ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

POST-GRADUATE PROGRAM

FINAL REPORT

NATURAL LANGUAGE PROCESSING

CHATBOT INTERFACE

GROUP MEMBERS:

ADITI PATIL

DHRUTIDHARA BEHARA

HEENA KHURANA

NEHA MENON

SAUMYA KOTHARI

GUIDELINES FOR FINAL REPORT

OBJECTIVES: It should contain all the 6 Sections mentioned in the project document. The report structure and brief requirements of each section is listed below.

1. *Summary of the problem statement, data, and findings*: Every good abstract describes briefly what was intended at the outset, and summarizes findings and implications.
2. *Overview of the final process*: Briefly describe your problem methodology. Include information about the salient features of your data, data pre-processing steps, the algorithms you used and how you combined techniques.
3. *Step-by-step walk through the solution:* Describe the steps you took to solve the problem. What did you find at each stage, and how did it inform the next steps? Build up to the final solution.
4. *Model evaluation*: Describe the final model in detail. What was the objective, what parameters were prominent, and how did you evaluate the success of your models?
5. *Comparison to benchmark*: How does your final solution compare to the benchmark you laid out at the outset? Did you improve on the benchmark? Why or why not?
6. *Visualizations*: In addition to quantifying your model and the solution, please include all relevant visualizations that support the ideas/insights that you gleaned from the data.
7. *Implications:* How does your solution affect the problem in the domain or business? What recommendations would you make, and with what level of confidence?
8. *Limitations*: What are the limitations of your solution? Where does your model fall short in the real world? What can you do to enhance the solution?
9. *Closing Reflections*: What have you learned from the process? What you do differently next time?

ABSTRACT

The problem statement consists of a database which is basically a record of accidents from 12different industrial plants in 03 different countries. It is an urgent need for industries/ companies around the globe to understand why employees still suffer some injuries/ accidents in plants. Based on which a chatbot utility has to be developed which can help the professionals to highlight the safety risk as per the incident description. This project facilitates the creation of a clickable GUI (graphic user interface) based chatbot interface for the same.

The data consists of 425 accident scenarios & 11 columns in the dataset.

```
[ ] # Get the top 5 rows
    display(industry_df.head())
```

	Unnamed: 0	Data	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party	Critical Risk	Description
0	0	2016-01-01 00:00:00	Country_01	Local_01	Mining	I	IV	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...
1	1	2016-01-02 00:00:00	Country_02	Local_02	Mining	I	IV	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...
2	2	2016-01-06 00:00:00	Country_01	Local_03	Mining	I	III	Male	Third Party (Remote)	Manual Tools	In the sub-station MILPO located at level +170...
3	3	2016-01-08 00:00:00	Country_01	Local_04	Mining	I	I	Male	Third Party	Others	Being 9:45 am. approximately in the Nv. 1880 C...
4	4	2016-01-10 00:00:00	Country_01	Local_04	Mining	IV	IV	Male	Third Party	Others	Approximately at 11:45 a.m. in circumstances t...

Shape of the data

```
[ ] print("Number of rows = {0} and Number of Columns = {1} in the Data frame".format(industry_df.shape[0], industry_df.shape[1]))

    Number of rows = 425 and Number of Columns = 11 in the Data frame
```

It was observed that there are a very few records in the dataset and hence we did upsampling of the data and applied different models and NN on the data to check for varioustypes of accuracies.

It was also observed that the data is skewed towards men.

OVERVIEW

On the data given, we first analysed the data. Findings from the analysis:

- Distribution of industry sectors differ significantly in each country.
- The number of accidents decreases as the Accident Level increases and increases as the Potential Accident Level increases.
- 44% Third party employees, 43% own employees and 13% Third party (Remote) employees working in this industry.
- Males have a higher accident level than females.
- Number of accidents is high in the beginning of the year and it keeps decreasing later.
- Number of accidents are very high in particular days like 4, 8 and 16 in every month.
- Number of accidents increased during the middle of the week and declined since the middle of the week.

After the analysis on the data, we performed the following NLP preprocessing steps:

- Converting description to lower case
- Replacing apostrophes to the standard lexicons
- Applying Lemmatizers
- Removing multiple spaces between word
- Removing stop words
- Removing punctuations

We performed the **visualisation WordCloud technique** for text data wherein each word is pictured with its importance in the context or its frequency.

We observed that there are many body-related, employee related, movement-related, equipment-related, and accident-related words.

We have performed following types of feature engineering techniques:

- Variable Creation - Word2Vec Embeddings - Word2vec takes as its input a large corpus of text and produces a vector space

```
Word2Vec(vocab=48, size=100, alpha=0.025)
[' ', 'e', 'i', 't', 'r', 'a', 'n', 'o', 's', 'l', 'c', 'd', 'p', 'm', 'g', 'u', 'h', 'f', 'b', 'y', 'v',
Word2Vec(vocab=48, size=100, alpha=0.025)
```

- Variable Creation - Glove Word Embeddings - It is based on matrix factorisation techniques on the word-context matrix.

```
print('Found %s word vectors.' % len(embeddings_index))
```

```
400000it [00:23, 17023.06it/s]Found 400000 word vectors.
```

- Variable Creation - TFIDF Features - Term Frequency (TF) is the total count of the unique words available within a document. Inverse Document Frequency (IDF) is used to identify the weights of the rare words or important words.
  Finally, both the generated matrices will be multiplied and another matrix will be obtained which will be having normalised weights available called TF-IDF.

	tfidf_activity	tfidf_area	tfidf_causing	tfidf_employee	tfidf_hand	tfidf_injury	tfidf_left	tfidf_operator	tfidf_right	tfidf_time
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.55	0.38	0.00	0.41	0.46	0.40	0.00	0.00	0.00

Variable Creation - Label Encoding - converting the labels into a numeric form so as to convert them into the machine-readable form.

	year	month	day	week_of_year	season	weekday	country	local	gender	industry_sector	emp_type	critical_risk	accident_level	potential_accident_level
0	2016	1	1	53	0	4	0	0	1	1	1	20	0	3
1	2016	1	2	53	0	5	1	1	1	1	0	21	0	3
2	2016	1	6	1	0	2	0	2	1	1	2	14	0	2

As there are very few records in the dataset, we have performed upsampling of data by:

- Upsampling minority classes
- Generating synthetic samples - SMOTE
- Variable transformation - Normalisation & Scaling
- Using PCA - Extracting Principal Components that capture about 95% of variance in data

We analysed various accuracy scores for different data using mentioned ML techniques.

<u>Output on the given dataset:</u>

	Method	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Multi-Class Logloss
1	LogisticRegression	0.74	0.73	0.56	0.73	0.63	0.88
2	RidgeClassifier	0.74	0.74	0.56	0.74	0.64	1.00
3	KNeighborsClassifier	0.75	0.70	0.57	0.70	0.63	6.48
4	SVC	0.74	0.74	0.54	0.74	0.63	0.92
5	DecisionTreeClassifier	0.99	0.64	0.66	0.64	0.65	12.34
6	RandomForestClassifier	0.95	0.73	0.56	0.73	0.63	3.07
7	BaggingClassifier	0.96	0.73	0.62	0.73	0.64	2.39
8	ExtraTreesClassifier	0.99	0.74	0.54	0.74	0.63	1.23
9	AdaBoostClassifier	0.73	0.74	0.67	0.74	0.66	1.23
10	GradientBoostingClassifier	0.94	0.68	0.55	0.68	0.61	0.96
11	LGBMClassifier	0.99	0.74	0.65	0.74	0.69	1.00
12	XGBClassifier	0.84	0.75	0.63	0.75	0.68	0.84

Output on upscaled data:

	Method	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Multi-Class Logloss
1	LogisticRegression	0.38	0.15	0.63	0.15	0.23	1.61
2	RidgeClassifier	0.64	0.07	0.01	0.07	0.01	1.00
3	KNeighborsClassifier	0.95	0.11	0.74	0.11	0.08	19.72
4	SVC	0.28	0.07	0.01	0.07	0.01	1.59
5	DecisionTreeClassifier	1.00	0.57	0.66	0.57	0.60	14.80
6	RandomForestClassifier	1.00	0.71	0.54	0.71	0.62	4.30
7	BaggingClassifier	1.00	0.73	0.59	0.73	0.65	2.42
8	ExtraTreesClassifier	1.00	0.74	0.54	0.74	0.63	1.32
9	AdaBoostClassifier	0.40	0.73	0.54	0.73	0.62	1.15
10	GradientBoostingClassifier	0.97	0.70	0.54	0.70	0.61	1.09
11	LGBMClassifier	1.00	0.73	0.54	0.73	0.62	1.36
12	XGBClassifier	1.00	0.75	0.59	0.75	0.66	0.98

Output on SMOTE dataset:

	Method	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Multi-Class Logloss
1	LogisticRegression	0.39	0.14	0.63	0.14	0.22	1.62
2	RidgeClassifier	0.73	0.07	0.01	0.07	0.01	1.00
3	KNeighborsClassifier	0.92	0.15	0.74	0.15	0.16	8.56
4	SVC	0.29	0.07	0.01	0.07	0.01	1.60
5	DecisionTreeClassifier	1.00	0.52	0.57	0.52	0.54	16.45
6	RandomForestClassifier	0.99	0.51	0.53	0.51	0.52	3.09
7	BaggingClassifier	1.00	0.60	0.58	0.60	0.59	2.28
8	ExtraTreesClassifier	1.00	0.67	0.57	0.67	0.62	1.38
9	AdaBoostClassifier	0.43	0.68	0.55	0.68	0.61	1.28
10	GradientBoostingClassifier	0.97	0.62	0.58	0.62	0.59	1.04
11	LGBMClassifier	1.00	0.74	0.67	0.74	0.66	1.16
12	XGBClassifier	0.99	0.54	0.61	0.54	0.57	1.17

Convert Classification to Numeric Problem

We first created a classification model that uses categorical columns and tf-idf features from accident description and label encoded target variable. We can use simple densely connected neural networks to make predictions.

Since we have ordinal relationship between each category in target variable, we have considered this one as numerical/regression problem and try to observe the ANN behaviour.

- Test Accuracy:50.00

Above one is under-fit model.

Multi-class classification - Target variable - One hot *encoded*

We created a classification model that uses categorical columns and tf-idf features from accident description and one-hot encoded target variable. We can use simple densely connected neural networks to make predictions.

- Train Accuracy: 73.95
- Test Accuracy: 73.81

This is a good fit model.

Multi-class classification - Target variable - One hot encoded with SMOTE data

We have created a classification model that uses categorical columns and tf-idf features from accident description and one-hot encoded target variable. We can use simple densely connected neural networks to make predictions.

- Train Accuracy: 20.00
- Test Accuracy: 7.14

This is over-fit model

Finally, we built the LSTM model-3 ways

1. Using description to predict Accident level
2. Using only Categorical Values to predict Accident Level
3. Using multiple inputs to predict Accident Level

STEP-BY-STEP WORKTHROUGH THE SOLUTION

STEP 4 - NLP Pre-processing for EDA

* 74% of data where accident description > 100 is captured in low accident level.
* Few of the NLP pre-processing steps taken before applying model on the data
    * Converting to lower case, avoid any capital cases
    * Converting apostrophe to the standard lexicons
    * Removing punctuations
    * Lemmatization
    * Removing stop words
* After pre-processing steps:
    * Minimum line length: 64
    * Maximum line length: 672
    * Minimum number of words: 10
    * Maximum number of words: 98
* From Word Cloud it was observed that:
    * There are many words related to body, employee, movement, equipment and accidents.
    * Body-related: left, right, hand, finger, face, foot and glove
    * Employee-related: employee, operator, collaborator, assistant, worker and mechanic
    * Movement-related: fall, hit, lift and slip
    * Equipment-related: equipment, pump, meter, drill, truck and tube
    * Accident-related: accident, activity, safety, injury, causing

STEP 5 - NLP Pre-processing for Model Building

* Using GloVe embeddings, converted word and sentences to vectorised form.
* Created a 'Term Frequency - Inverse Document Frequency ' df using TfidfVectorizer with only the top 30 most frequent terms
* All categorical variables were Label Encoded and a separate df was created with One Hot Encoded target variable.
* A final dataframe with all Label Encoded categorical variables and the tfidf vectors for the top 30 most frequent words was created.
* This dataframe was then split into Test and Train sets with a 20:80 ratio. Accident Level was the Target variable.

STEP 6 - Create Varied sets of data

* The X_train and y_train dataset was further used to create dataframes with varied transformations applied on them.
    * **Upsampling using Resample:** The minority classes were upsampled using the majority class. The end result was a dataset with the same count of data for all classes.
    * **Upsampling using Smote:** SMOTE, Synthetic Minority Oversampling Technique, is an oversampling technique where the synthetic samples are generated for the minority class. This again gave us a evenly distributed dataset.
    * **Scaling:** This ensures that all the columns in the dataset are in the range of 0 to 1
* At the end of this step, we had the following datasets:
    * **X_train, y_train:** Scaled data for training
    * **X_train_up, y_train:** Scaled and resampled data
    * **X_train_smote, y_train_smote:** Scaled data after SMOTE
    * **dummy_y:** One hot encoded target variable
    * **y_train_smote_dummy:** One hot encoded target variable after SMOTE

STEP 7 - Design, train and test ML classifiers

* These classifiers were run for three different types of data: LogisticRegression, RidgeClassifier, KNeighborsClassifier, SVC, DecisionTreeClassifier, RandomForestClassifier, BaggingClassifier, ExtraTreesClassifier, AdaBoostClassifier, GradientBoostingClassifier, LGBMClassifier, XGBClassifier
    * **Scaled data:** Here, the LGBMClassifier performed the best with 0.99 train accuracy and 0.74 test accuracy; DecisionTreeClassifier performed the worst with a train accuracy of 0.99 but a test accuracy of 0.64. H
    * **Data with Resampling:** RandomForestClassifier performed the best here with 100% train accuracy and 74% test accuracy while RidgeClassifier performed the worst with 63% train accuracy and 24% test accuracy.
    * **Data with SMOTE:** This dataset performed the worst in terms of accuracy with LGBMClassifier being the best model with 100% train accuracy and 65% test accuracy and RidgeClassifier being the worst with 71% train accuracy but only 24% test accuracy
    * Even though test accuracy was about 70% through out, these models show clear sign of overfitting.

STEP 8 - Design, train and test NN classifiers

* **Numeric Classification - Data with Label Encoded Target Variable:** Here, the inputs were X_train, y_train. Following is the model summary.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 50)                2150
_________________________________________________________________
dense_1 (Dense)              (None, 100)               5100
_________________________________________________________________
dense_2 (Dense)              (None, 150)               15150
_________________________________________________________________
dense_3 (Dense)              (None, 40)                6040
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 41
=================================================================
Total params: 28,481
Trainable params: 28,481
Non-trainable params: 0
_________________________________________________________________
```

* Train accuracy: 60.17963886260986
* Test accuracy: 50.0
* Accuracy: 0.4880952380952381
* Precision: 0.4880952380952381
* Recall: 0.4880952380952381
* F1 score: 0.4880952380952381

* **Multiclass Classification - Data with One-hot Encoded Target variable:** This is a multiclass classification model with One the encoded target variable. The inputs are X_train, y_train_dummy.

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_32 (Dense)             (None, 10)                430
_________________________________________________________________
dropout_18 (Dropout)         (None, 10)                0
_________________________________________________________________
batch_normalization_8 (Batch (None, 10)                40
_________________________________________________________________
dense_33 (Dense)             (None, 10)                110
_________________________________________________________________
dropout_19 (Dropout)         (None, 10)                0
_________________________________________________________________
batch_normalization_9 (Batch (None, 10)                40
_________________________________________________________________
dense_34 (Dense)             (None, 5)                 55
=================================================================
Total params: 675
Trainable params: 635
Non-trainable params: 40
_________________________________________________________________
```

* Train accuracy: 73.35329055786133
* Test accuracy: 73.8095223903656
* Accuracy: 0.7023809523809523
* Precision: 0.7283950617283951
* Recall: 0.7023809523809523
* F1 score: 0.7151515151515152

STEP 8 - Design, train and test NN classifiers [Cont.]

* **Multiclass Classification - Data with SMOTE and One-hot Encoded Target variable:** This is a multiclass classification model with One the encoded target variable with SMOTE applied. The inputs are X_train_smote, y_train_smote_dummy.

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_35 (Dense)             (None, 10)                430
_________________________________________________________________
dropout_20 (Dropout)         (None, 10)                0
_________________________________________________________________
batch_normalization_10 (Batc (None, 10)                40
_________________________________________________________________
dense_36 (Dense)             (None, 10)                110
_________________________________________________________________
dropout_21 (Dropout)         (None, 10)                0
_________________________________________________________________
batch_normalization_11 (Batc (None, 10)                40
_________________________________________________________________
dense_37 (Dense)             (None, 5)                 55
=================================================================
Total params: 675
Trainable params: 635
Non-trainable params: 40
_________________________________________________________________
```

  * Train accuracy: 52.06477642059326
  * Test accuracy: 19.0476194024086
  * Accuracy: 0.011904761904761904
  * Precision: 0.14285714285714285
  * Recall: 0.011904761904761904
  * F1 score: 0.021978021978021973

* Multiclass Classification - Data with One-hot Encoded Target variable gave us the best result in this step with about 73% test and train accuracy meaning that there was no overfitting or underfitting of data.

STEP 9 - Design, train and test LSTM classifiers

* **Model with Text Input:** Here, the description column is used to predict the target variable. Before using the columns, the following pre-processing is done on the column:
  * Converting description to lower case
  * Replacing apostrophes to the standard lexicons
  * Removing punctuations
  * Applying Lemmatizer
  * Removing multiple spaces between words
  * Removing stop words
  * Removing numeric values
  * Convert the words into their corresponding numeric indexes.
  * Sentences can have different lengths, and therefore the sequences returned by the Tokenizer class also consist of variable lengths, We padded our sequences using the max length.
  * Load the built-in GloVe word embeddings

```
Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 100)]             0
_________________________________________________________________
embedding_2 (Embedding)      (None, 100, 200)          553000
_________________________________________________________________
bidirectional_2 (Bidirection (None, 100, 256)          336896
_________________________________________________________________
global_max_pooling1d_2 (Glob (None, 256)               0
_________________________________________________________________
dropout_22 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_38 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_23 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_39 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_24 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_40 (Dense)             (None, 32)                2080
_________________________________________________________________
dropout_25 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_41 (Dense)             (None, 10)                330
_________________________________________________________________
dropout_26 (Dropout)         (None, 10)                0
_________________________________________________________________
dense_42 (Dense)             (None, 5)                 55
=================================================================
Total params: 933,513
Trainable params: 380,513
Non-trainable params: 553,000
_________________________________________________________________
None
```

  * Train accuracy: 73.95209670066833
  * Test accuracy: 73.8095223903656
  * Accuracy: 0.7380952380952381
  * Precision: 0.7380952380952381
  * Recall: 0.7380952380952381
  * F1 score:0.7380952380952381

STEP 9 - Design, train and test LSTM classifiers [Cont.]

* **Model using Categorical Values:** Here, we use all the categorical variables with the following transformations already applied on them:
  * Words to vectors and sentence to vectors transformation with the tfidf vectors for top 30 words.
  * Label encoding for all categorical variables.

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 12)]              0
_________________________________________________________________
dense_43 (Dense)             (None, 10)                130
_________________________________________________________________
dropout_27 (Dropout)         (None, 10)                0
_________________________________________________________________
batch_normalization_12 (Batc (None, 10)                40
_________________________________________________________________
dense_44 (Dense)             (None, 10)                110
_________________________________________________________________
dropout_28 (Dropout)         (None, 10)                0
_________________________________________________________________
batch_normalization_13 (Batc (None, 10)                40
_________________________________________________________________
dense_45 (Dense)             (None, 5)                 55
=================================================================
Total params: 375
Trainable params: 335
Non-trainable params: 40
_________________________________________________________________
None
```

  * Train accuracy: 73.95209670066833
  * Test accuracy: 73.8095223903656
  * Accuracy: 0.7380952380952381
  * Precision: 0.7380952380952381
  * Recall: 0.7380952380952381
  * F1 score:0.7380952380952381
* So, far the performance matrix for both the Text classifier and the Model suing Categorical values is exactly the same.

STEP 9 - Design, train and test LSTM classifiers [Cont.]

* **Model with Text Input and Categorical Values:** Here, the first submodel accepts textual input in the form of accident description. The second submodel will accept input in the form of meta information which consists of dense, batch norm and drop out layers. The output from the dropout layer of the first submodel and the output from the batch norm layer of the second submodel will be concatenated together and will be used as concatenated input to another dense layer with 10 neurons. Finally, the output dense layer will have five neuorns corresponding to each accident level.

```
Model: "model_5"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_7 (InputLayer)            [(None, 100)]        0
__________________________________________________________________________________________________
embedding_3 (Embedding)         (None, 100, 200)     553000      input_7[0][0]
__________________________________________________________________________________________________
bidirectional_3 (Bidirectional) (None, 100, 256)     336896      embedding_3[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_3 (GlobalM (None, 256)          0           bidirectional_3[0][0]
__________________________________________________________________________________________________
dropout_29 (Dropout)            (None, 256)          0           global_max_pooling1d_3[0][0]
__________________________________________________________________________________________________
dense_46 (Dense)                (None, 128)          32896       dropout_29[0][0]
__________________________________________________________________________________________________
dropout_30 (Dropout)            (None, 128)          0           dense_46[0][0]
__________________________________________________________________________________________________
input_8 (InputLayer)            [(None, 12)]         0
__________________________________________________________________________________________________
dense_47 (Dense)                (None, 64)           8256        dropout_30[0][0]
__________________________________________________________________________________________________
dense_50 (Dense)                (None, 10)           130         input_8[0][0]
__________________________________________________________________________________________________
dropout_31 (Dropout)            (None, 64)           0           dense_47[0][0]
__________________________________________________________________________________________________
dropout_34 (Dropout)            (None, 10)           0           dense_50[0][0]
__________________________________________________________________________________________________
dense_48 (Dense)                (None, 32)           2080        dropout_31[0][0]
__________________________________________________________________________________________________
batch_normalization_14 (BatchNo (None, 10)           40          dropout_34[0][0]
__________________________________________________________________________________________________
dropout_32 (Dropout)            (None, 32)           0           dense_48[0][0]
__________________________________________________________________________________________________
dense_51 (Dense)                (None, 10)           110         batch_normalization_14[0][0]
__________________________________________________________________________________________________
dense_49 (Dense)                (None, 10)           330         dropout_32[0][0]
__________________________________________________________________________________________________
dropout_35 (Dropout)            (None, 10)           0           dense_51[0][0]
__________________________________________________________________________________________________
dropout_33 (Dropout)            (None, 10)           0           dense_49[0][0]
__________________________________________________________________________________________________
batch_normalization_15 (BatchNo (None, 10)           40          dropout_35[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 20)           0           dropout_33[0][0]
                                                                 batch_normalization_15[0][0]
__________________________________________________________________________________________________
dense_52 (Dense)                (None, 10)           210         concatenate_1[0][0]
__________________________________________________________________________________________________
dense_53 (Dense)                (None, 5)            55          dense_52[0][0]
==================================================================================================
Total params: 934,043
Trainable params: 381,003
Non-trainable params: 553,040
__________________________________________________________________________________________________
None
```

* Train accuracy: 73.95209670066833
* Test accuracy: 73.8095223903656
* Accuracy: 0.7380952380952381
* Precision: 0.7380952380952381
* Recall: 0.7380952380952381
* F1 score:0.7380952380952381

* All three LSTM models performed exactly the same.

STEP 10 - Final Model Selection

* The next step was to select the model we would be using for our chat bot. We had to decide between the following models performing the best :
    * NN - Multiclass Classification - Data with One-hot Encoded Target variable
    * LSTM - Model with Text Input
    * LSTM - Model using Categorical Values
    * LSTM - Model with Text Input and Categorical Values
* Though, the NN model performed well, the Precision, Recall and F1 Score was behind the LSTM models.
* All of the LSTM models performed the same. We decided to go ahead with using **LSTM - Model with Text Input** because that model required the least inputs from the user. The user could only provide the description as the input and get the predicted Accident level as output.

STEP 11 - Building a Chat Bot

* This part of the code gave us a lot of issues. We had originally created the entire solution in Google Colab. But Tkinter does not work on Colab. We can tried using Flask but had no luck. The code was then downloaded to local and there were changes needed to the EDA graphs to get them running.
* The Chatbot takes the description from the Textbox on hitting Send and does the following steps:
    * Converts the string to df
    * Cleans the description using get_cleaned_desc function
    * Converts text to sequence
    * Loads pickled model
    * Predicts output using the loaded model
    * Converts the nXn output to 1X1
    * Converts the Quantitative output to Qualitative as :
        * I —> Minor
        * II —> Moderate
        * III—> Severe
        * IV —> Critical
        * V —> Fatal
    * Displays the output as a reply from Bot.

STOP

MODEL EVALUATION

The final solution uses Bidirectional LSTM model as the model to be pickled and used in the Chatbot.

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems. We can manage inputs in two ways using bidirectional LSTM, one from past to future and one from future to past and it differs this approach from unidirectional. BiLSTMs show quite good results as they understand the context better.

Our final solution only used the Accident Description column as the input. The following steps are performed on the column before feeding it to the model:

- Converting description to lower case
- Replacing apostrophes to the standard lexicons
- Removing punctuations
- Applying Lemmatizer
- Removing multiple spaces between words
- Removing stop words
- Removing numeric values
- Convert the words into their corresponding numeric indexes.
- Sentences can have different lengths, and therefore the sequences returned by the Tokenizer class also consist of variable lengths, we padded our sequences using the max length.
- Load the built-in Glove word embeddings

Model summary:

```
Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 100)]             0
_________________________________________________________________
embedding_2 (Embedding)      (None, 100, 200)          553000
_________________________________________________________________
bidirectional_2 (Bidirection (None, 100, 256)          336896
_________________________________________________________________
global_max_pooling1d_2 (Glob (None, 256)               0
_________________________________________________________________
dropout_22 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_38 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_23 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_39 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_24 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_40 (Dense)             (None, 32)                2080
_________________________________________________________________
dropout_25 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_41 (Dense)             (None, 10)                330
_________________________________________________________________
dropout_26 (Dropout)         (None, 10)                0
_________________________________________________________________
dense_42 (Dense)             (None, 5)                 55
=================================================================
Total params: 933,513
Trainable params: 380,513
Non-trainable params: 553,000
_________________________________________________________________
```

input_1: InputLayer
float32
input: [(None, 100)]
output: [(None, 100)]
embedding: Embedding
float32
input: (None, 100)
output: (None, 100, 200)
bidirectional(lstm): Bidirectional(LSTM)
float32
input: (None, 100, 200)
output: (None, 100, 256)
global_max_pooling1d: GlobalMaxPooling1D
float32
input: (None, 100, 256)
output: (None, 256)
dropout_4: Dropout
float32
input: (None, 256)
output: (None, 256)
dense_11: Dense
float32
input: (None, 256)
output: (None, 128)
dropout_5: Dropout
float32
input: (None, 128)
output: (None, 128)
dense_12: Dense
float32
input: (None, 128)
output: (None, 64)
dropout_6: Dropout
float32
input: (None, 64)
output: (None, 64)
dense_13: Dense
float32
input: (None, 64)
output: (None, 32)
dropout_7: Dropout
float32
input: (None, 32)
output: (None, 32)
dense_14: Dense
float32
input: (None, 32)
output: (None, 10)
dropout_8: Dropout
float32
input: (None, 10)
output: (None, 10)
dense_15: Dense
float32
input: (None, 10)
output: (None, 5)

Model evaluation is as follows:

- This model is good fit, it is identified by a training and validation loss that decreases to a point of stability with a minimal gap between the two final loss values. The loss of the model will almost always be lower on the training dataset than the validation dataset.

- We could see it accuracy continually rise during training. As expected, we see the learning curves for accuracy on the test dataset plateau, indicating that the model has no longer overfit the training dataset and it is generalized model.
- The model had similar scores for Precision, Recall and F1-score, about 74%, which shows that the model is the right fit without overfitting or underfitting.

COMPARISON TO BENCHMARK

- A benchmark study measures and compares usability metrics against a baseline study. In our case, with the very limited amount of data that we had, we used the Dummy Classifier as our baseline. Dummy Classifier is a classifier that makes predictions using simple rules. This classifier is useful as a simple baseline to compare with other (real) classifiers. Our final solution had to perform better than the baseline model. The Performance matrix using the dummy Classifier is as follows:

  - Test score: 0.5714285714285714
  - Accuracy: 0.6190476190476191
  - Precision: 0.6190476190476191
  - Recall: 0.6190476190476191
  - F1 score: 0.6190476190476191

- Some of the Machine learning models that we used, e.g., LogisticRegression using SMOTE data, fell way below this benchmark due to overfitting the data. The LSTM models have performed above the benchmark. This is because of the linguistic learning ability of the model. In addition to that. The preprocessing done on the data also allowed the model to concentrate on the important parts of the description without being led astray by the numerical and special characters.

VISUALIZATIONS

- Head (first 5 rows of data)

```
data.head()
```

	Unnamed: 0	Data	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party	Critical Risk	Description
0	0	2016-01-01 00:00:00	Country_01	Local_01	Mining	I	IV	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...
1	1	2016-01-02 00:00:00	Country_02	Local_02	Mining	I	IV	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...
2	2	2016-01-06 00:00:00	Country_01	Local_03	Mining	I	III	Male	Third Party (Remote)	Manual Tools	In the sub-station MILPO located at level +170...
3	3	2016-01-08 00:00:00	Country_01	Local_04	Mining	I	I	Male	Third Party	Others	Being 9:45 am. approximately in the Nv. 1880 C...
4	4	2016-01-10 00:00:00	Country_01	Local_04	Mining	IV	IV	Male	Third Party	Others	Approximately at 11:45 a.m. in circumstances t...

- Description of the data

```
data.describe(include=['object'])
```

	Data	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party	Critical Risk	Description
count	425	425	425	425	425	425	425	425	425	425
unique	287	3	12	3	5	6	2	3	33	411
top	2017-02-08 00:00:00	Country_01	Local_03	Mining	I	IV	Male	Third Party	Others	On 02/03/17 during the soil sampling in the re...
freq	6	251	90	241	316	143	403	189	232	3

- Table describing data types, unique values and missing values

	0	MissingVal	NUnique
Unnamed: 0	int64	0	425
Data	object	0	287
Countries	object	0	3
Local	object	0	12
Industry Sector	object	0	3
Accident Level	object	0	5
Potential Accident Level	object	0	6
Genre	object	0	2
Employee or Third Party	object	0	3
Critical Risk	object	0	33
Description	object	0	411

Variables and Pertinent Inferences

- Local
  - Highest manufacturing plants are located in Local_03 city and lowest in Local_09 city.

- Country
  - Percentage (%) of accidents occurred in respective countries: 59% in Country_01,31% in Country_02 and 10% in Country_03

- Industry Sector
  - Percentage (%) of manufacturing plants belong to respective sectors: 57% to Mining sector, 32% to Metal sector and 11% to others sector.

- Country + Industry Sector
  - Metals and Mining industry sector plants are not available in Country_03.
  - Distribution of industry sectors differ significantly in each country.

- Accident Levels
  - The number of accidents decreases as the Accident Level increases and increases as the Potential accident Level increases.

- Employee Type
  - 44% Third party employees ,43% own employees and 13% Third party (Remote) employees working in this industry.

- Gender + Employee Type
  - Proportion of third-party employees in each gender is equal, third party(remote) employees in each gender are not equal and own employees in each gender are not equal.

- Gender + accident Levels
  - Males have a higher accident level than females.
  - There are many low risks at general accident level, but many high risks at potential accident level

- Accident Levels + Employee Type
  - For both accident levels, the incidence of Employee is higher at low accident levels, but the incidence of Third parties seems to be slightly higher at high accident levels.

- Accident Levels + Calendar
  - Accidents are recorded from 1st Jan 2016 to 9th July 2017. Every month, there are a high number of accidents in 2016 and less in 2017.
  - Number of accidents is high in the beginning of the year and it keeps decreasing later.
  - Number of accidents are very high in particular days like 4, 8, and 16 in every month.
  - Number of accidents increased during the middle of the week and declined since the middle of the week.
  - Both of the two accident levels have the tendency that non-severe levels decreased throughout the year, but severe levels did not change much, and some of these levels increased slightly in the second half pf the year.
  - Both of the two accident levels are thought that non-severe levels decreased in the first and the last of the week, but severe levels did not change much.

Day Count
day
Days

Month Count
month
Months

Weekday Count
y
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Weekdays

- Critical Risk:
  - Most of the critical risks are classified as others.

Visualisation Word Cloud technique:

employee
hand
causing
left
right
operator
activity
assistant
worker
accident
equipment
collaborator
approximately
injury
moment
time
finger
work
floor
area
safety
circumstance
level
pipe
mesh
access
support
team
place
rock
pump
glove
meter
side
height
fall
metal
toward
performed
truck
cut
medical
mechanic
described
part
point
hit
one
structure
position
drill
bolt
slide
back
remove
platform
injured
generating
used
drilling
hitting
material
contact
edge
fragment
block
report
impacting
activities
using
cleaning
maintenance
foot
face
hopper
hose
water
front
inside
plate
made
two
car
falling
loading
step
loader
technician
chain
proceed
personnel
positioned
drainage
decide
tool
movement
without
pressure
slip
opening
wearing
allergic
carried
release
room
leave
machine
sheet
pulley
approx
due
key
moving
small
hydraulic
base
victim
immediately
event
last
leg
upper
scissor
tube
center
trying
ladder
weight
moved
returned
section
take
partner
mining
driver
helmet
reaction
perform
loaded
stopped
suddenly
carrying
cabin
making
release
machine
system
located
lip
bag
burn
glasses
shoulder
verify
turn
top
control
lift
ring
hospital
pressing
eye
pain
referred
gable
sleeve
pulling
hammer
load
type
welder
roof
wooden
direction
around
door
producing
struck
air
went
plant
feel
scoop
cover
mine
box
bee
stop
wall
tank
head
arm
bar
second
holding
bite
collection
vehicle
head
came
master
order
pressed
neck
balance
third
grating
staff
thumb
gate
cream
knee
cutting
wound
concrete
break
stepped
move
lower
come
lifting
removed
slick
valve
acid
near
sting
cable
released
impacting
lasted
cylinder
belt
help
instant
basket
system
zinc
center
silva
stopped
machine
release
room
personnel
proceed
positioned
leave
drainage
decide
tool
without
movement
pressure
slip
opening
wearing
allergic
carried
structure
one
bolt
drill
finger
time
floor
area
activity
assistant
employee
part
end
point
ground
impact
caused
region
surface
line
locomotive
steel
iron
person
piece
boot
projected
workshop
first
removing
guard
frame
hour
another
bee
stop
box
wall
mine
use
hammer
load
type
welder
roof
wooden
direction
around
door
producing
struck
air
went
plant
feel
scoop
cover
mine
partner
mining
driver
section
take
returned
moved
weight
maintenance
foot
face
hopper
hose
water
front
inside
plate
made
two
car
falling
loading
step
loader
technician
chain
proceed
personnel
positioned
drainage
decide
tool
movement
without

Machine Learning Models

1. Data with Label Encoding

	Method	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Multi-Class Logloss
1	LogisticRegression	0.74	0.73	0.56	0.73	0.63	0.88
2	RidgeClassifier	0.74	0.74	0.56	0.74	0.64	1.00
3	KNeighborsClassifier	0.75	0.70	0.57	0.70	0.63	6.48
4	SVC	0.74	0.74	0.54	0.74	0.63	0.90
5	DecisionTreeClassifier	0.99	0.64	0.66	0.64	0.65	12.34
6	RandomForestClassifier	0.95	0.73	0.56	0.73	0.63	3.07
7	BaggingClassifier	0.96	0.73	0.62	0.73	0.64	2.39
8	ExtraTreesClassifier	0.99	0.74	0.55	0.74	0.63	0.82
9	AdaBoostClassifier	0.73	0.74	0.67	0.74	0.66	1.23
10	GradientBoostingClassifier	0.94	0.68	0.55	0.68	0.61	0.96
11	LGBMClassifier	0.99	0.74	0.64	0.74	0.68	1.14
12	XGBClassifier	0.82	0.74	0.59	0.74	0.65	0.85

2. Data with Upsampling

	Method	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Multi-Class Logloss
1	LogisticRegression	0.74	0.26	0.68	0.26	0.35	1.77
2	RidgeClassifier	0.63	0.24	0.67	0.24	0.29	1.00
3	KNeighborsClassifier	0.95	0.50	0.63	0.50	0.55	11.75
4	SVC	0.97	0.56	0.59	0.56	0.58	1.31
5	DecisionTreeClassifier	1.00	0.61	0.65	0.61	0.63	13.57
6	RandomForestClassifier	1.00	0.74	0.64	0.74	0.67	3.09
7	BaggingClassifier	1.00	0.71	0.64	0.71	0.68	2.02
8	ExtraTreesClassifier	1.00	0.74	0.61	0.74	0.65	1.65
9	AdaBoostClassifier	0.40	0.61	0.59	0.61	0.60	1.30
10	GradientBoostingClassifier	0.97	0.57	0.63	0.57	0.60	1.06
11	LGBMClassifier	1.00	0.73	0.57	0.73	0.64	1.72
12	XGBClassifier	1.00	0.68	0.63	0.68	0.64	0.95

3. Data with SMOTE

	Method	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Multi-Class Logloss
1	LogisticRegression	0.77	0.26	0.57	0.26	0.33	1.92
2	RidgeClassifier	0.71	0.24	0.61	0.24	0.31	1.00
3	KNeighborsClassifier	0.92	0.32	0.55	0.32	0.38	16.38
4	SVC	0.95	0.43	0.54	0.43	0.47	1.51
5	DecisionTreeClassifier	1.00	0.46	0.60	0.46	0.52	18.50
6	RandomForestClassifier	0.99	0.57	0.59	0.57	0.58	3.67
7	BaggingClassifier	1.00	0.57	0.58	0.57	0.57	2.98
8	ExtraTreesClassifier	1.00	0.56	0.58	0.56	0.57	1.14
9	AdaBoostClassifier	0.58	0.55	0.56	0.55	0.55	1.33
10	GradientBoostingClassifier	0.96	0.55	0.58	0.55	0.56	1.20
11	LGBMClassifier	1.00	0.65	0.59	0.65	0.62	1.72
12	XGBClassifier	0.99	0.58	0.58	0.58	0.58	1.16

Here, we can see many classic cases of overfitting, with many models having around 100% train accuracy but only 73% test accuracy.

Artificial Neural Networks

1. Convert Classification to Numeric Problem

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 50)                2150
_________________________________________________________________
dense_1 (Dense)              (None, 100)               5100
_________________________________________________________________
dense_2 (Dense)              (None, 150)               15150
_________________________________________________________________
dense_3 (Dense)              (None, 40)                6040
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 41
=================================================================
Total params: 28,481
Trainable params: 28,481
Non-trainable params: 0
_________________________________________________________________
```

```
# evaluate the keras model
_, accuracy = model.evaluate(X_test, y_test, batch_size=8, verbose=0)
print('Test accuracy: %.2f' % (accuracy*100))
```

```
Test accuracy: 50.00
```

```
Text(0.5,1,'Training and validation accuracy')
```

```
Text(0.5,1,'Training and validation loss')
```

Above one is under-fit model, it can be identified from the learning curve of the training loss only. It is showing noisy values of relatively high loss, indicating that the model was unable to learn the training dataset at all and model does not have a suitable capacity for the complexity of the dataset.

2. <u>Multi-class classification-Target variable -one hot encoded</u>

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 10)                870
_________________________________________________________________
dropout (Dropout)            (None, 10)                0
_________________________________________________________________
batch_normalization (BatchNo (None, 10)                40
_________________________________________________________________
dense_6 (Dense)              (None, 10)                110
_________________________________________________________________
dropout_1 (Dropout)          (None, 10)                0
_________________________________________________________________
batch_normalization_1 (Batch (None, 10)                40
_________________________________________________________________
dense_7 (Dense)              (None, 5)                 55
=================================================================
Total params: 1,115
Trainable params: 1,075
Non-trainable params: 40
_________________________________________________________________
```

```
# evaluate the keras model
_, train_accuracy = model.evaluate(X_train, y_train_dummy, batch_size=8, verbose=0)
_, test_accuracy = model.evaluate(X_test, y_test_dummy, batch_size=8, verbose=0)

print('Train accuracy: %.2f' % (train_accuracy*100))
print('Test accuracy: %.2f' % (test_accuracy*100))
```

```
Train accuracy: 73.95
Test accuracy: 73.81
```

This one is good fit; it is identified by a training and validation loss that decreases to a point of stability with a minimal gap between the two final loss values. The loss of the model will almost always be lower on the training dataset than the validation dataset.

```
Text(0.5, 1.0, 'Training and validation accuracy')
```

We could see the accuracy continually rise during training. As expected, we see the learning curves for accuracy on the test dataset plateau, indicating that the model has no longer overfit the training dataset and it is generalised model.

3. <u>Multi-class classification – Target variable – One hot encoded with SMOTE data</u>

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 10)                870
_________________________________________________________________
dropout_2 (Dropout)          (None, 10)                0
_________________________________________________________________
batch_normalization_2 (Batch (None, 10)                40
_________________________________________________________________
dense_9 (Dense)              (None, 10)                110
_________________________________________________________________
dropout_3 (Dropout)          (None, 10)                0
_________________________________________________________________
batch_normalization_3 (Batch (None, 10)                40
_________________________________________________________________
dense_10 (Dense)             (None, 5)                 55
=================================================================
Total params: 1,115
Trainable params: 1,075
Non-trainable params: 40
_________________________________________________________________
```

```
# evaluate the keras model
_, train_accuracy = model.evaluate(X_train_smote, y_train_smote_dummy, batch_size=8, verbose=0)
_, test_accuracy = model.evaluate(X_test, y_test_dummy, batch_size=8, verbose=0)

print('Train accuracy: %.2f' % (train_accuracy*100))
print('Test accuracy: %.2f' % (test_accuracy*100))
```

```
Train accuracy: 20.00
Test accuracy: 7.14
```

```
Text(0.5, 1.0, 'Training and validation loss')
```

```
Text(0.5, 1.0, 'Training and validation accuracy')
```

This model is overfit model, it can be identified from the learning curve of the training and validation loss only.

LSTM Neural Network

1. Use description to predict "Accident Level"

```
print(model.summary())
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 100)]             0
_________________________________________________________________
embedding (Embedding)        (None, 100, 200)          612200
_________________________________________________________________
bidirectional (Bidirectional (None, 100, 256)          336896
_________________________________________________________________
global_max_pooling1d (Global (None, 256)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_11 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_12 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_13 (Dense)             (None, 32)                2080
_________________________________________________________________
dropout_7 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_14 (Dense)             (None, 10)                330
_________________________________________________________________
dropout_8 (Dropout)          (None, 10)                0
_________________________________________________________________
dense_15 (Dense)             (None, 5)                 55
=================================================================
Total params: 992,713
Trainable params: 380,513
Non-trainable params: 612,200
_________________________________________________________________
None
```

```
Text(0.5, 1.0, 'Training and validation loss')
```

```
Text(0.5, 1.0, 'Training and validation accuracy')
```

2. <u>Use only Categorical Values to predict “Accident level”</u>

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 12)]              0
_________________________________________________________________
dense_16 (Dense)             (None, 10)                130
_________________________________________________________________
dropout_9 (Dropout)          (None, 10)                0
_________________________________________________________________
batch_normalization_4 (Batch (None, 10)                40
_________________________________________________________________
dense_17 (Dense)             (None, 10)                110
_________________________________________________________________
dropout_10 (Dropout)         (None, 10)                0
_________________________________________________________________
batch_normalization_5 (Batch (None, 10)                40
_________________________________________________________________
dense_18 (Dense)             (None, 5)                 55
=================================================================
Total params: 375
Trainable params: 335
Non-trainable params: 40
_________________________________________________________________
None
```

```
Text(0.5, 1.0, 'Training and validation loss')
```

```
Text(0.5, 1.0, 'Training and validation accuracy')
```

3. <u>Using multiple inputs to predict "Accident Level"</u>

```
print(model.summary())
```

```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 100)]        0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 100, 200)     612200      input_3[0][0]
__________________________________________________________________________________________________
bidirectional_1 (Bidirectional) (None, 100, 256)     336896      embedding_1[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_1 (GlobalM (None, 256)          0           bidirectional_1[0][0]
__________________________________________________________________________________________________
dropout_11 (Dropout)            (None, 256)          0           global_max_pooling1d_1[0][0]
__________________________________________________________________________________________________
dense_19 (Dense)                (None, 128)          32896       dropout_11[0][0]
__________________________________________________________________________________________________
dropout_12 (Dropout)            (None, 128)          0           dense_19[0][0]
__________________________________________________________________________________________________
input_4 (InputLayer)            [(None, 12)]         0
__________________________________________________________________________________________________
dense_20 (Dense)                (None, 64)           8256        dropout_12[0][0]
__________________________________________________________________________________________________
dense_23 (Dense)                (None, 10)           130         input_4[0][0]
__________________________________________________________________________________________________
dropout_13 (Dropout)            (None, 64)           0           dense_20[0][0]
__________________________________________________________________________________________________
dropout_16 (Dropout)            (None, 10)           0           dense_23[0][0]
__________________________________________________________________________________________________
dense_21 (Dense)                (None, 32)           2080        dropout_13[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 10)           40          dropout_16[0][0]
__________________________________________________________________________________________________
dropout_14 (Dropout)            (None, 32)           0           dense_21[0][0]
__________________________________________________________________________________________________
dense_24 (Dense)                (None, 10)           110         batch_normalization_6[0][0]
__________________________________________________________________________________________________
dense_22 (Dense)                (None, 10)           330         dropout_14[0][0]
__________________________________________________________________________________________________
dropout_17 (Dropout)            (None, 10)           0           dense_24[0][0]
__________________________________________________________________________________________________
dropout_15 (Dropout)            (None, 10)           0           dense_22[0][0]
```

```
Text(0.5, 1.0, 'Training and validation loss')
```

```
Text(0.5, 1.0, 'Training and validation accuracy')
```

Chat User Interface

This output is as expected.

This output is as expected.

This output is wrong.

IMPLICATIONS

Through the course of this project, we have collected data from various sources to obtain the final dataset. The database comes from one of the biggest industries in Brazil and in the world. It is urgent need for industries/companies around the globe to understand why employees still suffer some injuries /accidents in plants. Sometimes they also die in such an environment. We have designed a ML/DL based chatbot utility which can help the professionals to highlight the safety risk as per the incident description.

The next steps would be that for every case, we can record and carefully collate the records. With these entries we can load them into the dataset. Currently we have only a 75% accuracy. We suggest recalibrating our data base before putting it into the world where it can properly be implementable.

We can also consider collecting data from different countries for the same use case to get a more robust dataset that can improve the model performance and provide rich predictions.

RECOMMENDATIONS

- In this project, we discovered that the main causes of accidents are mistakes in hand-operation and time-related factors.
- To reduce the occurrences of accidents, more stringent safety standards in hand-operation will be needed in periods when many accidents occur.
- I realized that the detailed information of accidents like 'Description' is so useful to analyse the cause.
- With more detailed information such as machining data (ex. CNC, Current, Voltage) in plants, weather information, employee's personal data (ex. age, experience in the industry sector, work performance), we can clarify the cause of accidents more correctly.
- With a greater number of observations than the current number of records = 425 so that we can feed more data into ML/ANN/NLP models to train, evaluate the performance of those models and get better results.
- There are quite a lot of Critical risk descriptions, but with the help of SME, we can decide whether this column has outliers or not and also SME can help us in understanding the data better.

LIMITATIONS

- We have a smaller number of observations to analyse the cause of accidents correctly and rather we should collect a greater number of observations to get better results.
- Less number of features available in dataset.
- Lack of access to quality data.

Where does our model fall short in the real world?

- Once we deploy the finalised model in production, we might get less F1-score as compared to productionalized model results.
- Since we are predicting the accident level, we need to be 100% sure or at least close to 100% so that we can prevent a lot of accidents in industry.

What can you do to enhance the solution?

- We can work on hyperparameter tuning and bootstrapping.
- We can also explore other Deep-Learning models like BART.
- We can also leverage Potential Accident level to see if it impacts the model performance in any way.

CLOSING REFERENCES

What did we learn from the process?

- How to work on Data science project to end-to-end.
- How to handle class imbalance dataset.
- How to build different ANN model architectures for handling multi-class classification problems.
- How to build different NLP architectures for handling both categorical and descriptive data.

What do we do differently next time?

- Perhaps we will explore more feature engineering and feature selection techniques.
- We will build the real chatbot using Streamlit or some other application.

CONCLUSION

- We have seven duplicate values in this dataset and dropped those duplicate values.
- We have no outliers in this dataset.
- We have no missing values in this dataset.
- Extracted the day, month and year from Date column and created new features such as weekday, weekofyear and seasons.
- Target variable – 'Accident Level' distribution is not equal (I: 309, II: 40, III: 31, IV: 30, V: 8).
    - Resampling techniques: Oversampling minority class
    - SMOTE: Generate synthetic samples
- By comparing the results from all ML methods with original data, we can select the best method as AdaBoost classifier with f1-score 65.38% with original data.
- Explored below options in Neural Networks:
    - Convert Classification to Numerical problem: achieved a test accuracy of 53.57% which is a bad result.
    - Multiclass classification - Target variable - One hot encoded: achieved a test accuracy of 73.81% and f1-score of 73.81% with original data + TF-IDF features from accident description column.
    - Create a model with Text inputs (accident description alone) only: surprisingly achieved a test accuracy of 73.81% and f1-score of 73.81% with original data.
    - Create a model with Categorical features only: achieved a test accuracy of 73.81% and f1-score of 72.28% with original data.
    - Create a model with Multiple Inputs (concatenated the layers from text input model and categorical features input model): surprisingly achieved a test accuracy of 73.81% and f1-score of 73.81% with original data.
- Finally bidirectional LSTM model can be considered to productionalized the model and predict the accident level.
- We have also experimented with using the 'Potential Accident Level' as the Target variable using all the mentioned techniques but the Accuracy and Confidence never rose above 20-25% in the Test set thus meaning that there is not enough relation between all the independent columns and the dependent variable ('Potential Accident Level').
- If we had more data, we could have had Validation Set as well.