

Final report for the submission of Industrial Chatbot

Shrav

Prepared by Sheshank Joshi with help from Harshavardhan Gorakh and Christen Abraham and under the valuable guidance provided by Mr. Vaibhav Kulkarni to whom the author is grateful.

NOTE :

Please note that this final reports builds and refers to the Interim report that is submitted before this and is given as an appendix to this documents. It is referred to everywhere in this document.

With unresolved differences in opinions and approaches between the team members, it was decided by the mentor and coordinator to split the team into two at the behest of the request made by the author. This report approach as listed in the report under appropriate heading.

Abstract

Note : The abstract given here is the replica of what is already given in the Interim report (please refer the appendix). Repeated here for the sake of quick revision.

Given the age of climate change and the value of capital investment in heavy goods industries, it has become extremely important to identify the disaster very early in the making. But it is also important to stop any incident/accident from worsening. Hence it becomes a necessity to analyze the early information or description by any person regarding the incident and assess the appropriate accident level. From that, taking other factors into consideration, employing the most modern methods of A.I and Natural Language processing, it is possible to assess the accident level from the information provided, along with some additional information that aids in assessing the threat level. In this regards attempts have been made to automate a part of this process, through a chatbot that is user friendly and third party accessible. As part of this challenge, an example data set provided by the problem statement (given below) and the possibilities and methods for development of such model has been taken.

Tools used :

A part from Standard Python Library, the following were the libraries used with appropriate versions of those libraries. The IDE used was Spyder 4 and python version 3.9.7.

<u>Library</u>	<u>Version</u>	<u>Purpose</u>
Gunicorn	20.1.0	Hosting Http requests on deployed server for Flask
tensorflow	2.5.0	Building Neural Networks for NLP Core Module
Wheel	0.35.1	For cloud deployment compatibilities
Pandas	1.3.2	For Data manipulation and handling
Textblob	0.15.3	In Language Model core for handling spelling errors
keras_self_attention	0.50.0	A special third party library of pre-configured layers for attention mechanism in NLP Models
Seaborn	0.11.2	For handy visualizations on data
Nltk	3.6.2	Used to build language models and vocabulary for text generation
Scipy	1.7.1	For statistical analysis of generated probabilities
Imbalanced_learn	0.8.0	A Scikit-Learn library extension for imbalanced class datasets
Pip	21.2.4	Standard python package installer for deployment on server
Scikit_learn	1.0	Main library for Supervised learning and data manipulation

Dataset Used :

Please refer to the appendix for the link and further information.

Project Team : Sheshank Joshi as a lonely front and backend programmer with Harshavardhan Gorakh providing backend support on documentation and purposes.

Please Note :

For Data description, Assumptions, Limitations and Exploratory Data Analysis, please refer to the Appendix for Interim Report documents.

There hasn't been any significant additions or changes made to the points already mentioned (in the previous report).

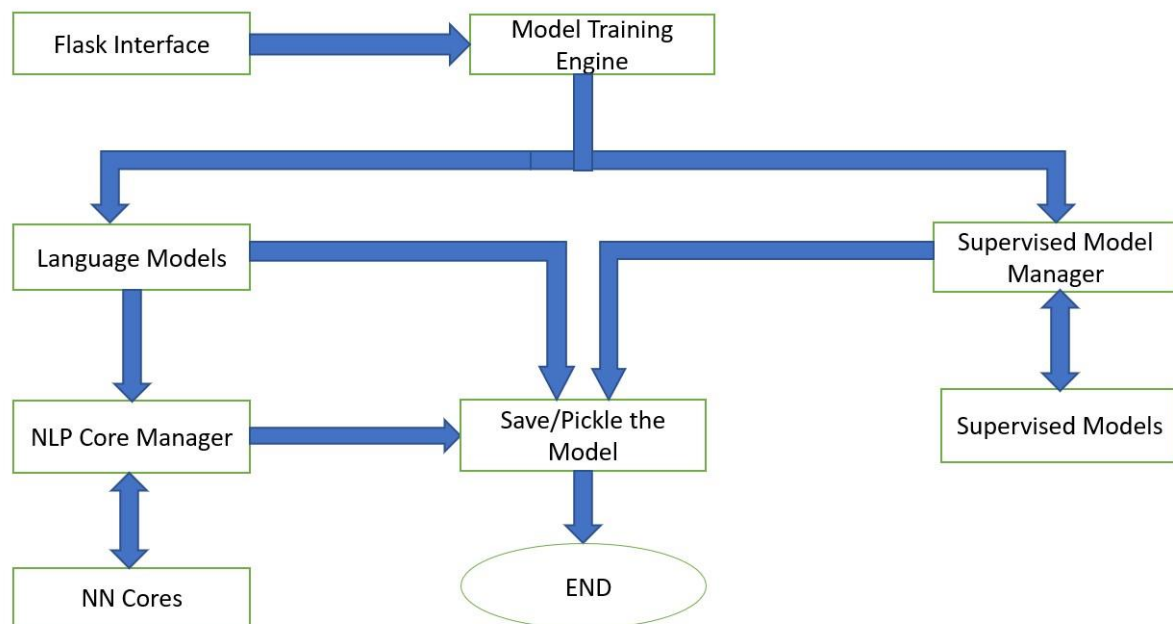
Comments on Hypothesis :

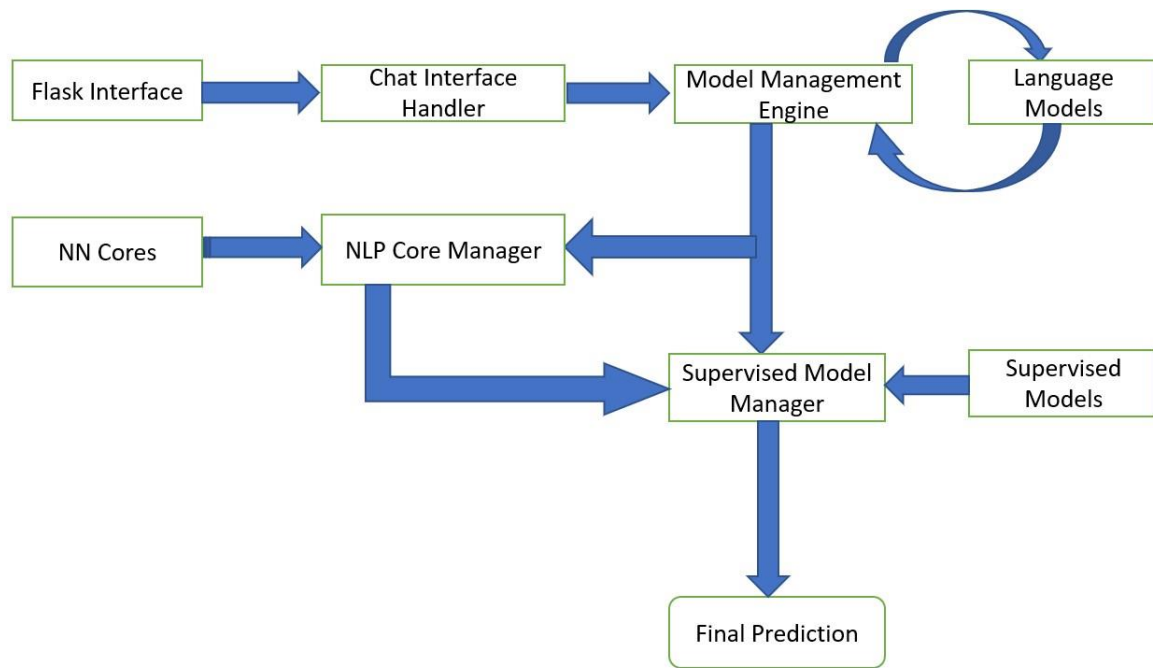
In addition to the already mentioned Hypothesis (in the appendix), it was observed that both Simple and Complex Hypothesis mentioned, are implementable. But the accuracy aspect of such an inclusion, i.e. supervised model building on top of Neural Network model conveniently ignores the probabilistic pathways and likelihood metrics for the models. A work-around has been implemented with detailed explanation at a later part in the this report.

Architecture of the entire project

For the sake of scalability and deployment mechanisms the following architecture with block diagrams have been thought about.

Background Processes : To implement a chatbot form of the actual project, it needs to be taken out of the format of a notebook and put in the deployable format i.e. individual modules and import as and when needed. Hence the entire project is split into two different phases 1. Training Phase 2. Run-time (prediction time) phase. Block diagrams representing them are given below.





Hence, all the backend information that was earlier reported as Jupyter Notebooks (accompanied with previous report) is now shifted to a new IDE viz. Spyder 4.2.5, for flexibility on deployment as well as smooth transition into .py format from .ipynb style of developing projects.

Please Note : The debugging information written, notes written during development are retained in the files containing source code to give the evaluator of this project, to have a complete picture of what is or was going on through author's mind and to explain in more detail each and every line or step of code written (hence help get a detailed feedback on things gone wrong, if any).

Briefed workings of each module are given under their respective sub-headings, are given below. To understand it better, they can be thought of something like OCI structure or layered structures of information flow top layer to bottom layer.

Flask Interface : This is placed in the file either in app.py file or main.py file, depending upon the file choice for deployment, though both are being tried at the time of generating this report. It has the flask interface that launches the code.

Chat Interface Handler : It is included in *dialogue.py*. It has a Chat_interface_handler class that is the entry point for predictions, and entire chatting engine. One has to initialize this object to access everything. It has a dialogue management engine, with one single entry point i.e. dialogue_in() method. It returns the appropriate intelligent answer. It also loads some sample dialogues from static responses generated by the trainer. It calls in a layer called model management engine that handles the information appropriately.

Model Management Engine : It is placed in the *model_management_engine.py* file with same class name. It brings all the different manages i.e. language model manager, the supervised model manager and the Neural Network Manager together and coordinate between them, especially loading the pre-trained models. IT IS REQUIRED THAT THE MODEL BE TRAINED FIRST IF USING THIS.

Model Training Engine : It is placed in model_training_engine.py file with the same class name. It takes care of selecting the best model available from the list of models initiated in the *NN_models.py* file i.e. the *NLP_core_manager.py* file. The metric is taken as loss i.e. the model that shows best

trends in reduced loss and accuracy. The automatic tuning of the selected or best model is not implemented in complete here, but the provision is in place with appropriate calls and everything in place. All it needs is populating the method appropriately. Primarily due to computing constraints the idea of automatic tuning aspect of training all the models (both NN Models and Supervised Classifier Models), the author of this report has deemed it inappropriate to current time frame to implement it. Hence the easy option of rough tuning and experimentation, manually by intelligent guesses, is undertaken by the time previous report was given.

NLP Core Manager : Given in *NLP_core_manager.py*, this is a prima-face interface and acts completely independently of any engines above it. Its defined in the class name NN_NLP. It brings core Neural Networks defined in NN_models.py file with different class names. It handles both prediction and training aspects. It also saves all the models and corresponding parameters. Refer to the actual code or use help() function to get more details on the class.

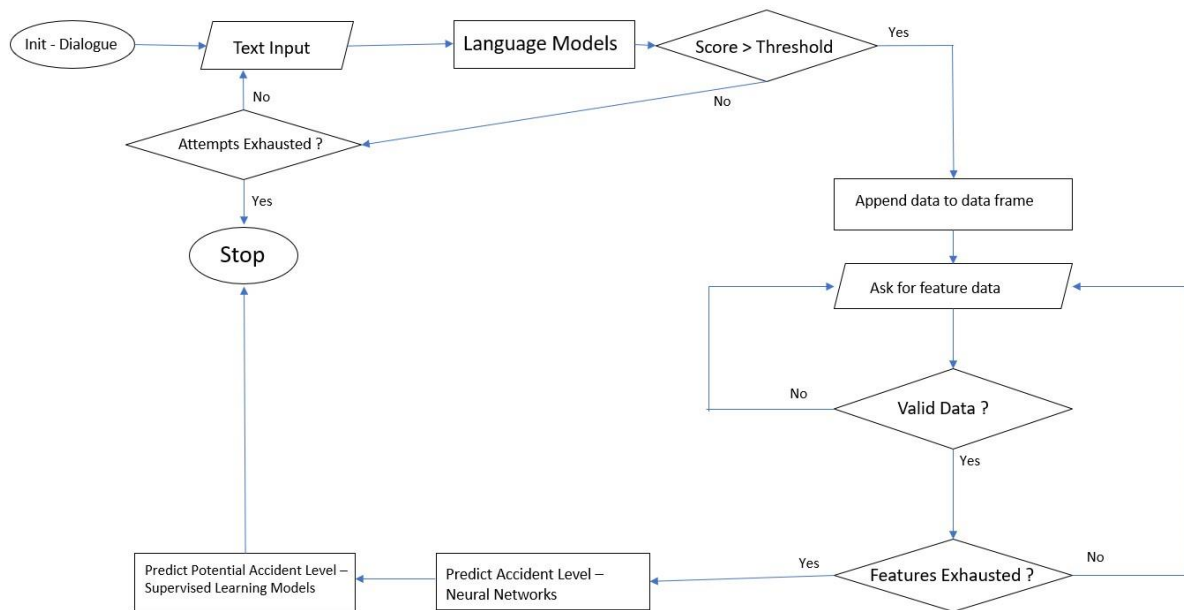
Language models : This is the most important part that acts as a pre-screening mechanism. The Basic idea is that chatbot should be able to identify the difference between a description for industrial accident and a random text, say a Nursery Rhyme. In that regards, there should be a mechanism of very strict Conditional frequency in terms of probability distribution possible to handle and decide a given set of words belong to a certain set of corpus or not. It eliminates irrelevant inputs and puts that extra touch to the entire model to make it working all for the right reason. More on how it is handled is explained as and when required i.e. in dialog flow. The two models implemented are 1. Maximum Likelihood Model 2. Kneser-Ney model (which uses Kneser-Ney smoothing methodology for probability distribution). More details can be found in NLTK library documentation. It is basically a conditional probability distribution for an n-gram based on history and exposure to the model.

NN Cores : These are listed in *NN_models.py*. There are five models, all of them are Models subclassed from Tensorflow Model, with various layer architectures (the actual layer architecture is given in the code that accompanies this report). Instead of going for saving model weights and model architectures separately, it is chosen to save entire model using the feature provided by the Tensorflow itself through its provision (for tensorflow 2.0 its Model.save) that saves the entire Network structure, weights and also losses, optimizations etc.

Supervised Models: Implemented in the file titled supervised_models.py, this contains supervised learning models listed as , 1. Logistic Regression, 2. Gaussian Naïve Bayes 3. K Neighbors Classifier 4. Support Vector Classifier 5. Decision Tree Classifier 6. Bagging Classifier 7. Adaboost Classifier 8. Gradient Boost Classifier 9. Random Forest Classifier 10. Multilayer Perceptron Classifier. Though automatic tuning for best parameters via Grid Search CV has been considered here, and scope for its implementation is provisioned for, the actual implementation is not undertaken.

Supervised Model Manager : It is subclassed on the core of Supervised Models module. All of them are trained and the appropriate metrics and their study is also implemented here. It is present as a class name sup_manager, within the file name supervised_core_manager.py. It handles both training, saving and prediction phases of the models. In addition to that, it conditions data like PCA transformation etc. for data processing and accuracy improvement. It automatically identifies the maximum number of features in PCA and turns and transforms data (based on maximum rate of explained variance ratio).

Chat Dialogue flow:



Explanation :

Chatbot takes in the dialogue input from the user, after first wishing the user after giving quick inputs. But the chatbot has to be initialized first after launching it by typing “next_item” as input. That will launch welcome message, though it works without that message too and can be used directly from the get go.

The language models trained, generate a probability distribution score for the given input texts, based on trained corpus. That score, if it par the threshold score, will be accepted as the input for Description column. If not the process is repeated, giving appropriate suggestions to the user, based on some examples from the original train dataset until 5 or 6 attempts have been finished. Each time user inputs data, it is incrementally added to the previous text input and finally when the score is part, the entire text is considered a single input and delegated to the appropriate module for further processing.

Additional Components involved :

A good deal of JavaScript had to be developed to be successfully able to handle server communication. For the sake of scalability and security, in the view of future uses, the POST method was preferred (primarily because of the lack of limit on the amount of text the user is flexible to input).

Ajax Communication was planned and successfully established that undertakes background communication, though insecure. Hence, it can be said that security perspective has been completely ignored here. Author is of the perspective that any effort in that regard is not required for the present project since its not going to be critical or deployed in real world situations.

Gunicorn library is inherently used (while deployment) for handling Http requests on server side.

HTML and CSS have been kept to the minimum by the author. Since, author is not an expert in either of them, they are developed to the minimum possible components and virtually no emphasis on aesthetics of the chat interface. Author assures that it is original code and hence no credits/references required.

A screen of the same has been placed here. Actual product can be checked/launched directly from the application folder from the entry point. Or you can check the HTML and JavaScript files in templates and static directories respectively (in compliance with Flask directives).

Benchmarks for the project :

At the outset of the project, the background engines are expected to function as one single unit. Their output is expected to be atleast 90%. But it didn't hit that but came close i.e. around 88%. The data points were found to be in-sufficient for efficient model training to happen. For that mitigation was carried out through Up-sampling and under-sampling the data points, but still it wouldn't introduce enough variety in the data, it can only be used to eliminate any class bias. Hence, it was observed that one single model isn't enough. Hence, for both Neural Networks and Supervised models, the method of using multiple models and using the maximum votes from different models, used to avoid the prediction errors. Its like Bagging classifier, only a little simpler.

Model Evaluation and Parameters :

Working of this whole project rested on a number of things. But the most important parameter has been the ability to differentiate an irrelevant text input from that of the actual description. The example of this has been discussed in the paragraph explaining the Language models. That parameter has been successfully achieved. There is a special function defined to debug the models in the chat interface handler that handily outputs the data related to the achieved score by any given text input.

For Neural Network models, If validation split has been ignored, the train score ought to achieve above 92% for any one single model, which is successfully achieved by Multi-head Attention model (93%).

Despite the efforts to decrease class imbalance while training, the best achievable score for any one single model hasn't been able to touch 95% which is a failure. Hence, it was considered a strong indication to go for multi-model prediction. Since models showed so much variety, it was understood to be better if used all of them concurrently.

Given the complications faced, it was decided that conservative estimates should be taken instead of going for accuracy. For e.g if probabilities associated with outputs are extremely close, say 45% level 1 and 42% level 2. On a conservative estimate, it is programmed to output Level 2. It can always be tuned.

Visualization and Evidence has already been provided in the interim report, attached in the appendix here. Please check out the details. Tuning parameters for all the modules are appropriately placed within those modules as class variables, wherever applicable.

Below is the screen shot of the chatbot in action.

Preventing Disasters one at a time

Was the Person injured Male or Female. Can you choose from below ? 0 . Female 1 . Male

Assuming you are the reporting person. Can you specify from below ? 0 . Employee 1 . Third Party 2 . Third Party (Remote)

I know it is tough. But, it is appreciated if you could answer what was the critical risk involved with accident, from your guess. Select from below frequently occurring risks. 0 . Not applicable 1 . Dues 2 . Blocking and isolation of energies 3 . Burns 4 . Chemical substances 5 . Confined space 6 . Cut 7 . Electrical Shock 8 . Electrical installation 9 . Fall 10 . Fall prevention 11 . Fall prevention (same level) 12 . Individual protection equipment 13 . Liquid Metal 14 . Machine Protection 15 . Manual Tools 16 . Others 17 . Plates 18 . Poll 19 . Power lock 20 . Pressed 21 . Pressurized Systems 22 . Pressurized Systems / Chemical Substances 23 . Projection 24 . Projection of fragments 25 . Projection/Burning 26 . Projection/Choco 27 . Projection/Manual Tools 28 . Suspended Loads 29 . Traffic 30 . Vehicles and Mobile Equipment 31 . Venomous Animals 32 . remains of choco

IV Thanks for using me. The above is the potential for your accident level based on the responses given by you. Please contact appropriate authorities. If you want to try again, please reset using the button given and start again. I am going to take a leave by ending this conversation now. Its a pleasure serving you.

reset [input field] Send

Please Note : Though the project for Grid Search CV and automatic tuning was implemented in previously presented notebook, its not practically feasible to implement the same on cloud train. So, it was not considered from the beginning stages itself of plan for the deployment.

Web Deployment :

The actual hosting of the code is undertaken at the URL given below

<https://chat-industrial-disaster-xdck3kjaqq-as.a.run.app> (run in South East Asia region for deployment feasibilities.)

<https://shravani-chatbot-xdck3kjaqq-el.a.run.app> (run in south asia region with Shrav being short form of original proposed chatbot name viz. Shravani)

At the time of filing the report there are some error generated while deployment, fundamentally created due to version incompatibilities, partially because the versions have changed since first adoption and final deployment stages. Problems encountered were primarily focussed on two aspects : 1. Due to multi-modal type choice, the start-up time and the computing resources required far exceed that of the limit placed on free resources offered by google 2. Cloud deployment is beyond the author's speciality on the subject. Hence, couldn't finish troubleshooting the errors within the time frame provided for the project. The corresponding error logs are accompanied with this report with the file name titled "cloud deployment error logs.txt".

Complete end to end training mechanism has been provided and implemented in Model Training engine. It is just a matter of initializing the Training Engine object with data loaded into a pandas data frame and arguments specifying it accordingly. A GUI designed for the same, via cloud computing, turned out of scope and beyond the time limits and capacity of the author, after several failed attempts. Alternatives couldn't be finished on time, which is deeply regretted.

Implications : This project has been designed and modularized in such a way that anything and everything can be modified without disturbing any other parts of the code and nothing is hard

coded. Everything can be tuned just one time during training phase. It is designed in such a way that, though it is not needed, the names of the feature columns hasn't been restricted nor are the names. Only the structure that NLP core with Supervised Core wrapped around its core is hard coded. Even in that, care has been taken such that models can always be replaced and modified. Hence, the scalability of the code is huge, and so is the model.

Any dataset that follows similar kind of abstract outline (as mentioned in assumptions column in the appendix), is able to use the code irrespective of what the columns represent. Even the dialogue handlers, saving parameters, everything is designed for the same.

Hence, the implications include the following for the Chatbot :

1. A Direct application of converting this chatbot into an IVRS system where anyone with a phone is able to report any industrial accident they see. The speech to text conversion adaptors can be used to apply this.
2. Interactive chatbots with the above, with Interactive responses indicating what exactly to do can be also indicated. Here it is not implemented. The chatbot stops at predicting the accident level and just informs the user to contact appropriate authorities.
3. If taken a step further, this can be used on a large to report any kind of accidents, not just industrial disasters. It might actually help the bystanders in arranging first-aid to the accident victims, instead of waiting for the staff to arrive.
4. If the NLP part of it is stressed further and not restricted to this domain alone, it can be used as a personal assistant to assess situations and working conditions are reported by the authorities. It can convert reports into actionable insights and classes.

Limitations :

Most of the limitations specified in the report earlier still apply here. In addition to that certain limitations have been identified while the working process here

1. Since part of speech implementation is skipped, it might lead to mis-representation of things
2. In language models, spell check is used with maximum probability. But that's not always the case where proper implementation should've been dependent on the context of occurrence of mis-spelled word.
3. Training additional data on top of the present pre-trained model is not yet verified or even tested for. Hence,
4. Since its not trained on large dataset, it might be very well confirmed to large biases, for which there are no positive ways to identify and correct other than the attempts already made.
5. Chatbot aspect of it, though is not of much focus, hasn't been handled to perfection where any input thrown at it is handled. Hence it can be said that this model is not for anyone who wants to have a chat. It is only for those who have seen it in action or the ones who are familiar with the conditions of its usage. For e.g. it doesn't greet the user back.
6. Visual aesthetics and certain aspects of CSS and HTML elements are not handled at, all though they are provided for. E.g. The reset button doesn't work, the description about authors doesn't actually carry any useful information about authors except describing their favourite animal.

Final Reflections :

It is a fantastic, exhausting, exhilarating journey from the beginning to the end. It made the author learn a whole new language called JavaScript. Thanks are in place, as mentioned at the beginning of the report.

This project is not a single person project, neither is it feasible to finish it in less than one month. Hence, had to make some compromises on the project. It is acceptable, but taking hard calls on things is essential in the very beginning of the project, to mitigate complications at the end. It was observed with respect to cloud deployment and version incompatibilities, where Docker Container could have been better.

Better team integration and standardized work environments are essentials, along with good communication skill and assertive leadership to lead the project in the right direction. There will always be differences in opinions and interpretations, but there can't be multiple right answers. Hence, there should be a clear path laid out at the very outset of any project that will decide how and where everything will go.

Interim Report for the project of Industrial Chabot

Abstract

Given the age of climate change and the value of capital investment in heavy goods industries, it has become extremely important to identify the disaster very early in the making. But it is also important to stop any incident/accident from worsening. Hence it becomes a necessity to analyze the early information or description by any person regarding the incident and assess the appropriate accident level. From that, taking other factors into consideration, employing the most modern methods of A.I and Natural Language processing, it is possible to assess the accident level from the information provided, along with some additional information that aids in assessing the threat level. In this regard attempts have been made to automate a part of this process, through a chatbot that is user friendly and third party accessible. As part of this challenge, an example data set provided by the problem statement (given below) and the possibilities and methods for development of such a model has been taken.

The problem statement can be broken down broadly into 2 parts – (1) To choose and train an ML/DL classifier model that can predict the severity (Potential Accident Level) of an accident provided supporting information. (2) Create an interactive chatbot which can converse with a user to extract all the necessary information and respond with the correct severity level of the incident reported.

Tools used: Python base with Tensorflow, Scikit-learn, pandas, numpy, matplotlib, jupyter notebook.

Note: The exact versions of the tools used, and code implemented can be found in the accompanying notebook.

Dataset used: <https://www.kaggle.com/ihtmstefanini/industrial-safety-and-health-analytics-database>

Project Team: Naseem Shaik , Aditya Vats, Harshavardhan Gorakh, Neerav Dahiya, Rakesh Gowda, Sheshank Joshi

Approach taken: We have decided to take two approaches to solving the classification problem. One approach focuses on an NLP model that tries to make its prediction based majorly on the text data provided. The other approach tries to employ supervised learning techniques on the non-textual data for predicting the target variable (in this case, Potential Accident Level). The NLP core model is handled by a team and Supervised models are handled by another team. A combined approach where a supervised model is wrapped around an NLP core, is also planned, subject to time constraints.

You can find the two aforementioned approaches in the two different notebooks provided.

Data description:

Apart from the description given from the source, it has been noticed that data contains 10 columns in total. Each column represents one feature each with the crux of the data being the description (with the name “Description”) in the dataset. All the columns are categorical, if not text. There are in total 413 data points which represent diverse, non-repetitive entries. Since countries have been aliased -so are the localities for data security – it hasn’t been explored any further. In the process of analysis of data, certain assumptions, with regards to data at hand, have been made as listed.

Assumptions:

1. Description for each entry describes only one event, and not multiple events occurring at the same time and same place.
2. Potential for any accident level can't be lesser than the assessed accident level, at any cost.

3. Given an accident level, all the other columns that are given i.e. all the columns except the description column, directly influence the chance of the escalation into a higher level.
4. Only factors given here influence the accident level turning into potential accident level. No other external factors are entertained.
5. The class imbalances noticed in the dataset are definitely indicative of the ground scenario and are considered as the attributes of (representative of) that particular feature. They are inseparable.
6. The Date that is logged into the database is the date on which the incident happened. In other words database entries made for each column are not other than the date the actual incident occurred.
7. Imbalances, noticed if any, are not caused by the data collection or database logging mistakes. It is genuinely reflective of the character of the feature. As mentioned in the matrix below
8. Description given for any incident in any country or any locality, is independent of the local nuances and variations within the usage of language (for e.g. accents, foreign language words etc.)
9. The Gender column given is the gender of the person that is mentioned in Employee or Third Party, which itself clearly indicates that an injury has happened and someone is hurt.
10. The accident level and the potential accident level, given the above point, is indicative of the threat posed to the concerned person/persons of interest and not the environment/factory/industry or other factors (which is assumed to be indicated by the column called "Critical Risk")
11. Given the above point, the fact that accident happened, is indicative that assessment for any accident level is dependent on the injury involving persons and not disaster caused by some other reason (involving no human element)

In the light of the above, it follows that certain limitations exist on any model we seek to base our model out of. Those are listed under limitations.

Limitations:

1. Data format for the model is fixed i.e. it should definitely have all the features as given in this dataset. Any extra data given, is not going to be entertained by the model in any case towards betterment of the existing model.
2. Any given feature, is supposed to follow the same guidelines to scaling their data boundaries as in the original dataset. Though it is not necessary, the desired results can only be obtained if the data is appropriately scaled. Hence, it is advised appropriately.
3. Scalability of the model to unseen data is subjected to severe limitations, particularly on unseen data. Model will not be robust enough to deal with unseen data from, say a different country or different locality or different industry. This can't be addressed by the model that is pre-trained on this particular dataset. The model weights and their performance is restricted to within the boundaries of this particular dataset. However, an appropriate mechanism for handling a custom dataset is being considered and worked upon right now.
4. If any foreign language words are found, they are eliminated. They don't add value to the performance of this model. No mechanism to handle such issues has been implemented.

Certain problems faced while dealing with data and the models, have been given in more detail in the accompanying data. The solutions to those problems and difficulties have enlightened us into coming up with the information given here.

In the light of the above, the dataset has been explored front and back thoroughly. The observations, inferences, Hypothesis, proofs and conclusions are handled with considerable detail in the accompanying notebook. The most important and implicative ones are touched upon in this report. Final reports on influences on the model and actual

performance evaluation of the model are, once again, given in considerable detail in the accompanying notebook. However, the findings are reported as seen here.

Summary of the problem statement, data and findings

Problem statement: In today's world, we have been hearing lots of Industrial accidents happening. It is an urgent need for industries/companies around the globe to understand why employees still suffer some injuries/accidents in plants. Sometimes they also die in such an environment. We have been given the dataset that comes from one of the biggest industries in Brazil and in the world. We need to design an ML/DL-based chatbot utility that can help the professionals to highlight the safety risk as per the incident description.

Data: The database comes from one of the biggest industries in Brazil with records of accidents from 12 different plants in 3 different countries where every line in the data is an occurrence of an accident. Dataset columns are as below:

- Date: timestamp or time/date information
- Countries: which country the accident occurred (anonymized)
- Local: the city where the manufacturing plant is located (anonymized)
- Industry sector: which sector the plant belongs to
- Accident level: from I to VI, it registers how severe was the accident (I means not severe but VI means very severe)
- Potential Accident Level: Depending on the Accident Level, the database also registers how severe the accident could have been (due to other factors involved in the accident)
- Genre: if the person is male or 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

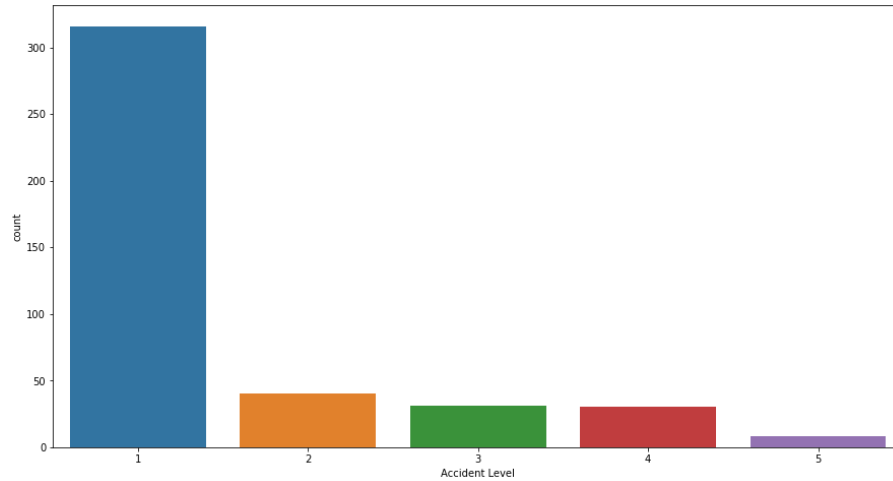
Summary of the Approach to EDA and Pre-processing

EDA

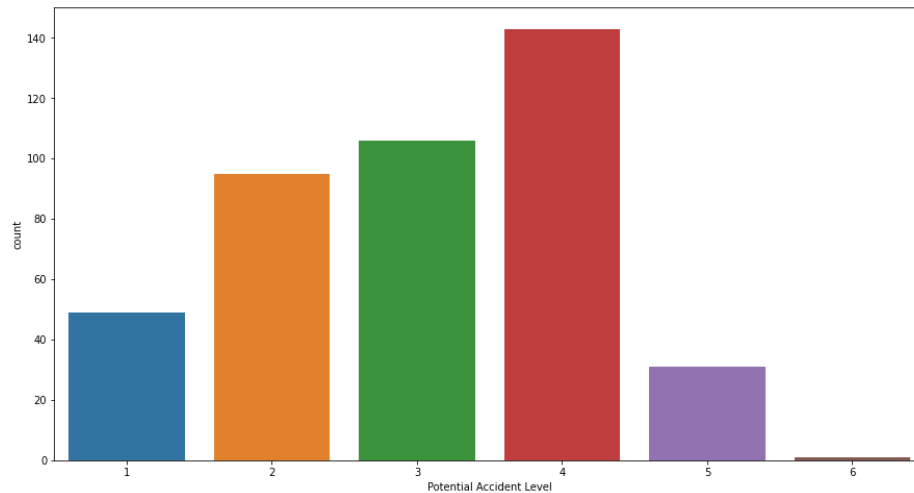
We have done univariate and bivariate analysis in multiple forms to identify what are the relationships if any between the data and what kind of distributions the data follows. Here are some of the Inferences about the data.

1. **The structure of the data:** There are 425 rows and 16 columns, out of which 7 rows are duplicates. Hence, the final data frame without duplicates has 418 rows and 16 columns.
2. **Missing values:** There is no missing data in any column and all the columns except Description are categorical although some of the extracted columns are certainly integers and the date column is converted to pandas DateTime object.
3. **Date:** The date column cannot be used as it is. Hence, we have extracted features from it - Year, month, weekday, week Of Year, day, Seasons (based on Brazil's season cycles since the data is from Brazil and surrounding countries).
4. **Description:** The description is a text column explaining the accident as text. There is a separate EDA and preprocessing done for description which is mentioned in detail later.
5. **Univariate Analysis Inferences:** Data is highly unbalanced in terms of most of the categorical columns.

- a. Accident Level 1 occurs a whopping 75% of the time.



- b. For Potential Accident Level, we see that the count of records increases with increasing levels up to 4 and then decreases. Thus, the Potential Accident level column seems to follow some amount of normal distribution.



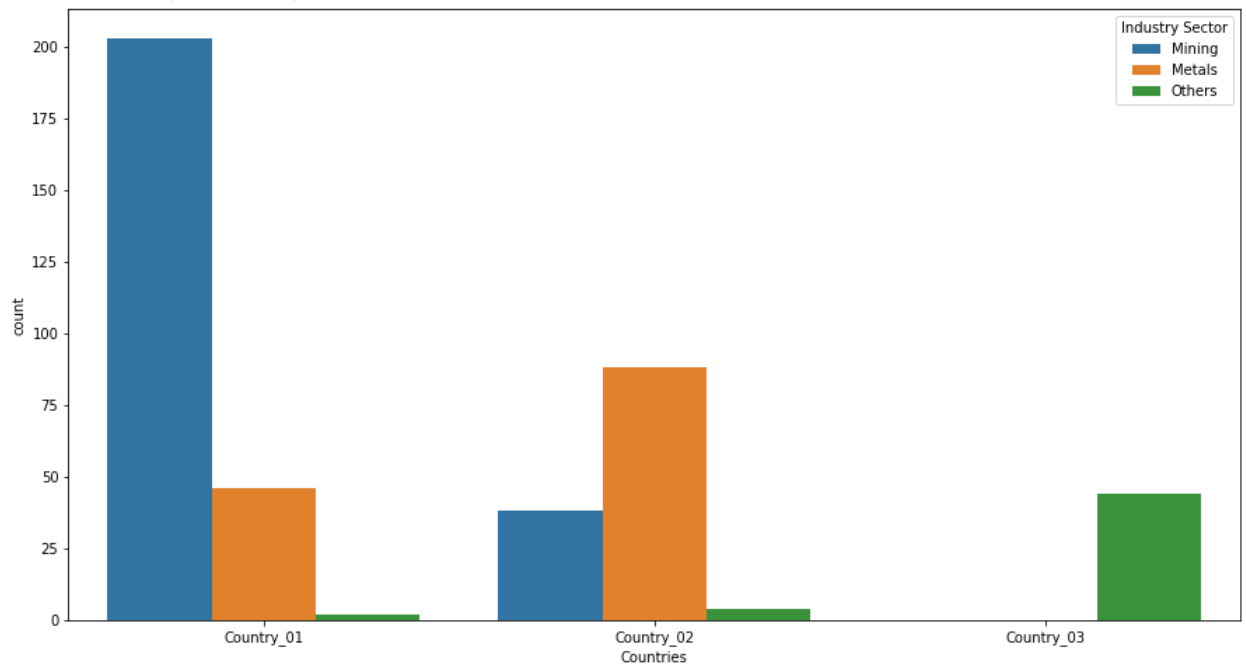
- c. Country Column has 3 distinct values.
d. Local has 12 distinct values with very little data in some of the categories, occurring in single digit.
e. The industry Sector has 3 distinct values.
f. The genre has 2 distinct values.
g. Employee or Third Party has 3 distinct values.
h. Critical Risk has 33 distinct values, with 25 categories having a single-digit count of records.
i. It seems the data is from January 2016 to July 2017. Other than that, the data seems to be well distributed between different months, years, days and weekdays.
j. The weekdays seem to be following a slightly normal distribution with more accidents occurring in mid-week rather than week start.

2. Multivariate Analysis:

- a. Correlation with Potential accident level
- i. Accident Level 0.502704
 - ii. Potential Accident Level 1.000000
 - iii. Year -0.004081
 - iv. Month -0.047853
 - v. Day 0.005721
 - vi. WeekofYear -0.033401

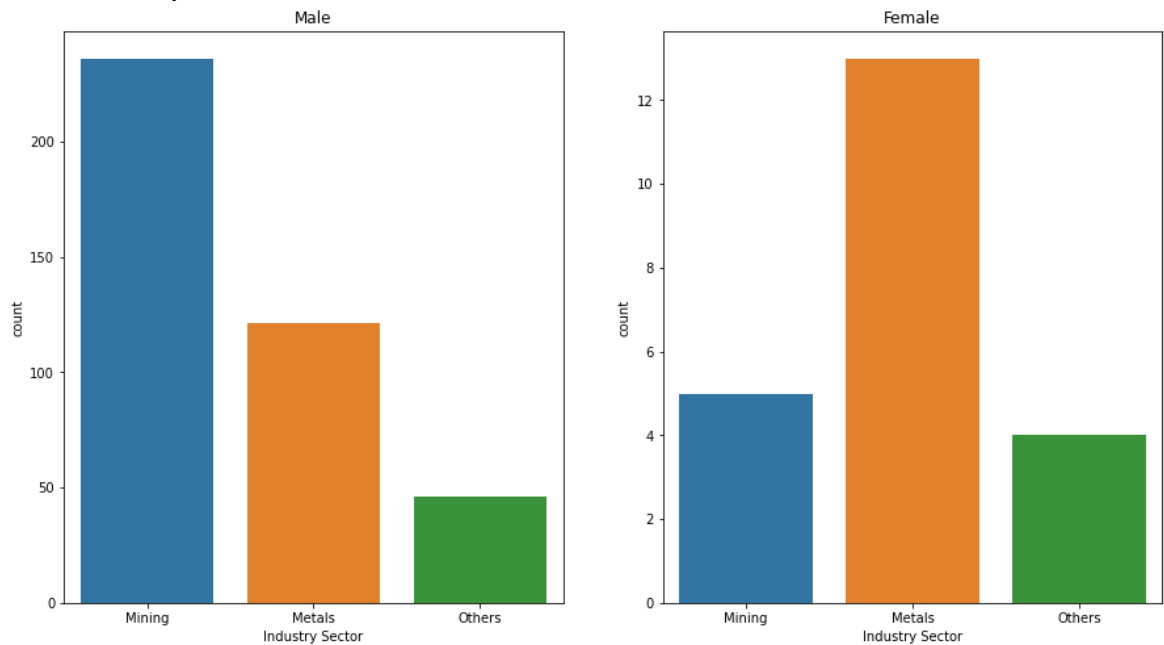
Potential Accident Level has strong positive correlation with Accident Level as expected but negligible or no correlation with the other numerical columns. These columns are basically categorical variables as opposed to true numerical variables and so correlation measure does not make much sense. For e.g. We do not expect the Potential Accident to rise/fall with increase in Month from Jan to Dec.

b. Country vs Industry sector



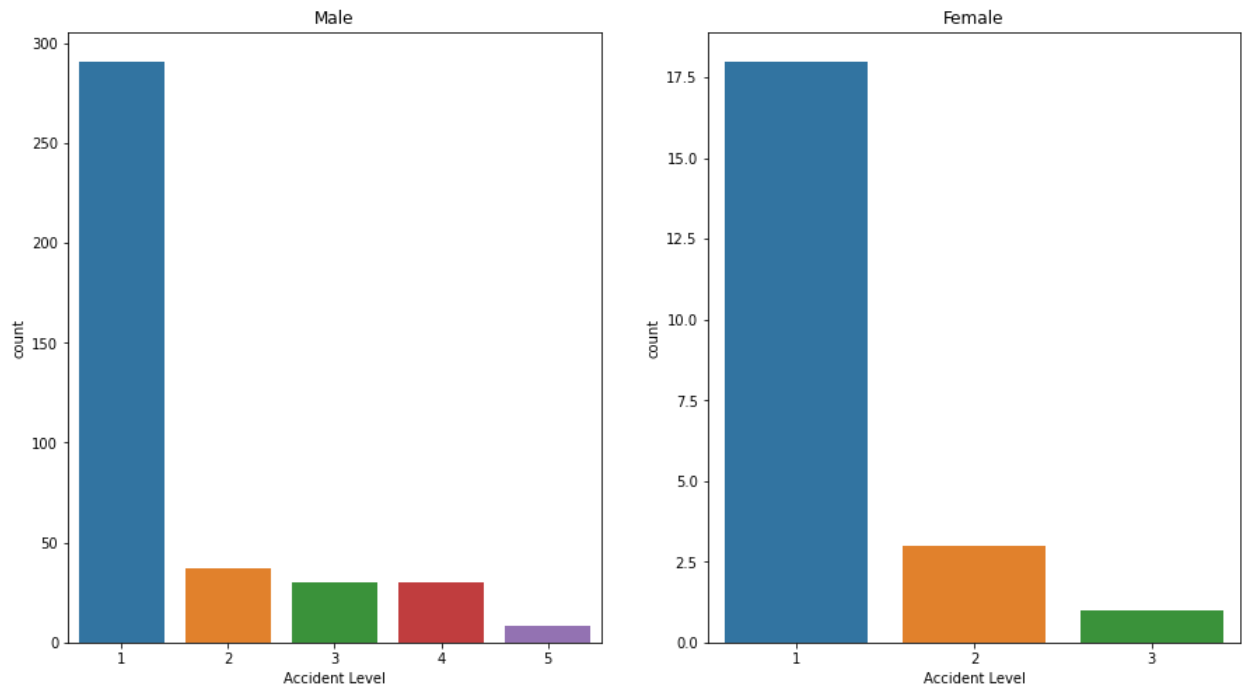
- i. **Inferences:** It seems each of the countries is focusing on one industry specifically.
1. Country 1 is focusing majorly on Mining
 2. Country 2 is focusing majorly on Metals
 3. Country 3 has only one category that has Others, and the remaining 2 countries don't have this sector information much but have it in minor

c. Industry sector vs Gender



- i. **Inferences:**
- ii. Male is majorly present in Mining
 - iii. Female are majorly present in Industry Sector

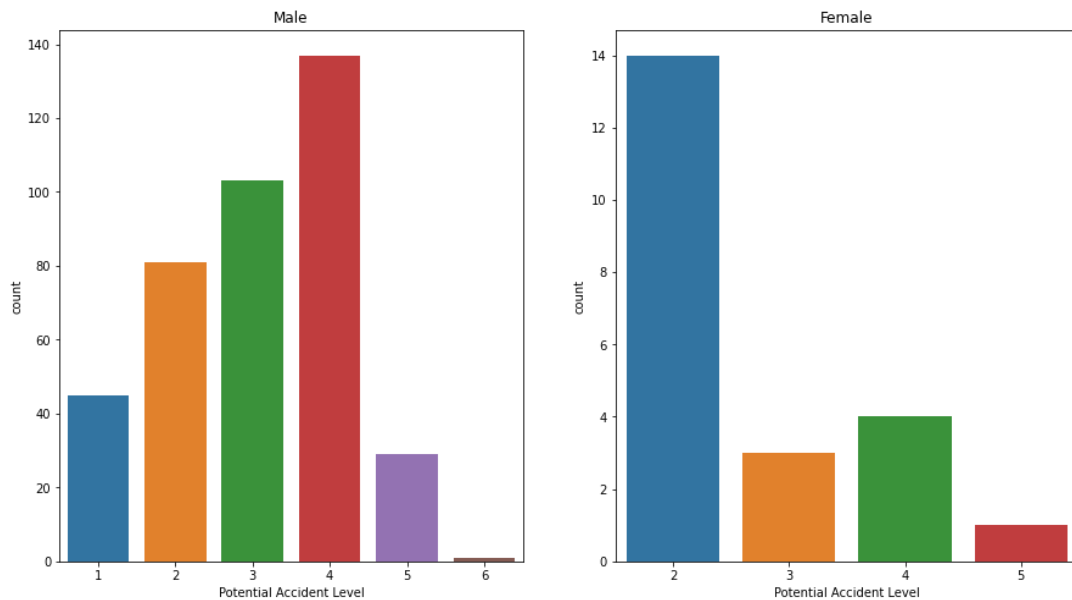
d. Accident level vs Gender



i. **Inferences:**

1. We see that the female has only 3 accident levels, which could signify that women are not put in high-risk jobs or they are generally more careful
2. The accident level percentage is more in 1 than 2 than 3 for female as well as male

e. **Potential Accident level vs Gender**



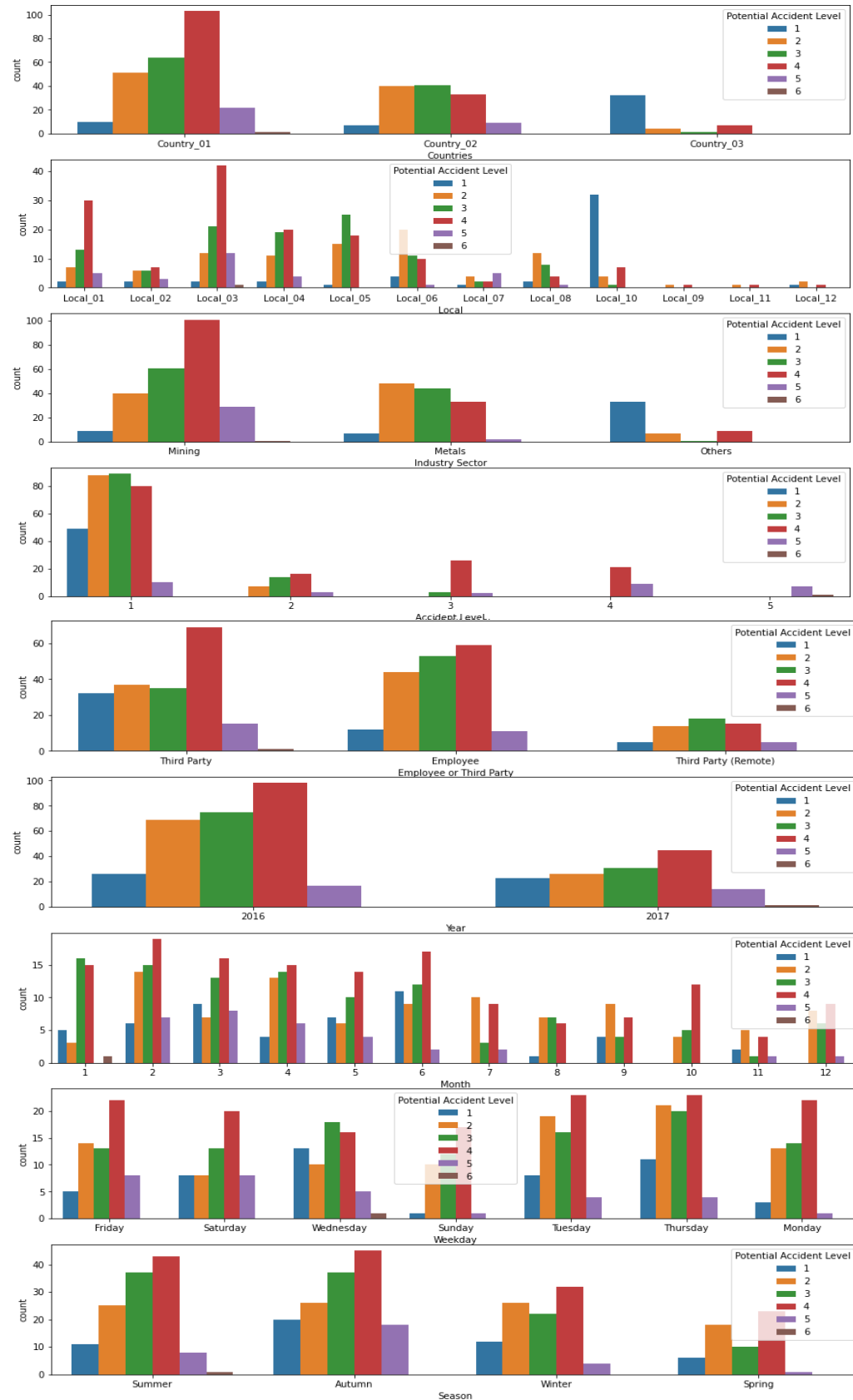
i. **Inferences:**

1. We see that the female potential accident level is decreasing as the accident level increases
2. for Male, the Potential accident level is increasing as the level increases
3. This concludes that gender is an important column for the model

3. **Bivariate Analysis of Potential Accident Level:**

- a. Let us do an analysis of the Potential Accident Level column with every other column. We have compared the Potential accident level with the below list of columns in our bivariate analysis to understand the relationship better:

- b. **Columns in comparison:** 'Countries', 'Local', 'Industry Sector', 'Accident Level', 'Employee or Third Party', 'Year', 'Month', 'Weekday', 'Season'

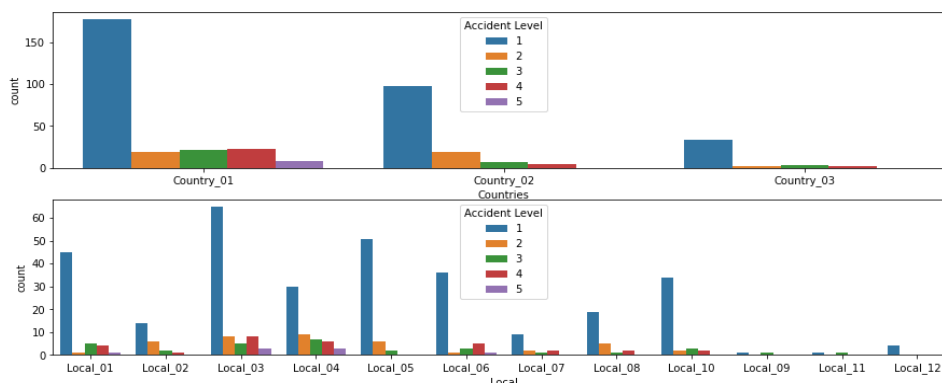


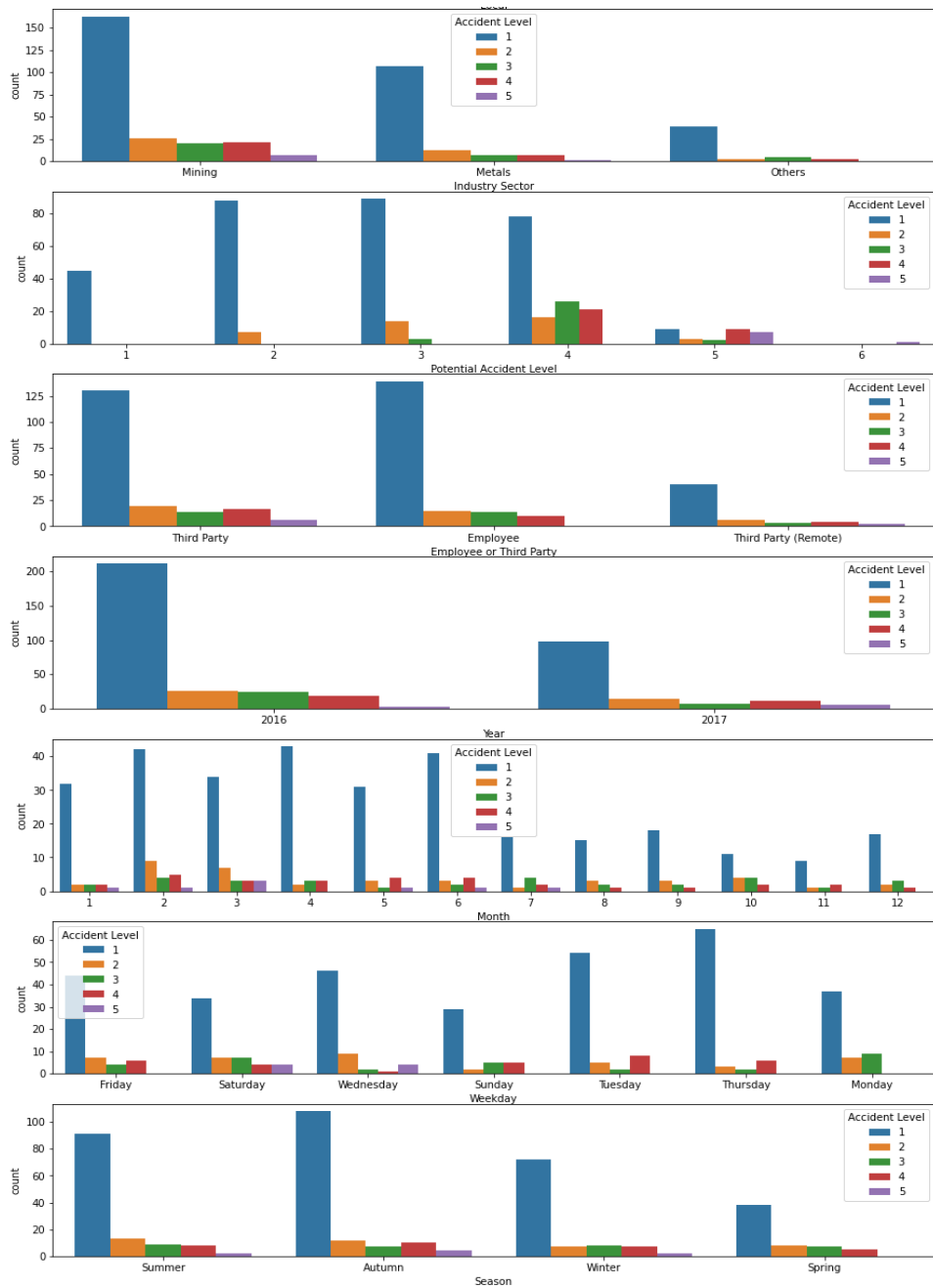
- c. **Inferences:** From the above graphs, we have made the below inferences on Potential Accident Level with respect to other columns

i. Country vs Potential Accident Level

1. Country 1 has Potential level 4 accidents more than other Potential levels. Also, we know that Country 1 has a higher mining industry sector, so we might see similar results in the industry sector too.
 2. Country 2 has Potential accident levels higher in the range of 2,3 and 4
 3. Country 3 has comparatively less number of accidents and majorly falling in accident level 1
- ii. Local vs Potential Accident Level
1. Most of the graphs look similar except for a few having peaked in one of the potential accident level categories which indicates that there must be a certain amount of correlation.
- iii. Industry sector vs Potential Accident Level
1. The Mining has majorly accidents falling in category 4 and country 1 which majorly has mining shows the similar graphs
 2. The metals follow the graph of Country 2 falling in the range of 2-4
 3. The correlation of Country with Industry sector seems to be really high and the way they are influencing the potential accident level is also the same
- iv. Accident level vs Potential Accident Level
1. Potential accident level is always greater than or equal to accident level which means the damage is controlled and stopped from reaching potential accident level.
 2. Most of the accident levels 1 are falling in the potential accident level categories of 2-4
- v. Employee or Third-party vs Potential Accident Level
1. Third-party has a sharp peak at potential accident level 4 showing they are more prone to severe accidents
 2. Employees are also having enough accident levels falling in the categories 2-4
 3. There are fewer data points for Third-party remote but their behavior is similar to Employees
- vi. Year vs Potential Accident level
1. The year 2016 and 2017 have similar correlations except that 2017 has a lesser dataset so lesser no. of accidents
 2. The year 2017 has one entry for potential accident level 6
 3. So by EDA, it seems that this column has very little correlation, during feature engineering we can look into this
- vii. Month vs Potential Accident level
1. The distribution of the graphs is slightly varying based on which quarter of the year it is in
 2. This justifies the creation of the season column
- viii. Weekday vs Potential Accident level
1. The distribution of Accident levels is highly varying with the weekdays and this column could be really useful and required while predicting the outcome
 2. For example, Thursday seems to be accomodating higher count of accidents from category levels 2-4
- ix. Season vs Potential Accident level
1. As we saw in the months, there is a quarterly correlation with the Accident level.
 2. Summer and Autumn have a higher count of accident levels compared with the others
 3. This column could be really useful while training the model

4. Bivariate Analysis of Accident Level:





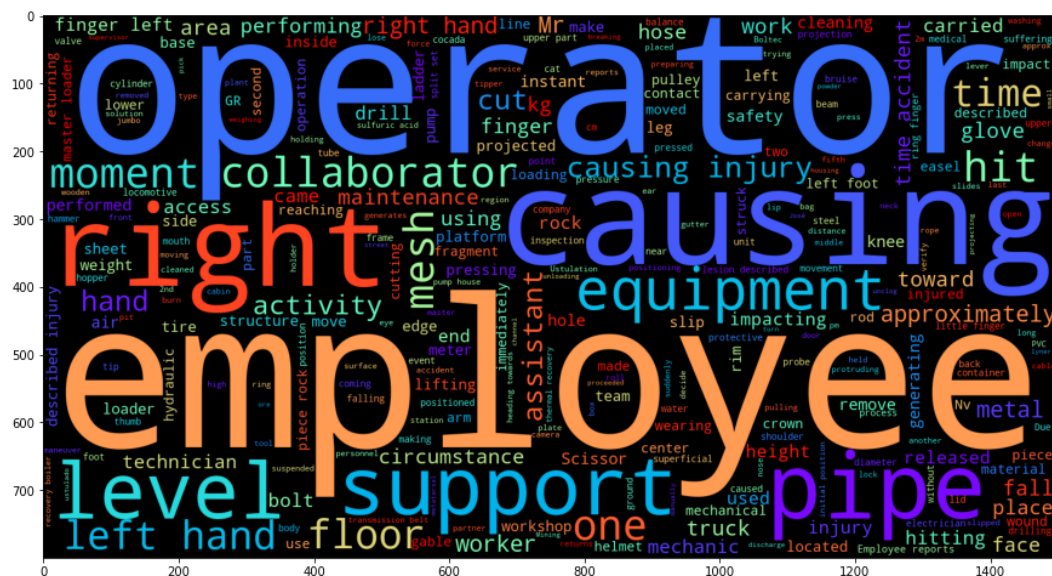
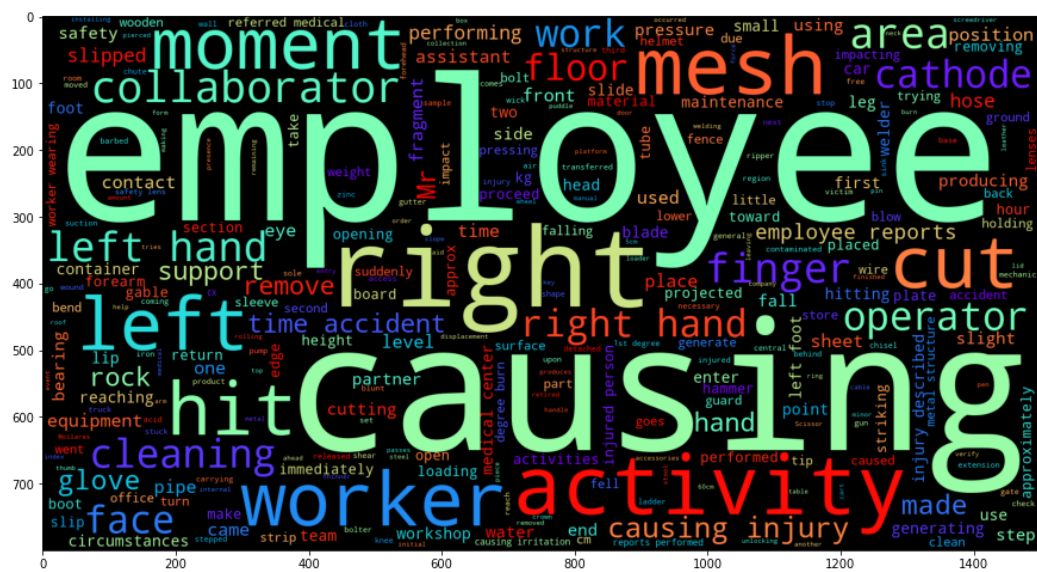
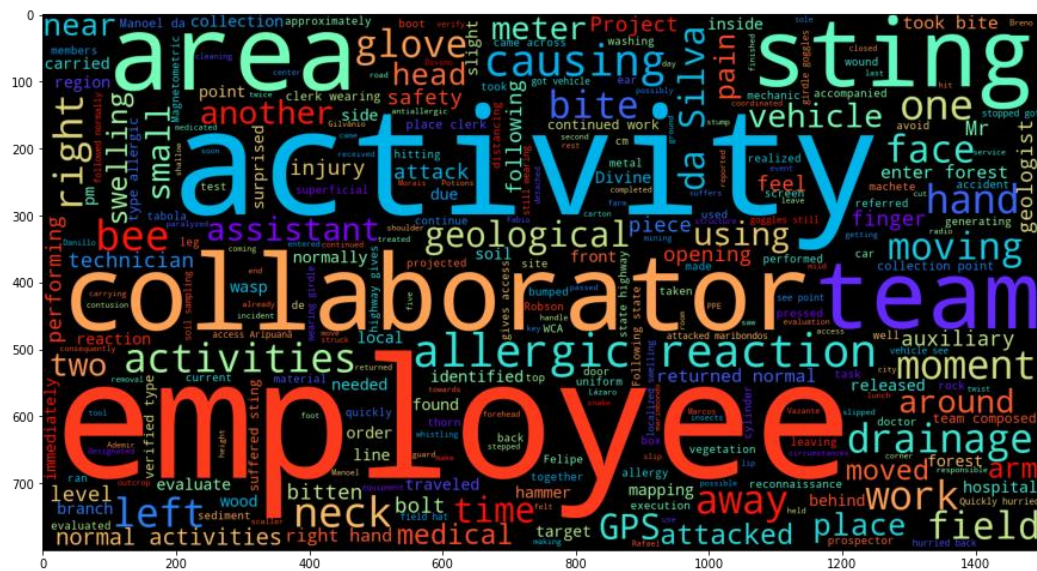
a. Inferences on Accident Level with respect to other columns

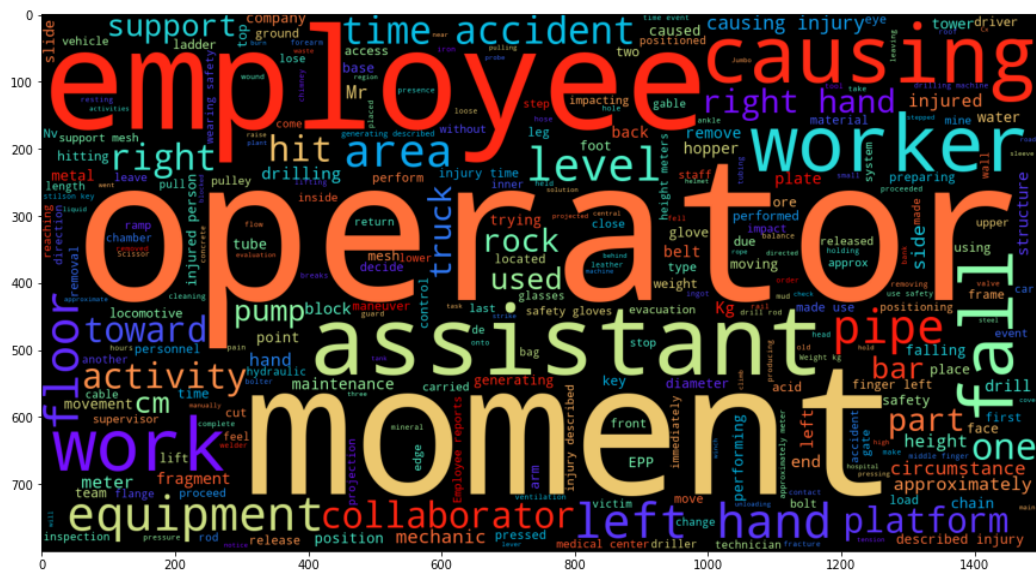
- Here, for all the columns, we see even in divided categories, accident level 1 is most frequent.
- The other accident levels are having almost similar distributions in each of the columns except for few differences which are not very clear in the graph because of the highly unbalanced dataset towards accident level 1

5. Word clouds

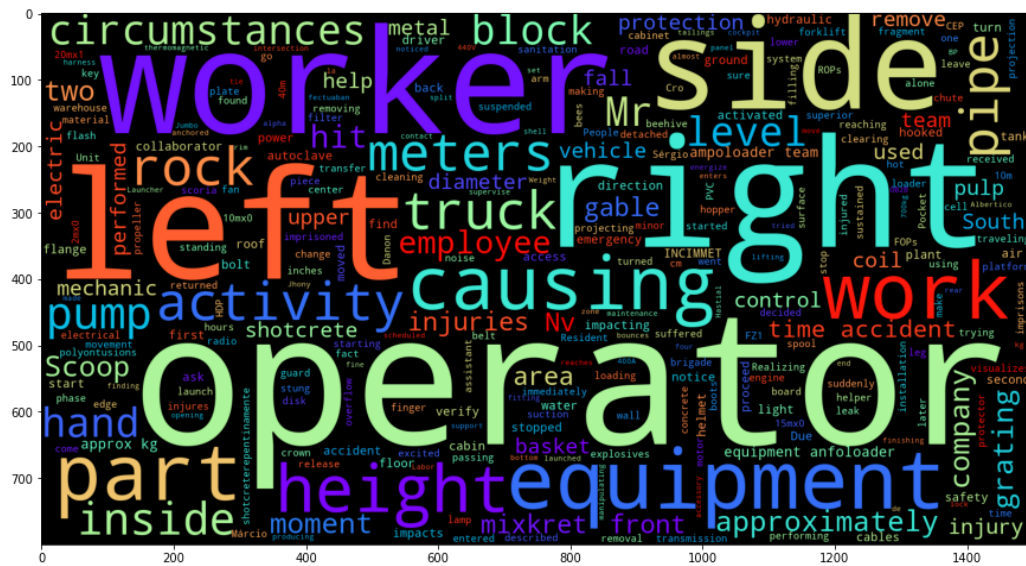
Let us get the most frequent appeared words in each of the potential accident levels

- Potential Accident Level - 1





e. Potential Accident Level - 5



f. Potential Accident Level - 6

Feature Selection:

We have 16 columns including the target column. Out of these columns, 6 are derived from the Date column namely Year, DayOfMonth, WeekOfYear, Month, WeekDay, Seasons. Based on the EDA we decide whether it is meaningful to keep or drop the features.

Features based on date columns:

1. **'Year'** cannot be a category as it keeps increasing with time unlike week/month. An entry with a new year value would not make sense to our model. Also there is no clear trend or pattern in the distribution YoY to keep it even as a numerical column. Hence it should be dropped.
2. As seen in the EDA the distribution of accidents based on **DayOfMonth** and **WeekOfYear** do not follow any pattern. The frequency of accidents have random fluctuations above and below a mean. Also, both columns have 31 and 52 distinct values respectively hence it will add a lot of dimensions and sparseness to our data. Information contained in these two columns will be better conveyed through WeekDay and Month columns. Thus these two columns will be dropped.
3. The distribution of Potential Accident Level (P.A.L) varies significantly w.r.t **'Month'**. For e.g (refer P.A.L vs Month chart) June has a high proportion of level1 accidents while month July has no level 1 accidents. July has a high proportion of level4 accidents. This column may capture external factors like change of season which could increase the chances of a particular type of accident to occur and is hence very meaningful to us.
4. When we club months together into **'Seasons'** we find that the distinction in the P.A.L distribution disappears or averages out (refer P.A.L vs season chart), which reduces the predictive power. The proportion of different Potential Accident Levels in the 4 seasons are very similar. Hence this column is dropped..
5. **'WeekDay'** (or Day of Week) gives us varying and distinct distributions of PALs which makes it helpful in prediction. For e.g. Sunday and Monday have a low proportion of Level1 accidents and Wednesday has a high proportion of Level 1 and 3 accidents compared to other days.

We keep 2 columns: Month and Weekday.

Non-date feature columns:

- Country, Local(ity), Industry Sector, Gender and Employee all have different distributions of Potential Accident Levels which means that these features are important and should translate to higher predictive power.

Observations:

We now have **1 numerical variable** (Accident variable) and **8 categorical variables**. The target Potential Accident level is a discrete numerical variable. It can also be taken as a continuous variable in case we use regression models. In our case, we will treat it as a categorical variable having 6 distinct classes (1-6).

- Accident Level can be treated as a numerical variable as it has a high direct correlation with Potential Accident Level. Higher AL means higher PAL.
- All other variables are categorical and will be one hot encoded.
- Critical Risk has too many distinct values and may not be suitable as a categorical variable for supervised learning models. It can instead be appended with the description text to be input into an NLP model.

One-Hot Encoding:

To be used as input with supervised learning classifier models like SVM, Logistic Regression, Decision Trees etc. the categorical variables will have to be converted to numerical form. Since none of these are ordinal variables we shall one-hot encode all 8 categorical columns namely:- 'Countries', 'Local', 'Industry Sector', 'Genre', 'Employee or Third Party', 'Month' and 'Weekday'.

Shape before one-hot encoding: (425, 9)

Shape after one-hot encoding: (425, 39)

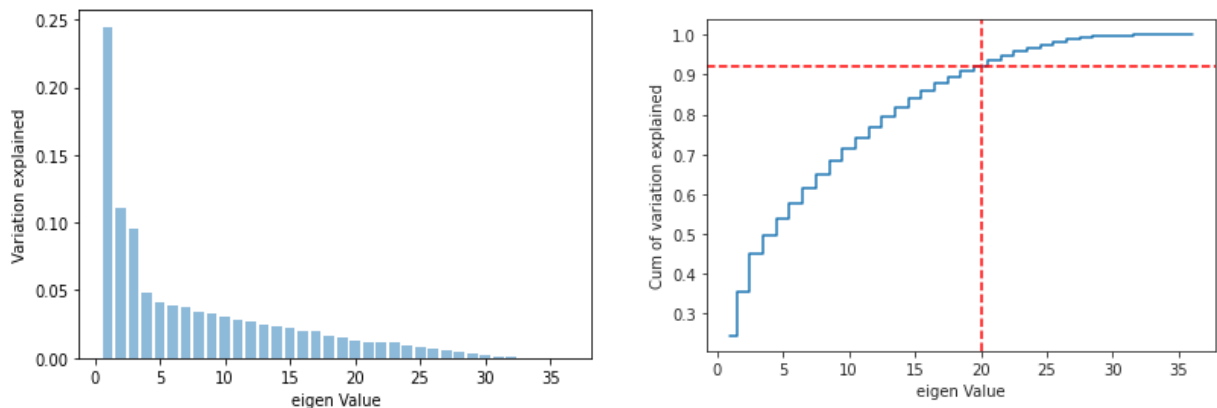
Dimensionality Reduction:

Case for PCA:

Consider the interaction between the independent variables Country and Industry [refer charts: (1) country vs industry; (2,3) bivariate Potential accident level vs Country, Industry]. Each industry is mostly restricted to one particular country. This results in the Potential Accident level distribution being very identical w.r.t both Country and Industry i.e both contain the same information with regards to the Target variable.

With PCA, the information containing all such interactions is transformed to a different feature space and only the components which hold the most information are selected. This results in reduced dimensions and also the suppression of noise along unnecessary dimensions.

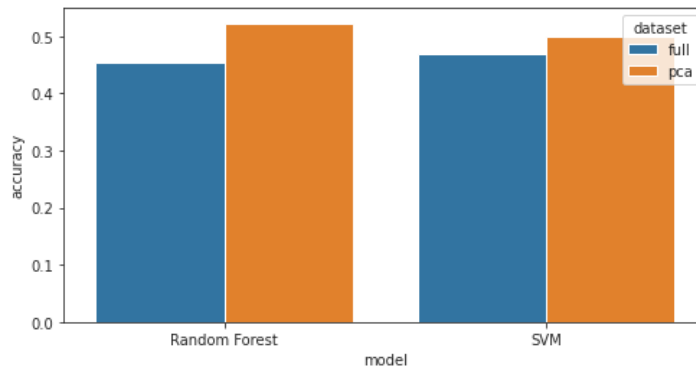
Analysis:



- PCA reveals that the first principal component contributes around 25% of the total variance.
- The first 20 components explain more than 92% of the variance. Thus we can reduce the number of features from 38 to 20.

PCA impact:

When tested with the Random-Forest model, the reduced PCA dataset produced better accuracy results (0.52 vs 0.45) and similarly also produced better results with SVM (0.50 vs 0.468) . Hence we shall use this reduced feature-set in all of our supervised models upstream.



Deciding Models and Model Building Based on the nature of the problem, decide what algorithms will be suitable and why?

Supervised Learning:

As we look to leverage the data using the predictive capabilities of machine learning, we know that there is no one size fits all approach. The machine learning algorithm we choose depends on the size, quality, and type of data as well as the project timeline and our overall goals.

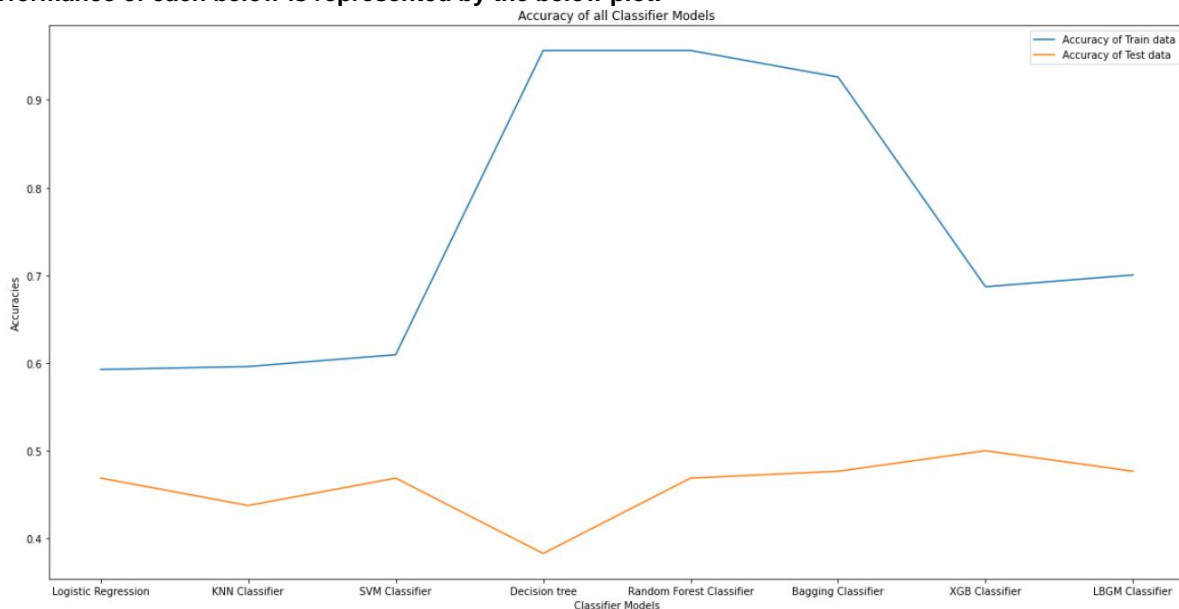
Here we are trying to predict the Potential Accident Level of a reported incident based only on non-text parameters. This can be treated as a classification problem and we can leverage various Multiclass classification techniques that fall under supervised learning.

Let's see the model working and performance of different models we have tried.

- Logistic Regression:** Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes. In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).
- KNN Classifier:** K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- SVM Classifier:** Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.
- Decision Trees:** Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. The two main entities of a tree are decision nodes, where the data is split and leaves, where we got outcome. In classification decision trees, the decision variable is categorical.

- **Random Forest Classifier:** As we know that a forest is made up of trees and more trees means more robust forest. Similarly, the random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.
- **Bagging Classifier:** A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples (or data) from the original training dataset – where N is the size of the original training set. Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.
- **XG Boost Classifier:** XG Boost or extreme gradient boosting is one of the well-known gradient boosting techniques(ensemble) having enhanced performance and speed in tree-based (sequential decision trees) machine learning algorithms. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree-based algorithms are considered best-in-class right now.
- **Light GBM:** Light GBM is a fast, distributed, high performance gradient boosting framework based on decision tree algorithms, used for ranking, classification and many other machine learning tasks. Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise. So, when growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms. Also, it is surprisingly very fast, hence the word 'Light'.

Performance of each below is represented by the below plot:



Results Summary:

	Model	train-accuracy	test-accuracy
6	XGB Classifier	0.686869	0.500000
7	LBGM Classifier	0.700337	0.476562
0	Logistic Regression	0.592593	0.468750
2	SVM Classifier	0.609428	0.468750
4	Random Forest Classifier	0.956229	0.445312
1	KNN Classifier	0.595960	0.437500
5	Bagging Classifier	0.932660	0.406250
3	Decision tree	0.956229	0.382812

- Of all the models, the XGBoost model performed the best on our test set giving us 50% accuracy.
- This is followed by LightGBM, Logistic Regression and SVM which give an accuracy of around 46-47%.
- Even though ensemble models like Random Forest, Bagging classifier and Decision trees performed very well on training data, these models were not able to reflect the same performance on test data.

We try to tune our model parameters using Grid search CV to select best parameter values.

Hyper Parameter Tuning using Grid Search CV:

In almost any Machine Learning project, we train different models on the dataset and select the one with the best performance. However, there is room for improvement as we cannot say for sure that this particular model is best for the problem at hand. Hence, our aim is to improve the model in any way possible. One important factor in the performances of these models are their hyperparameters, once we set appropriate values for these hyperparameters, the performance of a model can improve significantly.

Grid Search CV: GridSearchCV is a function that comes in Scikit-learn's model selection package. So, an important point here to note is that we need to have the Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters. As for its working, we pass predefined values for hyperparameters to the GridSearchCV function. We do this by defining a dictionary in which we mention a particular hyperparameter along with the values it can take.

Hyperparameters of different models:

```

Model name: Logistic Regression
Hyperparameters: {'C': 1.0, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ratio': None, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': 1, 'penalty': 'l2', 'random_state': None, 'solver': 'lbfgs', 'tol': 0.0001, 'verbose': 0, 'warm_start': False}

Model name: Naive Bayes Classifier
Hyperparameters: {'priors': None, 'var_smoothing': 1e-09}

Model name: K-Nearest Neighbor Classifier
Hyperparameters: {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 5, 'p': 2, 'weights': 'uniform'}

Model name: SVM Classifier
Hyperparameters: {'C': 1.0, 'break_ties': False, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 3, 'gamma': 'scale', 'kernel': 'rbf', 'max_iter': -1, 'nu': 0.1, 'probability': False, 'random_state': None, 'shrinking': False, 'tol': 0.0001, 'verbose': 0, 'warm_start': False}

Model name: Decision Tree Classifier
Hyperparameters: {'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_features': None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction': 0.0, 'random_state': None, 'splitter': 'best', 'verbose': 0}

Model name: Random Forest Classifier
Hyperparameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_features': 'auto', 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction': 0.0, 'n_estimators': 100, 'n_jobs': None, 'oob_score': False, 'random_state': None, 'verbose': 0, 'warm_start': False}

Model name: Bagging Classifier
Hyperparameters: {'base_estimator': None, 'bootstrap': True, 'bootstrap_features': False, 'max_features': 1.0, 'max_samples': 1.0, 'n_estimators': 10, 'n_jobs': None, 'oob_score': False, 'random_state': None, 'verbose': 0, 'warm_start': False}

Model name: XGB Classifier
Hyperparameters: {'base_score': 0.5, 'booster': 'gbtree', 'colsample_bylevel': 1, 'colsample_bynode': 1, 'colsample_bytree': 1, 'gamma': 0, 'learning_rate': 0.1, 'max_depth': 3, 'max_delta_step': 1, 'max_leaves': 31, 'min_child_weight': 1, 'missing': None, 'monotone_constraints': None, 'n_estimators': 100, 'n_jobs': None, 'num_parallel_tree': 1, 'objective': 'binary:logistic', 'random_state': None, 'reg_lambda': 1.0, 'scale_pos_weight': 1.0, 'silent': 0, 'sketch_reduce': False, 'subsample': 1.0, 'tree_method': 'exact', 'verbose': 0, 'warm_start': False, 'watchdog': 10}

Model name: LBMG Classifier
Hyperparameters: {'boosting type': 'gbdt', 'class weight': None, 'colsample bytree': 1.0, 'importance type': 'split', 'learning rate': 0.1, 'max depth': -1, 'min child weight': 1, 'min loss': 0.0001, 'min split loss': 0.0001, 'min split weight': 0.0001, 'min weight': 0.0001, 'n_estimators': 100, 'n_jobs': None, 'num_parallel_tree': 1, 'objective': 'binary:logistic', 'random state': None, 'reg lambda': 1.0, 'scale pos weight': 1.0, 'silent': 0, 'subsample': 1.0, 'tree method': 'exact', 'verbose': 0, 'warm start': False, 'watchdog': 10}

```

Performance of models after Hyper parameter tuning:

Performing model optimizations...

Estimator: Logistic Regression

Best params: {'clf__C': 0.1, 'clf__penalty': 'l2', 'clf__solver': 'liblinear'}

Best training accuracy: 0.518

Test set accuracy score for best params: 0.469

Estimator: Support Vector Machine

Best params: {'clf__C': 100, 'clf__gamma': 0.001, 'clf__kernel': 'rbf'}

Best training accuracy: 0.512

Test set accuracy score for best params: 0.477

Estimator: Random Forest

Best params: {'clf__bootstrap': True, 'clf__criterion': 'entropy', 'clf__max_depth': None, 'clf__max_features': 'sqrt', 'clf__min_samples_leaf': 4, 'clf__min_samples_

Best training accuracy: 0.505

Test set accuracy score for best params: 0.484

Estimator: XG Boost

Best params: {'clf__colsample_bytree': 0.5, 'clf__gamma': 0.3, 'clf__learning_rate': 0.01, 'clf__max_depth': 11, 'clf__min_child_weight': 1}

Best training accuracy: 0.522

Test set accuracy score for best params: 0.469

Estimator: LGBM Classifier

Best params: {'clf__colsample_bytree': 0.9, 'clf__learning_rate': 0.01, 'clf__num_leaves': 6, 'clf__subsample': 0.7}

Best training accuracy: 0.488

Test set accuracy score for best params: 0.453

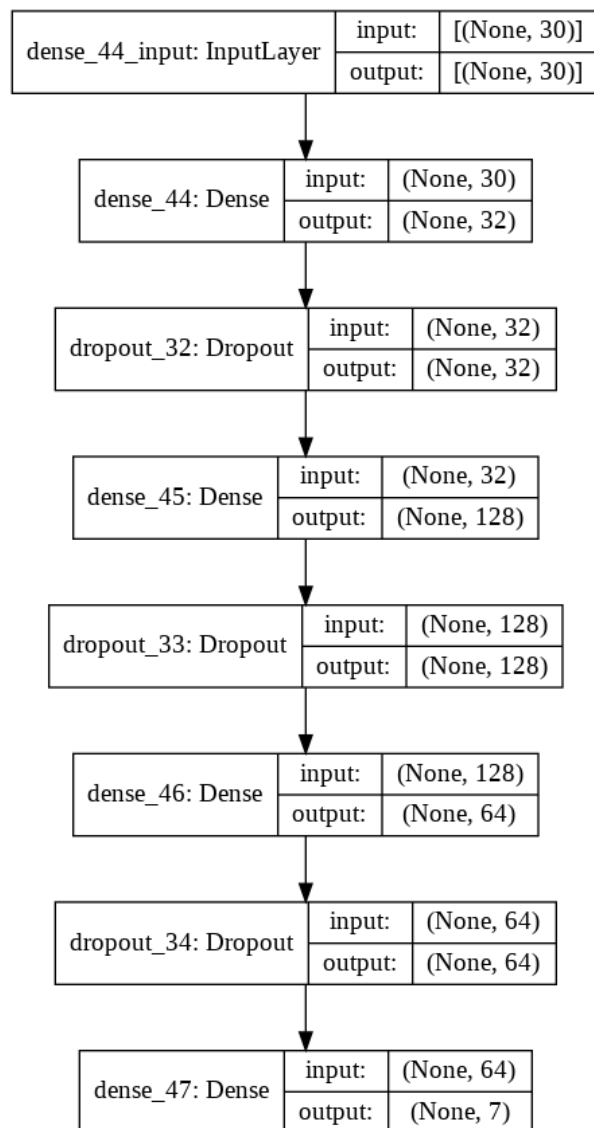
Classifier with best test set accuracy: Random Forest

Observations:

- We can observe that almost all the models selected for tuning have given similar performance in terms of accuracy.
- The highest accuracy score of 0.484 was reported for Random Forest classifier followed by 0.477 for SVM and 0.469 for XGBoost and LR.
- These scores reported are mean Cross Validated accuracy scores and hence are a bit lower than the accuracy score produced by validating on a static test-set. But they are a better representative of real world estimates.
- To possibly extract a few more percentage points in accuracy, bayesian optimization can be tested to get closer to optimal values.

Neural network models

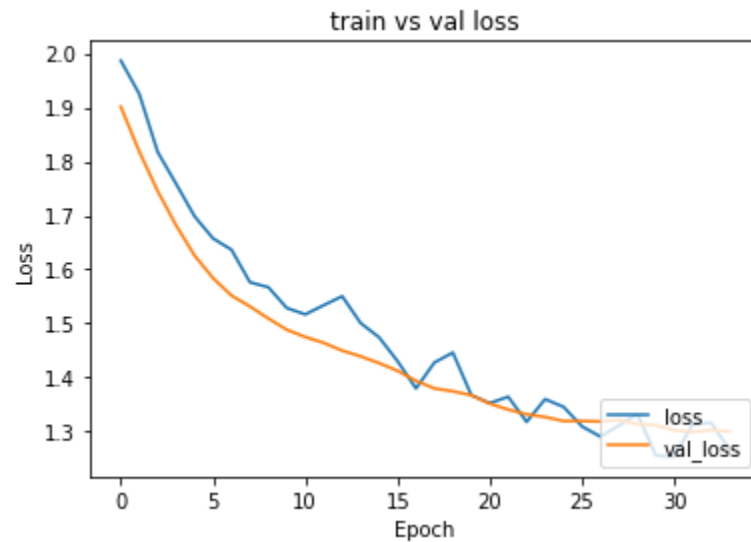
- Added a column called Description Sentiment Score to get some information about Description as well for the model.
 - Used PCA to get 20 features having 92% explained variance and 30 features with 99% explained variance.
 - Used Hyperparameter tuning to identify the best number of hidden layers and number of neurons in each layer as well as dropout percentages for each layer.
- a. The architecture of the best-tuned model based on accuracy using the above step is given in the below diagram:



- b. Accuracy vs epochs plot for training:



- c. Loss vs epochs plot for training:



- d. We got best validation accuracy of 50% for 92% variance features and around 52.06% for 99% variance model.

NLP Core Model

Data Exploration:

Data generated in different industries situated in different localities in different countries. Accident level and potential accident levels are target variables. Not much information is given on Potential Accident Level. Accident Levels are reported as it happened and based on accident level and few other factors, the potential accident level has been reported.

Current Industrial Practice:

Before we proceed further we can take a look at current industrial practices.

The risk matrix is defined as various levels of consequence and categories of probabilities. It is used to record anticipated, hypothetical scenarios that describe how a hazard could result in harm or damage. Looking at the data the Accident Level (AL) is generated based on reported injury and Potential Accident Level (PAL) is generated after engineering judgment, experience, past incidents and its consequences and QRA (Quantitative Risk Assessment)

techniques, and various engineering tools. As per Companies policy (or discretion) the severity is calculated and converted into words as follows

Probability Categories	Definitions
A (high probability) 1 incident per 3 months	Repeated incidents (dehydration, slip, non-avoidable operator error etc)
B (twice/year)	Isolated incidents (season change etc)
C (once/10 year)	Possibility of occurring sometime (equipment maintenance, turn around etc)
D (once/100 years)	Not Likely to occur (major equipment failure, loss of valuable material, natural calamity, catastrophic events)
E (Low Probability almost zero) (once/1000 years)	Practically impossible (temperature below 0 degree Celsius in Mumbai)

It has been observed that this particular data has 12 localities. And are associated with particular industry sectors and countries. As mentioned in the tree diagram below. Looking at the chart below we can see that Country 03 has only the “Others” industry sector associated with Local-10 and Country 02 has less data related to “Others” industry. It can be seen that Locality is an independent variable on which Industry and Country are dependent. It shows limitation on data.



Inferences:

1. Text corpus can give us the industry of the particular incident, even if it is not mentioned by the user. So, it can be used to fill missing data, if not provided by the user.
2. For other kind of guesses based on description, we have to follow secondary inferences or implications. It needs to be studied in more detail. In the preliminary studies, it is not clear if locality or country can be guessed based on the description alone, though it might be possible to implement it.
3. Text used for LSTM needn't have more than three or four units because none of the sequences larger than that are worth it. Further experimentation needed to verify this.
4. Backward propagation seems to be not that important because sequence of occurrence of events seems less important than entire description of the events. That's because sequential or chained descriptions (in-depth descriptions) tend to use less and less of unique words (because they will be referred to by pronouns etc.)
5. None of the Industrial descriptions show significant deviations from the mean word count difference, implying that events happening in sequence is just about average. So, a normal Network with a normal propagation would suffice.
6. It has been observed that the length of the description doesn't translate at all into the possible misjudgement for a country in the final model.
7. However, it is to be noted that studying the description given for proper formatting, and verifying threshold parameters of the model, will have to be integrated into the model during the training phase itself. So, for example if the bot is used for anything else other than its intended purpose, it should simply choose not to respond, or politely deny the request.
8. It has to be taken care that when description is given by the user it is enough: assurance that the description is enough to make a proper guess with proper probability becomes extremely important. Hence, a mechanism of asking for a more detailed description if the given input is insufficient for probability par score is required. Hence the implementation is to be in that regard.
9. Other ideas like Accent detection, Named entity recognition to identify the Nouns and from there guessing a country etc. have been deemed impossible with the scarcity of the data at hand.

Hypothesis:

Nature of Hypothesis is complex (Relation between two or more independent variables and two or more dependent variables (Accident Level and Potential Accident Level are dependent variables, Description, Locality etc. are independent variables, creating a non-linearity in the data).

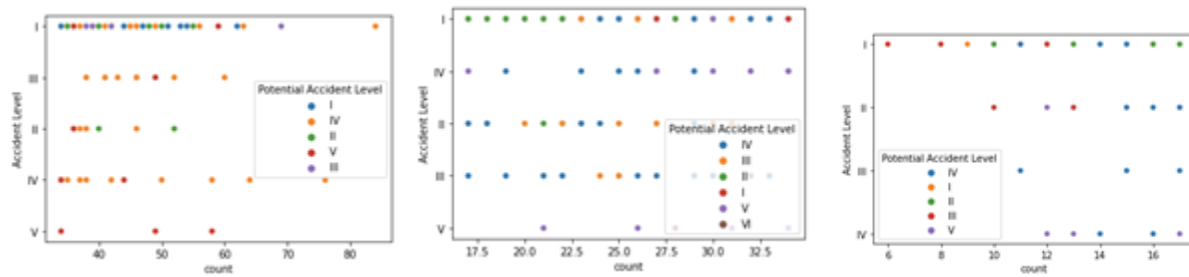
For any model to work, there should be a discernible pattern in the data, though not readily quantifiable.

1. Simple Hypothesis (One independent Variable and One dependent variable): Based on the particular words and their probability distributions in the Description Column of Dataset accident Level can be determined; A language model can be constructed to handle unseen data or typo errors; A language specific language register can be created.
2. Complex Hypothesis: Based on Description alone Accident Level can be determined; based on Locality, Description, Critical Risks and Accident Level Potential Accident Level can be determined; Based on Accident level, with the given accuracies of the model a supervised model can be constructed.

To prove these hypotheses we are testing the usage of the NLP model and LSTM technique.

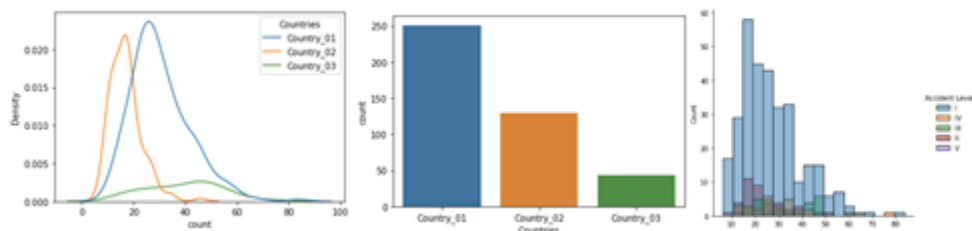
Evidence:

Word count



The above pictures indicate the normalized structure of description length with respect to accident level in the foresight of potential accident level. The order of the pictures is Q1,interquartile,Q3. This supports the language model hypothesis and is independent of country or locality. More information can be found in the accompanying notebook.

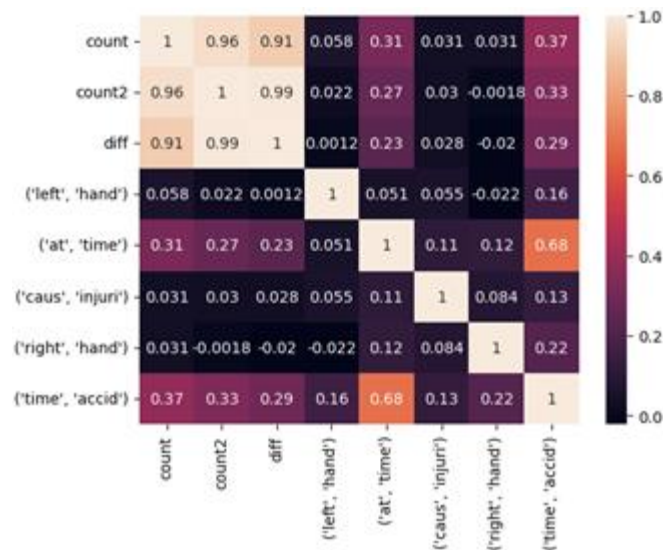
Country



These demonstrate that language description is independent of the country, and the variations observed here are because of the variations in data and not the description. The description length has got nothing to do with the level of accident assessed or the potential accident assessed.

Language models

Though not included here, it has been found that the Bigram language model is feasible to handle the text data. Trigram and skip-gram model is observed to be not feasible in the current scenario. More details are mentioned in the accompanying notebook. An example of the Conditional frequency distribution of Bigrams, for accident level 1, showing a healthy conditional frequency distribution indicating the models and their feasibility.



Corpus Data

It has been found that certain exclusive corpus is possible based on the type of industry that is under consideration and their corresponding probabilities are given as below for one industry. This is evidence enough to undertake the language model for classification and be confident about the handling of similar data with similar boundaries that is not in the corpus

```

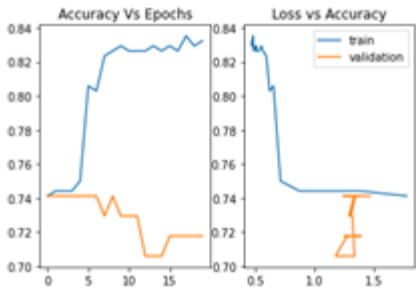
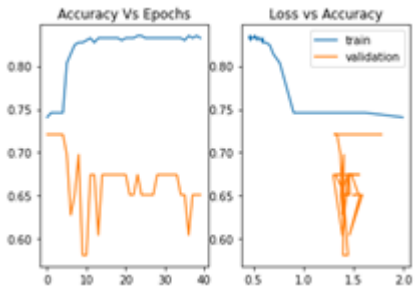
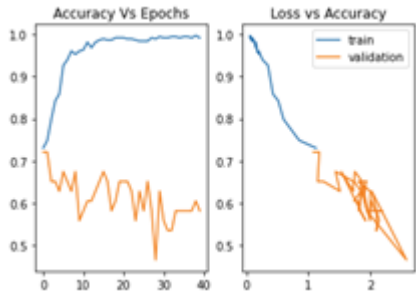
These are the Exclusive words in Metal Industry
cathod      14
boiler      6
thermal     6
degre       6
ingot       6
..
stepladd    1
command     1
stood       1
kiln        1
brush       1
Length: 227, dtype: int64

```

Models Tested

An Overview of the models tested, that are found feasible and appropriate for the given confinements and considerations of the data are given below, while the important ones are elaborated.

	loss	accuracy	val_loss	val_accuracy	model
0	0.623329	0.856266	1.215647	0.698529	Simple LSTM
1	0.666894	0.842147	1.567490	0.640116	Simple BiLSTM
2	0.645883	0.819830	1.451042	0.665116	Attention LSTM
3	0.402029	0.898691	1.492306	0.656395	Multi-Head Attention
4	0.180592	0.959359	1.836161	0.613372	Attention Conv BiLSTM

<p>Simple LSTM model</p>		<p>Didn't carry the potential. But found to be of best fit to the data. It is positively responding and reducing the loss.</p>
<p>Self-Attention LSTM</p>		<p>The amount of loss is topped off. So, in future scaling of data, this might not show proper trend. The good point is that training happened pretty quickly.</p>
<p>Self-Attention with Convolution, LSTM and Bidirectional</p>		<p>Faced the same issue as Self-Attention mechanism. The Good point is the reducing loss in train. So, model can be improved with more data. So, it can be readily deployed and retrained.</p>

Note: Two more models i.e. Multi-headed Attention and Bidirectional have been tested too, that can be verified from the notebook that accompanies this report.

Conclusions:

There is no specific pattern, highly indicating that there is a complete mismatch between the training data and the validation data, since it is much more susceptible to the information variations and non-normalized data. Thus indicating that splitting the data is causing irreparable damage and loss of valuable information from the training data set.

Final Report:

It has been clearly observed across both the NLP Core model and the classification model, both of them suffer with the same problem viz. lack of data. The problem requires approaches and ideas as described below. But it is to be noted that any further exploration of choices to include outside data has been voted out of favour by the team. So, the maximum limit for various scores are expected to be not much beyond what is already listed in this report.

Steps to improve the NLP model:

The Following steps are being taken

1. Synthetic data generation is a difficult task. Attempts are being made to generate text from within the language corpus to improve the model. Feasibility and applicability is to be checked.
2. Maximum description length after some cleaning process is tuneable parameter. To improve the accuracy of the model it is being tinkered with. Embedding dimension is tuneable hyper parameter, which is also being explored with to improve the model.
3. Due to limited data and hardware capability, model improvement requires a different approach. It is thus being explored on overcoming those difficulties.

Model Improvement and Further Explorations:

Below are a few ways we could use to improve the model performance.

Ensembling: As we know that the NLP model is using only the Description column and supervised learning models use the rest of the columns except the description and these two models have an exclusive understanding of the dataset. So applying a few ensemble techniques could help us in improving the overall accuracy of the model.

Two-stage classification: We can also use the confidence levels of the NLP model to decide if the prediction goes through the supervised models and compare the accuracies. Two-stage classification models could be helpful.

Given the above, a chatbot is not complete without proper deployment and real-world interfacing. Hence, the attempts are being made as of filing of this report. The attempts include, but not restricted to the following:-

1. A Graphical User Interface via a HTML Forms that lets us train the chatbot on custom data.
2. Methods, involving, but not restricted to, using flask app serving as a chatbot server on a Container as a Service Cloud server.
3. Localized user interface using tkinter that lets us automate the whole process involving NLP and chatbot building.
4. Language models with auto-rectify and common problem addressing are being considered as at the time of filing of this report. Feasibility and accuracy of those remains to be seen.