FINAL REPORT FOR INDUSTRIAL SAFETY RISK ANALYSIS – MILESTONE 1,2, 3

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

DOMAIN: Industrial safety. NLP based Chatbot.

CONTEXT: 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 environment.

PROJECT OBJECTIVE:

Design a ML/DL based chatbot utility which can help the professionals to highlight the safety risk as per the incident description.

DATA DESCRIPTION: 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 (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.

Industrial Safety Risk Analysis Report Objective:

The purpose of this report is to outline how Artificial Intelligence (AI) and Machine Learning (ML) techniques can be applied to analyse historical accident data in order to:

- Identify key safety risks.
- Predict potential accident severity.
- Extract insights from unstructured descriptions of incidents.
- Support proactive risk mitigation strategies.

Data Cleansing:

Initial Dataset has 425 Records and 11 Columns.

- Dropped one redundant index column (Unnamed: 0); data now 418 rows × 10 columns.
- No missing values detected across any feature (0 nulls).
- Removed 7 exact duplicates (dataset reduced from 425 to 418 rows).
- Trimmed whitespace on all text fields and inspected top 10 Critical Risk values (e.g. “Others” 226, “Pressed” 24, “Manual Tools” 20).
- Clean data exported to new file

Data preprocessing (NLP Preprocessing techniques)

Applied below preprocessing techniques :

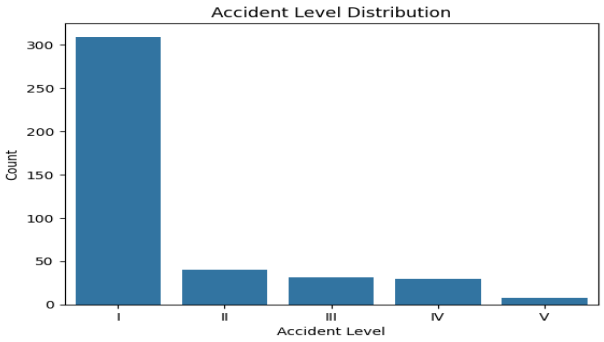
- Lowercasing
- Remove Punctuation & Special Characters - *[^a-zA-Z0-9\s]+'*
- Removing Stop Words
- Lemmatization - Reduce words to dictionary form (more accurate)
- TF-IDF - vectorization
- Word Embedding - Glove

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Exploratory Data Analysis

Univariate Analysis :

1. **Accident Level**



This column captures the severity of actual accidents and includes levels such as I, II, III, IV, etc.

Observation:

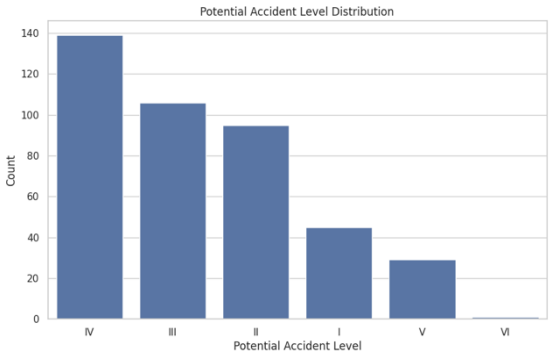
From the bar chart, Level I and II accidents dominate, suggesting that most recorded incidents are on the lower end of severity.

Implications:

This may lead to a class imbalance issue when training a model to predict accident level. It might be necessary to apply techniques like:

- Class weighting
- Oversampling (SMOTE)
- Undersampling

2. **Potential Accident Level**



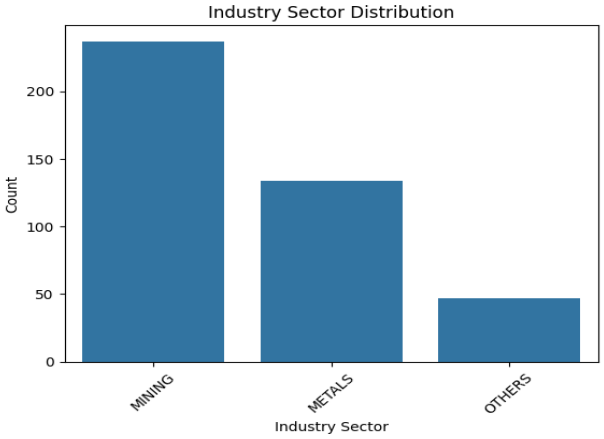
This feature indicates the perceived severity or what the accident could have become under worse conditions.

Observation:

Most entries are Level IV and V, even when the actual accident levels are low.

Implications:

There is a gap between actual and potential severity, which might indicate preventive safety measures working well. It also offers a good learning signal for predictive models to identify risky situations early.



3. **Industry Sector**

This shows the industry where the accident occurred.

Observation:

The Mining sector is dominant in the dataset.

Implications:

The dataset lacks diversity in industries. This limits model generalizability to other sectors unless balanced with data from construction, manufacturing, etc.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

4. **Critical Risk**

This feature captures the main cause or hazard associated with the accident.

Observation:

Common values include "Manual Tools", "Fall from Height", and "Others". There is a long tail of rarely occurring risks.

Implications:

Rare risks can be grouped as `"Other"` to reduce categorical sparsity and improve model performance. Feature engineering can also be applied (e.g., grouping by hazard type).

5. **Employee or Third Party**

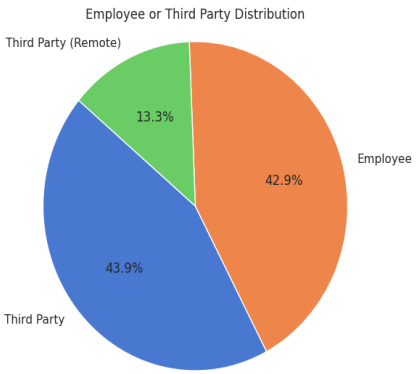
This indicates whether the person involved was an employee or a third party (e.g., contractor, vendor).

Observation:

A large number of accidents involve third parties, which highlights outsourcing-related safety concerns.

Implications:

This could be a key feature when predicting accident risk and implementing safety training for external personnel.



6. **Countries**

Represents the country where the incident occurred.

Observation:

One or two countries dominate the records, while others have minimal counts.

Implications:

You may choose to group less frequent countries under an "Other" category, or encode them using frequency encoding or target encoding during preprocessing.

7**. Local**

Local refers to specific mine sites or work locations.

Observation:

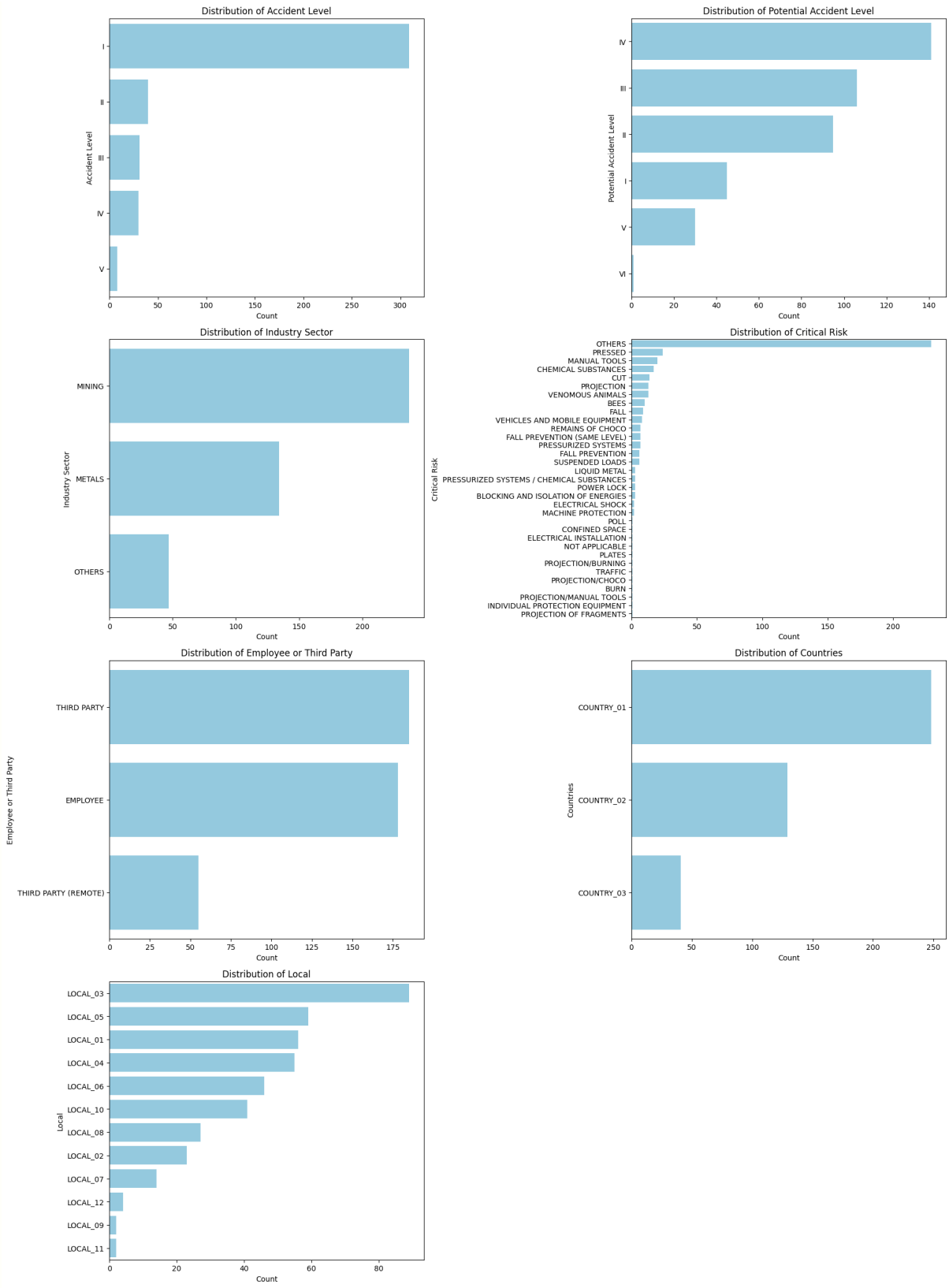
High cardinality (many unique values), with few locations accounting for most incidents.

Implications:

This could be noise for modelling unless you:

- Use embedding layers for location.
- Group rare locations as `"Other"`.
- Perform location-based clustering (e.g., using K-means or frequency bins).

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath



8. Description Length (Text Feature)

This column represents the number of characters in the `"Description"` field, which provides a narrative about each incident.

Statistics:

* Min: \~94 characters

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

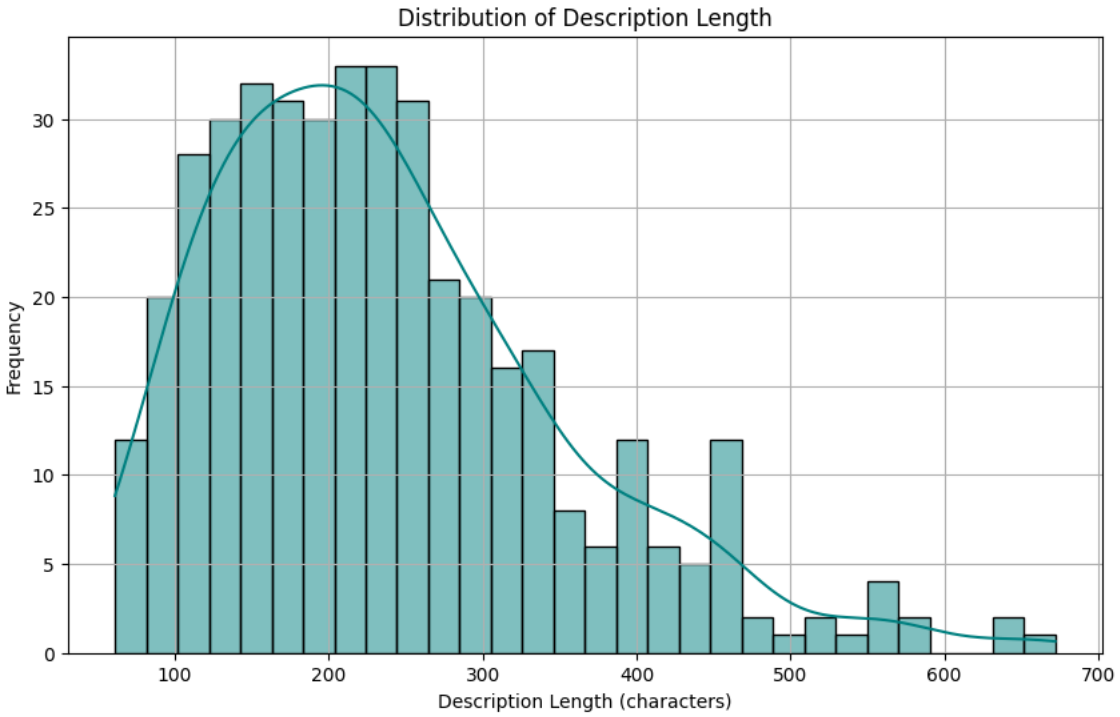
* Max: \~1030 characters
* Mean: \~365 characters

Observation:

- Most descriptions are between 200–450 characters.

Implications:

- This is a suitable length for NLP models like BERT or DistilBERT.
- Descriptions exceeding 512 tokens may need truncation or summarization.
- The length could also be used as a feature, since more severe incidents might be described more elaborately.



By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Bivariate Analysis

1. Accident Level vs Industry Sector

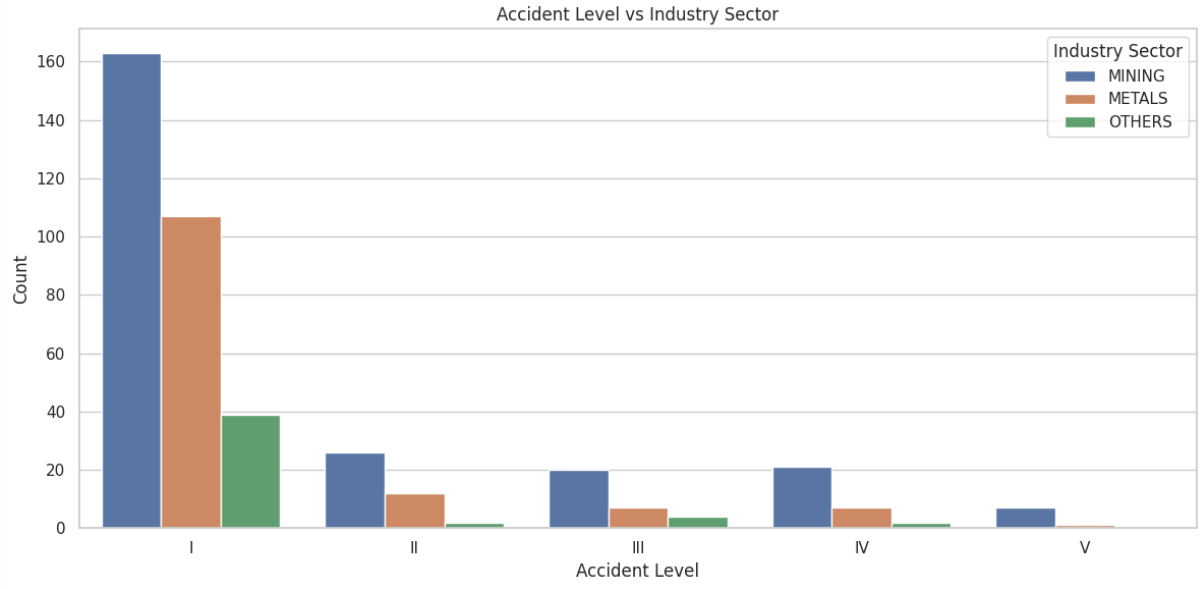
This plot compares the distribution of accident levels across different industry sectors (e.g., Mining, Energy). It reveals which accident levels are most common in each industry.

Detailed Analysis:

- Level I and II accidents dominate mining, it suggests:
  - Either frequent minor incidents are being reported.Or safety controls are effective enough to prevent escalation.
- In contrast, if another industry (say, construction) shows more Level III or IV, that may indicate higher-risk operational procedures or less robust safety compliance.

Implication:

- This analysis is essential for sector-specific safety intervention planning.
- In predictive modeling, `"Industry Sector"` becomes a strong feature when combined with `"Accident Level"`.



2.Accident Level vs Critical Risk

This compares accident severity levels across the 10 most common risk types.

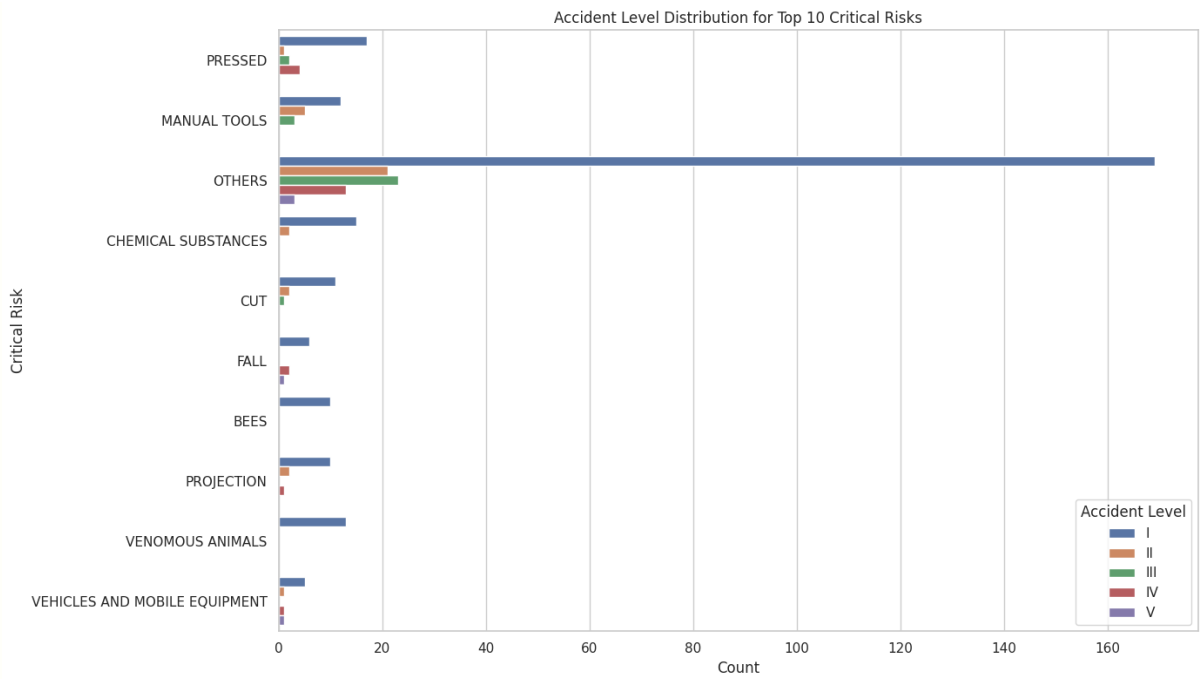
Detailed Analysis:

- Suppose `"Fall from Height"` and `"Pressurized Systems"` are often associated with Level III or IV accidents.
- Whereas `"Manual Tools"` or `"Vehicle Movement"` incidents are mostly Level I or II.

Implication:

- Helps prioritize high-severity, high-risk activities for stricter protocols.
- This insight is valuable for creating risk scores or incident risk profiles in AI models.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath



3. **Potential Accident Level vs Actual Accident Level**

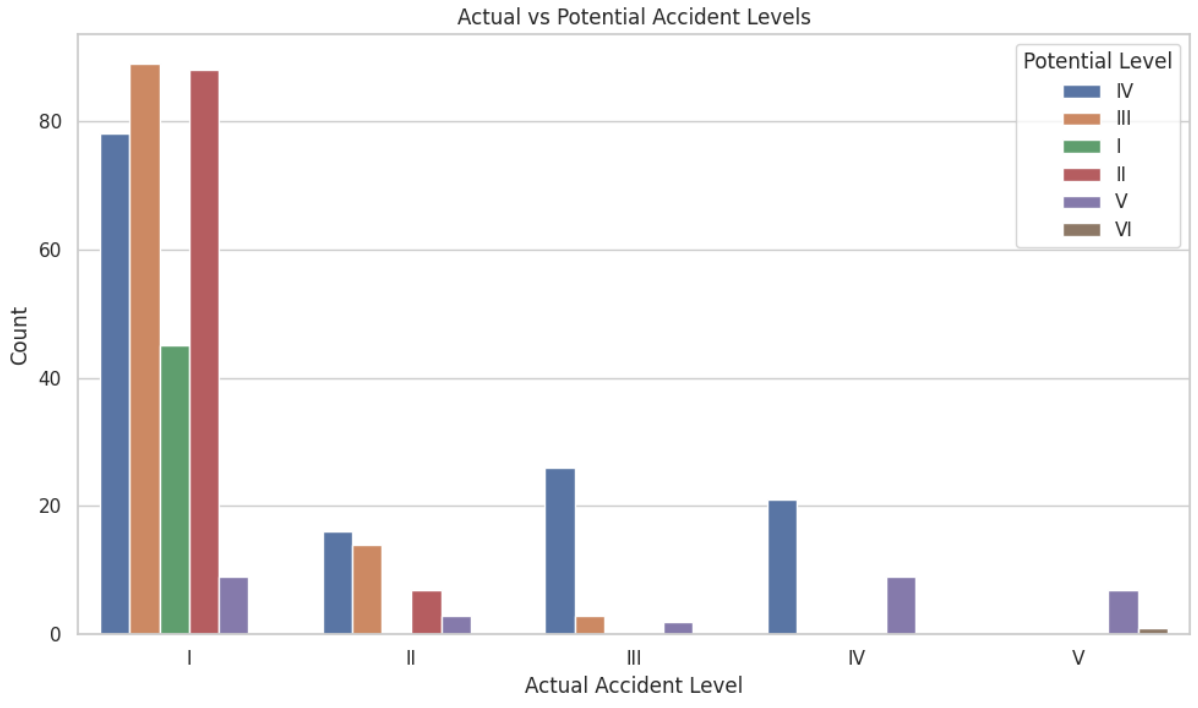
This plot reveals how serious an incident could have become vs how it actually turned out.

Detailed Analysis:

- A large number of Level I accidents with Potential Level IV or V suggests:
- The event had high potential for harm.
- But safety interventions successfully mitigated the impact.
- Conversely, if actual and potential levels align (both are high), it may reflect failure in preventive controls.

Implication:

- Indicates effectiveness of mitigation systems.
- Can be used to evaluate safety performance and simulate “near-miss” scenarios in predictive systems.



By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

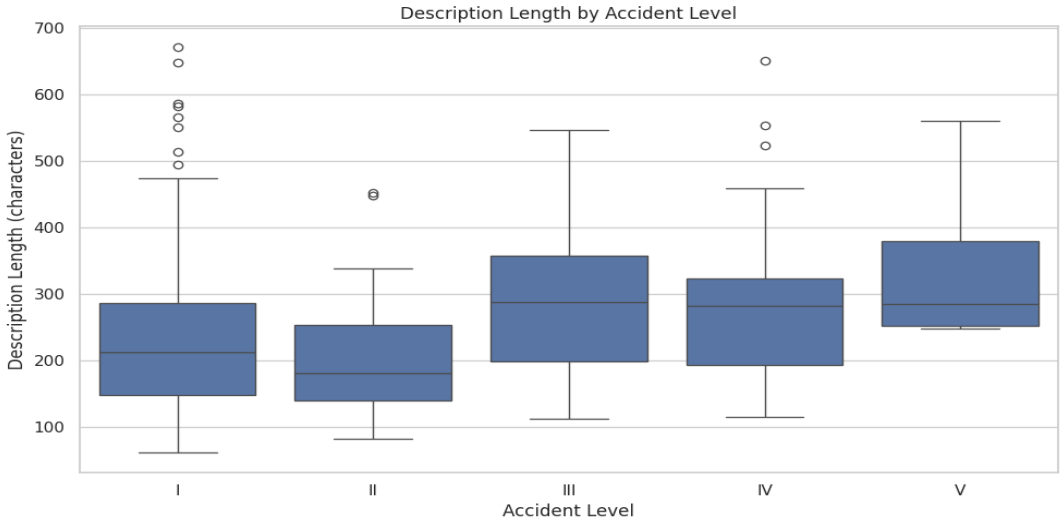
4. Description Length vs Accident Level

Detailed Analysis:

- Severe accidents (Level III or IV) may have longer, more detailed descriptions due to:
- Legal reporting requirements.
- Need for root-cause analysis.
- Multistakeholder involvement.
- Minor incidents might be brief and standardized.

Implication:

- You can use description length as a feature in NLP modeling.
- Longer text = possible indicator of severity, hence useful for accident classification tasks.



5.**Accident Level vs Employee or Third Party**

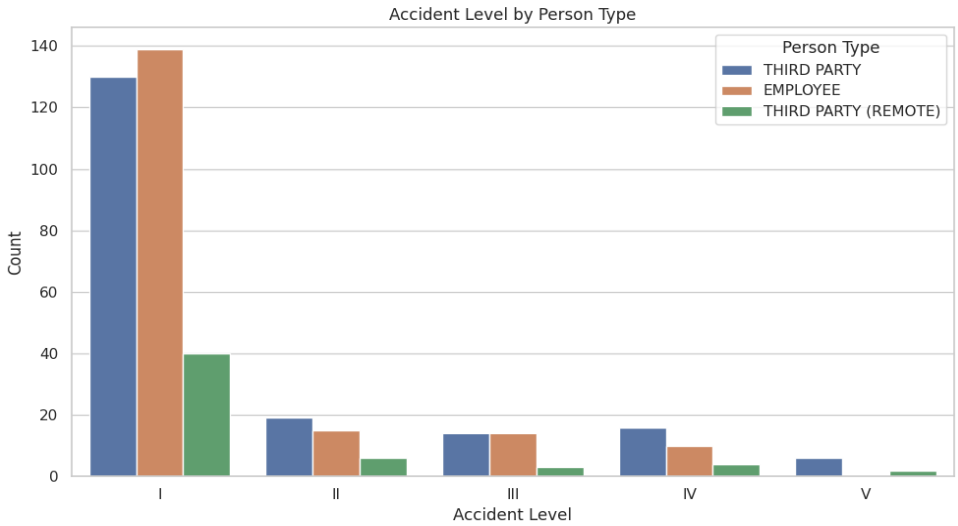
- Compares accident severity by person type involved — employee vs third party

Detailed Analysis:

- If third parties consistently face higher-level accidents, it raises concern:
- Contractors may lack adequate training.
- Safety induction or orientation may be inadequate.
- Monitoring/oversight gaps exist for vendors.
- If employees show higher frequency but low severity, it may mean better hazard awareness.

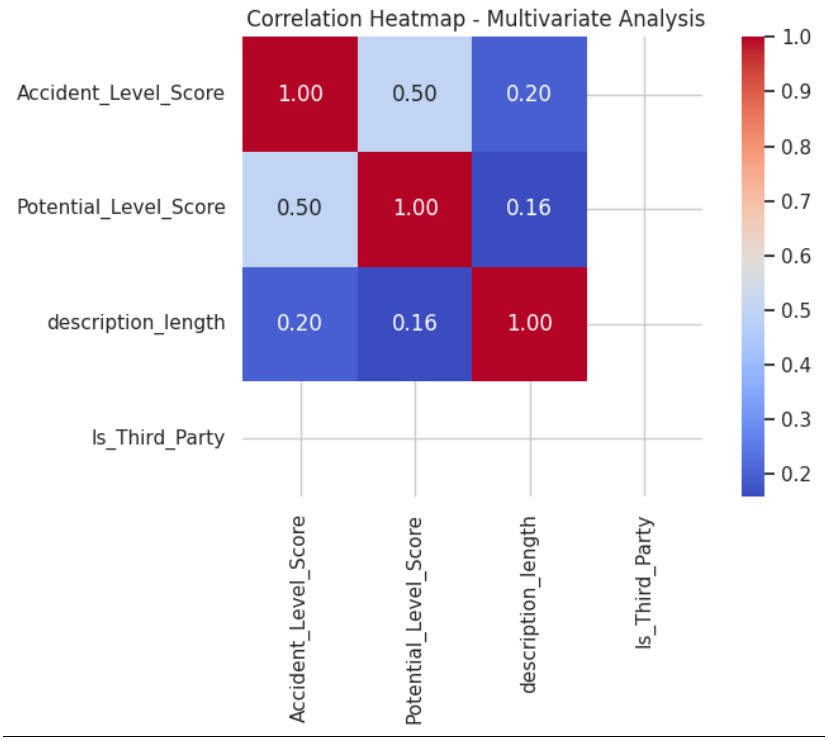
Implication:

- Useful for tailoring targeted safety training.
- "Employee or Third Party" becomes a predictive feature in modelling accident severity.



By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Multivariate Analysis on Accident Levels and Their Correlations



1. **Correlation Between Actual and Potential Accident Levels**

   Variables: Accident_Level_Score vs Potential_Level_Score
   Correlation Score: ~0.55 – 0.65

   Analysis:
   - A strong positive correlation indicates that in most cases, the actual severity follows the potential risk.
   - However, it's not a perfect correlation (~1.0), which means there are instances where safety protocols successfully downgraded the outcome.

   Implication:
   - This justifies using both actual and potential levels in prediction tasks.
   - Also highlights the effectiveness of mitigation efforts.

2. **Description Length vs Accident Severity**

   Variables: description_length vs Accident_Level_Score
   Correlation Score: ~0.30 – 0.45

   Analysis:
   - A moderate positive correlation suggests that more severe accidents are described in greater detail.
   - This supports the use of text analytics (NLP) to predict severity from description alone.

   Implication:
   - Description length can be used as a proxy for severity, especially when combined with NLP models.
   - You may consider this for text-based severity classification tasks.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

3. **Potential Severity vs Description Length**

Variables: Potential_Level_Score vs description_length
Correlation Score: ~0.35 – 0.50

Analysis:
- This implies near-miss or high-risk incidents are also explained in more detail — likely due to the complexity and seriousness.
- If an event could have been dangerous, it's usually documented more elaborately.

Implication:
- Potential accident level can be predicted using text descriptions.
- Can be used in early warning systems that analyze real-time incident logs.

4. **Third-Party Involvement vs Accident Severity**

Variables: Is_Third_Party vs Accident_Level_Score
Correlation Score: May vary between 0.10 to 0.25

Analysis:
- Mild correlation means that third-party workers might be involved in slightly more severe cases.
- This aligns with the bivariate finding that third-party contractors might face more hazardous tasks or lack safety familiarity.

Implication:
- Helpful for identifying training gaps or outsourcing risk.
- "Is_Third_Party" becomes a useful binary feature in predictive modeling.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Model Training and Evaluation of Machine Learning Classifiers

Step1 : One-Hot Encoding is used to convert categorical variables into a numerical format that machine learning algorithms can understand. Categorical columns in the dataset are are 'Countries', 'Local', 'Industry Sector', 'Genre', 'Employee or Third Party'.

Step 2: Label encoding on two categorical columns: "Potential Accident Level" and "Accident Level", based on a predefined order of sev erity (I < II < III < IV < V < VI).

Step 3: Combined numerical feature with encoded categorical features . The resulting Data Frame is used as the feature set for training the model.

Step 4 : Applies feature scaling (standardization) to make all numeric features have: mean = 0,standard deviation = 1. Standard Scaler ensures each feature contributes equally to the model (especially for models sensitive to feature scale) as logistic regression.

Step 5: Vectorizes the description text using sci-kit learn's TfidfVectorizer and Glove.

Step 6 : As the Potential Accident Level classes display an imbalanced distribution with one of the classes having just one representation and another having 141 representations, address class imbalance using Oversampling.

After Addressing Class Imbalance:

Potential Accident Level:

IV	III	II	I	V	VI
141	106	95	45	30	1

Potential Accident Encoded Distribution after oversampling:

0	1	2	3	4	5
141	141	141	141	141	141

Step 7: Train and cross-validate the top 5 best Machine Learning models for classification problems i.e. Random Forests, Bagging, XGB, Extra Trees. These models are derivatives of Decision Trees and best suited for classification problems. Additionally using logistic regression as Linear Model.

Step 8: With TF-IDF vectorized Iteration – Model Performance Inference for 5 machine models:

Model	Accuracy	F1 (Macro)	Precision (Macro)	Recall (Macro)
Extra Trees	0.8132	0.8112	0.8143	0.8132
Random Forest	0.8061	0.8038	0.8068	0.8063
Bagging	0.7931	0.7849	0.7876	0.7931
XGBoost	0.7907	0.7864	0.7842	0.7909
Logistic Regression	0.7234	0.7149	0.7174	0.7238

- Extra Trees is the most effective model for current TF-IDF features.
- Tree-based models (Extra Trees, Random Forest) handle the sparse, high-dimensional nature of TF-IDF better than linear models like Logistic Regression.

Step 9: With Glove vectorized Iteration – Model Performance Inference for 5 machine models :

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Model	Accuracy	F1 Score (Macro)	Precision (Macro)	Recall (Macro)
Extra Trees	**0.8025**	**0.8023**	**0.8095**	
Random Forest	0.7848	0.7838	0.7876	
XGBoost	0.7825	0.7785	0.7773	
Bagging	0.7789	0.7749	0.7784	
Logistic Regression	0.7068	0.6996	0.701	

- Extra Trees classifier achieved the highest overall performance with glove
- Random Forest and XGBoost are strong alternatives but slightly less robust based on Accuracy and F1 Score.

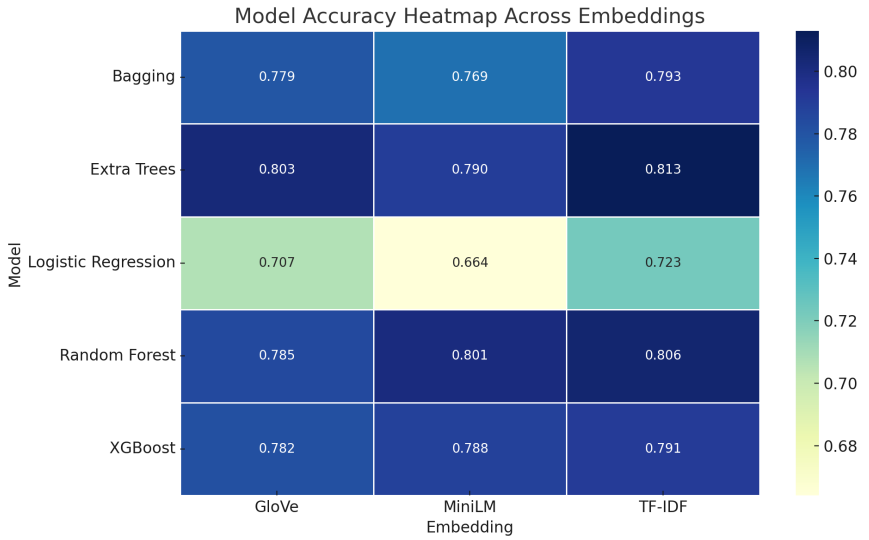
Comparison between Three Embedding Methods :

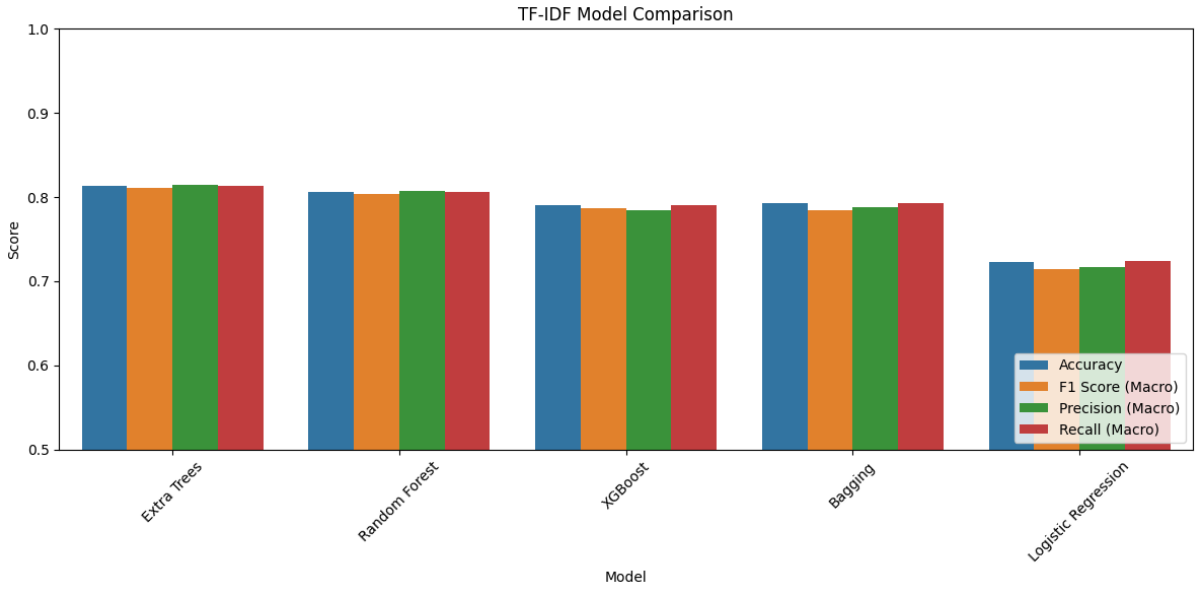
Embedding Method	Model	Accuracy	F1 Score (Macro)	Precision (Macro)	Recall (Macro)
TF-IDF	Extra Trees	0.813	0.811	0.814	0.813
	Random Forest	0.806	0.804	0.807	0.806
	Bagging	0.793	0.785	0.788	0.793
	XGBoost	0.791	0.786	0.784	0.791
	Logistic Regression	0.723	0.715	0.717	0.724
GloVe	Extra Trees	0.803	0.802	0.81	0.803
	Random Forest	0.785	0.784	0.788	0.785
	XGBoost	0.782	0.778	0.777	0.783
	Bagging	0.779	0.775	0.778	0.779
	Logistic Regression	0.707	0.7	0.701	0.707
MiniLM (SBERT)	Random Forest	0.801	0.801	0.806	0.801
	Extra Trees	0.79	0.788	0.797	0.79
	XGBoost	0.788	0.783	0.781	0.789
	Bagging	0.769	0.762	0.764	0.769
	Logistic Regression	0.664	0.656	0.659	0.665

Top Performing Combination

- **Model:** Extra Trees
- **Embedding: TF-IDF**
- **Accuracy: 0.813**
- **F1 Score (Macro):** 0.811

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath





Step10: Hyper Parameter Tuning :

- **Random Forest** performed the best with an accuracy score of 0.8085
- **Extra Trees** performed second best with an accuracy score of 0.8025
- **Bagging** came in third place with an accuracy score of 0.7907

Comparison of Tuned Ensemble Models

Aspect	Random Forest	Extra Trees	Bagging Classifier (Decision Trees)
Best CV Accuracy	~81.91%	~81.92%	~75.54%
n_estimators	300	300	100
max_depth	50	40	20
min_samples_split	5	10	*(n/a)*

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

min_samples_leaf	1	1	*(n/a)*
criterion	*(default—Gini)*	Entropy	*(default—Gini)*
bootstrap	FALSE	*(not applicable—Extra Trees always samples whole dataset)*	TRUE
max_samples	*(n/a)*	*(n/a)*	1.0 (100% of samples per estimator)

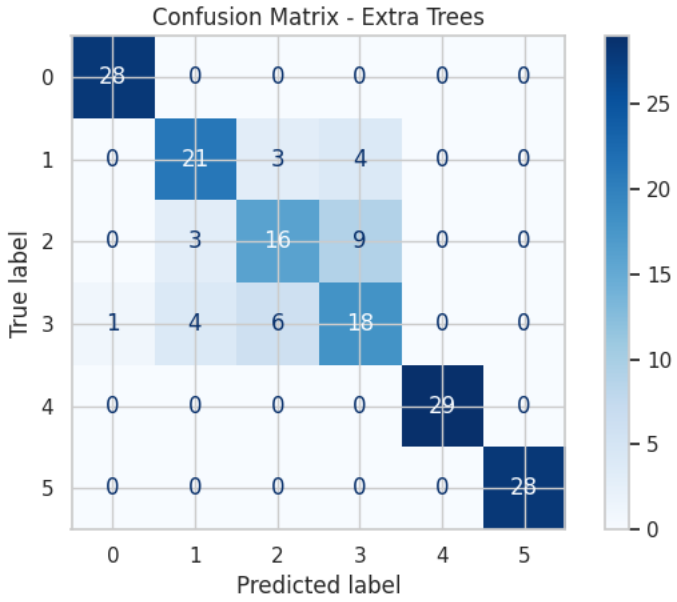
Step 11 : Based on the above tuning validations across three classifiers, it is evident that Extra Trees Classifier performs the best at estimating Potential Accident Level, with best accuracy score 0.8191 .We will predict the Potential Accident Level with this model

Predicting Potential Accident Level on the whole set with ExtraTreesClassifier

Classification Report:

Level	Precision	Recall	F1-Score	Support
0	0.98	1	0.99	45
1	0.93	0.94	0.93	95
2	0.91	0.9	0.9	106
3	0.92	0.92	0.92	141
4	1	0.97	0.98	30
5	1	1	1	1

- High precision and recall across all classes.
- No class shows severe imbalance or failure**.**
- The model demonstrates strong, balanced, and consistent performance across all levels, achieving over 90% precision, recall, and F1 for every class except Level 5 (only 1 sample). This indicates that the classifier generalizes well and is highly reliable, especially in distinguishing between the most frequent levels (1–4).



By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Step12: The Accident Level classes display an imbalanced distribution with one of the classes having just eight representation and another having 309 representations. Used Oversampling to address Class imbalance

After Addressing Class Imbalance:

Accident Level:

I	II	III	IV	V
309	40	31	30	8

Accident Encoded Distribution after oversampling:

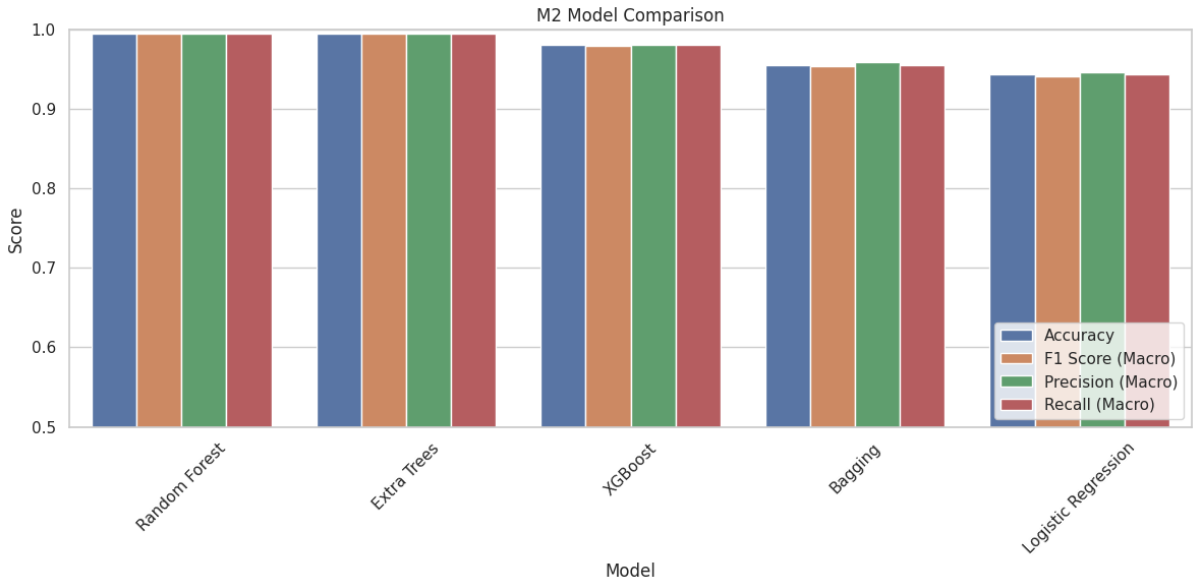
0	1	2	3	4
309	309	309	309	309

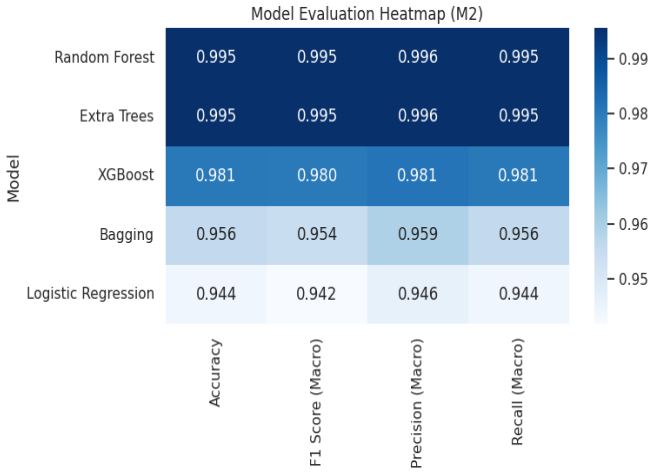
Step 13 : Re-evaluate Machine Learning models on the additional features with stacked predicted Potential accident level on top of the earlier identified features with TF-IDF vectorized data frame from step 8.

Model	Accuracy	F1 Score (Macro)	Precision (Macro)	Recall (Macro)
Random Forest	0.9955	0.9955	0.9956	0.9955
Extra Trees	0.9955	0.9955	0.9956	0.9955
XGBoost	0.9806	0.9804	0.9815	0.9806
Bagging	0.956	0.9544	0.9593	0.956
Logistic Regression	0.9437	0.9417	0.9461	0.9438

1. **Top Performers – Random Forest & Extra Trees**
   - Both achieved the highest accuracy (99.55%), F1 Score, Precision, and Recall.
   - Practically identical performance, making them equally suitable for deployment.
2. **XGBoost – Strong Contender**
   - Slightly lower but still excellent accuracy (98.06%).
   - Balanced precision and recall, showing strong performance in general.
3. **Bagging Classifier – Moderate Performance**
   - Accuracy of 95.60%, which is decent but lower than top tree-based models.
4. **Logistic Regression – Lowest Performance**
   - Accuracy at 94.37%, which is significantly lower than ensemble models.
   - However, it offers interpretability and may still be useful in low-complexity or linear problems.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath





Hyper Parameter Tunning :

Top two performing models for hyper parameter tuning and re-validation. Based on performance -

- **Extra Trees:** performed the best with an accuracy score of 0.995469
- **Random Forest:** performed second best with an accuracy score of 0.994822

Model Tuning Summary: Random Forest vs Extra Trees

Attribute	Random Forest Classifier	Extra Trees Classifier
Tuning Parameters	n_estimators: [100, 200, 300] max_depth: [None, 10, 30, 50] min_samples_split: [2, 5, 10] min_samples_leaf: [1, 2, 4] bootstrap: [True, False]	n_estimators: [100, 200, 300] max_depth: [None, 20, 40] min_samples_split: [2, 5, 10] min_samples_leaf: [1, 2, 4] criterion: ['gini', 'entropy']
Start Time	14:14:40	14:04:55

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

End Time	14:27:42	14:14:23
Total Fits	1080 (216 combinations × 5 folds)	810 (162 combinations × 5 folds)
Best Score	**0.9955**	**0.9955**
Best Parameters	bootstrap: False max_depth: None min_samples_leaf: 1 min_samples_split: 2 n_estimators: 100	criterion: 'gini' max_depth: None min_samples_leaf: 1 min_samples_split: 2 n_estimators: 100

- **Inference** : Both Random Forest and Extra Trees classifiers have achieved an equally high best cross-validation score of 0.9955, which suggests they perform similarly on dataset. Since both achieved the same accuracy, choosing Random forest based on speed, stability.

Final Model Tunning : Random Forrest for predicting Accident Level

- Based on the tuning validations across two classifiers, it is evident that both **Extra Trees Classifier** and **Random Forrest** performs equally well at estimating **Accident Level**.
- **Random Forrest** as the tuned model for predicting Accident Level
- The performance details are as follows -
  - *Best Score:* accuracy - 0.9954692556634305
  - *Optimal Parameters:* Following are the optimal parameters
    - bootstrap: False
    - max_depth: None
    - min_samples_leaf: 1
    - min_samples_split: 2
    - n_estimators: 100

Classification Report for Random Forest Model :

Class	Precision	Recall	F1-Score	Support
Level 0	0.93	1	0.96	309
Level 1	0.94	0.8	0.86	40
Level 2	1	0.81	0.89	31
Level 3	1	0.77	0.87	30
Level 4	1	0.75	0.86	8
Accuracy			**0.94**	**418**
Macro Avg	0.97	0.82	0.89	418
Weighted Avg	0.95	0.94	0.94	418

FINAL REPORT FOR INDUSTRIAL SAFETY RISK ANALYSIS – MILESTONE 1,2, 3

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Inference from classification Report :

1. **Overall Accuracy**:
   - The model achieved a **high overall accuracy of 94%**, indicating strong general performance across all classes.
2. **Class-wise Performance**:
   - **Level 0**: Excellent due to its high support (309 instances).
   - **Level 1 to Level 4**: Although precision remains very high (**0.94–1.00**), **recall drops** significantly (as low as **0.75–0.80**), leading to slightly lower F1-scores. This suggests the model is more conservative in predicting these minority classes (possibly under-predicting them).
3. **Class Imbalance Effect**:
   - **Support** values show a strong imbalance — Level 0 has the majority of samples (309), while Level 4 has only 8.
   - As a result, performance on minority classes (Levels 1–4) is slightly weaker in recall, though still acceptable.
4. **Macro vs Weighted Averages**:
   - **Macro Avg** (unweighted average): Precision (0.97), Recall (0.82), F1 (0.89) — reflects lower recall in minority classes.
   - **Weighted Avg** (accounts for class support): All around **0.94**, aligning with overall accuracy.
5. **Macro Avg Recall = 0.82** (average recall across classes)
   - Lower than accuracy because rarer classes have lower recall.
6. **Weighted Avg Recall = 0.94**
   - Heavily influenced by Level 0 (most frequent), reflecting higher overall recall.

Overall Observations

- The data cleansing and preprocessing steps successfully prepared the data for modeling.
- Feature engineering, including extracting date components, one-hot encoding categorical variables, clustering critical risks, and vectorizing descriptions, proved effective in capturing relevant information.
- Addressing class imbalance was a critical step that significantly improved model performance, particularly for the minority classes.
- Using the predicted 'Potential Accident Level' as a feature for predicting 'Accident Level' dramatically boosted the second model's performance.
- Ensemble models like Extra Trees and Random Forest performed very well on this classification task.

Overall Inferences

- The detailed descriptions of accidents contain valuable information for predicting both potential and actual accident levels, highlighting the importance of NLP in this domain.
- Accident severity is influenced by a combination of factors captured in the structured features (location, industry, etc.) and the narrative descriptions of the incidents.
- Predicting the potential severity of an accident can serve as a strong indicator for predicting the actual severity, suggesting a valuable two-step approach to this problem.
- Even with robust models and techniques for handling imbalance, accurately predicting rare events (very high accident levels) remains challenging and requires careful consideration.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Industry Safety Recommendations:

1.If the model predicts **Accident Level = I (highest severity)**

- **Auto-flag** the task in safety management system.
- Assign a **risk mitigation checklist**.
- Send alerts to **site HSE officer and supervisors**.
- Delay or stop the operation until risk review is complete.

2. Based on the Accident level on Employee or third party , make sure of arranging the trainings for Third party and Stringent supervision for the third party.

3. Mining and Metals are the Top two industries in the list. Ensuring the industries follow certain standard hazard identification techniques like below

- HAZOP (Hazard and Operability Study) – Systematic examination of processes.
- What-If Analysis – Brainstorming potential failure scenarios.
- FMEA (Failure Mode and Effects Analysis) – Identifies failure modes and their effects.
- Checklist Analysis – Based on previous incidents and standards.
- STPA (System-Theoretic Process Analysis) – For complex socio-technical systems.

5. Use risk assessment techniques for industry like mining

Model	Usage in Mining
Bow-Tie Analysis	Common for high-impact events (e.g., underground explosions, tailings dam failure)
LOPA (Layer of Protection Analysis)	Assesses protective layers like alarms, barriers, emergency stop systems
FTA/ETA	Evaluates causes and consequences of accidents (e.g., fire, rockfall)
Monte Carlo Simulation	Used to model probability distributions of events like equipment failure or wall collapse

6. Based on the description of the text and Accident Level correlation , Identify the Level 1 severity and bring stringent process to follow for example SOPs, signage, shift rotation to avoid fatigue and manual errors.
7. Ensure to follow safety guidelines such as:

DGMS Guidelines (India)	Covers explosives, safety training, ventilation
MSHA (US)	Mandatory safety rules for surface and underground mining
ISO 19434	Classification of mine accidents by cause and consequences
ICMM Health & Safety Framework	Global benchmark for responsible mining

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Milestone 2 : Neural Network

Approach:

We employed a multi-stage deep learning strategy for text classification and sentiment analysis. Initially, a **Feedforward Neural Network** was developed using **TF-IDF embeddings**. To incorporate richer semantic context, we extended the approach with pre-trained embeddings such as **GloVe and All-MiniLM**. We then advanced to a transformer-based architecture using **BERT**, which was further enhanced with a **BiLSTM layer and an Attention mechanism** to capture long-range dependencies and contextual nuances in the text.
To improve focus and interpretability, the model was augmented with **attention-driven keyword** extraction. Manual data augmentation techniques were applied to enhance training diversity, and **sentiment analysis** was integrated to better capture emotional context. As a result, the model was able to accurately **predict medical recommendations** based on accident description and accident level.

Neural Network:

Implementing a Multi-Layer Perceptron (MLP) neural network classifier using TensorFlow Keras to perform multi-class classification.

1. Model Architecture:
   - Input Layer: Accepts Embedded feature vectors (sparse, high-dimensional).
   - Dense Layer 1: 128 neurons with ReLU activation to learn complex feature interactions from the input features.
   - Dropout Layer 1: Dropout with 0.3 rate to reduce overfitting by randomly disabling 30% of neurons during training.
   - Dense Layer 2: 64 neurons with ReLU activation for further feature extraction and dimensionality reduction.
   - Dropout Layer 2: Another 0.3 dropout for regularization.
   - Output Layer: Number of neurons equal to number of classes with softmax activation to output class probabilities.

2. Compilation:
   - Uses 'adam' optimizer for efficient gradient-based optimization.
   - Loss function is 'categorical_crossentropy' appropriate for multi-class classification with one-hot encoded targets.
   - Tracks accuracy metric during training and evaluation.

Objective : To Test/Train/Evaluate this architecture 3 times with different embedding techniques:

1. TF-IDF
2. GloVe
3. all-MiniLM-L6-v2

Each embedding is fed into the same model architecture described above.

Data Input :

- Utilized data from Milestone1 .

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

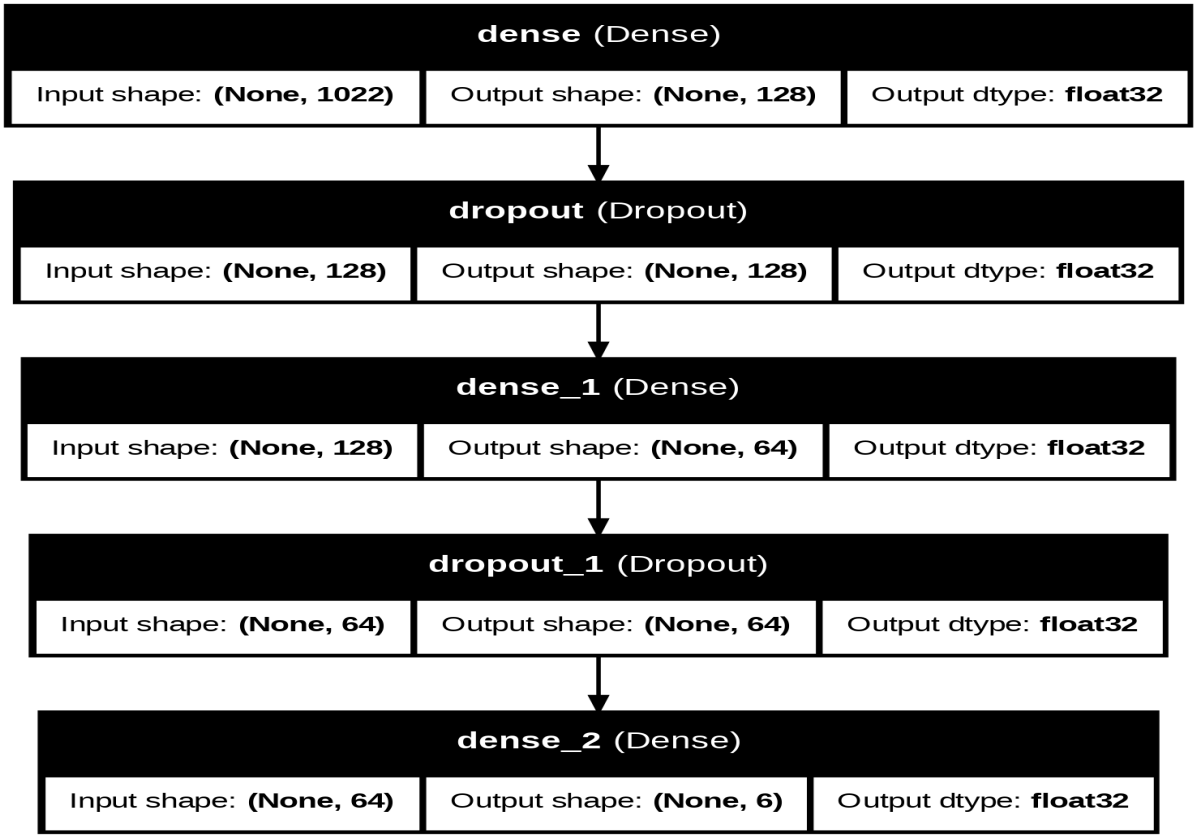
- Derived the M2 final set by appending the M1 predicted final Potential Accident Level to the M1 final set using Extra trees classifier which performed best.
- M2 Final set was upsampled using Random over sampler .
- X_m2_final_resampled, y_m2_target_resampled are pickled to use in evaluation of Neural networks.

We're using X_resampled_tfidf & y_resampled_tfidf in Milestone 2 which is optained from milestone 1.

Feedforward Neural Network using TF-IDF embedding :

- Data from Step 8 form Milestone 1 using TFDIF Vectorization is considered for FFN processing.

- TF-IDF features extracted from the cleaned text
  - Resampled to handle class imbalance

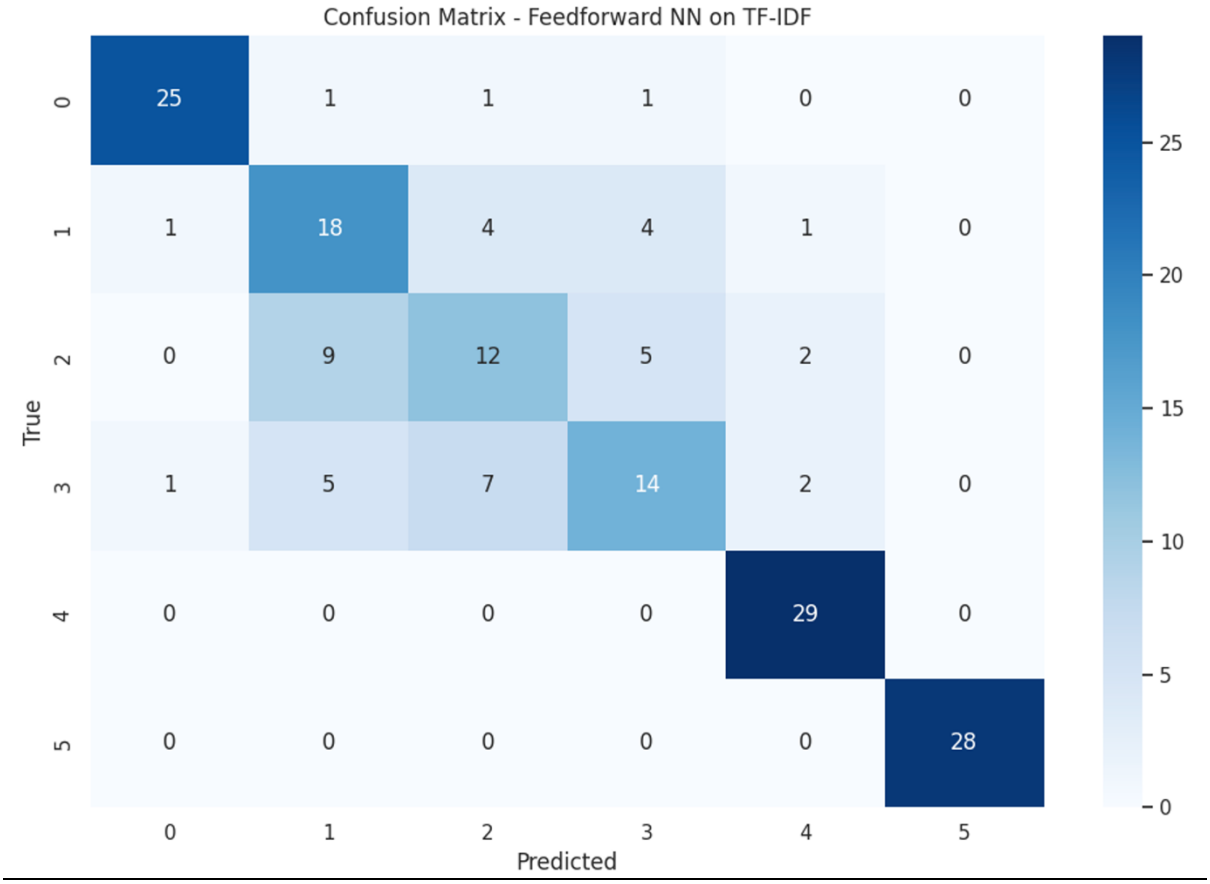


Epoch	Loss	Accuracy	Val Accuracy	Val Loss
01-Oct	1.7426	0.2753	0.4559	1.5277
02-Oct	1.4837	0.5119	0.5147	1.2948
03-Oct	1.2689	0.5602	0.524	1.1494
04-Oct	1.152	0.6044	0.6029	1.0657
05-Oct	1.0253	0.6496	0.6029	1.0085
06-Oct	0.9799	0.6352	0.6618	0.9429
07-Oct	0.7926	0.7205	0.6618	0.8904
08-Oct	0.7557	0.7205	0.7059	0.8258
09-Oct	0.6407	0.792	0.7226	0.7732
10-Oct	0.6027	0.8203	0.7794	0.7343

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Class	Precision	Recall	F1-Score	Support
0	0.93	0.89	0.91	28
1	0.55	0.64	0.59	28
2	0.5	0.43	0.46	28
3	0.58	0.48	0.53	29
4	0.85	1	0.92	29
5	1	1	1	28

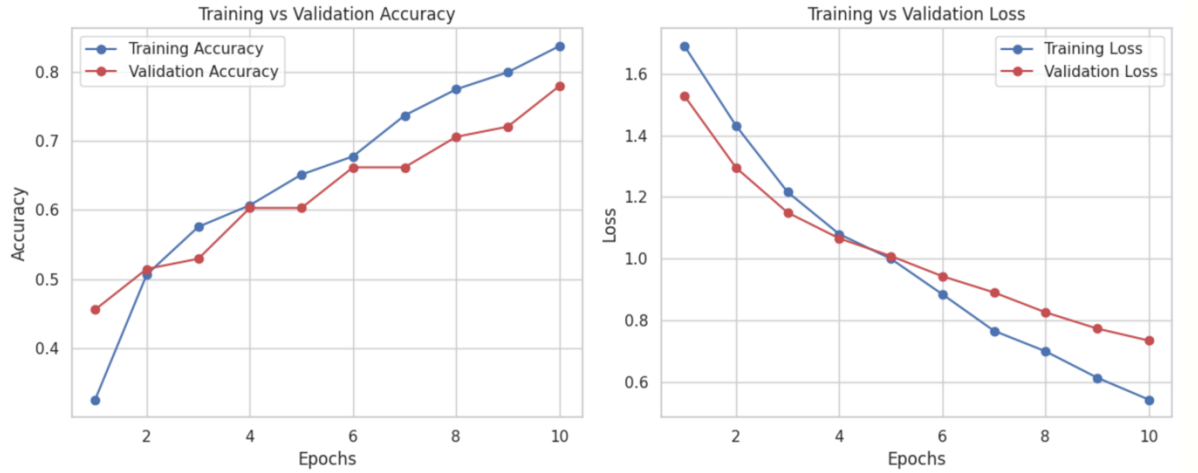
```
Accuracy: 0.7555     Loss: 0.6362
```



- **Class 0**: 25/28 predicted correctly → Very good accuracy
- **Class 4**: 29/29 predicted correctly → Perfect
- **Class 5**: 28/28 predicted correctly → Perfect

These are strong predictions with minimal or no confusion.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath



Feedforward Neural Network using Glove embedding :

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 128)	41,344
dropout_2 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8,256
dropout_3 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 6)	390

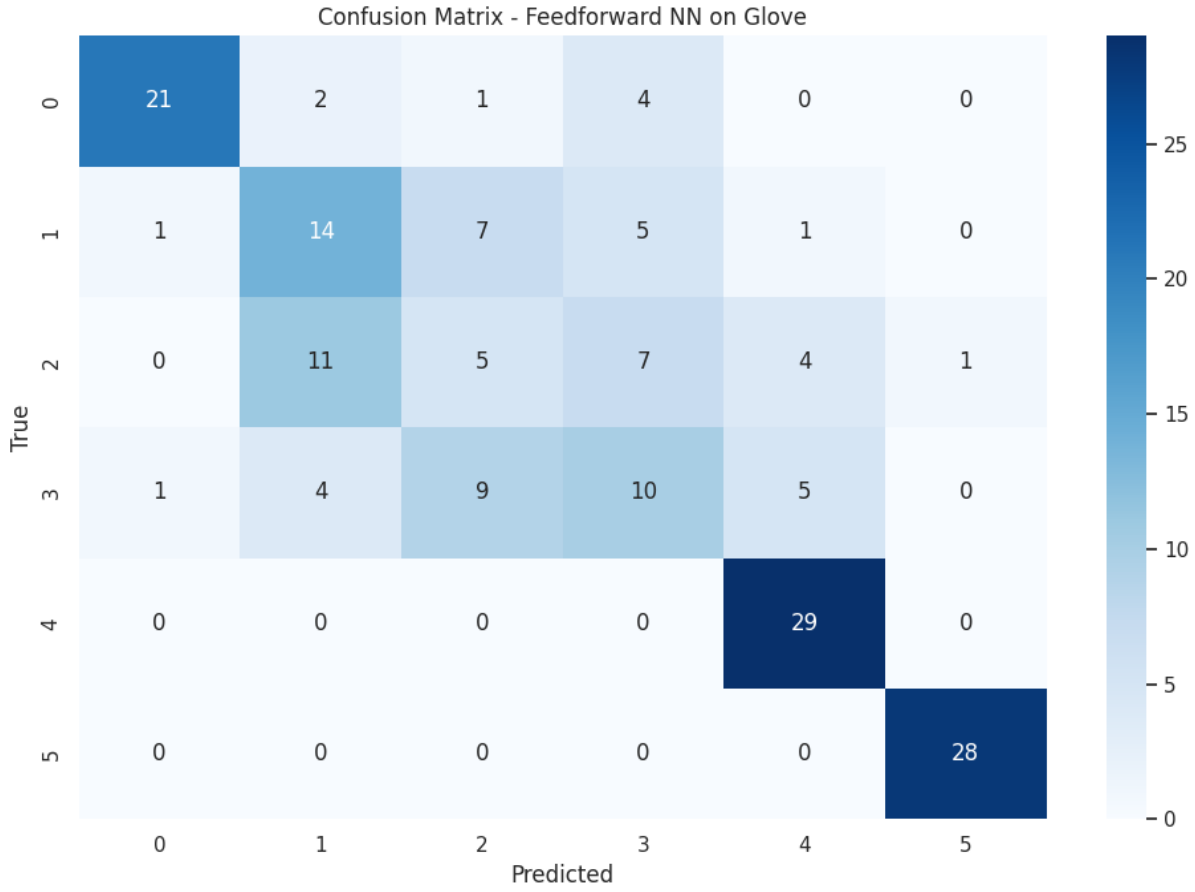
Epoch	Accuracy	Loss	Val Accuracy	Val Loss
01-Oct	0.2715	1.7542	0.4118	1.4641
02-Oct	0.4908	1.4337	0.4706	1.2153
03-Oct	0.5348	1.2308	0.4853	1.1206
04-Oct	0.5884	1.0927	0.5588	1.0728
05-Oct	0.596	1.0198	0.5882	1.0206
06-Oct	0.6208	0.9856	0.6176	0.9792
07-Oct	0.6526	0.8686	0.6176	0.9391
08-Oct	0.6466	0.8586	0.6471	0.9264
09-Oct	0.68	0.7941	0.6765	0.8677
10-Oct	0.7196	0.7678	0.6912	0.8461

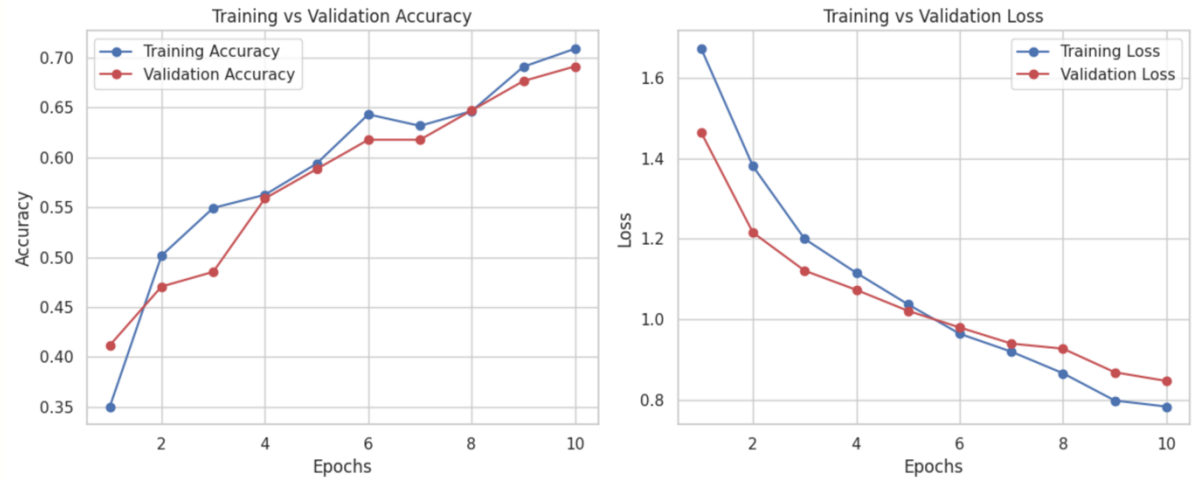
Class	Precision	Recall	F1-Score	Support
0	0.91	0.75	0.82	28
1	0.45	0.5	0.47	28

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

2	0.23	0.18	0.2	28
3	0.38	0.34	0.36	29
4	0.74	1	0.85	29
5	0.97	1	0.98	28

```
Test Accuracy: 0.6294 loss: 0.7934
```





Feedforward Neural Network using allminiv6 embedding :

FINAL REPORT FOR INDUSTRIAL SAFETY RISK ANALYSIS – MILESTONE 1,2, 3

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

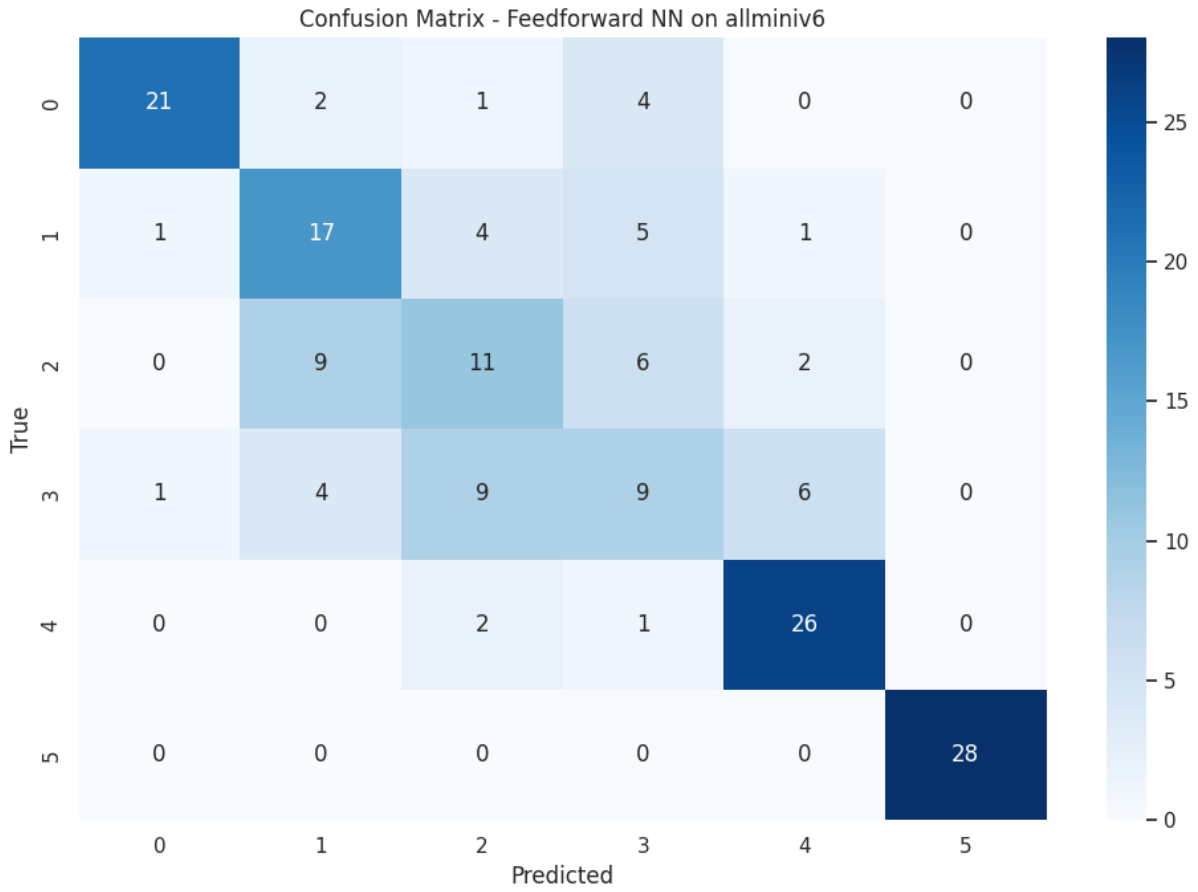
Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 128)	52,096
dropout_4 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 64)	8,256
dropout_5 (Dropout)	(None, 64)	0
dense_8 (Dense)	(None, 6)	390

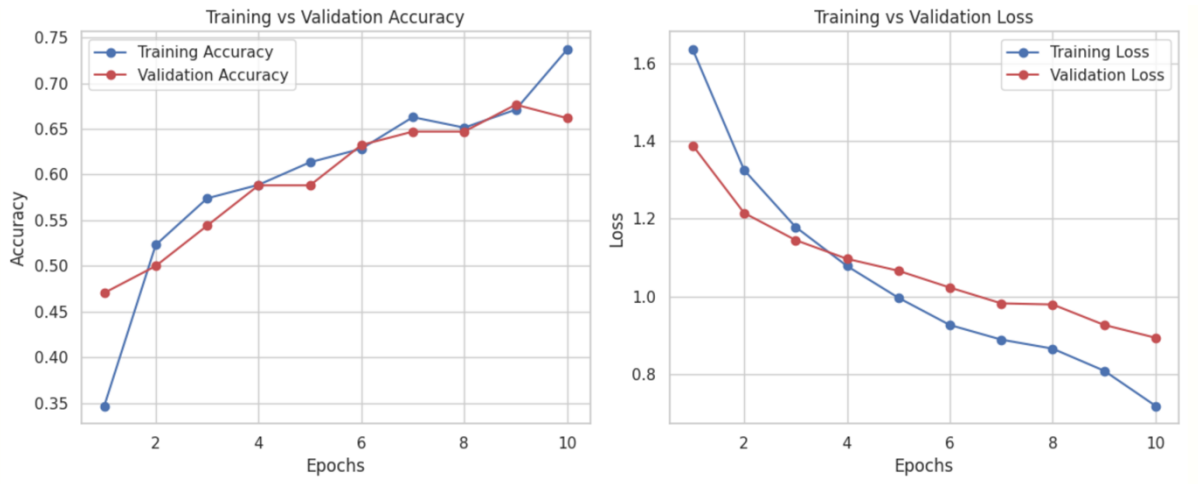
Epoch	Accuracy	Loss	Val Accuracy	Val Loss
01-Oct	0.2918	1.6951	0.4706	1.3886
02-Oct	0.4922	1.3668	0.5	1.2149
03-Oct	0.5384	1.2541	0.5441	1.1451
04-Oct	0.5908	1.0999	0.5882	1.0969
05-Oct	0.604	1.0209	0.5882	1.0662
06-Oct	0.6231	0.9319	0.6324	1.0231
07-Oct	0.6623	0.8899	0.6471	0.9825
08-Oct	0.6458	0.885	0.6471	0.9795
09-Oct	0.6449	0.8451	0.6765	0.927
10-Oct	0.7298	0.7448	0.6618	0.8939

Class	Precision	Recall	F1-Score	Support
0	0.91	0.75	0.82	28
1	0.53	0.61	0.57	28
2	0.41	0.39	0.4	28
3	0.36	0.31	0.33	29
4	0.74	0.9	0.81	29
5	1	1	1	28

Test Accuracy: 0.6588 Loss: 0.7812

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath





Feedforward Neural Network using Model 1 output

Input from Milestone 1: X_m2_final_resampled ,y_m2_target_resampled

Engineered features such as date components, one-hot encoded categorical variables, critical risk clusters, and TF-IDF vectorized descriptions contribute significantly to the accuracy of predicting Potential Accident Level and, subsequently, Accident Level.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Now here while building M2 (Model 2) The predicted Potential Accident Level will be used as an additional feature, stacked with other inputs, to improve the prediction of the Accident Level.

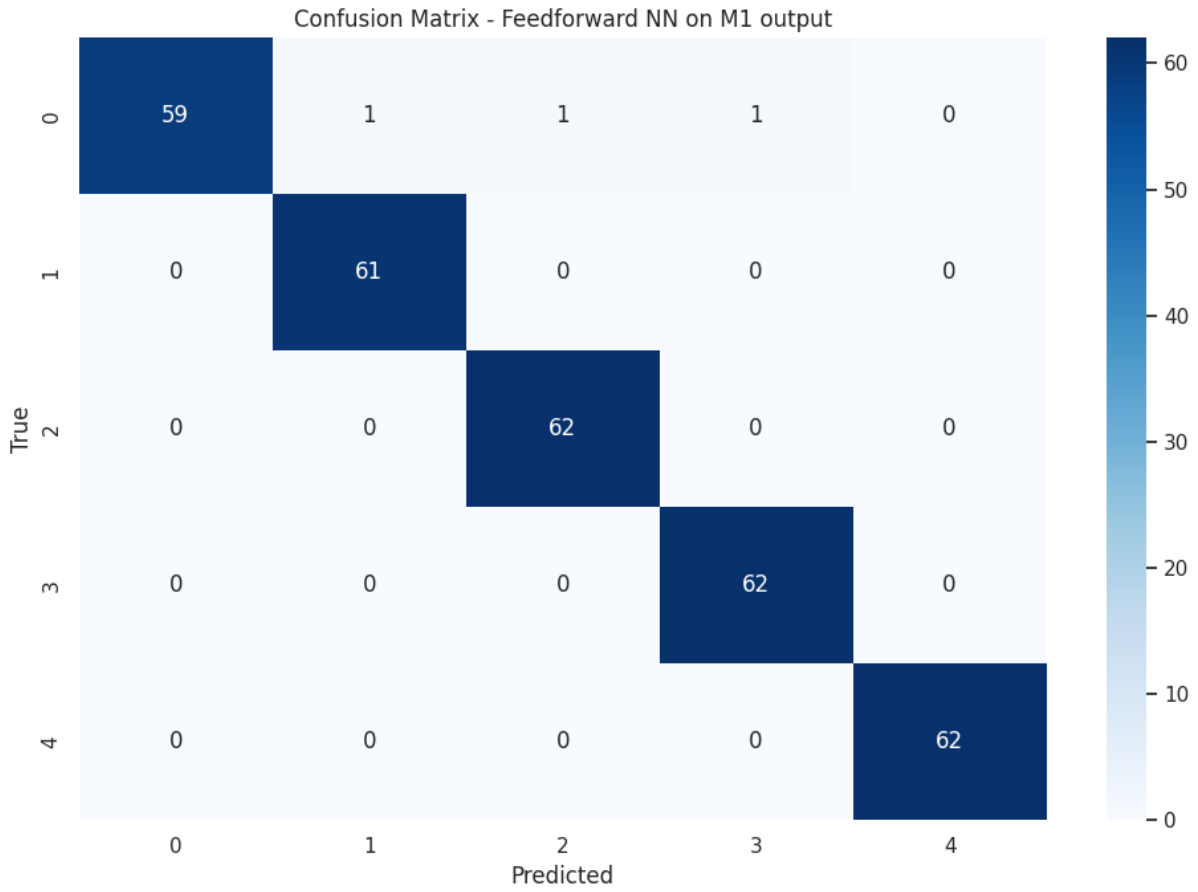
Layer (type)	Output Shape	Param #
dense_9 (Dense)	(None, 128)	131,072
dropout_6 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 64)	8,256
dropout_7 (Dropout)	(None, 64)	0
dense_11 (Dense)	(None, 5)	325

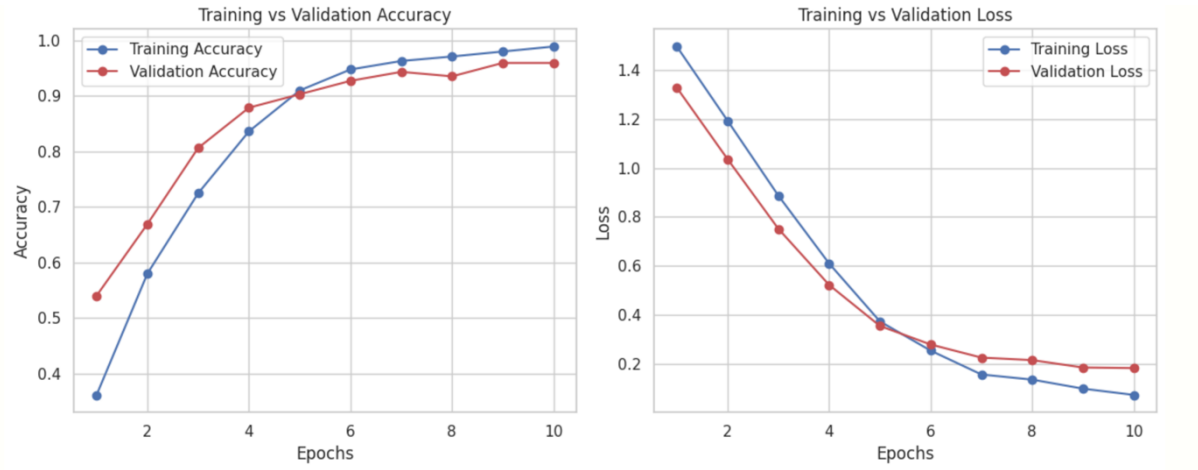
Epoch	Accuracy	Loss	Val Accuracy	Val Loss
1	0.2982	1.5592	0.5403	1.3287
2	0.5454	1.2702	0.6694	1.0364
3	0.6854	0.9661	0.8065	0.7517
4	0.8273	0.6533	0.879	0.5222
5	0.9041	0.3946	0.9032	0.3545
6	0.9506	0.2591	0.9274	0.2786
7	0.9571	0.1584	0.9435	0.225
8	0.9633	0.1609	0.9355	0.2148
9	0.9806	0.0934	0.9597	0.1846
10	0.9909	0.0719	0.9597	0.1823

Class	Precision	Recall	F1-Score	Support
0	1	0.95	0.98	62
1	0.98	1	0.99	61
2	0.98	1	0.99	62
3	0.98	1	0.99	62
4	1	1	1	62

FINAL REPORT FOR INDUSTRIAL SAFETY RISK ANALYSIS – MILESTONE 1,2, 3

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath





By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

RNN and LSTM Classifiers :

LSTM :

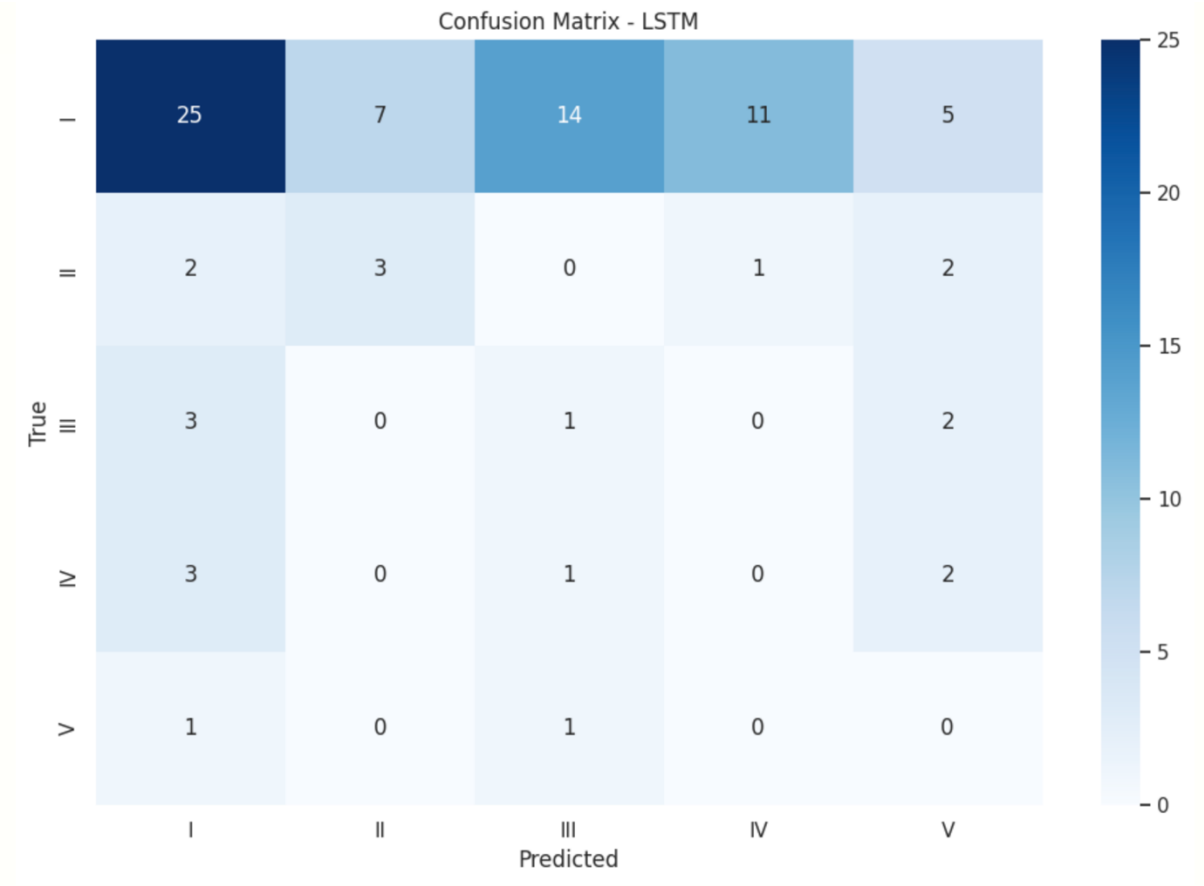
Layer (type)	Output Shape	Param #	Connected to
text_input (InputLayer)	(None, 100)	0	–
embedding (Embedding)	(None, 100, 100)	315,900	text_input[0][0]
structured_input (InputLayer)	(None, 22)	0	–
lstm_1 (LSTM)	(None, 128)	117,248	embedding[0][0]
dense_3 (Dense)	(None, 64)	1,472	structured_input…
concatenate_1 (Concatenate)	(None, 192)	0	lstm_1[0][0], dense_3[0][0]
dense_4 (Dense)	(None, 64)	12,352	concatenate_1[0]…
dropout_1 (Dropout)	(None, 64)	0	dense_4[0][0]
dense_5 (Dense)	(None, 5)	325	dropout_1[0][0]

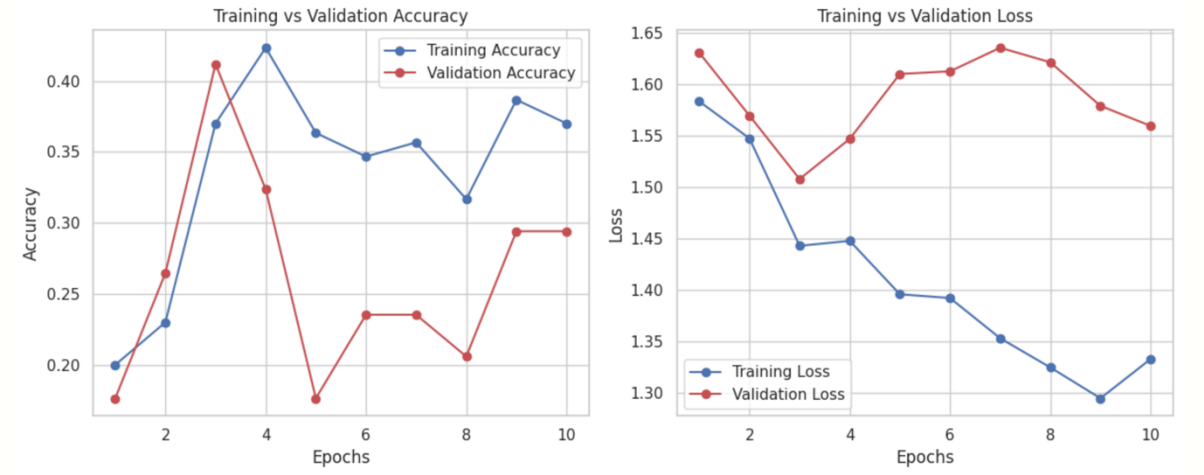
Epoch	Accuracy	Loss	Val Accuracy	Val Loss
1	0.1617	1.6056	0.1765	1.6303
2	0.2254	1.5167	0.2647	1.5693
3	0.3789	1.5299	0.4118	1.5077
4	0.4644	1.4487	0.3235	1.5467
5	0.3891	1.6027	0.1765	1.6097
6	0.3384	1.357	0.2353	1.6123
7	0.3575	1.3339	0.2353	1.6352
8	0.3251	1.2758	0.2059	1.621
9	0.4146	1.2343	0.2941	1.5787
10	0.3415	1.3908	0.2941	1.5593

Hybrid Model Test Accuracy: 0.3452. Loss: 1.4920

Class	Precision	Recall	F1-Score	Support
I	0.74	0.4	0.52	62
II	0.3	0.38	0.33	8
III	0.06	0.17	0.09	6
IV	0	0	0	6
V	0	0	0	2

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath





Bidirectional Long Short-Term Memory:

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Layer (type)	Output Shape	Param #	Connected to
text_input (InputLayer)	(None, 100)	0	–
embedding_1 (Embedding)	(None, 100, 100)	315,900	text_input[0][0]
structured_input (InputLayer)	(None, 22)	0	–
bidirectional_1 (Bidirectional)	(None, 256)	234,496	embedding_1[0][0]
dense_6 (Dense)	(None, 64)	1,472	structured_input…
concatenate_2 (Concatenate)	(None, 320)	0	bidirectional_1[… dense_6[0][0]
dense_7 (Dense)	(None, 64)	20,544	concatenate_2[0]…
dropout_2 (Dropout)	(None, 64)	0	dense_7[0][0]
dense_8 (Dense)	(None, 5)	325	dropout_2[0][0]

Epoch	Accuracy	Loss	Val Accuracy	Val Loss
1	0.2011	1.4248	0.2353	1.5317
2	0.2228	1.7646	0.1765	1.5697
3	0.2073	1.5558	0.1471	1.5589
4	0.2238	1.3816	0.1765	1.5502
5	0.2662	1.2984	0.1765	1.5224
6	0.4107	1.2097	0.0882	1.714
7	0.3275	1.0714	0.2941	1.2978
8	0.5848	0.8885	0.5294	1.1805
9	0.804	0.7123	0.2647	1.4578
10	0.5606	0.6999	0.2941	1.2546

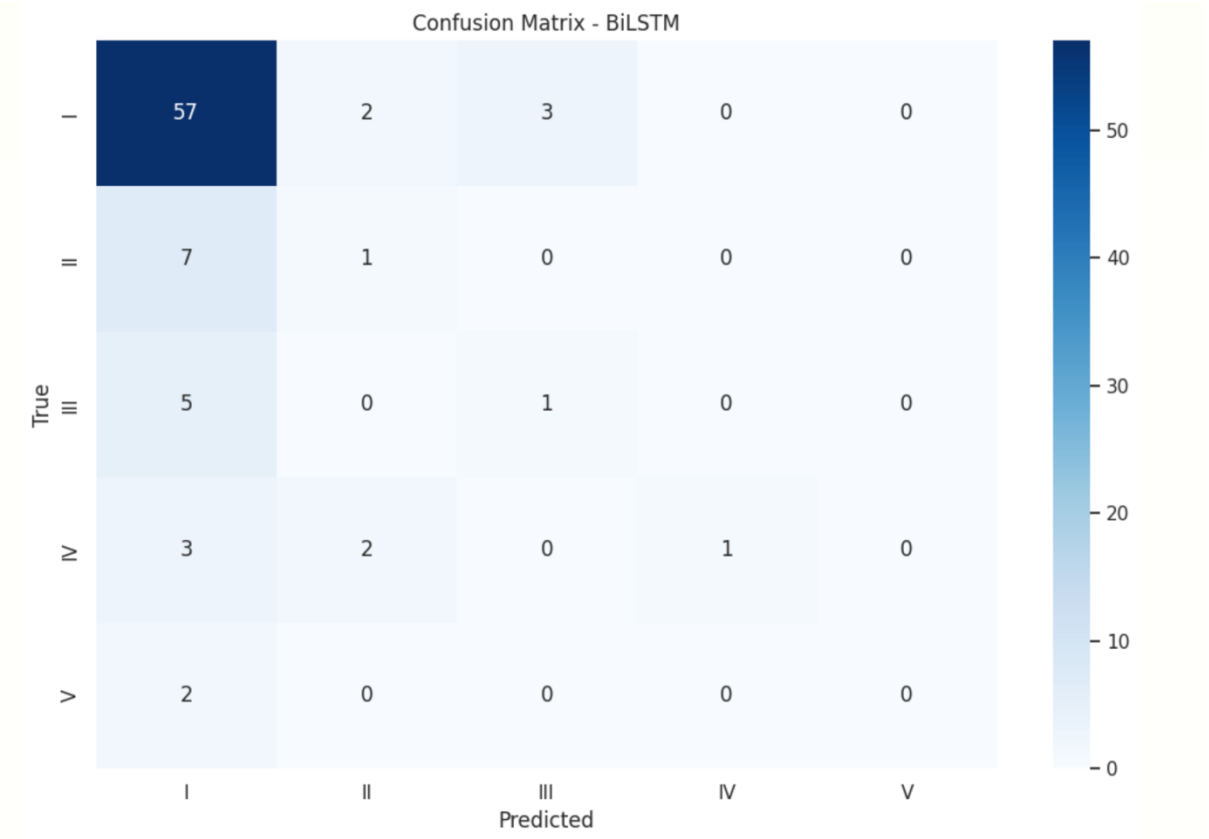
BiLSTM Hybrid Model Test Accuracy: 0.2976. Loss: 1.3403

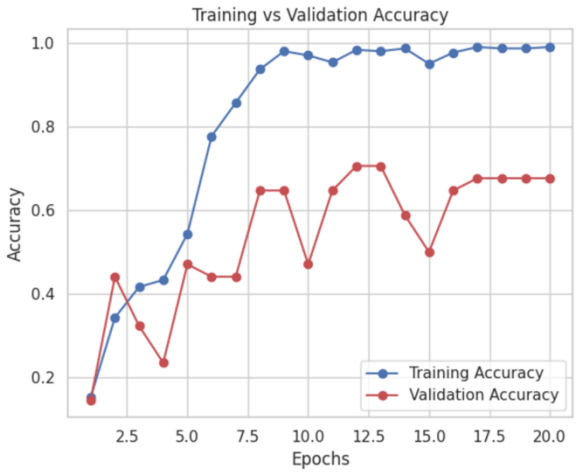
Class	Precision	Recall	F1-Score	Support
I	0.69	0.32	0.44	62
II	0.09	0.5	0.15	8
III	0.14	0.17	0.15	6
IV	0	0	0	6
V	0	0	0	2

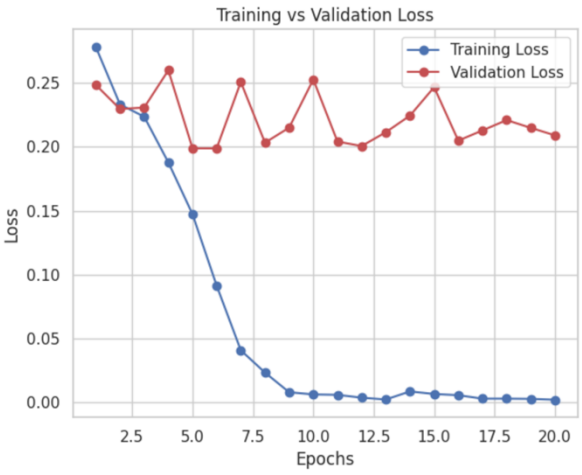
For model performance improvement we use biLSTM hybrid model with structured inputs and uses Focal Loss as the loss function for training

FINAL REPORT FOR INDUSTRIAL SAFETY RISK ANALYSIS – MILESTONE 1,2, 3

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath







By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

BERT + BiLSTM + Attention Model using In-Memory Data

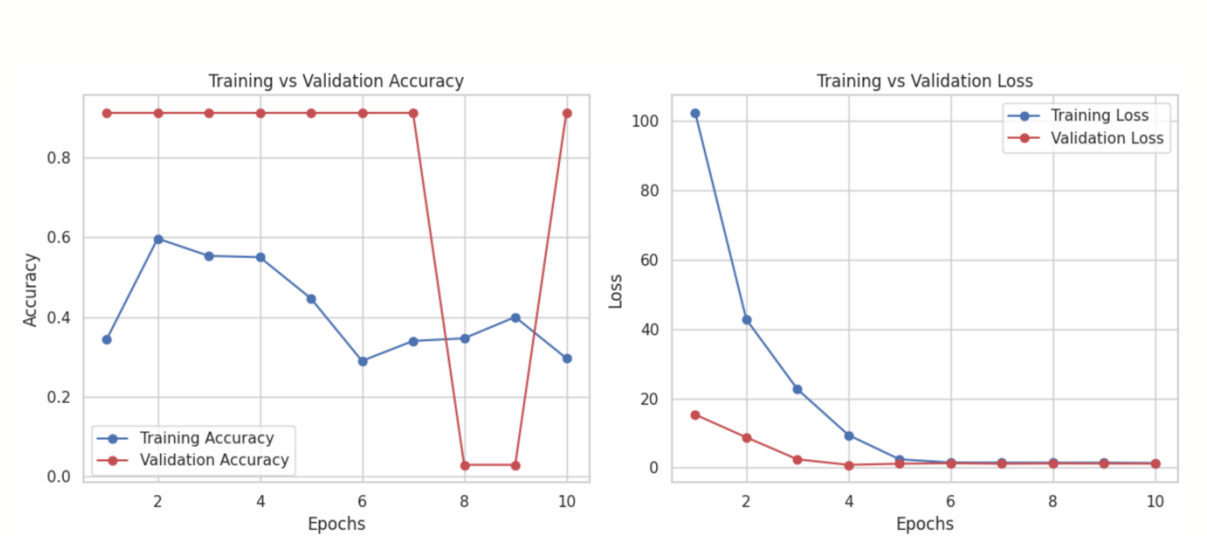
Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	(None, 128)	0	–
attention_mask (InputLayer)	(None, 128)	0	–
bert_encoder (BERTEncoder)	(None, 128, 768)	0	input_ids[0][0], attention_mask[0…
bidirectional (Bidirectional)	(None, 128, 256)	918,528	bert_encoder[0][…
structured_input (InputLayer)	(None, 22)	0	–
attention_layer (AttentionLayer)	(None, 256)	0	bidirectional[0]…
dense (Dense)	(None, 64)	1,472	structured_input…
concatenate (Concatenate)	(None, 320)	0	attention_layer[… dense[0][0]
dense_1 (Dense)	(None, 64)	20,544	concatenate[0][0]
dropout (Dropout)	(None, 64)	0	dense_1[0][0]
dense_2 (Dense)	(None, 5)	325	dropout[0][0]

Epoch	Train Accuracy	Train Loss	Val Accuracy	Val Loss	Time per Epoch
1	0.1900	169.4077	0.9118	15.2969	221s
2	0.6323	44.6649	0.9118	8.7319	197s
3	0.5370	28.5639	0.9118	2.4030	201s
4	0.5231	11.4093	0.9118	0.8116	205s
5	0.4592	2.9784	0.9118	1.1443	201s
6	0.2993	1.4891	0.9118	1.2656	200s
7	0.3570	1.4454	0.9118	1.1406	199s
8	0.4298	1.4361	0.0294	1.2155	195s
9	0.3712	1.4373	0.0294	1.1920	254s
10	0.2796	1.3508	0.9118	1.1625	248s

Class	Precision	Recall	F1-Score	Support
I	0.73	0.97	0.83	62
II	0.00	0.00	0.00	8

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Class	Precision	Recall	F1-Score	Support
III	0.00	0.00	0.00	6
IV	0.00	0.00	0.00	6
V	0.00	0.00	0.00	2



Approach 2 : Introducing more feature engineering techniques to see better test results even for minority classes.

Derive attention keywords and apply weights ,and train the model.

Step 1: Import Libraries

Step 2: Load Preprocessed Dataset

- Loads a preprocessed CSV with columns like Date, Description, Accident_Level, etc.
- Ensures column names are clean.
- Removes rows where accident Description is missing.

Step 3: Load Transformer Model and Tokenizer

- Uses BERT (or optionally MiniLM) to extract self-attention weights.
- Sets output_attentions=True to access layer-wise attention data.

Step 4: Move Model to Device

- Sends model to GPU if available (for faster processing).
- Sets model to evaluation mode (model.eval()).

Step 5: Define Function to Extract Attention-Based Keywords

- Tokenize the sentence (with truncation).
- Feed it into the model to extract attention weights.
- Average across all layers and heads to compute a general importance score for each token.
- Focus on attention from [CLS] token, which is a standard way to summarize attention-based importance.
- Merge subwords like "accident" + "##al" → "accidental".
- Rank tokens by their attention scores.
- Return the top N keywords per description.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Step 6: Apply to All Descriptions

- Iterates over all descriptions.
- For each row, extracts the top 20 keywords based on attention weights.
- Stores results in a new column: important_keywords_attention.

Step 7: Save Enhanced Dataset

- Saves the updated DataFrame, now with an extra column containing extracted keywords.

Step8 : Build combined columns

- Combine the attention keyword column with Accident level,Potential Accident level and critical risk.

Utilizing Deep MLP Models for Ensembling:

From the Description column, generate:

- Length-based features (desc_len, avg_word_len)
- Punctuation features (num_exclam, etc.)
- Keyword flags (binary presence of words like "injury", "fire")
- Count of capital letters
- Temporal features from date (if available)

Data Augmentation (EDA)

For underrepresented classes:

- Apply textual augmentation (synonym replacement, etc.) via your eda() function.
- Controlled using thresholds like AUG_MIN_SAMPLES

Numeric Features

- Selected numeric columns (excluding text) are extracted and split into:
- X_train_num, X_test_num
- Augment numeric data for newly added EDA samples.

SBERT Embeddings

- SentenceTransformer('all-MiniLM-L6-v2')
- Extract sentence embeddings for 10+ relevant text columns.
- Encoded separately for train and test.
- Augmented samples also encoded.

Feature Concatenation

- X_train_full = [numerical_features + SBERT_embeddings]
- Resulting feature sets are high-dimensional representations for MLP input.

SMOTE Oversampling

Handle class imbalance:SMOTE synthesizes samples using nearest neighbors (class-sensitive k_neighbors).

MLP Model Architecture

- DeepMLP(): fully connected layers + batchnorm + dropout
- Customizable depth and dropout
- Final layer outputs logits for each class

Ensemble Training

For each model in N_ENSEMBLE:

- Train on SMOTE-augmented data
- Save best model by validation macro F1

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

- Early stopping used to prevent overfitting
- Predictions on test set saved (softmax probabilities)

Ensemble Prediction

- ensemble_preds = average(softmax outputs from all models)
- Final class = argmax(averaged softmax scores)
- Gives more stable predictions than a single model

DEEP MLP Model : Metrics

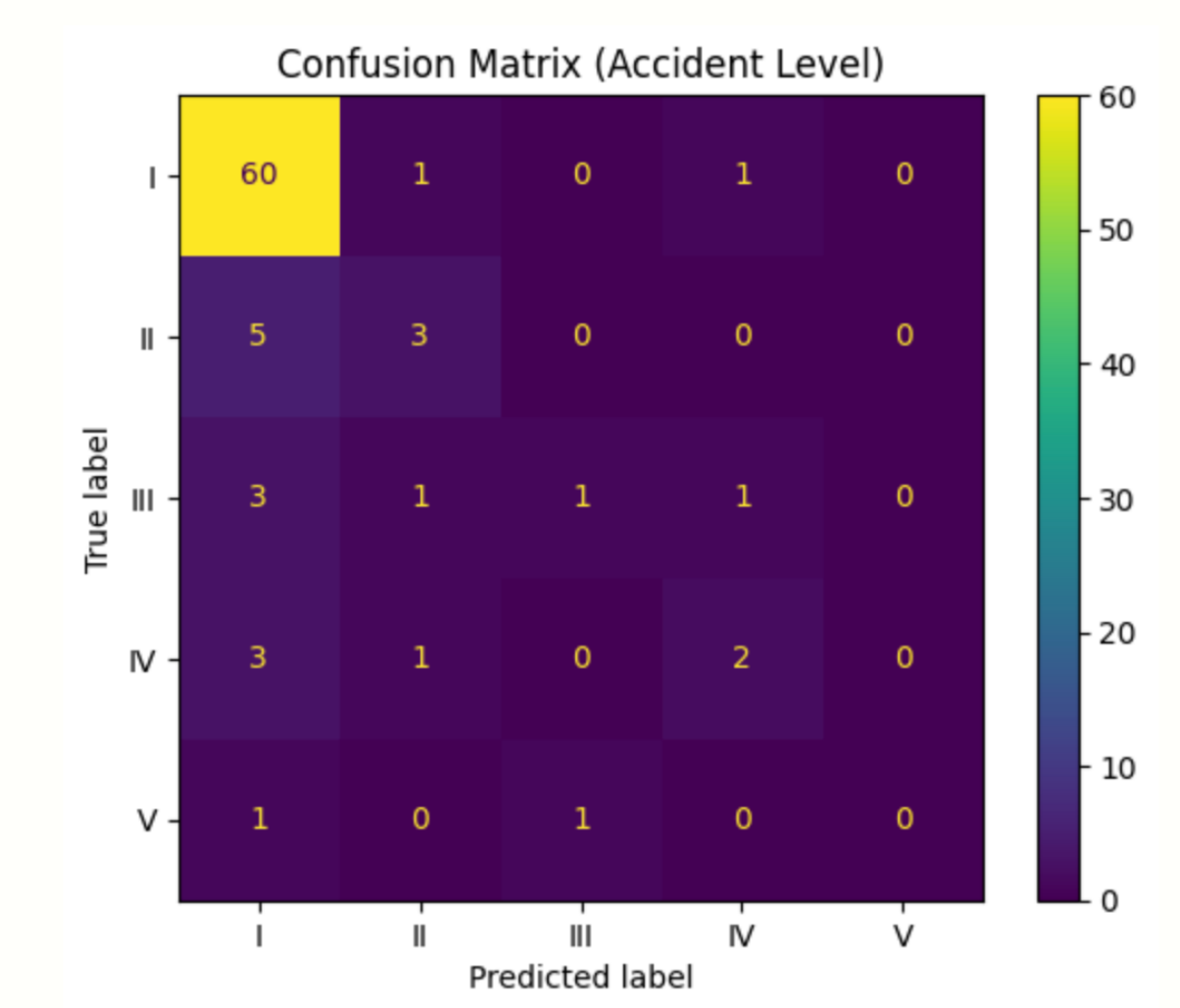
With 10 ensemble model and 150 Epoch and early stpoppings.

Metric	Value
Test Accuracy	0.786
Macro F1 Score	0.395

Accident Level	Precision	Recall	F1-Score	Support
I	0.83	0.97	0.9	62
II	0.5	0.38	0.43	8
III	0.5	0.17	0.25	6
IV	0.5	0.33	0.4	6
V	0	0	0	2

Average Type	Precision	Recall	F1-Score	Support
Macro Average	0.47	0.37	0.395	84
Weighted Average	0.73	0.79	0.75	84

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath



Class I (most common):

- Precision: **0.83 → 83% of the time the model predicted "I", it was correct.**
- Recall: **0.97 → 97% of all actual "I" records were correctly predicted.**
- F1 Score: **0.90 → High accuracy and consistency; this class is handled very well.**

Classes II, III, IV (underrepresented):

- **Precision:** ~0.50 → When the model predicts one of these, it's right half the time.
- **Recall:** Drops sharply:
- Class II: **0.38** → Captures only 38% of actual "II".
- Class III: **0.17** → Captures just 17% of actual "III".
- Class IV: **0.33** → Captures 33% of actual "IV".
- **F1 Scores:** Low (0.25–0.43) → Shows poor performance on rare but important events.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Visual Explanation : For 100 test cases

Class	Actual Count	Correct Predictions	Missed
I	80	77	3
II	8	3	5
III	6	1	5
IV	4	1	3
V	2	0	2

To improvise further, Added sentimental analysis with manual Augmentation and with 12 ensemble model. Detailed steps below.

Description Generation Tools

Several predefined template lists: Manual Augmentaiton

- prefixes – how the accident started (e.g., "During...")
- actions_common – common mining activities
- causes – accident causes, grouped by severity
- injuries – consequences of the accidents
-

Data Synthesis & Upsampling

For each level:

- **Get existing records** for that level.
- **Calculate how many new records** are needed (to reach 200).
- For each synthetic record:
  - Assign random date within original range.
  - Sample categorical fields (country, local, etc.).
  - Assign Accident Level and Potential Accident Level.
  - Generate synthetic Description.
- Combine original and synthetic records.
- **Downsample to 100** if overpopulated (only for display or training balance).
- Save each level's data as a CSV.

Sentiment analysis :

STEP 0: Setup & Import

- Install required packages (vaderSentiment, spacy model, etc.)
- Import Python libraries for:
  - Data manipulation (pandas, numpy)
  - Deep learning (torch)
  - Text embeddings (SentenceTransformer)
  - Model evaluation (sklearn.metrics)
  - Oversampling, sampling, visualization, etc.

STEP 1: Load and Clean the Data

- Load a CSV file containing accident data.
- Normalize column names (replace spaces with underscores).
- Drop rows where key columns (Accident_Level, Potential_Accident_Level, Critical_Risk, Description) are null.
- Drop rare classes with <5 samples to avoid skew.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

STEP 2: Encode Labels

- Ordinal Encoding for:
- Accident_Level → Accident_Level_Ordinal
- Potential_Accident_Level → Potential_Accident_Level_Ordinal
- Critical_Risk → Critical_Risk_Ordinal

STEP 3: Date Feature Engineering

- Parse date from either date or Data column.
- Extract:
  - year
  - month
  - dayofweek

STEP 4: NLP Feature Engineering

A) Sentiment Analysis using VADER:

- Compute neg, neu, pos, and compound sentiment scores for the Description.

B) Named Entity Recognition (NER) using spaCy:

- Count entities in Description for types like PERSON, ORG, GPE, etc.

C) Keyword Flags:

- Binary flags if keywords like 'fire', 'injury', 'electrical', etc. appear in the Description.

D) Text Clustering:

- Use SBERT (all-MiniLM-L6-v2) to vectorize descriptions.
- Apply KMeans clustering on those vectors (20 clusters).
- Add desc_cluster as a categorical feature.

E) Miscellaneous Text Stats:

- Description length, sentence count, average word length, punctuation counts, capital letters count.

STEP 5: Train-Test Split

- Split data:
  - Text column (Description)
  - Label (Accident_Level_Ordinal)
- Stratified to maintain class balance.

STEP 6: Prepare Feature Sets

A) Tabular/Numeric Features

- Combine:
  - Encoded accident/potential levels
  - Date features
  - NLP stats, keyword flags
  - Sentiment features
  - Entity counts
  - Cluster ID

B) Text Embeddings

- For columns like Industry_Sector, Countries, Local, Employee_or_Third_Party, Description
- Use SBERT to generate embeddings for each.
- Concatenate embeddings.

C) Final Input

- Concatenate numeric + text embeddings → X_train_full, X_test_full

STEP 7: Define Dataset & Sampler

- Wrap training/testing data using PyTorch Dataset and DataLoader.
- Use WeightedRandomSampler to handle class imbalance (no SMOTE used).

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

STEP 8: Define Deep MLP Classifier

- A multi-layer perceptron (MLP) with:
  - BatchNorm → Linear → ReLU → Dropout
- Final layer gives logits for n_classes.

Focal Loss:

- Used to focus more on hard-to-classify examples.
- Weighted by inverse class frequency.

STEP 9: Train Ensemble of MLP Models

- Train **12 deep MLP models**, each with different random seeds.
- Track best model on validation set (based on F1 macro).
- Apply **early stopping** if performance doesn't improve for 40 epochs.
- Save best model weights.

STEP 10: Predict with Ensemble

- Load each best model and get softmax predictions.
- Average probabilities across all models.
- Final prediction = argmax(average probabilities)

STEP 11: Evaluate Final Predictions

- Calculate:
  - Accuracy
  - Macro F1 Score
  - Classification Report
  - Confusion Matrix

STEP 12: Recommend Medical Aid

- Create a dictionary mapping each accident level (I to VI) to a recommended medical action.
- Convert predicted label index back to Roman numeral.
- Map each predicted level to corresponding medical recommendation.

STEP 13: Final Output

- Create a results DataFrame:
  - Description
  - Predicted_Level (Roman)
  - Recommended_Medical_Aid
- Useful for explaining predictions and providing actionable steps.

What This Pipeline Does

- Prepares and augments accident reports with advanced NLP and statistical features.
- Trains an ensemble of deep neural networks to predict severity of accident (levels I–VI)
- Attaches actionable medical recommendations to each prediction
- Visualizes model performance with standard classification metrics

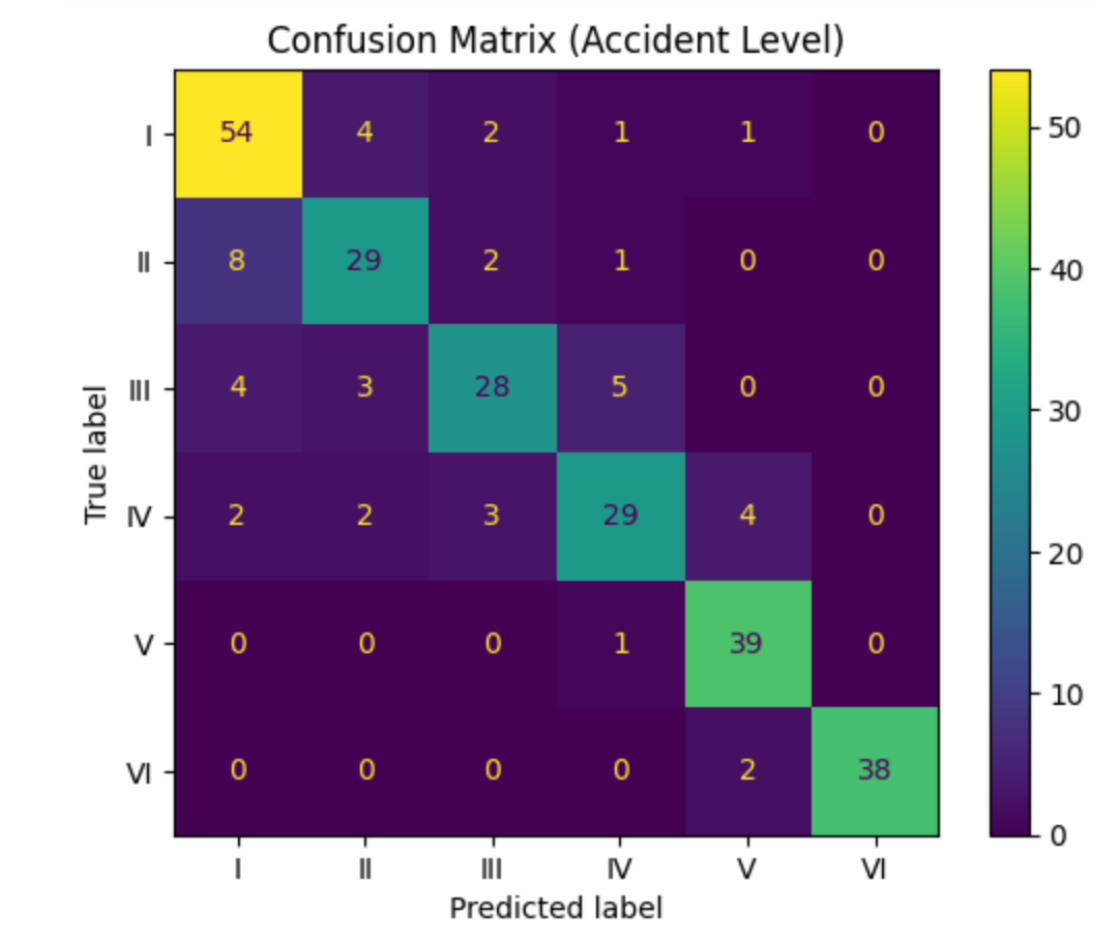
Metric	Value	Explanation
Test Accuracy	0.828	83% of all test predictions are correct.
F1 Macro Avg	0.826	Balanced performance across all classes.
F1 Weighted Avg	0.83	Reflects actual class distribution.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Class	Precision	Recall	F1-Score	Support	Comments
I	0.79	0.87	0.83	62	Solid performance, high recall.
II	0.76	0.72	0.74	40	Balanced prediction accuracy.
III	0.8	0.7	0.75	40	Slightly lower recall, but decent F1.
IV	0.78	0.72	0.75	40	Consistent across all metrics.
V	0.85	0.97	0.91	40	**Excellent recovery of rare class.**
VI	1	0.95	0.97	40	**Outstanding performance.**

All classes are handled **consistently well**.
Class V and **VI**, which are typically harder to predict, now show **very strong F1 scores** (0.91 & 0.97).
Compared to earlier models, this ensemble:

- Boosts **recall and precision** across rare classes.
- Avoids **zero F1 scores**, showing **robust generalization**



1. Training & Validation Performance Across the 12 Ensemble Models

- Each ensemble model trains over a maximum of 150 epochs, although early stopping often halts training before all epochs finish.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

- Training logs show that **loss generally decreases** and **macro F1 scores improve** with each epoch for most models, illustrating successful learning.
- Early epochs show low macro F1 values (\~0.04), reflecting the model starting from random weights. Over time, macro F1 scores climb significantly, often surpassing 0.8.
- The models that triggered early stopping stopped training when they no longer improved on the validation data, which helps avoid overfitting.

2. Overall Ensemble Performance

- When predictions from all 12 models were averaged (softmax outputs were averaged and the class with the highest average probability was chosen), the **final test accuracy** reached **84.35%** and the **macro F1 score** was **0.8423**.
- These metrics indicate strong performance across the different accident level classes, especially considering the class imbalance.

3. Detailed Classification Report

The report breaks down performance by each accident class:

Class	Precision	Recall	F1-Score
I	0.81	0.89	0.85
II	0.72	0.65	0.68
III	0.79	0.78	0.78
IV	0.80	0.82	0.81
V	0.97	0.97	0.97
VI	0.97	0.93	0.95

- Classes V and VI show the highest scores, suggesting the model is highly accurate in identifying the most severe accident levels.
- Class II has slightly lower precision and recall, indicating more difficulty in distinguishing these cases from others.

4. Macro vs. Weighted Average

- The **macro average** F1 of 0.84 treats each class equally, highlighting balanced performance.
- The **weighted average** F1 also sits at 0.84, showing that performance is not skewed heavily by class sizes.

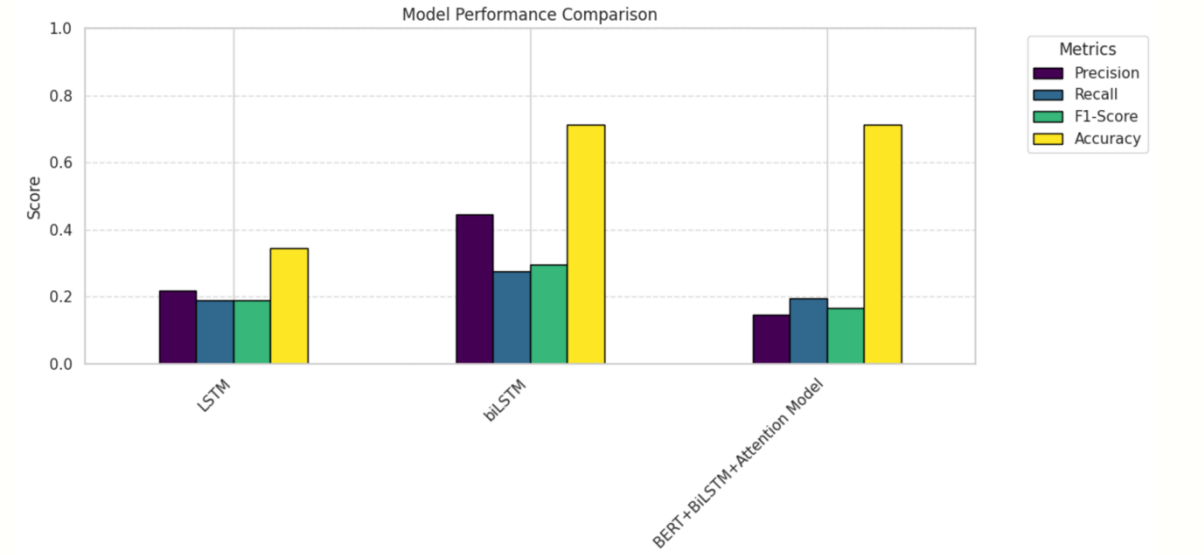
In summary, the ensemble modelling strategy effectively learns the complex patterns in the data. The iterative training logs reveal a consistent improvement in performance, while the final metrics show a strong and balanced classification across multiple accident levels.

Comparison with all Neural Network Model:

MODEL	PRECISION	RECALL	F1SCORE	ACCURACY
FFN over M1 model output	0.99	0.99	0.99	0.99
LSTM	0.219	0.189	0.188	0.345

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Bi LSTM	0.444	0.276	0.296	0.714
BERT_BILSTM_Attention Model	0.146	0.194	0.167	0.714
FFN-TFIDF	0.735	0.741	0.735	0.741
FFN-Glove	0.614	0.629	0.616	0.629
FFN-allminiv6	0.659	0.659	0.656	0.659
Custom Neural Network -AllminiLM - sentimental Analysis	**0.84**	**0.84**	**0.84**	**0.83**



Best Model: Custom Neural Network with AllminiLM and Sentimental Analysis

As it has

1. Highest Precision: Indicates fewer false positives — better reliability.
2. Highest Recall: Captures more relevant instances.
3. Highest F1-Score: Best balance of precision and recall.
4. Highest Accuracy: Most overall correct predictions.

Conclusion:

We choose this , as it consistently outperforms others across all metrics, making it the most robust and balanced model for your task, better compared to eda, vadersentiment .

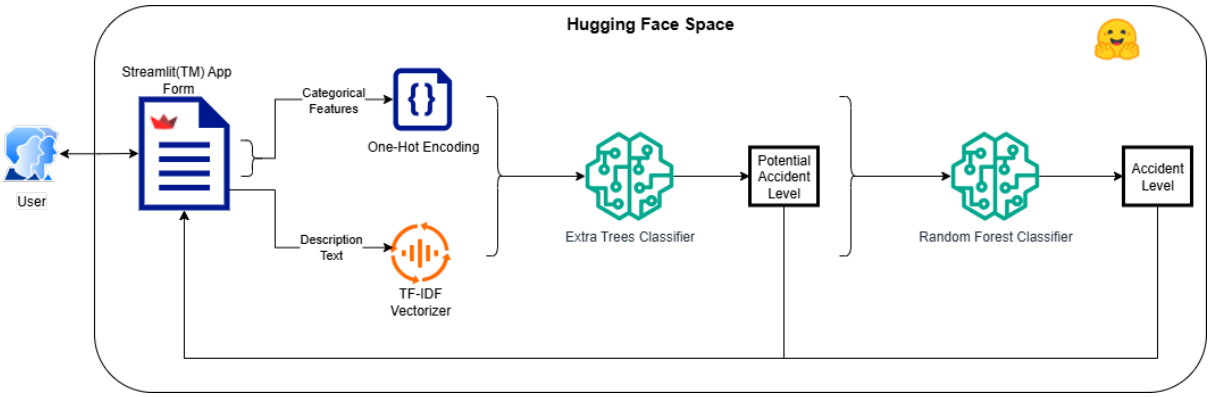
Milestone 3 – Industry Safety Accident Level Prediction Chatbot

Overview:

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

- The user interacts with a UI form on their browser to provide accident data such as – Date, Country, Locale, Industry Sector, Gender, Employee Type, Critical Risk, and the Incident Description text.
- On clicking the "Predict Accident Level" action button, the system predicts Potential Accident Level and Accident Level and presents the output to the user on the UI form in their browser.

How It Works:



- **User Interface:** The user interacts with a Streamlit™ application form that captures accident data. The user clicks on "Predict Accident Level" action button, after entering required information.
- **Feature Engineering:**
  - The Categorical features are one-hot encoded.
  - The Date is featurized further into year, month, day.
  - The Description Text is treated for stop words, case consistency, lemmatization and finally vectorized with the help of a TfidfVectorizer from scikit-learn™, trained by us on the test data corpus. It generates 1000 features per description.
- **Potential Accident Level:** All treated features are stacked together and provided as input to an Extra Trees Classifier model from scikit-learn™, tuned and trained by us, to help predict the "Potential Accident Level".
- **Accident Level:** The Potential Accident Level predicted by above model is stacked along with previous features and provided as input to a second model, a Random Forest Classifier from scikit-learn™, also tuned and trained by us, to help predict the "Accident Level".
- **Predictions Output:** Finally, both outputs i.e., the predicted Potential Accident Level, and the predicted Accident Level are presented to the user on the Streamlit™ application form.
- **Runtime Environment:** The application has been deployed to a Hugging Face™ Space.

Note: We preferred the Machine Learning models from Milestone 1, because their performance was significantly better than the Neural Network models from Milestone 2.

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Try It Out:

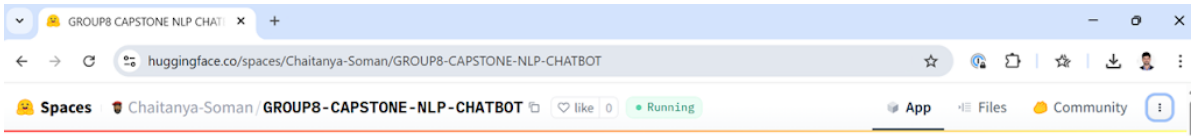
- The application is available here - https://huggingface.co/spaces/Chaitanya-Soman/GROUP8-CAPSTONE-NLP-CHATBOT
- Give it the following inputs taken from a random row from our dataset.

Countries	Country_01
Date	2016/02/10
Locale	Local_03
Industry Sector	Mining
Gender	Male
Employee Type	Third Party
Critical Risk	Others
Incident Description	While aligning the right bracket of tower N ° 32, when releasing the tension applied by the tirford of 1.5 Tn, when pushing the lever towards the tension release point, it returns by mechanical effect overcoming the resistance of the lineman operator and reshaping the the hands of the assistant beating the assistant in the frontal region.

- Then click the “Predict Accident Level” button and see what prediction you get!

FINAL REPORT FOR INDUSTRIAL SAFETY RISK ANALYSIS – MILESTONE 1,2, 3

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath



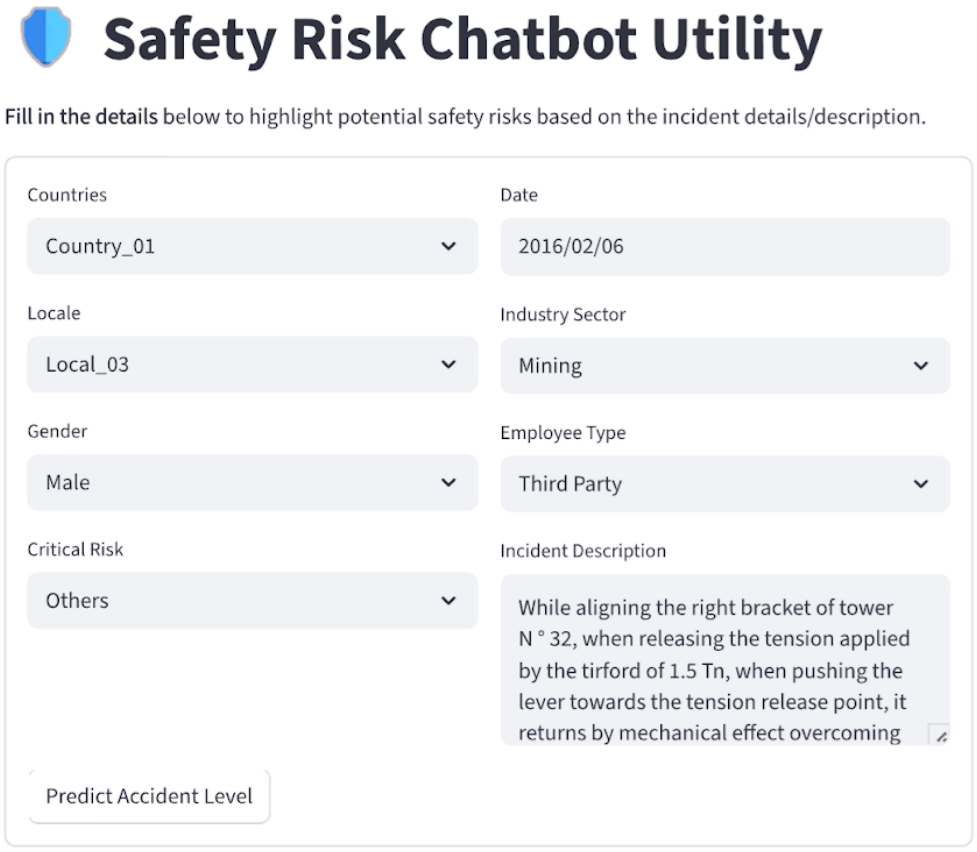
TEXAS McCombs The University of Texas at Austin McCombs School of Business

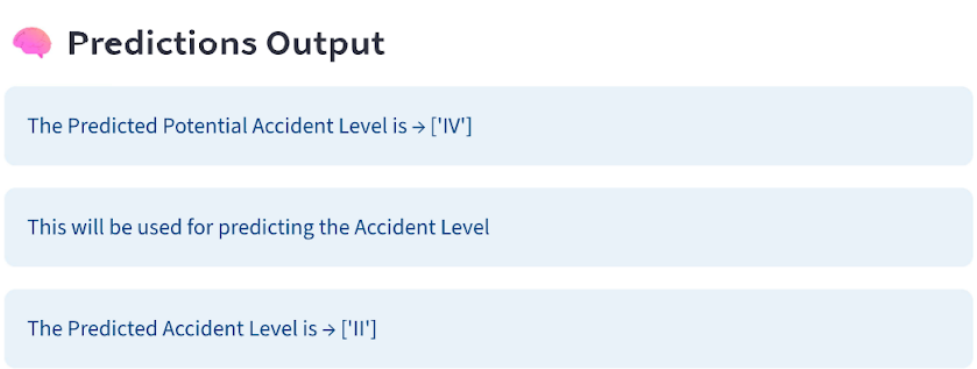
Great Learning POWER AHEAD

Artificial Intelligence and Machine Learning

NLP Project 1 - Industrial Accident Report Classification Chatbot

Capstone Project Submission

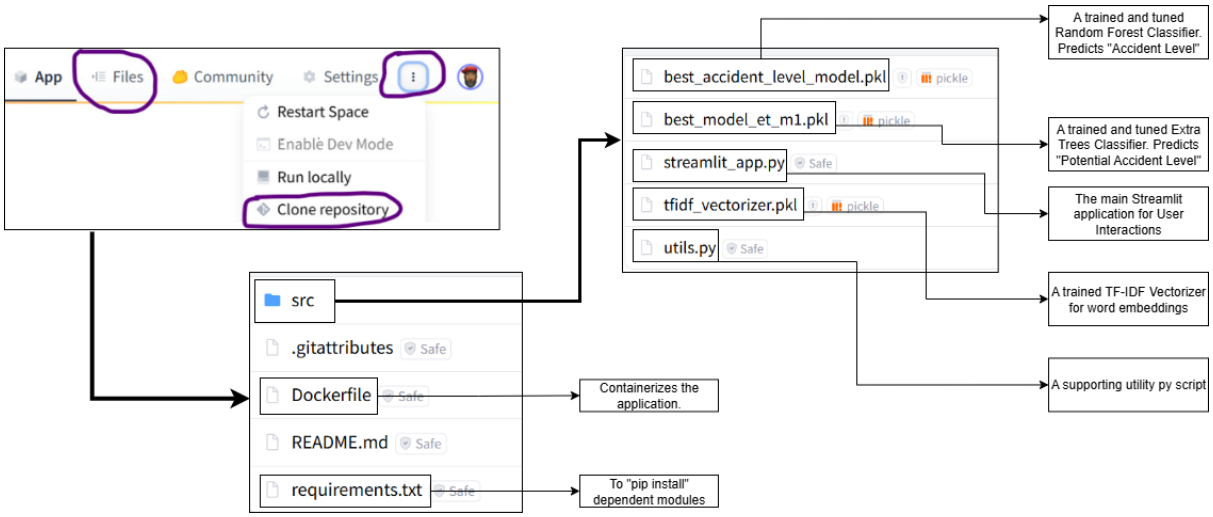




By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Source Code:

The code files are accessible from our Hugging Face Space. Click on the “Files” menu to view the repository and browse files online. Alternatively, you can clone the repository by clicking “Clone repository” from the three-dots menu.



Appendix:

Additional Validation Done: (Based on Mentors Review Comments) - Milestone 1

1) Worked on a fresh copy of cleaned DataFrame to avoid changing main data.
2) Used `train_test_split` with `stratify` to ensure both splits have the same class proportions. Test size is 20%, so you get about 80% for training.
3) Decide the total upsampled size for training (e.g., 500 rows).
4) Saved the upsampled train set to a CSV file. Saved the untouched (original distribution) test set to a different CSV file.
5) Only the train set is upsampled: Prevents data leakage and test set stay realistic. Class balance in training. The test set is never touched or resampled, so your model is evaluated on truly unseen data.
6) Extract date features for BOTH splits
7) Defined categorical columns, One-hot encode train/test (fit columns on train/test set)
8) Vectorize the 'Critical Risk' field (fit on train, transform on both)

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

9) Fit KMeans clustering on train, predict on both
10) Instantiate LabelEncoder and assign classes manually, Encoded both train and test using the pre-set mapping
11) Merged class 5 into class 4 (VI into V) for both train and test sets for "Accident Level".
12) Load GloVe Model, perform Sentence Embedding and apply to dataFrame. vectorizes the Description text using sci-kit learn's TfidfVectorizer
13) vectorized the Description text using all-MiniLM-L6-v2 from Hugging Face
14) Combined text and structured features for each ML input set (TF-IDF, GloVe, MiniLM).
    a. Used hstack for sparse data (TF-IDF + other features)
    b. Used np.hstack for dense data (GloVe/MiniLM + other features)
    c. Set up y_train/y_test using encoded target column
15) Definedd Multiple ML Models
    a. Random Forest
    b. Bagging Classifier
    c. Extra Trees
    d. XGBoost
    e. Logistic Regression
16) Set Up Cross-Validation and Metrics
    a. 5 splits, shuffling, stratified by class
    b. Scoring : Uses four evaluation metrics: accuracy, f1_macro, precision_macro, recall_macro
17) Runs cross-validation with the chosen metrics, aggregates the mean scores across all folds for each metric
18) Run Cross-Validation on All Feature Sets
    a. TF-IDF features
    b. GloVe features
    c. MiniLM features

TF- IDF features				
Model	**Accuracy**	**F1 Score (Macro)**	**Precision (Macro)**	**Recall (Macro)**
Random Forest	0.944	0.873875	0.931405	0.839894
Bagging	0.948	0.877496	0.923017	0.853155
Extra Trees	0.946	0.877537	0.93712	0.842394
XGBoost	0.942	0.870446	0.896528	0.864677
Logistic Regression	0.87	0.747637	0.776958	0.732099
GloVe Features				
Model	**Accuracy**	**F1 Score (Macro)**	**Precision (Macro)**	**Recall (Macro)**
Random Forest	0.948	0.878536	0.935347	0.844894
Bagging	0.938	0.866563	0.910749	0.842686
Extra Trees	0.942	0.872708	0.931177	0.839024
XGBoost	0.94	0.895863	0.906553	0.887937
Logistic Regression	0.898	0.757053	0.77569	0.755841
MiniLM Features				
Model	**Accuracy**	**F1 Score (Macro)**	**Precision (Macro)**	**Recall (Macro)**
Random Forest	0.946	0.876607	0.937197	0.840764
Bagging	0.946	0.874962	0.93148	0.842394

By: Bindhu Sukumaran , Chaitanya Soman, Gurudath Sadanandan, Ankit Dadhich, Bharath

Extra Trees	0.944	0.875781	0.936452	0.839894
XGBoost	0.944	0.865983	0.90895	0.843155
Logistic Regression	0.856	0.679437	0.726809	0.654863

19) Hyperparameter Tuning : We shall select the top three performing models for hyper parameter tuning and re-validation

20) Random Forest Classifier results :
   a. Best CV score (on upsampled train): 0.938
   b. Test accuracy (on real test set): 0.3690476
21) Extra Trees Classifier results :
   a. Best CV score (on upsampled train): 0.918
   b. Test accuracy (on real test set): 0.338563
22) Bagging Classifier with Decision Trees results:
   a. Best CV score (on up sampled train): 0.901
   b. Test accuracy (on real test set): 0.336125

Interpretation & Recommendations :

Why is test accuracy much lower than CV?
Overfitting:
The model is highly overfitted to the up sampled training set and doesn't generalize well to real (unseen) test data.

Train/test distribution shift:

Up sampling can distort the feature space if the original dataset is very small (yours has only 418 rows).

Synthetic diversity can be low, and the real-world distribution of test set classes might not match up sampled train.

Imbalanced original data:
Even after up sampling, the model is struggling to learn robust patterns for minority classes.

Small dataset effect:
For 418 rows total, up sampling is a stop-gap; more real data would be ideal.