

CAPSTONE PROJECT NLP CHATBOT INTERFACE INTERIM REPORT

Group 6

Team Members

Amit Pudkey

Arul Kumar Raman

Ananya Sathyanarayana

Kushal Voona

Shrey Rathi

Shobhit Verma

Table of Contents

Problem Statement.....	3
Introduction.....	3
Project Objective.....	3
Data Description.....	3
Data Preprocessing.....	4
Data Cleaning.....	4
Feature Engineering.....	4
Text Preprocessing.....	4
Exploratory Data Analysis.....	5
Summary of Data.....	5
Univariate Analysis.....	6
Word clouds.....	11
Multivariate Analysis.....	13
Model Building Experiment.....	17
Approach.....	17
Model Performance.....	19
Hyperparameter Tuning.....	21
Compare Model Performance post tuning.....	22
Conclusion.....	25

Problem Statement

Introduction

In today's industrial world, keeping workers safe remains a pressing challenge. Even with modern safety measures, employees continue to face workplace accidents that can lead to injuries or worse. Our focus is on a major Brazilian industry where understanding why accidents happen - and preventing them - is crucial for protecting workers' lives. While companies collect extensive data about these incidents, making sense of this information quickly and effectively is difficult for safety professionals. This project introduces an intelligent chatbot that uses Natural Language Processing to help these professionals rapidly analyze accident reports and identify safety risks, ultimately working towards a safer workplace for everyone.

Project Objective

The primary objective is to design a machine learning (ML) based chatbot utility. This tool is intended to help safety 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.

Data Description

The dataset is obtained from one of the biggest industries in Brazil and in the world. It is a record of accidents from 12 different plants in 3 different countries.

Data Components of the columns

- Time-based Information
 - Data: timestamp or time/date information
- Geographic Information
 - Countries: which country the accident occurred (anonymised)
 - Local: the city where the manufacturing plant is located (anonymised)
 - Industry sector classification

- Accident Classification
 - Accident level: Severity levels (I to V)
 - Potential accident levels: Severity levels (I to VI)
 - Critical Risk: some description of the risk involved in the accident
- Personnel Information
 - Genre: if the person is male or female
 - Employee or Third Party: if the injured person is an employee or a third party
- Descriptive Elements
 - Description: Detailed description of how the accident happened.

Data Preprocessing

Data Cleaning

- Initial data contained 425 rows and 11 columns
- Columns were renamed to follow a more standard convention like Date to Date, Countries to Country, Local to City, using '_' as a separator in column names
- Removed Unnamed:0 column as it was unique and more like a duplicate index
- Identified 7 duplicate rows after removing above column, these were dropped
- Further identified 7 rows with same Description, these were also dropped
- There was only one row with Potential Accident Level = VI, this was updated to the closest severity i.e. Level V
- Post data cleaning, dataset contains 411 Rows and 10 Columns

Feature Engineering

- Extract the Year, Month, Date, Day from the Date Columns
- Add number of words in Description as a new column -Word_Count
- Now Data set has 411 rows and 15 columns

Text Preprocessing

- Checked for non-ASCII and HTML tags, found no discrepancy
- Used WordNet and NLTK to perform below operations:
 - Removed special characters
 - Converted all to lower case
 - Trimmed white spaces
 - Removed stop words
 - Apply Lemmatizer

Exploratory Data Analysis

Summary of Data

Defined a method `detailed_summary` that can display all the statistics of numerical and categorical variables in the given data frame

Code snapshot:

```
def detailed_summary(df):  
    ...  
    Print a summary of the given dataframe to display no of records, columns,  
    datatype and names of each column.  
    Also list the Missing values, Duplicate values, if any along with other details  
    ...  
  
    print(f'Data set has {df.shape[0]} rows and {df.shape[1]} columns')  
    print(f'Various datatypes present in the dataset are:\n{df.dtypes.value_counts()}')  
    summary = pd.DataFrame(df.dtypes, columns = ['Datatypes'])  
    summary = summary.reset_index()  
    summary['Missing_values'] = df.isnull().sum()  
    summary['Missing_values'].fillna(0,inplace=True)  
    summary['Unique_values'] = df.nunique().values  
    summary['Duplicate_values'] = df.duplicated().sum()  
    summary['First_value'] = df.loc[0].values  
    summary['Second_value'] = df.loc[1].values  
    return summary
```

Sample output as below:

`detailed_summary(df)`

Data set has 425 rows and 11 columns
Various datatypes present in the dataset are:
object 9
int64 1
datetime64[ns] 1
Name: count, dtype: int64

	index	Datatypes	Missing_values	Unique_values	Duplicate_values	First_value	Second_value
0	Unnamed: 0	int64	0.0	425	0	0	1
1	Date	datetime64[ns]	0.0	287	0	2016-01-01 00:00:00	2016-01-02 00:00:00
2	Country	object	0.0	3	0	Country_01	Country_02
3	City	object	0.0	12	0	Local_01	Local_02
4	Industry_Sector	object	0.0	3	0	Mining	Mining
5	Accident_Level	object	0.0	5	0	I	I
6	Potential_Accident_Level	object	0.0	6	0	IV	IV
7	Gender	object	0.0	2	0	Male	Male
8	Employee_Type	object	0.0	3	0	Third Party	Employee
9	Critical_Risk	object	0.0	33	0	Pressed	Pressurized Systems

While removing the drill rod of the

Univariate Analysis

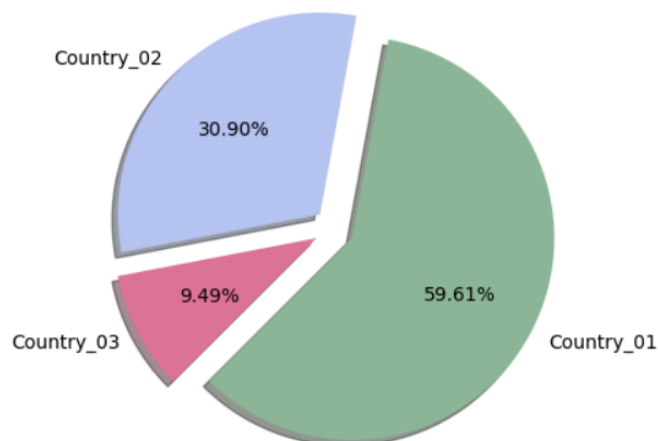
Helper method to plot distribution of categorical features

Defined a method to visualise either a pie chart or a count plot based on the number of unique values in the given categorical column

Snapshot of the code:

```
def visualise_cat_features(df,col):  
    ...  
    Plots a pie chart if feature has less than 5 unique values  
    Plots a countplot if it has >=5 and <=20 unique values  
    Prints an error message if it has >20 unique values  
    ...  
  
    num_values = df[col].nunique()  
    values_count = df[col].value_counts()  
  
    if num_values <= 4:  
        #pie  
        colors = ['#8EB897', '#B7C3F3', '#DD7596', '#4F6272' ]  
        plt.pie(values_count, labels = values_count.index, colors=colors, autopct='%0.2f%%',  
                shadow = True, explode = [0.1]*len(values_count.index), startangle = -135)  
        plt.title(f"Distribution of {col}")  
        plt.show()  
  
    elif num_values <= 20:  
        #countplot  
        plt.figure(figsize = (12,5))  
        ax = sns.countplot(df,x=col,order=values_count.index,palette='Set2')  
  
        # Calculate percentages  
        total = len(df[col])  
  
        # Add labels on the bars  
        for p in ax.patches:  
            count = p.get_height()
```

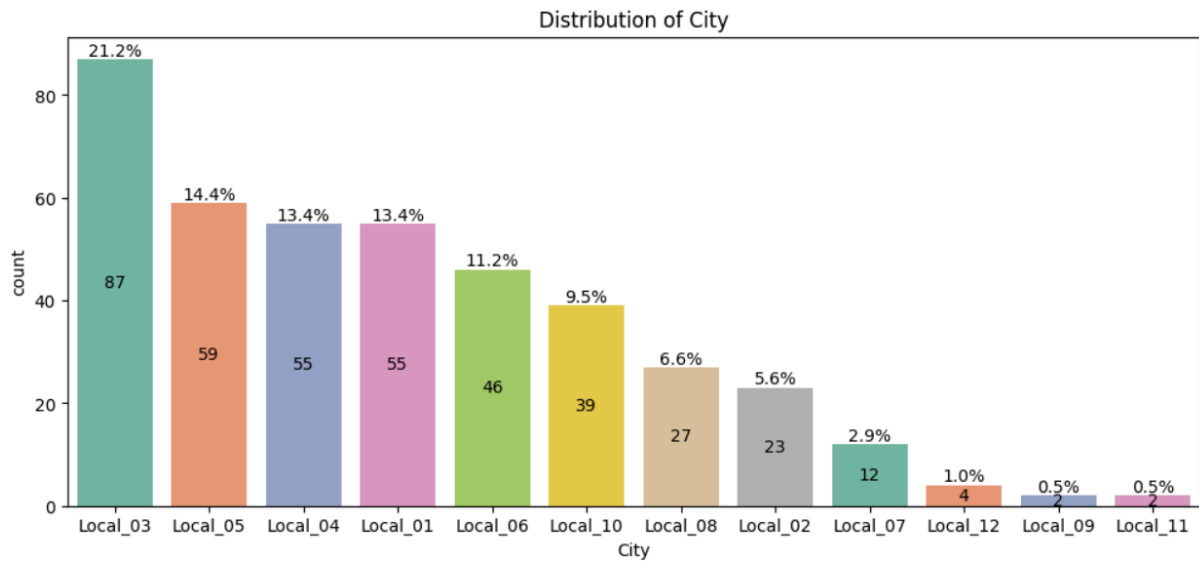
Analysis by Country



- Country 1 has reported almost 60% of accidents. Probably because there are more cities with industrial plants in this country. This needs to be verified in Bivariate analysis

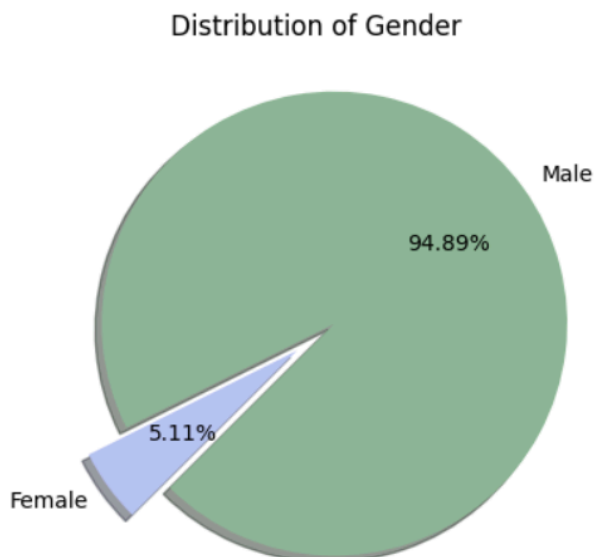
- Country 2 and Country 3 report approx 30% and 10% accidents respectively

Analysis by City



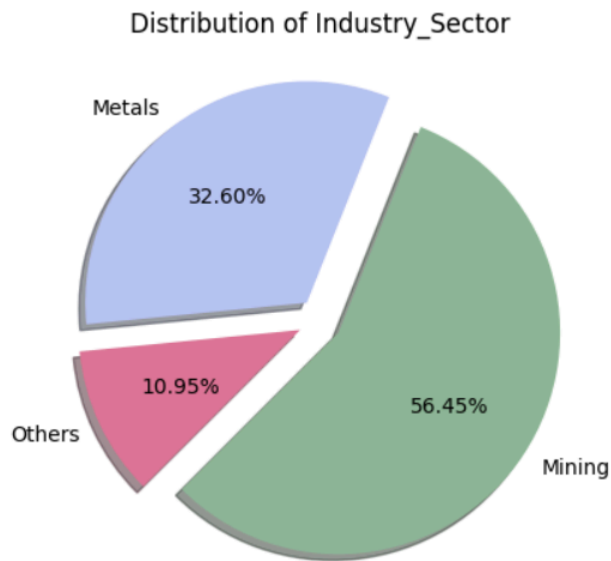
- 21% accidents of total accidents reported from Local 3
- Local 5, Local 4 and Local 1 are next major cities with accidents
- Local 2, Local 9 and Local 11 have reported least number of accidents (less than 2% together)

Analysis by Gender



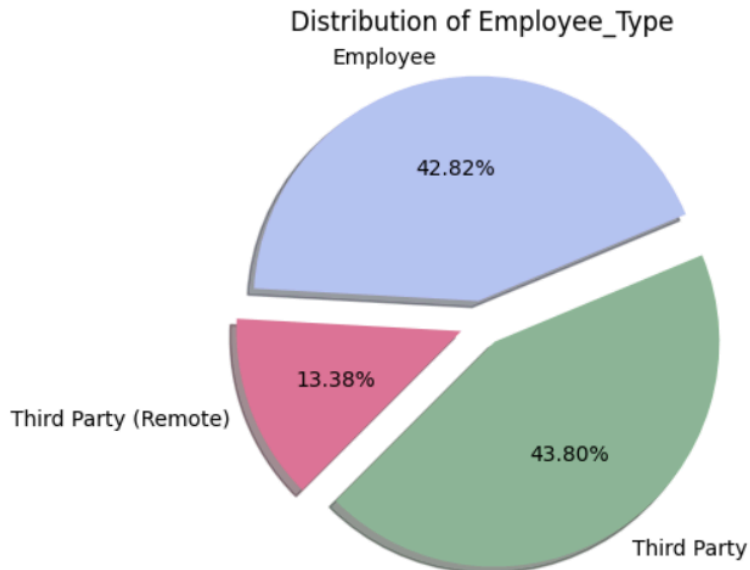
- 95% employees who reported accidents are Male compared to 5% Females

Analysis by Industry Sector



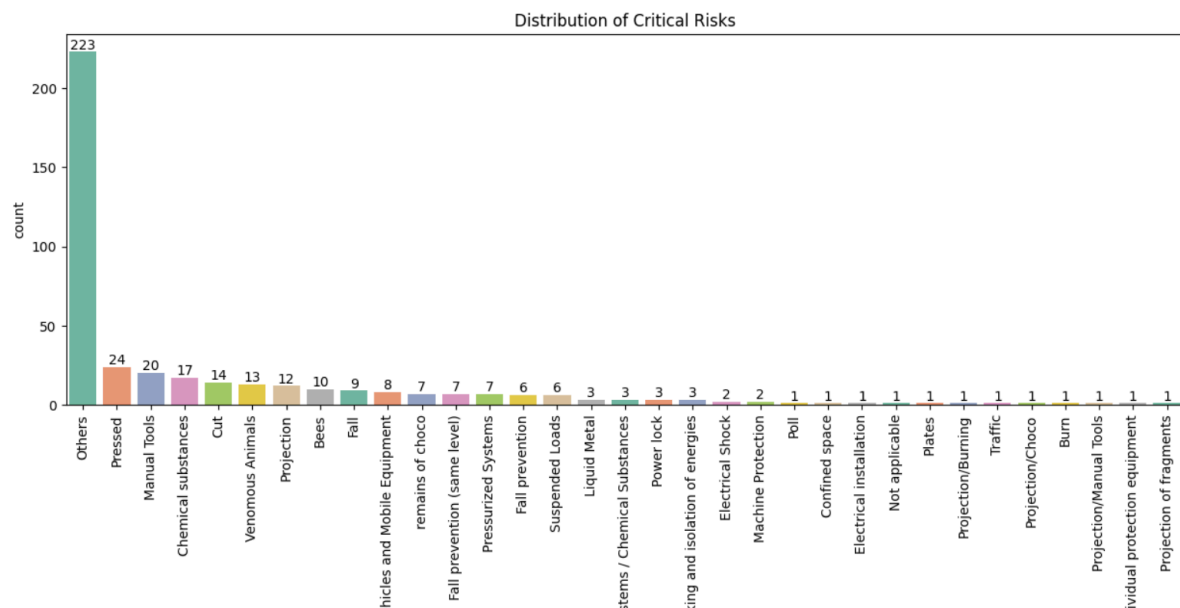
- Mining industry contributes to almost 56% accidents
- Metals and Others industries contribute 33% and 11% accidents (approx)

Analysis by Employee Type



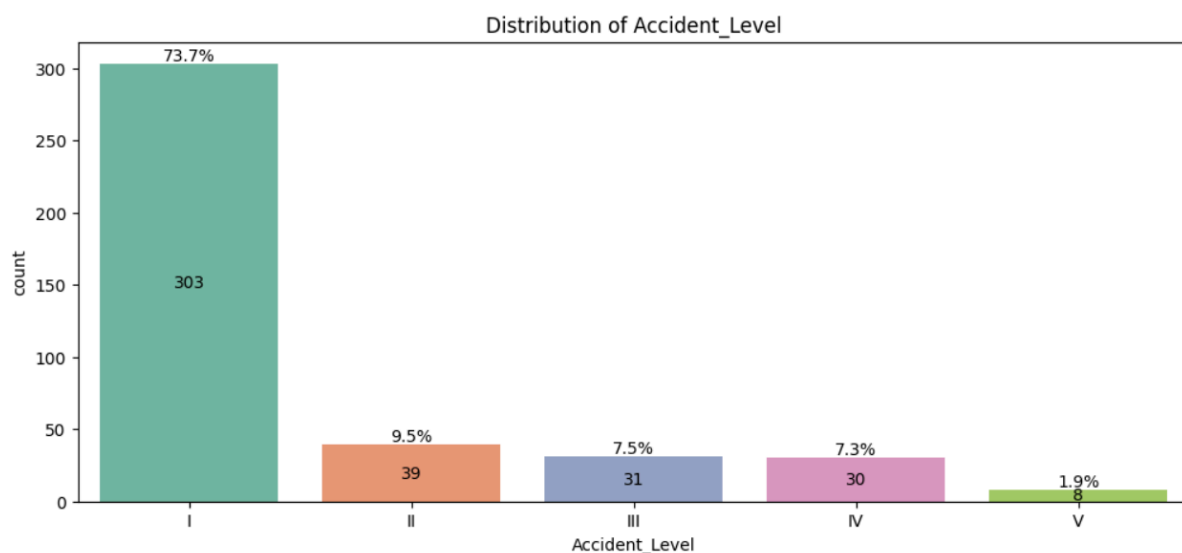
- 44% of accidents are recorded by Third party (possibly contract) employees
- 43% accidents are recorded by Direct employees
- Least accidents (13%) are recorded by Third Party Remote employees, possibly because they are not working on location in the plants

Analysis by Critical Risk



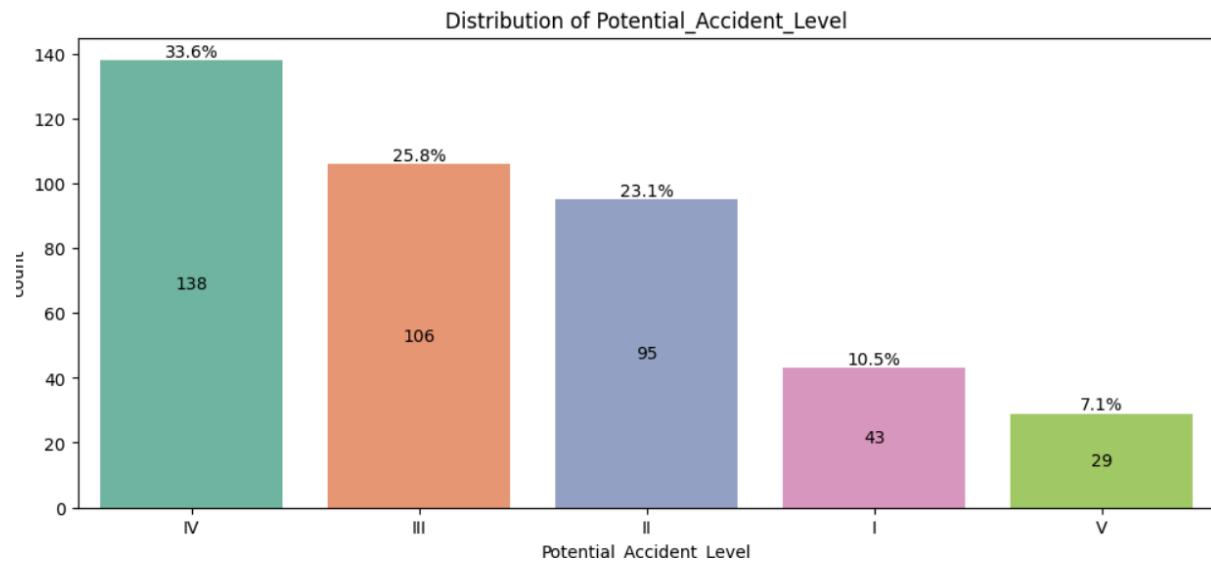
- More than 50% of accidents are categorised as Others.
- The other significant critical risks are Pressed, Manual Tools, Chemical Substances, Cut, Venomous Animals, Projection, Bees and Fall

Analysis by Accident Level



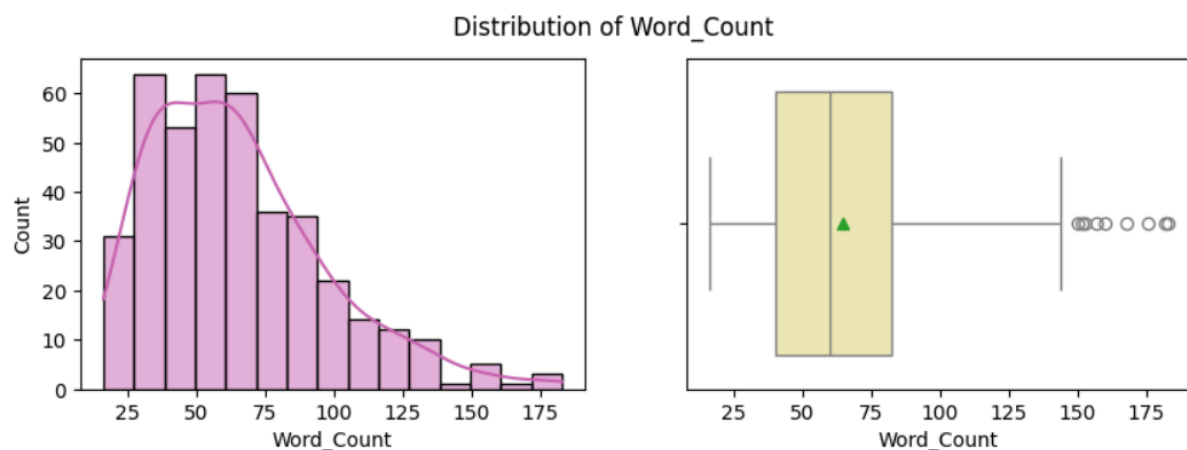
- 74% of the accidents are categorised as Level I accidents indicating a low severity
- 2% are classified as Level V (most severe)

Analysis by Potential Accident Level



- There is a mismatch observed between Accident Level and Potential Accident Level
- Most of the accidents categorised as Level I Accident are of higher severity, this can be seen from the reduced number of Level I accidents according to Potential Accident Level
- This indicates that Potential Accident Level will indicate a better categorisation of accidents as compared to Accident Level and hence a better target for predicting accident severity

Analysis by Word Count



- Distribution of Word Count is right skewed, indicating most of the descriptions contain fewer than 75 words
- 50% of the descriptions are around 60 words or lesser
- Few descriptions contain more than 150 words



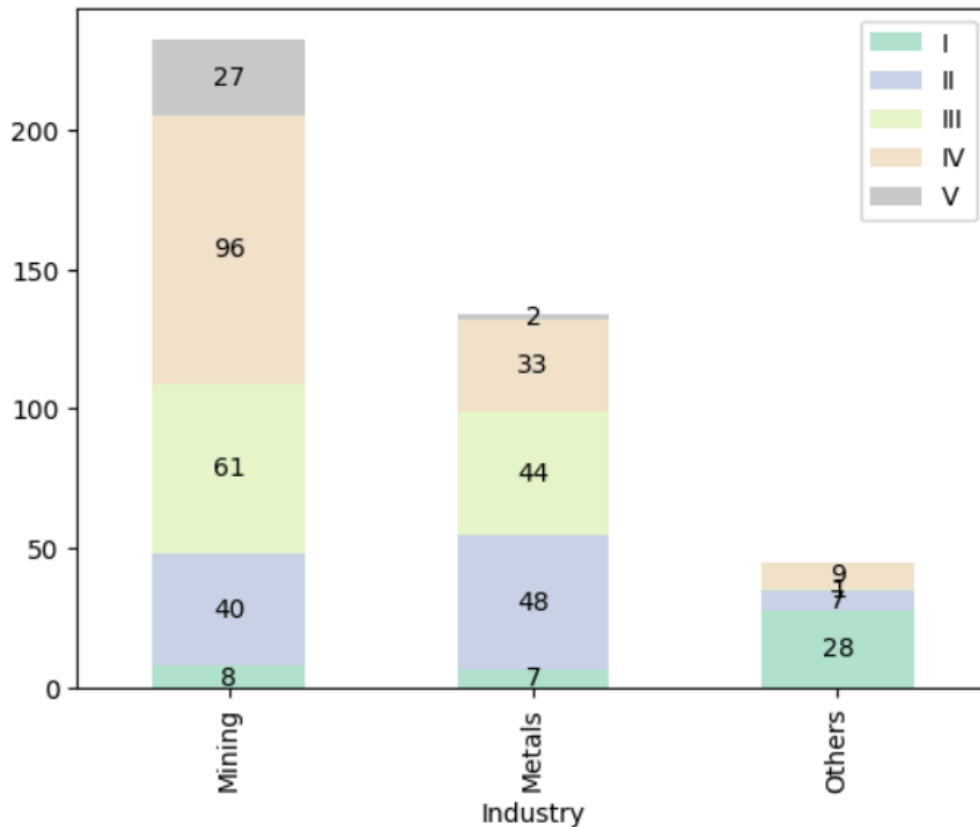
Word Cloud for Verbs used in Description



- Most of the descriptions contain nouns like operator, employee - people involved in the accident
- Most of the descriptions contain verbs like hit, carry, injured, fell, drill which is most likely the action performed when the accident occurred

Multivariate Analysis

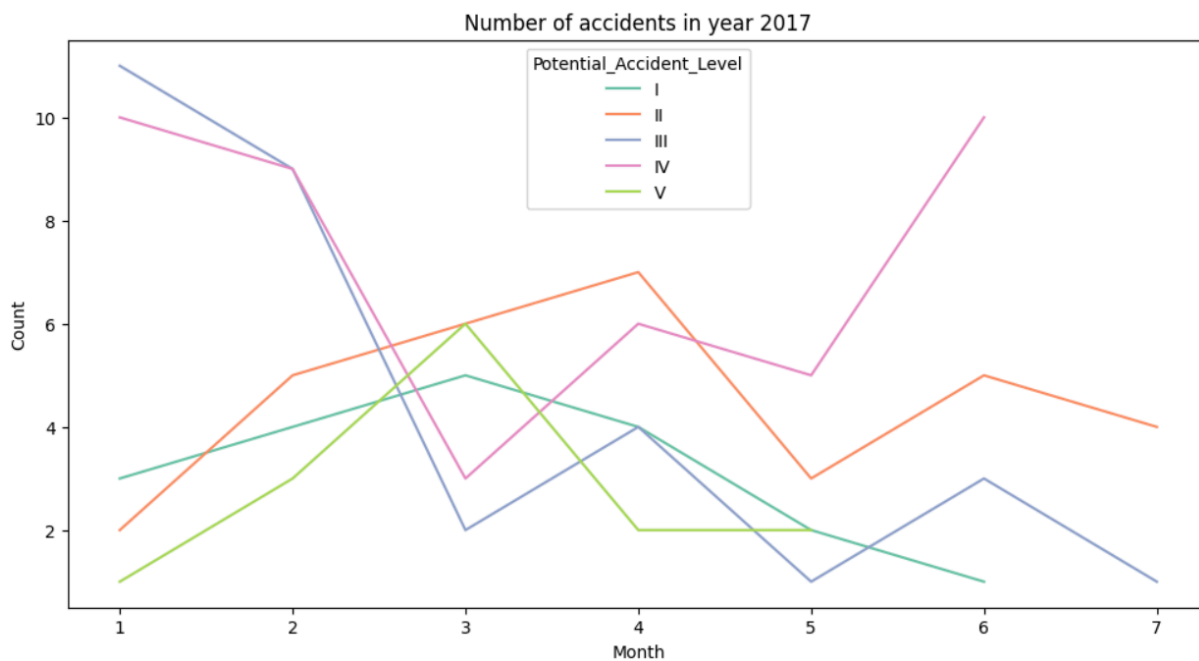
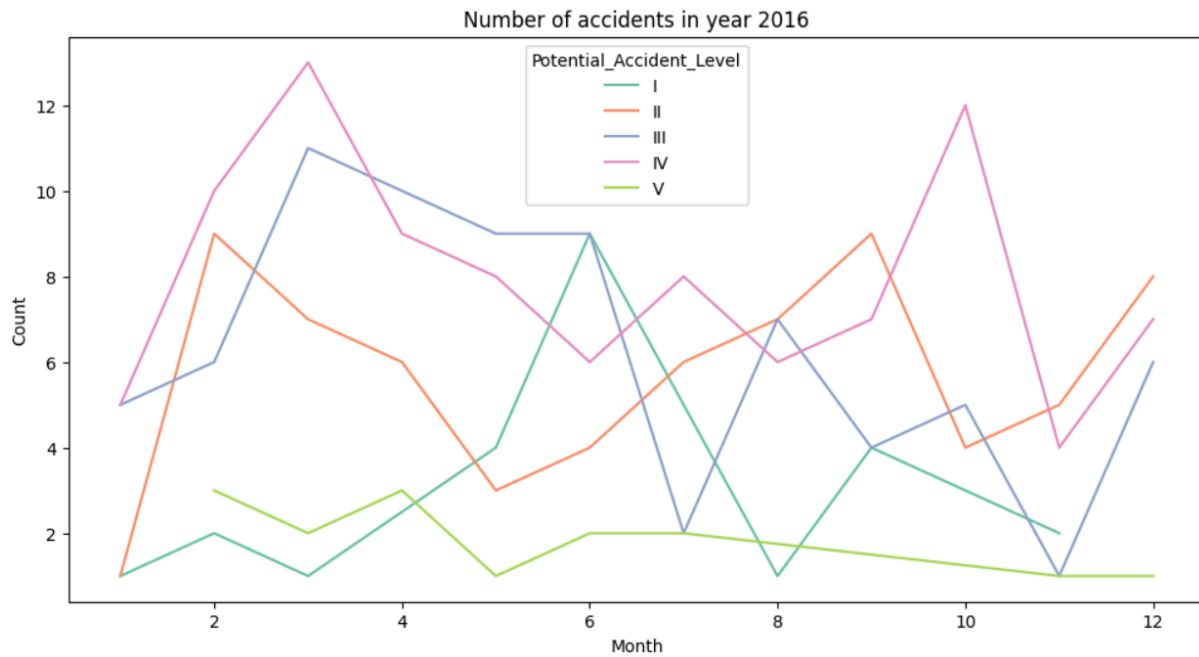
Analysis of industry Sector vs Potential Accident Level



- The most severe accidents (Level V,4,3) are seen only in Mining & Metals industry sectors, majority in Mining and 1 in 'Others' industry
- The 'Others' industry sector has least accidents, with majority being the least severe ones (Level I).
- Accident Levels II accidents have occurred more in Mining and Metals

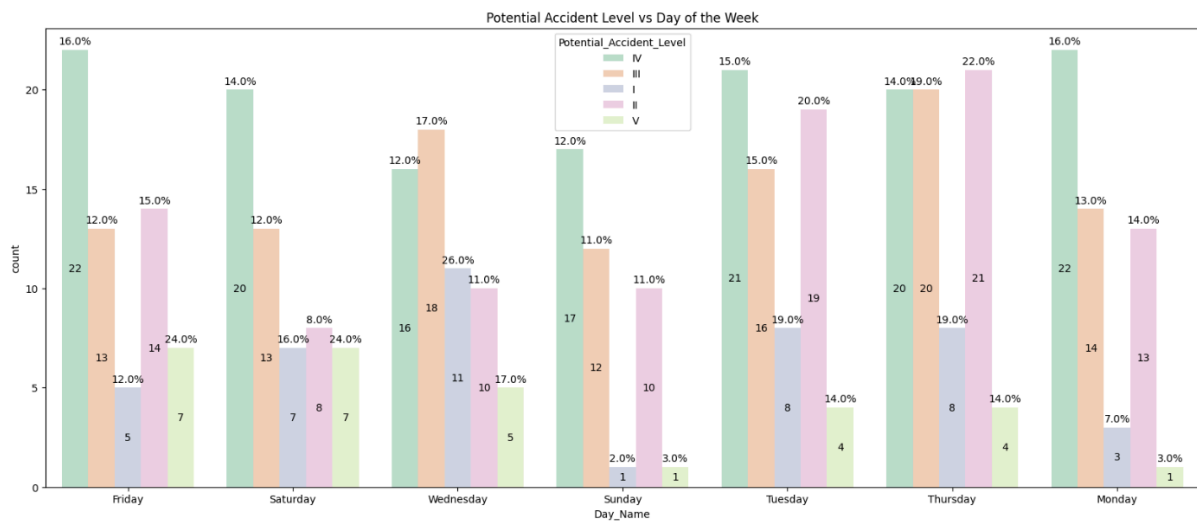
Analysis by Time period of accidents

Since the data contained accident details for years 2016 and 2017, we plotted the temporal distribution separated by Potential Accident Level



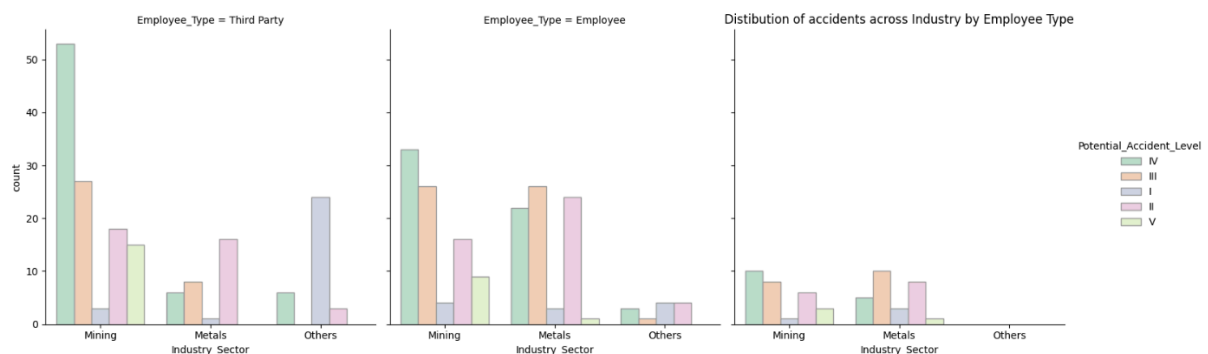
- Accident details are recorded for January 2016 to July 2017
- There is a decreasing trend in accidents for year 2017 as compared to year 2016
- Severe accidents are observed across different months in each year
- Most severe accidents (Level V) were observed in February and April in 2016 while in March for year 2017

Analysis of Potential Accident Level by Day of the week



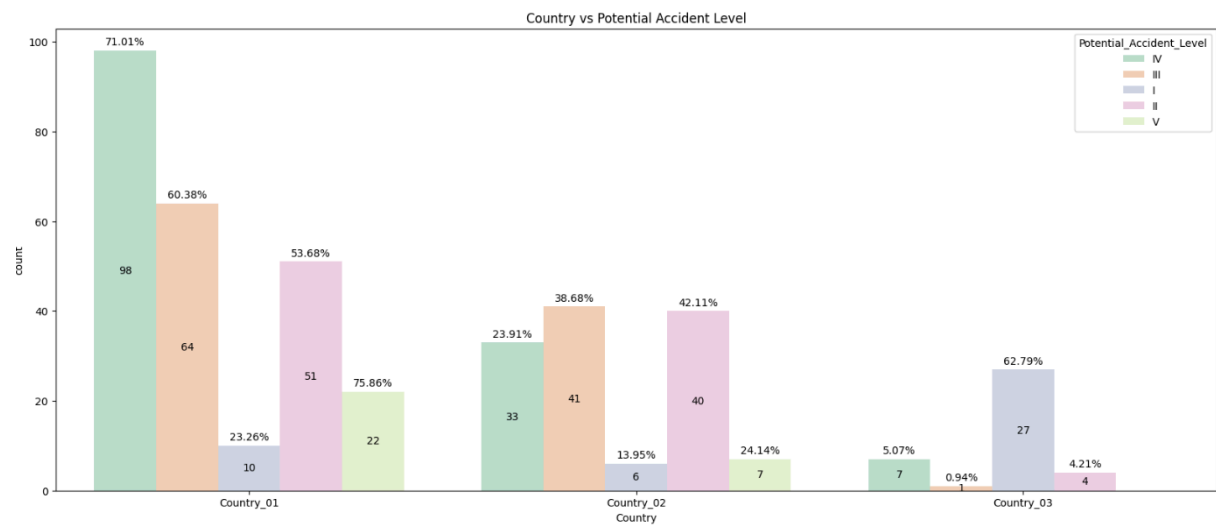
- Accidents are more prone to occur on Thursday and Friday
- Most severe (Level V) accidents are prone to occur on Friday or Saturday
- Least severe (Level I) accidents may occur on any day as per data given its distribution

Analysis of Potential Accident Level by Employee Type across Industry sectors



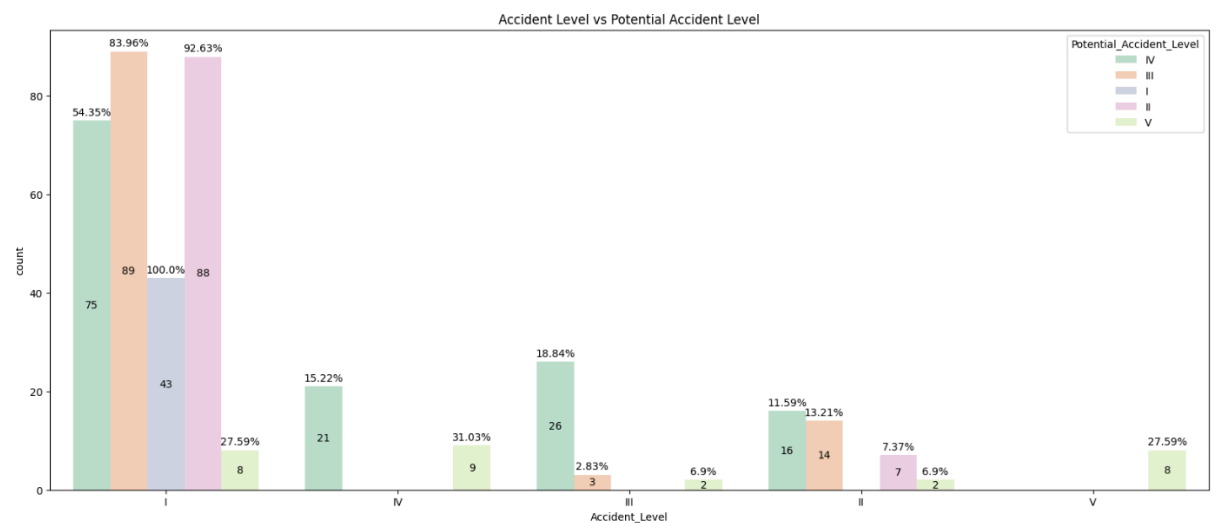
- Third Party employees reported many accidents compared to other employee types across all industries
- Others industries do not have any accidents reported by Remote Third-Party employees
- Most severe accidents are also reported by Third party employees

Analysis of Potential Accident Level by Country



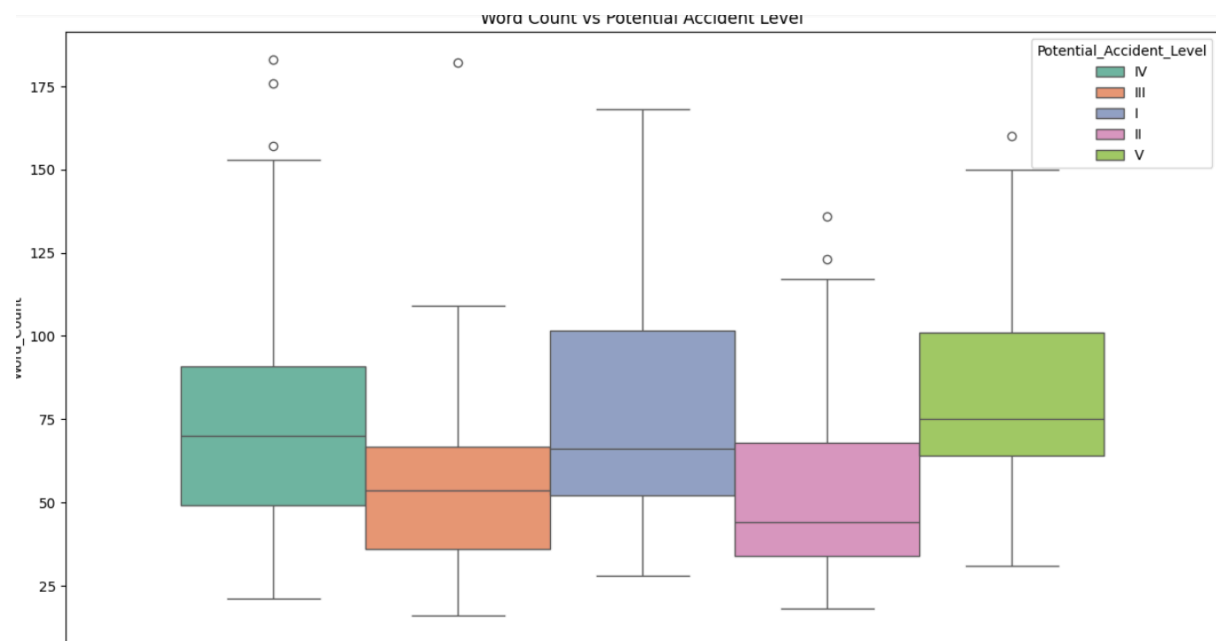
- Country 1 has reported most accidents with majority of Level IV accidents
- Country 3 has reported the least number of accidents of which most are low severity accidents
- Most severe accidents (Levels IV and V) are reported in Country 1 and Country 2

Analysis of Potential Accident Level by Accident Level



- Accident Level tends to categorise accidents on lower severity than Potential Accident Level
- Most of the accidents reported as Level I Accident are actually of a higher severity
- 92% of Potential Accident Level II are categorised as Level I accidents
- 84% of Potential Accident Level III are categorised as Level I accidents
- Only 8 accidents were categorised with highest severity in both Accident Level and Potential Accident Level

Analysis of Potential Accident Level by Word Count



- Description of Levels I, IV and V in general longer than those of Levels II and III
- Longest description has around 200 words and belongs to Potential Accident Levels III and IV

Model Building Experiment

Approach

The aim of the project is to predict the accident severity based on its description so that we can utilize NLP techniques and build a chatbot that can help in increasing the safety of industrial workers.

Key points

We will solve this problem using Traditional ML techniques.

This can be categorised as a multiclass classification to determine accident severity (Level I to Level V).

Based on the EDA, we know Accident Level is highly imbalanced (74% of accidents are classified as Level I) and hence may not capture the actual severity like Potential Accident Level. Hence, we will use the Potential Accident Level as a target.

Text embeddings:

We will generate embeddings for accident Description (after text preprocessing) using 4 different techniques: TF-IDF, Word2Vec, Glove and Sentence Transformer

ML Models and Model Training process:

We will explore the below traditional ML models on each of the above embeddings one at a time and compare the performance.

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Classifier
- XGBoost Classifier

Each of the above models will be included as a part of a Scikit-Learn pipeline.

During experimentation, we found models which include other relevant features in addition to the accident description perform slightly better. So the final training dataset will include the text embeddings along with selected features from the dataset.

The Dropped features include 'Date', 'Day_Name', 'Accident_Level', 'Description' and 'Processed_Description'

The best performing 3 models will be selected for Hyper parameter tuning using GridSearchCV. Post tuning, we will record their performance on the test set using classification report, metrics and confusion metrics.

Evaluation Metrics:

- Models will be evaluated on Accuracy, Precision, Recall and F1 -Score against Train and Validation data
- Stratified Cross Validation will be performed to record Mean F1-scores across different folds
- The key metric of evaluation will be F1-Score

Snapshot of Data split

```
[ ] #define dataframe containing select columns from original data
feature_df = df.drop(['Date', 'Day_Name', 'Accident_Level', 'Potential_Accident_Level', 'Description', 'Processed_Description'], axis=1)
```

```
[ ] feature_df.head(2)
```



	Country	City	Industry_Sector	Gender	Employee_Type	Critical_Risk	Year	Month	Day	Word_Count
0	Country_01	Local_01	Mining	Male	Third Party	Pressed	2016	1	1	80
1	Country_02	Local_02	Mining	Male	Employee	Pressurized Systems	2016	1	2	54



Combine embeddings and other features

```
[ ] X_tfidf = tfidf_df.join(feature_df) #TF-IDF embeddings + relevant features

#encode the target
y = df['Potential_Accident_Level']

encoder = OrdinalEncoder(handle_unknown='use_encoded_value', unknown_value=-1)
y = encoder.fit_transform(y.values.reshape(-1,1))

[ ] #Split data into temp and test, further split temp to train and validation
X_temp, X_test_tfidf, y_temp, y_test_tfidf = train_test_split(X_tfidf, y, random_state = SEED, stratify = y, test_size=TEST_SIZE)
X_train_tfidf, X_valid_tfidf, y_train_tfidf, y_valid_tfidf = train_test_split(X_temp, y_temp, random_state = SEED,
stratify = y_temp, test_size = TEST_SIZE)
```

Model Performance

Model performance on TF-IDF

	Accuracy Train	Precision Train	Recall Train	F1 Score Train	Accuracy Test	Precision Test	Recall Test	F1 Score Test	CV Score Mean
TFIDF Logistic Reg	1.000000	1.000000	1.000000	1.000000	0.348485	0.355556	0.298304	0.310333	0.336414
TFIDF DecisionTree	1.000000	1.000000	1.000000	1.000000	0.272727	0.219372	0.258065	0.229976	0.233550
TFIDF RandomForest	1.000000	1.000000	1.000000	1.000000	0.257576	0.352235	0.207772	0.223620	0.275571
TFIDF SVM	0.866412	0.917886	0.748193	0.780828	0.363636	0.231303	0.238930	0.201176	0.225583
TFIDF Xgboost	0.996183	0.997753	0.997059	0.997389	0.242424	0.205466	0.193705	0.193571	0.190359

Model performance on Word2Vec

	Accuracy Train	Precision Train	Recall Train	F1 Score Train	Accuracy Test	Precision Test	Recall Test	F1 Score Test	CV Score Mean
W2Vec DecisionTree	1.000000	1.000000	1.000000	1.000000	0.409091	0.370909	0.380081	0.367378	0.212932
W2Vec Xgboost	1.000000	1.000000	1.000000	1.000000	0.378788	0.275138	0.278197	0.263717	0.218080
W2Vec Logistic Reg	0.652672	0.711491	0.626925	0.657806	0.318182	0.251483	0.279226	0.258767	0.259717
W2Vec RandomForest	1.000000	1.000000	1.000000	1.000000	0.378788	0.247500	0.262959	0.247094	0.228404
W2Vec SVM	0.366412	0.246592	0.227547	0.163578	0.333333	0.133333	0.202674	0.118824	0.143684

Model performance on Glove

	Accuracy Train	Precision Train	Recall Train	F1 Score Train	Accuracy Test	Precision Test	Recall Test	F1 Score Test	CV Score Mean
Glove RandomForest	1.000000	1.000000	1.000000	1.000000	0.484848	0.427778	0.393522	0.396498	0.281966
Glove Logistic Reg	0.927481	0.946150	0.947447	0.946647	0.378788	0.329470	0.325643	0.326526	0.278342
Glove Xgboost	1.000000	1.000000	1.000000	1.000000	0.378788	0.321508	0.317622	0.306841	0.301549
Glove DecisionTree	1.000000	1.000000	1.000000	1.000000	0.333333	0.283843	0.283402	0.280356	0.243387
Glove SVM	0.553435	0.583214	0.428556	0.444570	0.363636	0.330926	0.241604	0.215775	0.183752

Model performance on Sentence Transformer

	Accuracy Train	Precision Train	Recall Train	F1 Score Train	Accuracy Test	Precision Test	Recall Test	F1 Score Test	CV Score Mean
ST Logistic Reg	1.000000	1.000000	1.000000	1.000000	0.469697	0.415556	0.421523	0.413483	0.364433
ST Xgboost	1.000000	1.000000	1.000000	1.000000	0.409091	0.390870	0.356552	0.360519	0.324161
ST SVM	0.832061	0.918094	0.734113	0.780773	0.424242	0.396364	0.353216	0.342424	0.287357
ST DecisionTree	1.000000	1.000000	1.000000	1.000000	0.378788	0.322381	0.338370	0.328360	0.220041
ST RandomForest	1.000000	1.000000	1.000000	1.000000	0.378788	0.420907	0.309300	0.322527	0.294620

Compare all model performance

Snapshot of best performing models:

	Accuracy Train	Precision Train	Recall Train	F1 Score Train	Accuracy Test	Precision Test	Recall Test	F1 Score Test	CV Score Mean
ST Logistic Reg	1.000000	1.000000	1.000000	1.000000	0.469697	0.415556	0.421523	0.413483	0.364433
Glove RandomForest	1.000000	1.000000	1.000000	1.000000	0.484848	0.427778	0.393522	0.396498	0.281966
ST Xgboost	1.000000	1.000000	1.000000	1.000000	0.409091	0.390870	0.356552	0.360519	0.324161
ST SVM	0.832061	0.918094	0.734113	0.780773	0.424242	0.396364	0.353216	0.342424	0.287357
ST DecisionTree	1.000000	1.000000	1.000000	1.000000	0.378788	0.322381	0.338370	0.328360	0.220041
Glove Logistic Reg	0.927481	0.946150	0.947447	0.946647	0.378788	0.329470	0.325643	0.326526	0.278342
ST RandomForest	1.000000	1.000000	1.000000	1.000000	0.378788	0.420907	0.309300	0.322527	0.294620
W2Vec Xgboost	1.000000	1.000000	1.000000	1.000000	0.393939	0.338629	0.318228	0.312261	0.244017
TFIDF Logistic Reg	1.000000	1.000000	1.000000	1.000000	0.348485	0.355556	0.298304	0.310333	0.336414
Glove Xgboost	1.000000	1.000000	1.000000	1.000000	0.378788	0.321508	0.317622	0.306841	0.301549
Glove DecisionTree	1.000000	1.000000	1.000000	1.000000	0.333333	0.283843	0.283402	0.280356	0.243387
W2Vec DecisionTree	1.000000	1.000000	1.000000	1.000000	0.333333	0.241832	0.250924	0.241103	0.226927
TFIDF DecisionTree	1.000000	1.000000	1.000000	1.000000	0.272727	0.219372	0.258065	0.229976	0.233550

Snapshot of least performing models:

TFIDF DecisionTree	1.000000	1.000000	1.000000	1.000000	0.272727	0.219372	0.258065	0.229976	0.233550
TFIDF RandomForest	1.000000	1.000000	1.000000	1.000000	0.257576	0.352235	0.207772	0.223620	0.275571
Glove SVM	0.553435	0.583214	0.428556	0.444570	0.363636	0.330926	0.241604	0.215775	0.183752
TFIDF SVM	0.866412	0.917886	0.748193	0.780828	0.363636	0.231303	0.238930	0.201176	0.225583
TFIDF Xgboost	0.996183	0.997753	0.997059	0.997389	0.242424	0.205466	0.193705	0.193571	0.190359
W2Vec SVM	0.366412	0.246592	0.227547	0.163578	0.333333	0.133333	0.202674	0.118824	0.143684

Observations:

- Sentence transformer embedding models are the best performing among all embeddings, followed by Glove
- Word2Vec and TF-IDF embeddings are least performing among the embeddings
- Logistic Regression, Random Forest and XGBoost models' performance is better than those of SVM and Decision Tree
- Best performing 3 models are:
 - Logistic Regression on Sentence Transformer
 - ✓ F1- Score (41%) and Mean CV score (36%)
 - Random Forest on Glove
 - ✓ F1- Score (39%) and Mean CV score (28%)
 - XGBoost on Sentence Transformer
 - ✓ F1- Score (36%) and Mean CV score (32%)
- Least performing models are:
 - Decision Tree on Word2Vec
 - ✓ F1- Score (18%) and Mean CV score (25%)
 - Logistic Regression on Sentence Transformer
 - ✓ F1- Score (11%) and Mean CV score (13%)

Hyperparameter Tuning

The Best performing 3 models are selected for Hyper param tuning using GridSearchCV.

This process was handled using a single method that could fit the pipeline with the given model, tune the parameters and display the metrics before and after tuning

Model performance of Logistic Regression on Sentence Transformer

	Accuracy	Precision	Recall	F1 Score
ST Logistic Regression Base Train	1.000000	1.000000	1.000000	1.000000
ST Logistic Regression Base Valid	0.469697	0.415556	0.421523	0.413483
ST Logistic Regression Best Train	0.996183	0.997753	0.996721	0.997217
ST Logistic Regression Best Valid	0.469697	0.408333	0.418849	0.409442

Model performance of Random Forest on Glove

	Accuracy	Precision	Recall	F1 Score
Glove RandomForest Base Train	1.000000	1.000000	1.000000	1.000000
Glove RandomForest Base Valid	0.439394	0.375579	0.378813	0.372893
Glove RandomForest Best Train	0.690840	0.601741	0.582583	0.584758
Glove RandomForest Best Valid	0.348485	0.268328	0.263153	0.257907

Model performance of XGBoost on Sentence Transformer

	Accuracy	Precision	Recall	F1 Score
ST XGBoost Base Train	1.000000	1.000000	1.000000	1.000000
ST XGBoost Base Valid	0.409091	0.390870	0.356552	0.360519
ST XGBoost Best Train	0.961832	0.976283	0.949013	0.961573
ST XGBoost Best Valid	0.318182	0.316667	0.264451	0.274190

Observations:

- Logistic Regression on Sentence Transformer embedding is still overfit after tuning
- XGBoost on ST performs better than Random Forest on Glove

Compare Model Performance post tuning

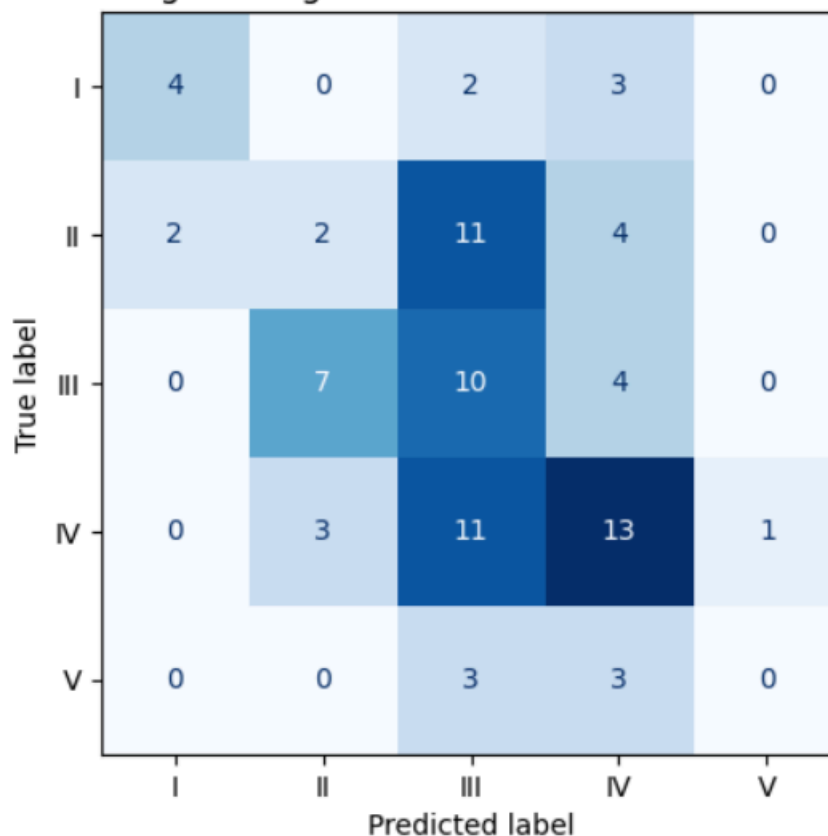
We will compare performance of each of the 3 top models after tuning using GridSearch

Model performance of Logistic Regression on Sentence Transformer

ST Logistic Regression Test Set Classification Report

	precision	recall	f1-score	support
0.0	0.67	0.44	0.53	9
1.0	0.17	0.11	0.13	19
2.0	0.27	0.48	0.34	21
3.0	0.48	0.46	0.47	28
4.0	0.00	0.00	0.00	6
accuracy			0.35	83
macro avg	0.32	0.30	0.30	83
weighted avg	0.34	0.35	0.33	83

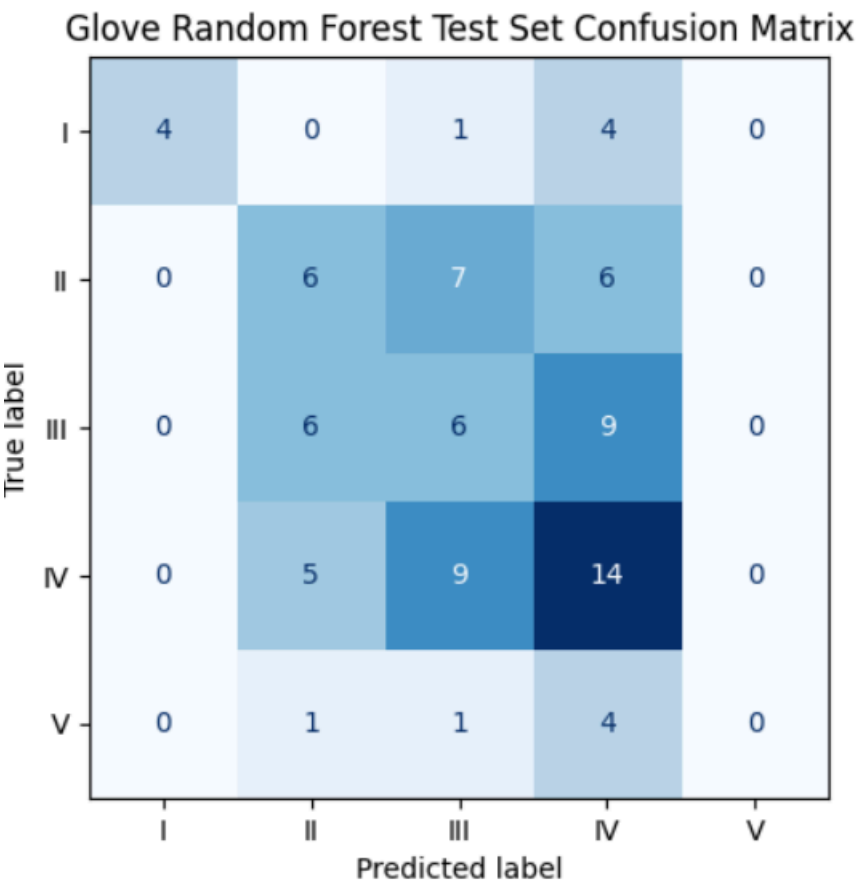
ST Logistic Regression Test Set Confusion Matrix



Model performance of Random Forest on Glove

Glove Random Forest Test Set Classification Report

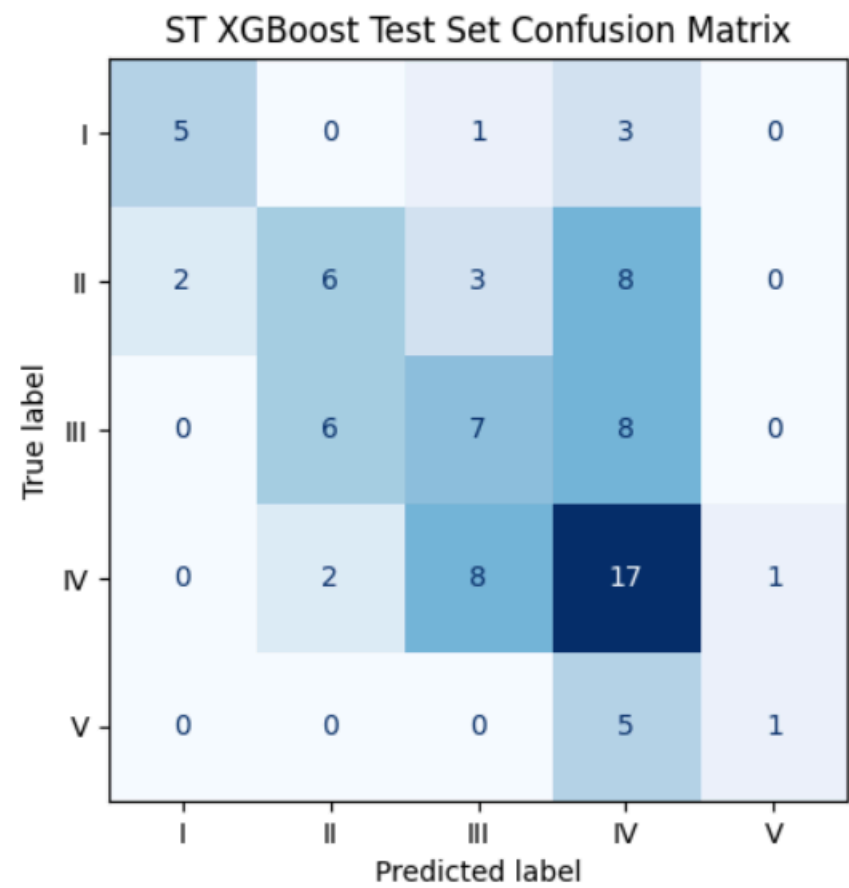
	precision	recall	f1-score	support
0.0	1.00	0.44	0.62	9
1.0	0.33	0.32	0.32	19
2.0	0.25	0.29	0.27	21
3.0	0.38	0.50	0.43	28
4.0	0.00	0.00	0.00	6
accuracy			0.36	83
macro avg	0.39	0.31	0.33	83
weighted avg	0.38	0.36	0.35	83



Model performance of XGBoost on Sentence Transformer




ST XGBoost Test Set Classification Report

	precision	recall	f1-score	support
0.0	0.71	0.56	0.63	9
1.0	0.43	0.32	0.36	19
2.0	0.37	0.33	0.35	21
3.0	0.41	0.61	0.49	28
4.0	0.50	0.17	0.25	6
accuracy			0.43	83
macro avg	0.49	0.40	0.42	83
weighted avg	0.44	0.43	0.42	83



Conclusion

Summary of performance after hyper param tuning

	Accuracy	Precision	Recall	F1 Score	
ST Logistic Regression	0.349398	0.317017	0.298037	0.295984	
Glove Random Forest	0.361446	0.392342	0.309190	0.327429	
ST XGBoost	0.433735	0.485182	0.395698	0.416278	

- XGBoost on Sentence Transformer embedding is the best performing model after hyper param tuning with F1 Score is 41% on the test set
- Random Forest on Glove embedding is next with F1 Score of 32%
- Least performing model is Logistic Regression on Sentence Transformer embedding with F1 Score of 29%
- Mis-classification is highest for Potential Accident Levels III and IV among all models but mostly in Logistic Regression on Sentence Transformer embedding
- Results can be improved by obtaining more training data (if feasible) and including advanced techniques like deep learning which can interpret the contextual information in accident description in a better way
- Another option is to use SMOTE to handle data imbalance or use data augmentation (NLP augmentation) to increase model performance