related travel.

3. **Industry Sector and Country Analysis:**



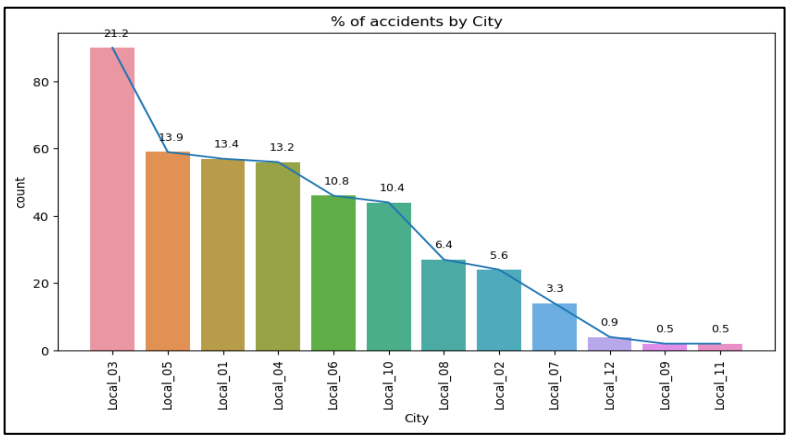
The Mining industry experiences the highest number of accidents, followed by the Metals industry.

4. **Country Analysis:**



Country_01 has the highest number of accidents, contributing to nearly 90% of the total accidents.

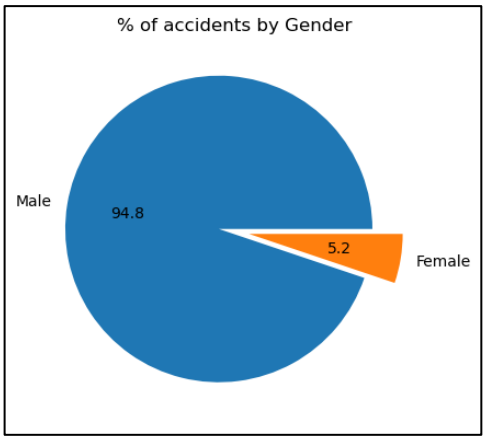
5. **City Analysis:**



A count plot and line plot together indicate that about 61% of the accidents occur in

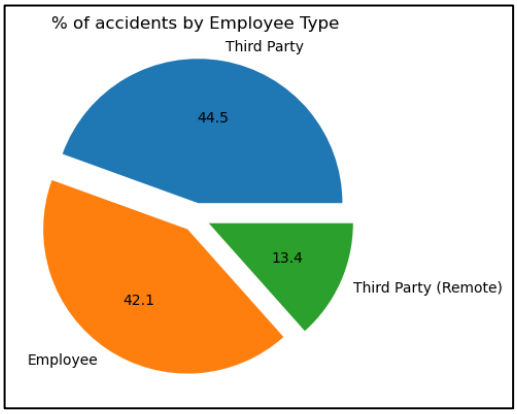
plants located in Local_01, Local_03, Local_04, and Local_05. Local_03 alone accounts for 21% of the accidents. The bottom most 6 cities - Local_02, Local_07, Local_08, Local_09, Local_11, Local_12 cities witnessed only 17% of the total accidents.

6. **Gender Analysis:**



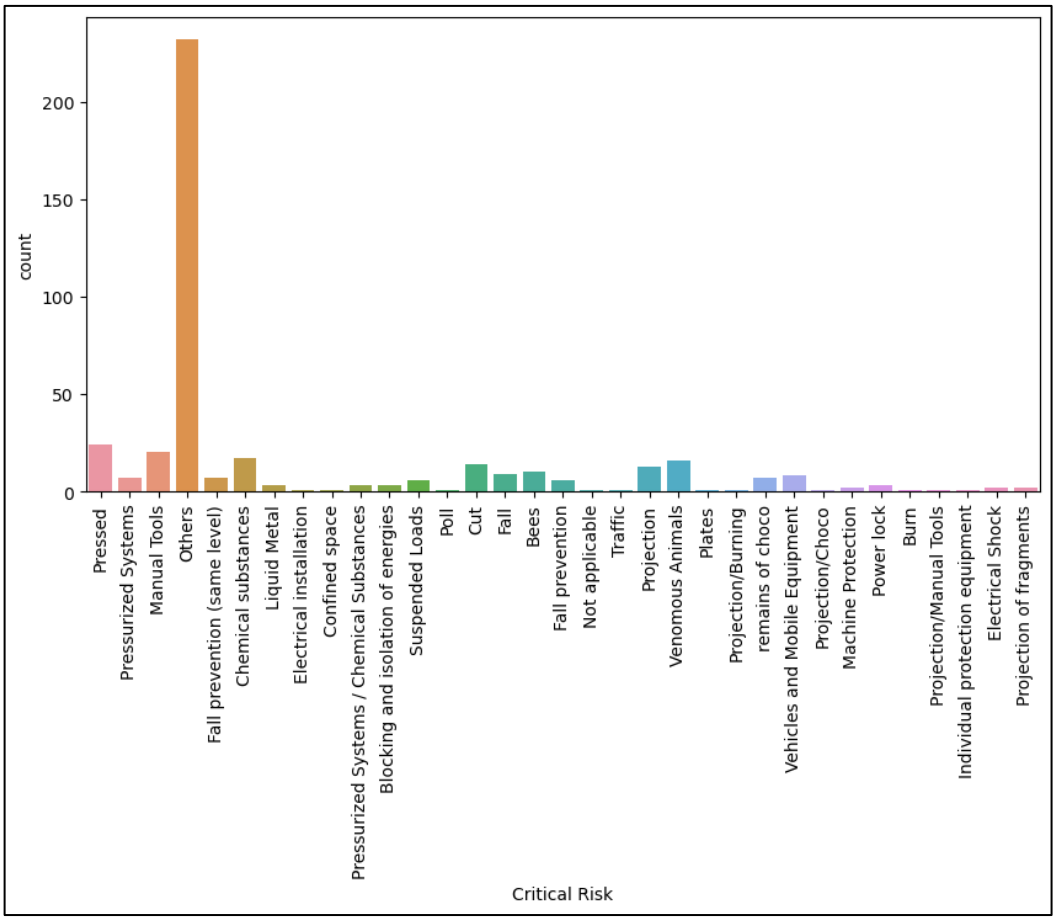
Only about 5% of the accidents involve females, potentially reflecting the gender distribution in industrial jobs.

7. **Employee Type Analysis:**



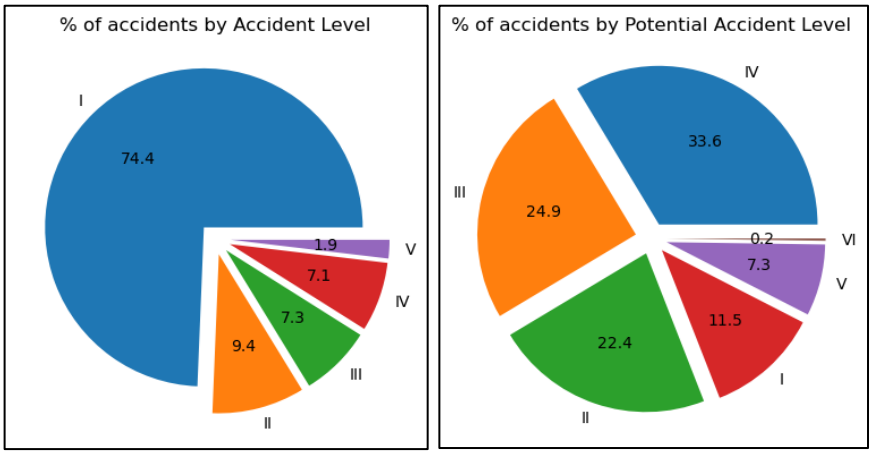
Third Party Remote workers encounter the fewest accidents, with overall 58% of the accidents involving Third Party employees, which could potentially be the case as they'd largely be working off-site.

8. **Critical Risk Analysis:**



The 'Others' category encompasses about 50% of the risks in the dataset. Common risks include Pressed, Manual Tools, Chemical Substances, Venomous Animals, Projection, and Cut.

9. **Accident Level & Potential Accident Level Analyses:**

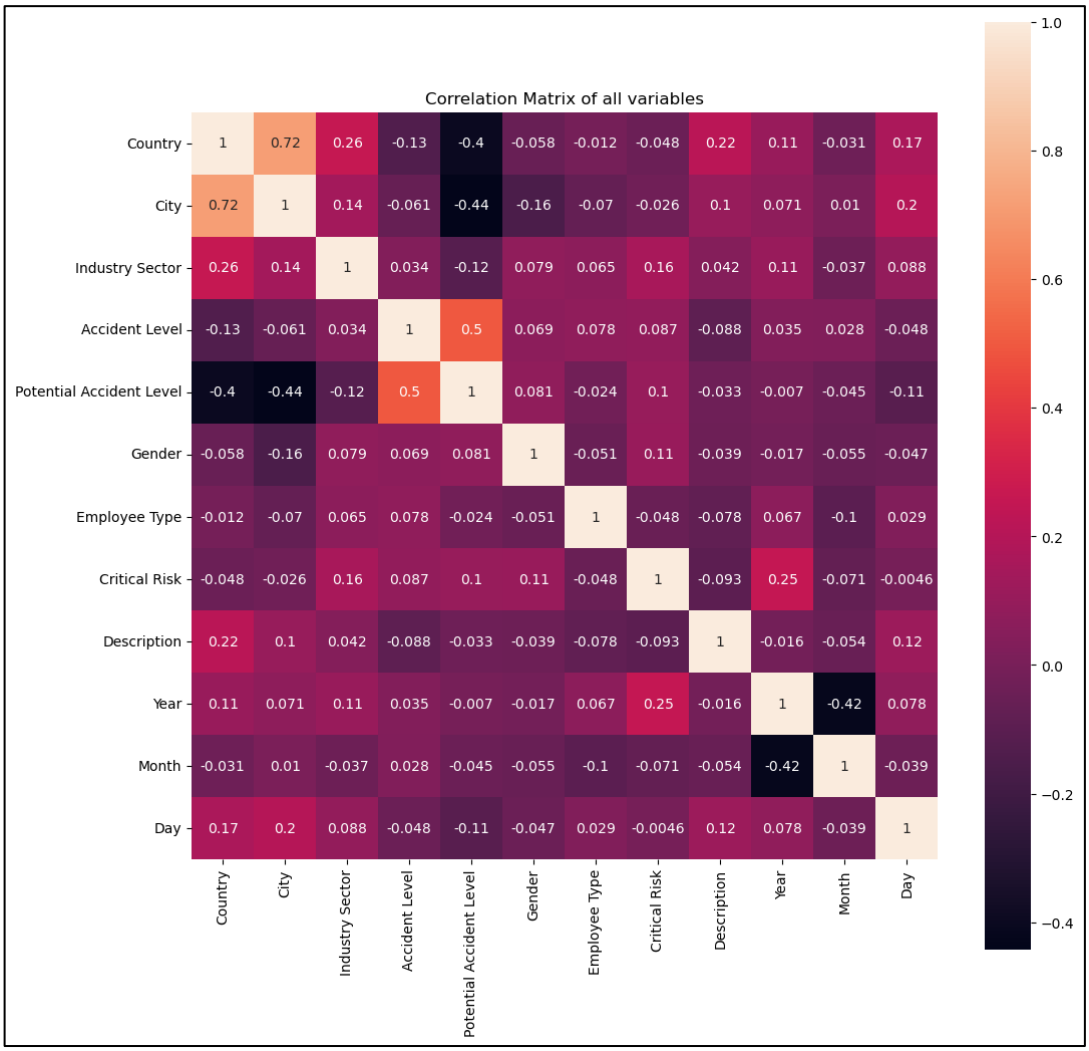


The majority of accidents are classified as Level I, occurring due to minor incidents. High risk or severe accidents (Level V) account for only 2%.

Approximately 81% of accidents could potentially be moderately severe (Levels II, III, and IV), and only 7.5% could have been the most severe (Level V). To simplify the comparison between Accident Level and Potential Accident Level, we adjusted Level VI to Level V in the data, as Level VI has only 1 accident recorded.

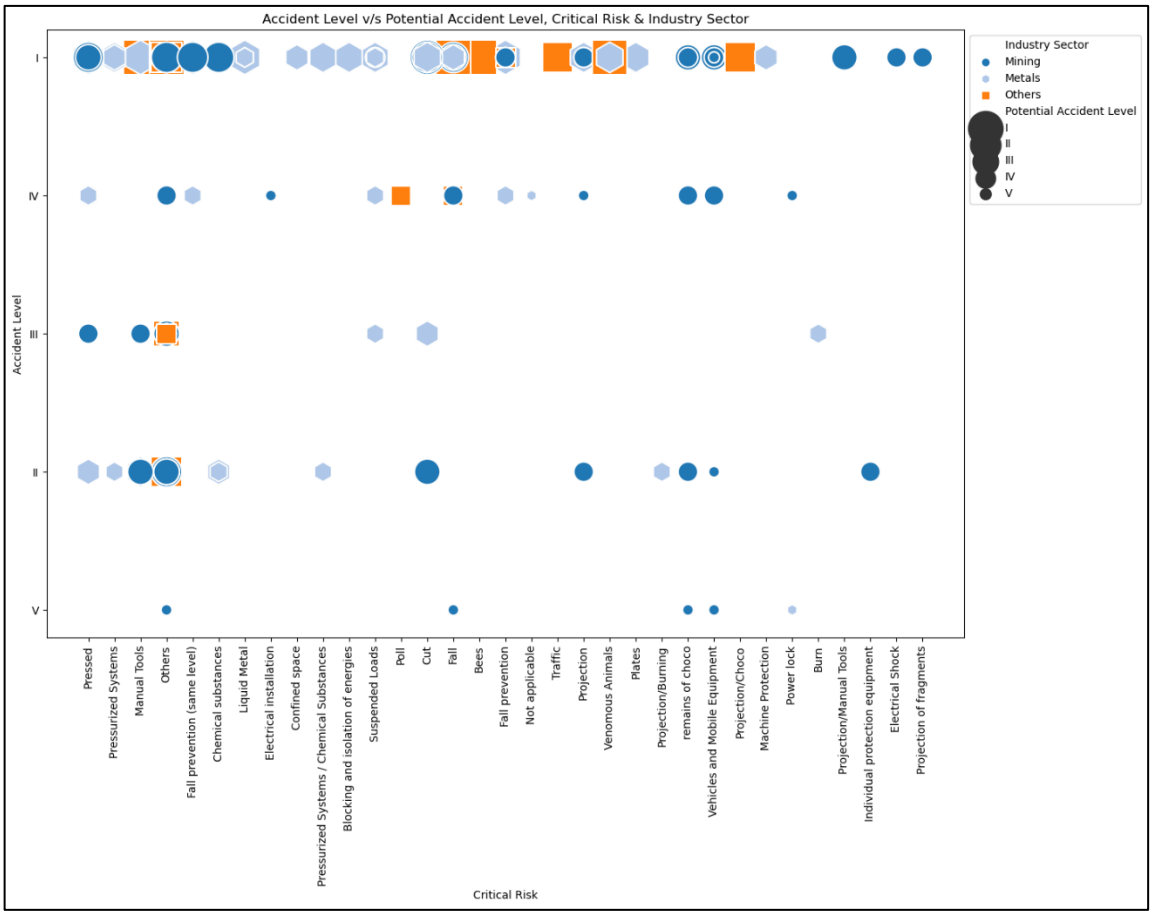
Multivariate Analysis

10. **Correlation Analysis:**



A heatmap of the correlation matrix shows the highest correlation between City and Country. Accident Level and Potential Accident Level share only a 50% correlation. Other variables exhibit low to moderate correlation.

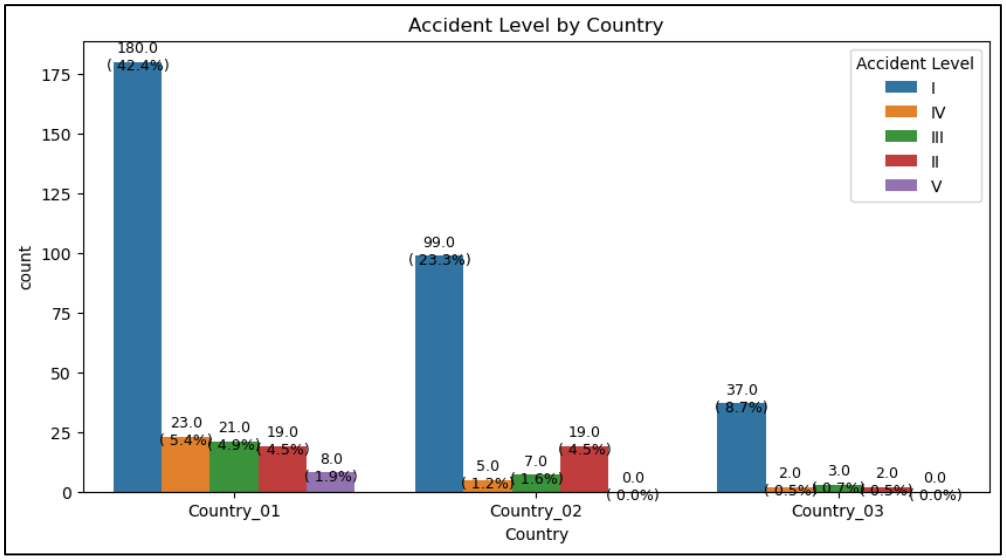
11. **Scatter Plot Analysis:**

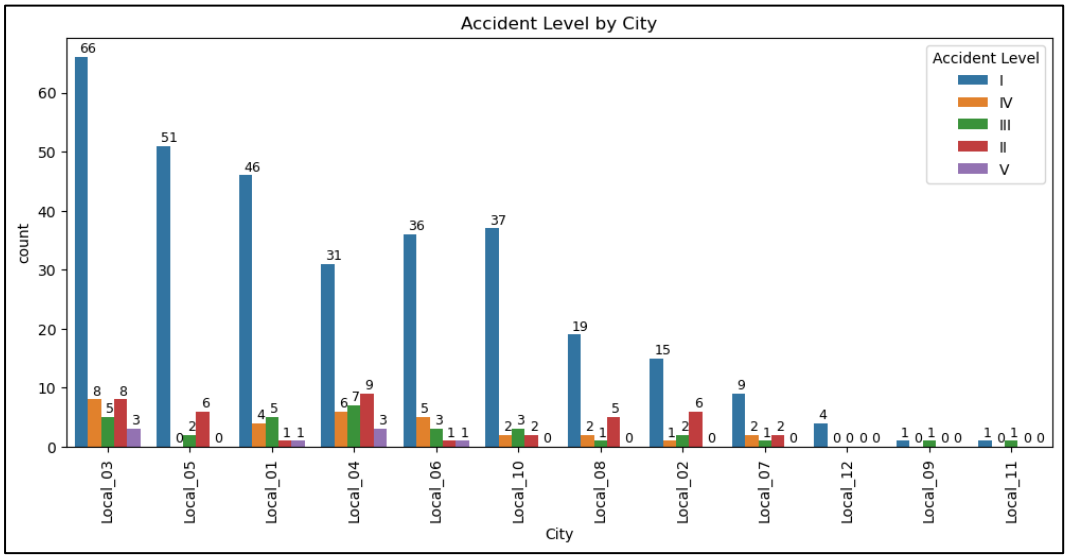


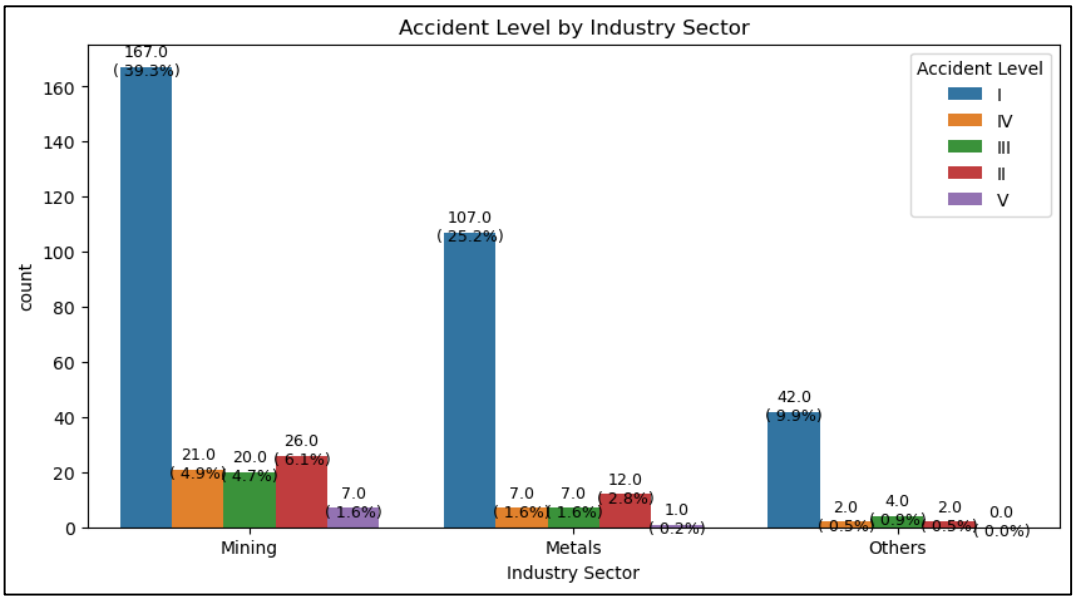
A scatter plot comparing Accident Level, Potential Accident Level, Critical Risk, and Industry Sector reveals that most accidents are of the least severity (Level I), of which many could have been potentially more severe (Levels II & III). The most severe accidents (Level V) are predominantly in the Mining industry, except for 'Power Lock' in the metals industry. The following risks are only seen in Metals industry: Pressurized Systems, Liquid Metal, Confined Space, Chemicals Substances, Blocking and isolation of engines, Suspended Loads, Burn, Machine Protection, with majority of them observed in Level I accidents.

Bivariate Analysis

12. Accident Level by Country, City & Industry Sector

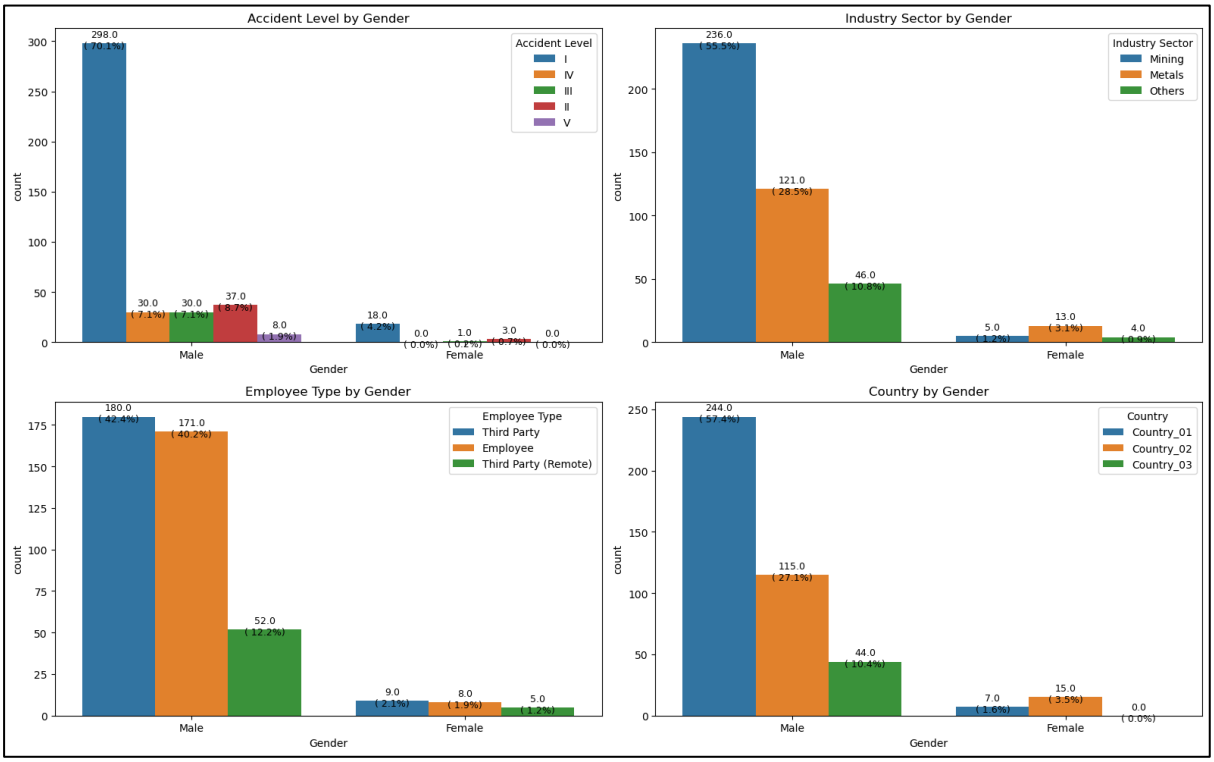






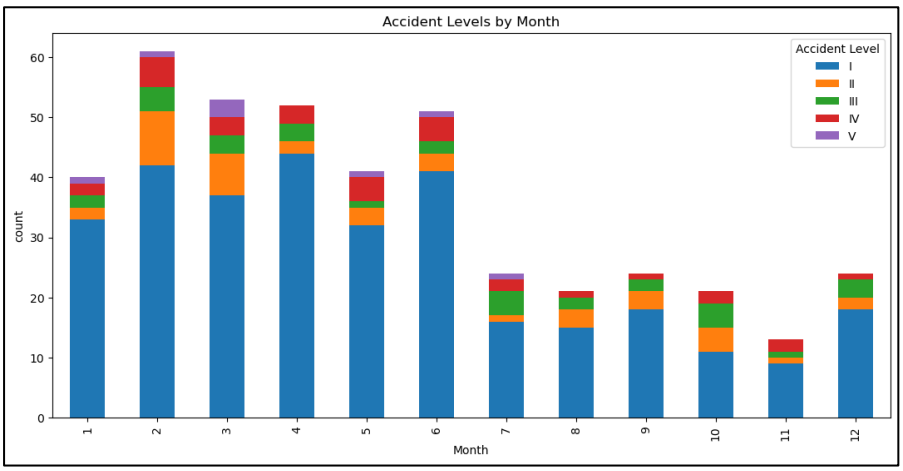
Count plots by Country, City, and Industry Sector indicate trends in accident levels across different categories. Country_01 has the majority of accidents, primarily of lower severity (Level I). Local_9, Local_11, Local_12 cities have only 2-4 accidents, could be due to low population or low staff in these plants, but we do not have any data to confirm that. Although country_02 has maximum number of plants but witnesses 2nd highest number of accidents, majority of which being least severity accidents (Level I). Plants in Local_01, Local_03, Local_04, & Local_06 cities only have faced the highest severity accidents (Level V). The risks associated could be Fall, Vehicles & Mobile Equipment, Remains of Choco & Power Lock and some are categorized under 'Others'.

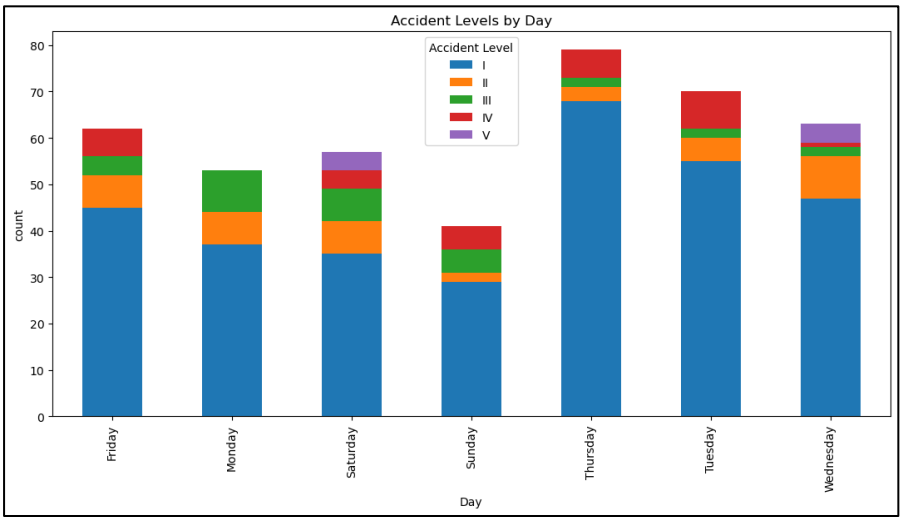
13. Different factors by Gender:



There are no females who suffered from an accident in Country_03, but even in other 2 countries together, only 6% of the accidents have females involved. About 60% of the females met with accident in Metals industry, and 23% in Mining industry, while the rest in 'Others'. 87% of the men, with almost equal distribution, are either direct Employees or Third Party employees.

14. Accident Levels by Months & Days:





Majority of potentially least severe accidents (level I) are seen in first half of the year. But these numbers don't include data post 9th July 2017, hence this does not provide a full view. Majority of moderate severity accidents (levels II, III, IV) are seen in February & March. Monday has witnessed only Level I, II & II accidents, while Level IV accidents have mostly occured on Tuesday, Thursday, Friday & Sunday.

Final Observations:

- The most severe accidents occur primarily in March, particularly on Wednesdays and Saturdays.
- The analysis highlights the concentration of moderate to high severity accidents in specific cities and industries, with various industry & non-industry specific risks involved.
- Gender analysis shows a significant difference in the distribution of accidents, with a higher occurrence among men.

Proposed Approach and Theory (Data Cleaning, Model Selection for Machine Learning Model)

The approach involves several stages, starting with data cleansing to ensure quality and reliability. This is followed by preprocessing techniques specific to NLP, preparing the data for analysis. The subsequent phase involves designing and testing basic machine learning classifiers, moving towards more complex neural network classifiers, including RNN or LSTM models. The selection of the most suitable model is based on their performance, aiming for the most effective tool for the chatbot's functionality.

In the "Data Preprocessing (NLP Preprocessing Techniques)" section of the analysis, we performed several steps to process and analyse the 'Description' column of the dataset. These steps are crucial for understanding the causes of accidents detailed in the descriptions. Here's a breakdown of the steps taken:

Data Copying and Initial Display:

Created a copy of the dataset and displayed the initial few entries of the 'Description' column.

Checking for Non-ASCII Characters:

Implemented a function to check if there are any characters beyond the ASCII character set in the descriptions. The outcome indicated no such characters were present.

Checking for HTML Tags and URLs:

Implemented a function to check for the presence of HTML tags or URLs in the text. The result showed no HTML tags or URLs.

Text Preprocessing:

- Lowercased all text to maintain uniformity.
- Handled accented characters and Unicode using Unicode normalization.
- Retained only alphanumeric characters, removing any special characters.
- Removed English stop words to focus on relevant words.
- Applied lemmatization to reduce words to their base or dictionary form.

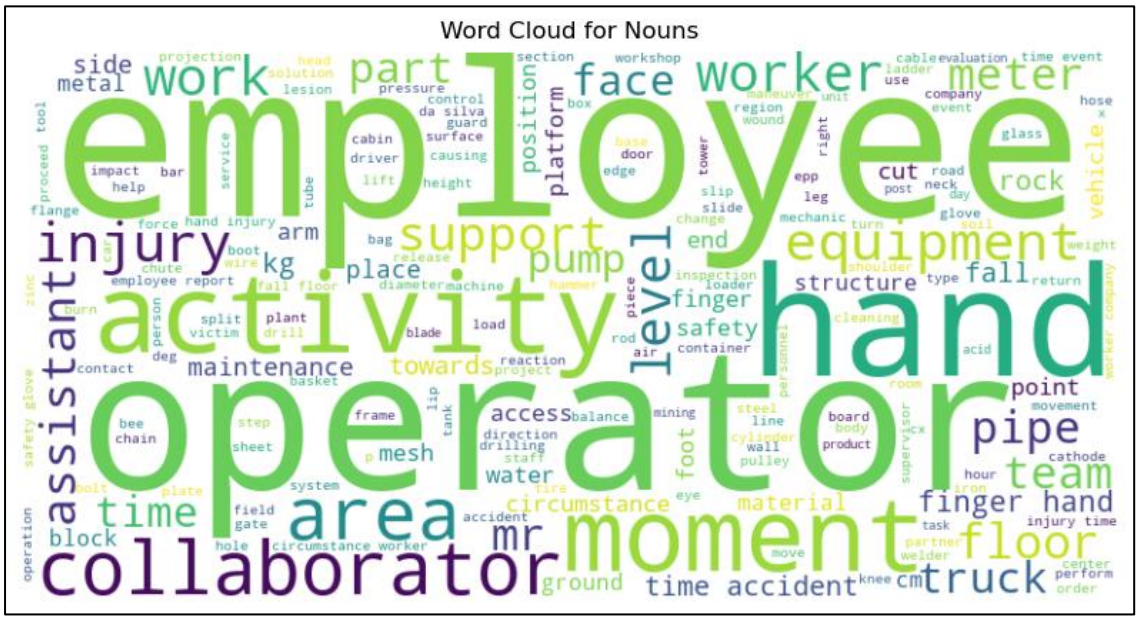
Word Cloud Generation:

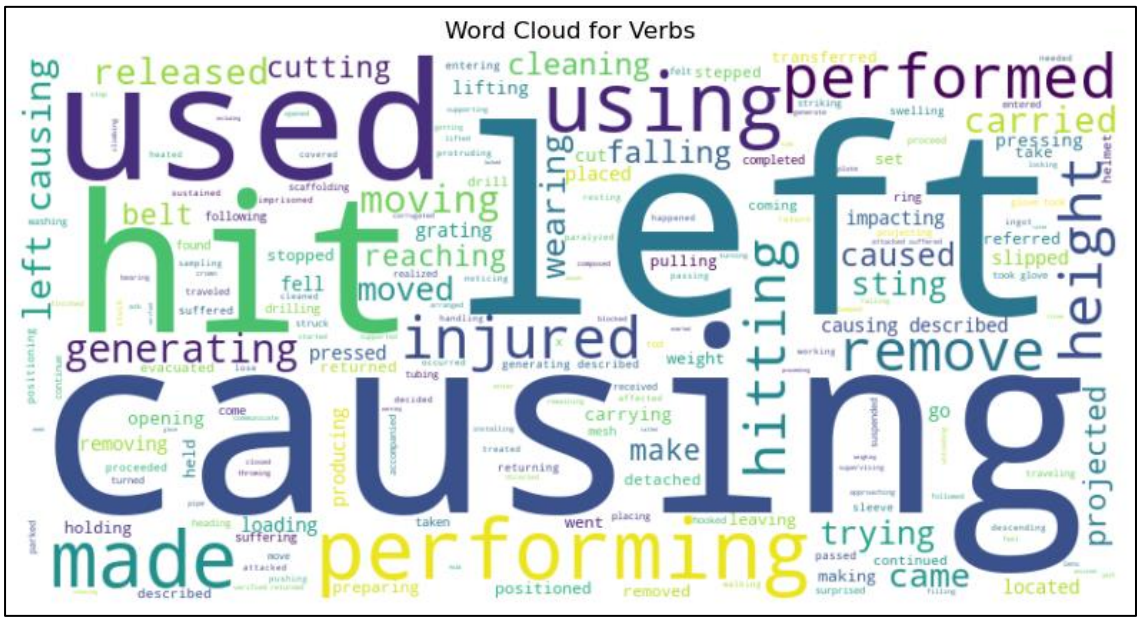
Generated a Word Cloud from the pre-processed descriptions to visually represent the most common words in the accident descriptions. This helps in quickly identifying key themes or frequent terms.

POS (Part-of-Speech) Tagging:

Performed POS tagging on the descriptions to categorize words into their respective parts of speech.

Extracted and generated separate Word Clouds for nouns and verbs. This helped in identifying common actions (verbs) and objects or subjects (nouns) associated with the accidents.





Observations:

- Noted that there were many action-related words (verbs) such as 'falling', 'cutting', 'drilling', which indicate the nature of the accidents.
- Identified various nouns like 'truck', 'pipe', 'bee', which point towards the objects involved in the accidents.
- This comprehensive preprocessing and analysis provides a clearer understanding of the causes and nature of the accidents in the dataset, which is essential for developing strategies to mitigate such incidents in the future.
-

Further Steps were taken to prepare the data for NLP model building:

Tokenization Using SpaCy:

- Loaded the English language model from SpaCy (en_core_web_sm).
- Converted the 'Description' column to string type.
- Applied SpaCy's tokenization to the 'Description' column, creating a new column 'Tokenized Text' that contains the tokens for each description.

Initial DataFrame Display:

- Displayed the DataFrame showing the first few rows, including columns like 'Date', 'Country', 'City', 'Industry Sector', 'Accident Level', 'Potential Accident Level', 'Gender', 'Employee Type', 'Critical Risk', 'Description', 'Year', 'Month', 'Day', 'POS_Tags', and 'Tokenized Text'.

N-gram Generation:

- Defined a function generate_ngrams to create n-grams (uni-grams, bi-grams, and tri-grams) from the tokenized text.
- Applied this function to the 'Description' column to create three new columns: 'Uni-grams', 'Bi-grams', and 'Tri-grams', each containing the respective n-grams for each description.

Most Frequent N-grams Identification:

- Extracted the most frequent uni-grams, bi-grams, and tri-grams using Python's Counter method.
- Displayed the top 20 most frequent n-grams in each category.

Vector Encoding with TF-IDF:

- Utilized TfidfVectorizer from sklearn, setting max features to 1000 and specifying English stopwords, with n-gram range set to (2, 2) (bi-grams).
- Transformed the 'Description' column into a TF-IDF matrix and converted it into a DataFrame (tfidf_df), which includes the TF-IDF scores for the top features.

Identification of Top Features:

- Summarized the TF-IDF scores for each feature across all documents.

- Sorted and displayed the top 30 features based on their aggregated TF-IDF scores.
- The following table gives the feature bi-gram and the TF-IDF score for them:

Feature	TF-IDF Score
left hand	16.026339
right hand	13.926010
causing injury	13.242063
time accident	10.120484
employee report	8.380265
finger left	7.739658
injury described	6.511859
medical center	6.181397
left foot	5.524003
right leg	5.479867
described injury	5.202557
finger right	4.763424
hand causing	4.761339
cm cm	4.331312
generating injury	4.217899
injured person	4.186464
hit right	4.007624
fragment rock	3.973550
middle finger	3.919644
circumstance worker	3.892055
support mesh	3.778769
right foot	3.681881
injury time	3.656111
da silva	3.262092
ring finger	3.258298
causing cut	3.255090
left leg	3.049839
employee used	3.027345
accident employee	2.984388
safety glove	2.938843

Vector Encoding with Word2Vec:

- Initialization of Word2Vec Model: Leveraged the Word2Vec implementation from the gensim library. The model was configured to create vectors of size 1000 for each word, ensuring a comprehensive and detailed representation. The window size was set to 5 to consider a balanced context of surrounding words. The min_count parameter was set to 1, allowing the model to learn representations for all words, even those with a single occurrence.
- Training on Tokenized Text: Applied the Word2Vec model to a list of tokenized sentences, tokens_list, derived from the 'Description' column of our dataset. This list represented the corpus for training, where each entry corresponded to a preprocessed and tokenized description.
- Sentence Vector Generation: Developed a custom function, get_sentence_vector, to compute sentence vectors. This function iteratively processed each word in a sentence, summing their respective Word2Vec vectors if present in the model's vocabulary. This approach effectively created a cumulative vector representation for each sentence.
- Vector Integration in DataFrame: Executed the get_sentence_vector function across all tokenized sentences in tokens_list. The resulting sentence vectors were then stored in a new column, 'Description_Vector', within the original DataFrame (pp_data). This integration facilitated a seamless combination of the original textual data with its corresponding vectorized representations.
- Dataframe Transformation: The transformed DataFrame, now enriched with sentence vectors, provided a dual view - the original text descriptions alongside their vector representations. This format allowed for a more nuanced analysis and application in downstream tasks such as clustering, classification, or similarity searches.

Saving the Cleansed Data:

- Saved the processed DataFrame to a .csv file, ensuring that all the preprocessing steps and feature extractions are preserved for future use in modelling or further analysis.

Performance of the Model (Stats of Performance and Configuration Comparisons mostly by Grid Search Mechanism)

The model's performance is evaluated through a series of tests and comparisons. The primary method for optimization is the grid search mechanism, which helps in fine-tuning the model parameters to achieve the best results. The performance metrics are likely to include accuracy, precision, recall, and F1 scores, among others. These statistics provide a quantitative measure of the model's effectiveness in accurately identifying and categorizing industrial safety risks through the chatbot interface.

The following steps were followed for building a machine learning model and comparing the results:

Modelling using TF-IDF vectors:

Data Splitting:

- The TF-IDF matrix (tfidf_matrix) was used as the feature set (X), and the 'Accident Level' column from the preprocessed data (pp_data) served as the target variable (Y).
- The dataset was split into training and testing sets using a 70-30 split ratio, with stratification based on the target variable to ensure a balanced representation of each class.

Classifier Training and Evaluation:

- Five different classifiers were selected: Logistic Regression, Random Forest, K-Nearest Neighbor, Support Vector Machine, and AdaBoost.
- Each classifier was trained on the training set and then used to make predictions on the test set.

Performance Evaluation:

- The performance of each classifier was evaluated using accuracy, precision, recall, and F1-score metrics.
- A confusion matrix for each classifier was also generated to understand the distribution of predictions across different accident levels.

Results Compilation:

- The accuracy of each classifier was compiled into a DataFrame for a clear comparison.
- Here is a tabulated summary of the results:

Name	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	74.22	55.0	74.0	63.0
Random Forest	74.22	55.0	74.0	63.0
K-Nearest Neighbor	74.22	55.0	74.0	63.0
Support Vector Machine	74.22	55.0	74.0	63.0
AdaBoost	71.88	55.0	72.0	62.0

Observations:

- All the models, except AdaBoost are providing the same accuracy of 74%, 2% higher than AdaBoost. But for such a highly imbalanced dataset, it's better to use weighted average of Precision, Recall & F1-Score as performance metrics.
- All the models provide very low performance, with only 55% Precision & 62-63% F1-Score, as only accidents of Level I have been correctly predicted and all the accidents in Level II - V are incorrectly predicted as Level I.
- This is due to the dataset being small and highly imbalanced with majority of the accidents from Level I.

Modelling using Word2Vec vectors:

Data Preparation:

- The Word2Vec vectors (X_wv) are extracted from the 'Description_Vector' column of the preprocessed data (pp_data).
- The target variable (Y) is presumably another column from the same dataset, though its nature isn't specified in your description.

Splitting Training and Testing Data:

- The dataset is divided into training and testing sets using the train_test_split function.
- A 70-30 split is applied, meaning 70% of the data is used for training and 30% for testing.
- The split is stratified based on the target variable (Y) to ensure each class is represented proportionally in both training and testing sets.

Classifier Training and Evaluation:

- Five different classifiers are used: Logistic Regression, Random Forest, K-Nearest Neighbor, Support Vector Machine, and AdaBoost.
- Each classifier is trained on the training set (x_train_wv, y_train) and then used to make predictions on the test set (x_test_wv).

Performance Evaluation:

- The performance of each classifier is assessed using several metrics: accuracy, precision, recall, and F1-score.
- A confusion matrix is also generated for each classifier to visualize the distribution of predictions across different classes.

Results Compilation:

- The accuracy of each classifier is compiled into a DataFrame for easy comparison.
- The table shows the performance metrics for each model: accuracy, precision, recall, and F1-score.

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Logistic Regression	74.21875	55.0	74.0	63.0
Random Forest	74.21875	55.0	74.0	63.0
K-Nearest Neighbor	74.21875	55.0	74.0	63.0
Support Vector Machine	74.21875	55.0	74.0	63.0
AdaBoost	71.87500	56.0	72.0	63.0

Observations:

- In our results, all classifiers except AdaBoost exhibit the same accuracy (74.21%), precision (55.0%), recall (74.0%), and F1-score (63.0%).
- AdaBoost shows slightly lower accuracy (71.87%) but slightly higher precision (56.0%).
- The high recall but low precision suggests that the models are biased towards predicting the majority class, as indicated by the confusion matrices which show a large number of true positives for class 'I' and zero for other classes.
- We can therefore observe that there is no difference in the model performance on both the Vectorization methods: TF-IDF & Word2Vec. Hence, going forward we can choose either one, we're selecting TF-TDF here as the importance of the words in a sentence matters to understand the cause of the accident and identify the Accident Level.

These findings highlight the challenge in predicting the accident level accurately due to the skewed nature of the dataset. The similar performance across diverse models suggests that the issue might not be the model selection but the underlying data distribution. Future steps could include addressing this imbalance, possibly through techniques like oversampling, under sampling, or synthetic data generation.

Modelling using TF-IDF vectors and Grid Search:

Initialization of Classifiers:

The function defines a dictionary classifiers_dict with different machine learning classifiers like Logistic Regression, Random Forest, K-Nearest Neighbor, Support Vector Machine, and AdaBoost, each initialized with their respective settings.

Parameter Grid Definition:

A param_grid dictionary is created, specifying the hyperparameters to be tuned for each classifier. For example, C values for Logistic Regression and SVC, n_estimators and max_depth for Random Forest, n_neighbors for K-Nearest Neighbor, and parameters for AdaBoost.

Grid Search and Model Fitting:

For each classifier, the function performs Grid Search using GridSearchCV. It fits the model on the training data (x_train, y_train) with various combinations of parameters defined in param_grid.

Selection of the Best Model:

After the Grid Search, the best model is selected based on the highest accuracy score. grid_search.best_estimator_ gives the best model for each classifier.

Model Evaluation:

The best model is then used to predict on the test data (x_test). The predictions are evaluated using various metrics like accuracy, precision, recall, and F1-score.

The classification_report and confusion_matrix functions are used to generate detailed performance reports and confusion matrices for each model.

Compilation of Results:

The function collects the performance metrics for each classifier and stores them in a dataframe. This includes the model name, accuracy, precision, recall, and F1-score.

Output:

Finally, the function returns the dataframe with the performance metrics for each classifier.

The following is the performance report of the various machine learning models:

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	74.21875	55.0	74.0	63.0
Random Forest	74.21875	55.0	74.0	63.0
K-Nearest Neighbor	74.21875	55.0	74.0	63.0
Support Vector Machine	74.21875	55.0	74.0	63.0
AdaBoost	71.87500	55.0	72.0	62.0

In this case, after performing this process, it is observed that there was no significant difference in model performance after using Grid Search. The maximum accuracy achieved was 74.2% and the F1-score was 63%, which suggests that the hyperparameter tuning did not substantially improve the performance of the models over the baseline. This could indicate

that either the models are already optimized with the default parameters, or the features used for training are not sufficiently informative to achieve higher performance, or that the dataset itself is challenging in a way that limits model performance.

EOR