|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | Capstone Project-Final Report on Chatbot | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | AkibBharathChandanKomal |  | | | |
|  | | | |  |  | | | |
|  | | |  | | |  | | |

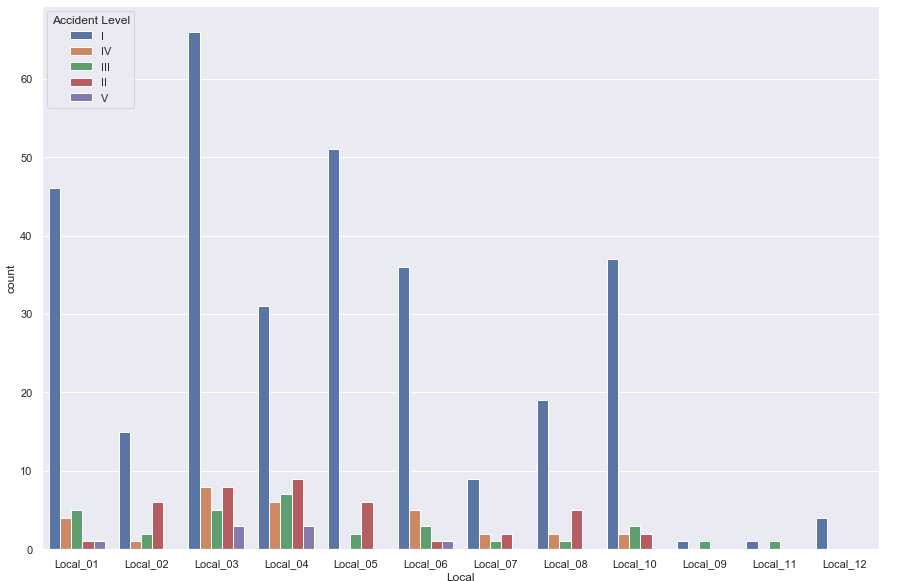


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | |  | |
|  | Mentor- Rohit Raj | | |  | |
|  |  |  |  | |  | |
|  | Problem Statement Summary: One of the biggest industries in world suffering with the employee’s accident and major injuries and sometimes employee’s also die in such environment. Our task is to create NLP based Chatbot so that professional get details of safety risk as per accident description. | | |  | |
|  | Dataset & Observations Dataset Columns Overview:   * *Data*: timestamp or time/date information( appropriately renamed to date) * *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) * *Description:* Detailed description of how the accident happened. * *Potential Accident Level*: Depending on the Accident Level, the database also registers how severe the accident could have been (due to other factors involved in the accident) * *Genre*: if the person is male of female(appropriately renamed to gender) * 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 |  |  | |  | |
|  | Observations based on the dataset:   1. Dataset has 425 rows and 11 columns   //   1. Mining sector is impacted by all level of accident, most accidents have recorded with minimum severity level for mining sector. | | | |  | |

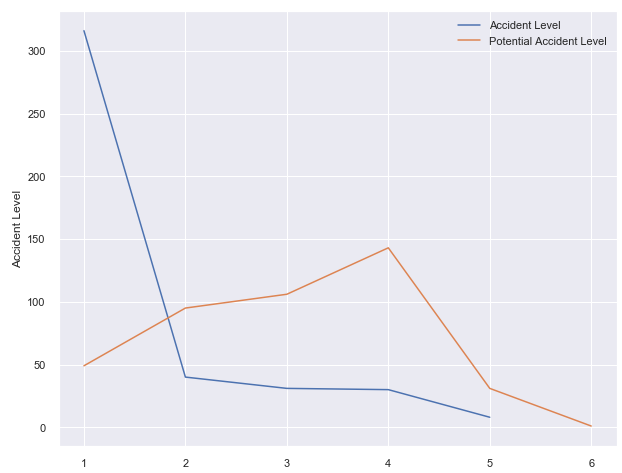
1. Men are involved in all level of accident where female are involved in 1st and 2nd level only which is minimal accident level.

//

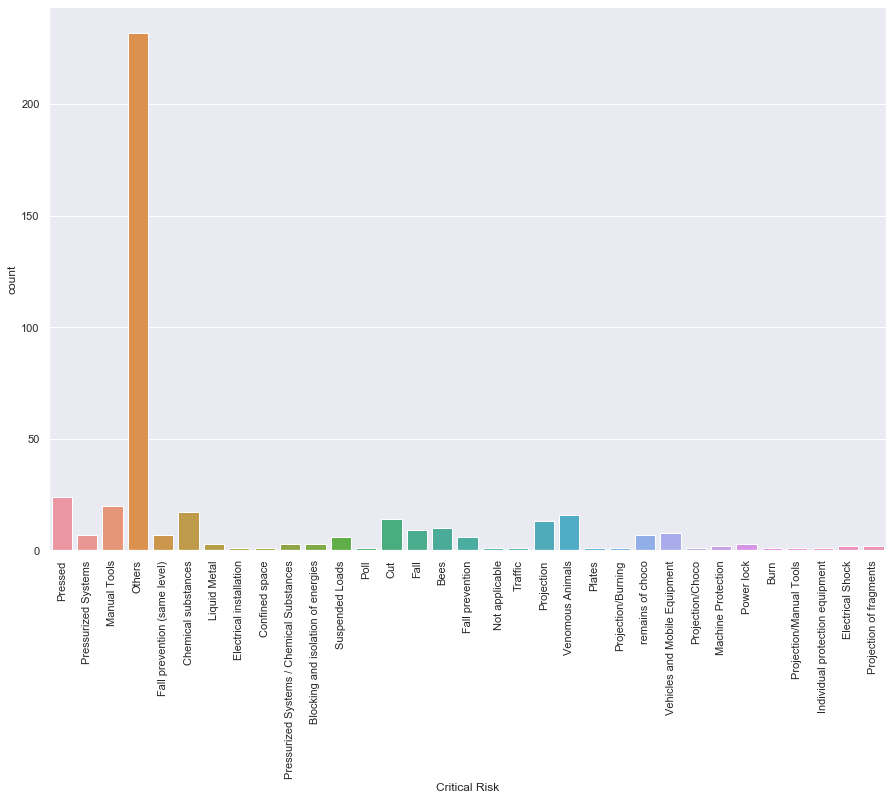
1. Most of accident causes is not defined properly as it is selected as OTHER but few major accident causes are “Pressed”, “Manual Tools”, “Chemical substances”, “Cut”, “Bees”, “Venomous Animal”
2. City number 3 is involved in most of accident where city number 12 is involved in minimum accident.
3. City number 3 and 4 are involved in all type of accident level.



1. Most and all type of accident levels are recoded in mining sector followed by metal industry sector.
2. We can observe that most of accident have minimal accident level but high potential risk.

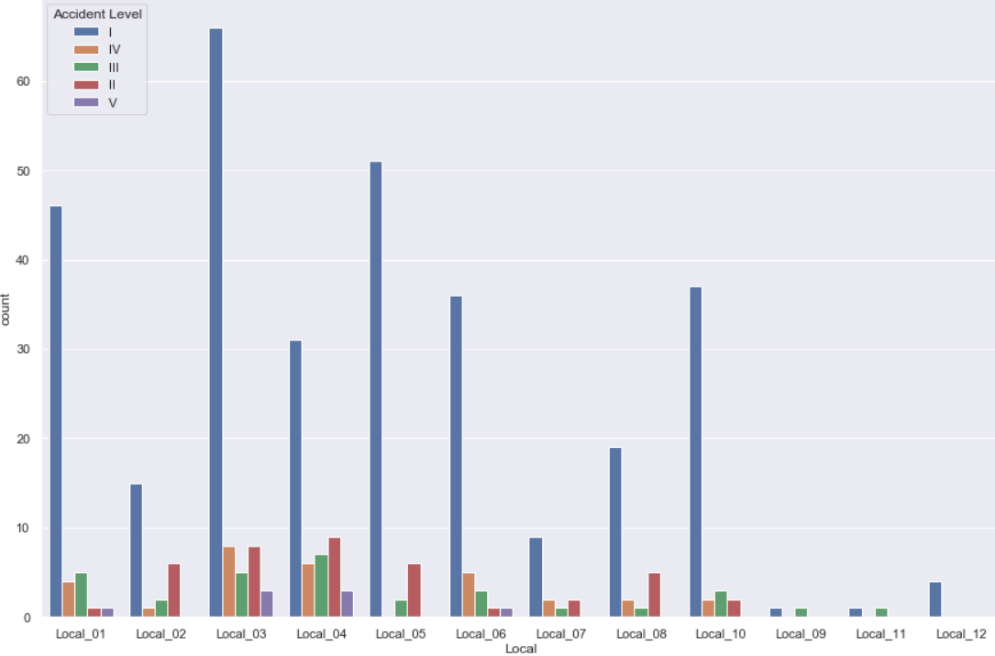


1. There are high number of accidents have been recorded in 2017-07 and 2017-07



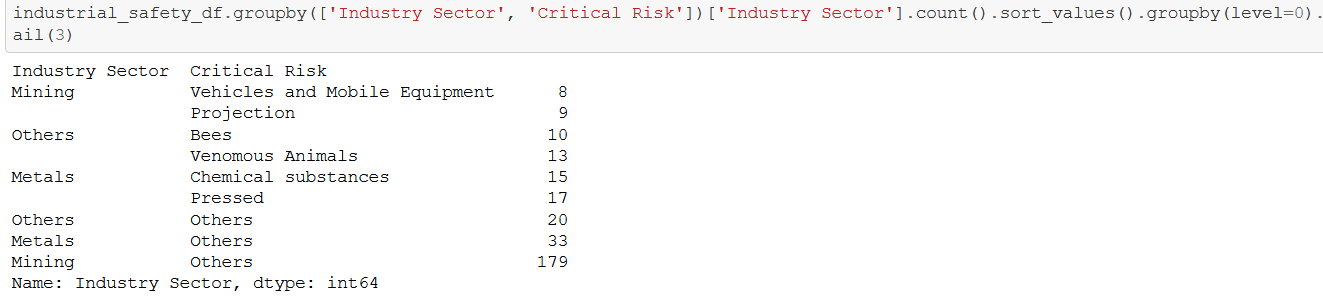
* This is the details of critical risk counts as per the data set. Most of them happened where it’s

described as *others*.



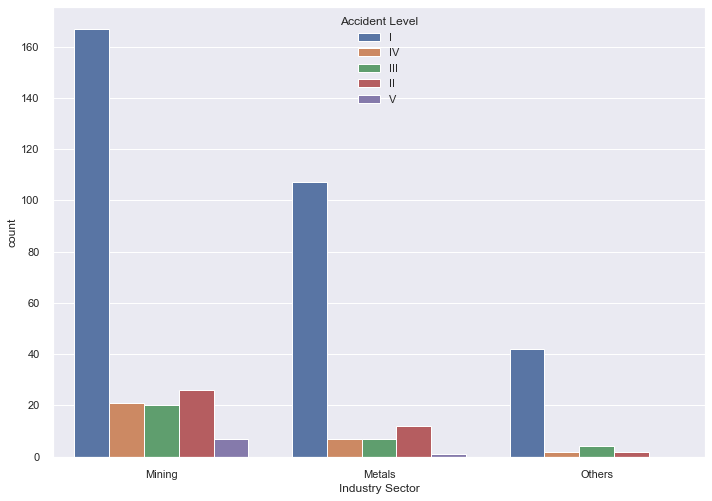
* For each local level-I accident has happened most.
* Local\_03 has reported most of the accidents.

Here is another view of Critical risk over each industry:



* ***Mining***industry reported most of the risk

We will take another look on how risky Mining industry is:



* So, *Mining* is the riskiest industry and followed by *Metals* and *Others*.
* Also, all type of accidents count is much higher in Mining as compare to the rest.
* In Metal only Accident Level I high but others are significantly less.

1. The dataset is rich with data containing from 2016-2018. It has three years of data as per the below time series.



1. Word cloud depiction: Employee, left, causing, hand, operator, activity are most frequent words in the dataset



|  |  |
| --- | --- |
|  | Data Cleaning:  * We updated the column names based on contextual relevance: data was updated to date and Genre was updated to Gender. * We did not have any null value. * Removed irrelevant data (Numbers and Punctuation) and converted text to lower case and tokenize data. * Removed stop words. * We have performed stemming and lemmatization. Stemming lead to a few absurd words being introduced in our data , we then utilized lemmatization which led to better results. * We removed all unwanted tokens which were lesser than two characters in size. * After applying lemma and above mentioned steps, the final data was added in the *Updated Description* column.  Data Encoding:  * Bag of Words**:** The text is represented as the bag of its words, disregarding grammar and even word order but keeping multiplicity.      * TF-IDF encoding: erm frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus      * GLoVe encoding: GLoVe stands for “Global Vectors”. It captures both global statistics and local statistics of a corpus, in order to come up with word vectors      Briefings:  * In our implementation we have noticed for LSTM related model , GLoVe was more efficient than using bag of words , and for Fastext we used TF-IDF. * As we are dealing with text data here, we have considered cleansing the text data ‘Description’ so that it can be encoded properly and fed to the machine. * We have chosen *Description* column as the most salient feature as it determines the potential accident level and the accident level. * Critical Risks also could be chosen as a target but as the dataset doesn’t have a good distribution for the 33 unique values ,Hence , we choose Potential Accident Level as the target. |

|  |  |
| --- | --- |
|  | Walk Through the Solution:  * The goal is to predict the potential accident level. As the very first step after getting the data preprocessed and encoded the text data with TF-IDF ,GloVe,BOW. * As a next step we tried building a random forest classifier on the text data encoded. * As random forest failed to rise up to the expectation, we have tried with the text data with stop words and also with BOW encoded. But that didn’t change the result much. We still were stuck with a poor score. * By using hyperparameter tuning also nothing changed much. * The failure of the previous stages had led us in search of alternate models and we came across FastText. * In the next few steps, we have tried building up a solution based on FastText. First, we preprocessed the text as FastText requires. * In the process of building FastText we did split the data with 80:20 and came up with a model initially with 75% of accuracy. * Next, we implemented uni directional and bi directional LSTM, GRU ,Leaky Relu with GLoVe encoding, with batch normalization and dropouts. * In order to optimize and reach the target score with LSTM we created dummy matrix of size same as target column label encoding to divide data into test and train properly. * With the help of previous steps, we got a descent score of 75%. * But our target was to create a model of 90% or more. As LSTM after tuning and optimizations didn’t give us the expected outcome, we switched back to FastText. * As the final step we did some hyper parameter tuning on the FastText by changing the n-grams and epochs, we finally reached there. * At this point we had all our model ready and evaluated. As FastText performed best on the test data we have chosen the FastText as our model to go forward. * From here, we worked on different APIs and finally able to integrate our model with an interactive web application.   So, to summarize, as we approached different ways to solve the problem, we finally found it through FastText after working with multiple models mentioned above. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Model Evaluation:   Below table gives us a clear picture on how we decided our final model.   |  |  |  | | --- | --- | --- | | **Model Name** | **Accuracy** | **Comment** | | Random Forest Classifier | 41.40% | We got very bad accuracy with RFC | | FastText |  |  | | FastText | 99.70% | We got very good accuracy with FastText model with epoch of 300 | | Bidirectional LSTM | 73% | We got decent accuracy with bidirectional LSTM(both with and without dropout and batch normalization)but there is huge difference between accuracy and validation accuracy | | LSTM | 67% | As compare to above model accuracy is low and same is above huge difference between accuracy and validation accuracy (both with and without dropout and batch normalization) | | LSTM with LeakyReLU | 73% | as compare to above model accuracy is low and same is above huge difference between accuracy and validation accuracy | | GRU | 0.9% | Both accuracy and validation accuracy were very low |  * We decided FastText as the final model. This is a supervised classifier and is an open-source, free, lightweight that allows users to learn text representations and text classifiers. It works on standard, generic hardware. * In order to preprocess FastText requires ***\_\_label\_\_*** on each target value to train a model. * To improve the performance, we did some hyperparameter tuning with ***wordNgrams***and ***epoch****.* * As the dataset was small and it’s difficult to up sample the text data, we did validate the performance of our model on both the test and with the whole dataset. It performed well with very high precision and recall.  Model Details:  * **Preparation**: * Fastest expect “\_label\_” in column, added same in target column. * Saved data as text file * Splitting data into test and train in ratio of 20:80 * **Model:** * fasttext.train\_supervised('fastext.datas-fasttext-train.txt', wordNgrams = 2, epoch=75) * Score: (340, 0.9970588235294118, 0.9970588235294118)  Model Success: Here is the classification report attached which clearly indicates that FastText is the right model to choose.    It’s clear that our model has performed very well in case of positively identifying all categories of target label. |

#### 

#### Comparison to Benchmark:

The target was to reach 90% with good precision and recall with a model. With FastText we have achieved more than that and even though for \_\_label\_\_III precision could be bit better but as it has performed very well with others, we can wind up saying FastText has very well set the standard here and may be exceeded the benchmark we set as a team.

#### Chatbot: A chatbot is a software application used to conduct an on-line chat conversation via text in lieu of providing direct contact with a live human agent. A chatbot is a type of software that can automate conversations and interact with people through messaging platforms.

Our chatbot is mainly based on the below components:

* Fasttext based model
* Flask based API
* Dot net Based UI component

The details for model selection have already been discussed , the API and UI details have been discussed below

#### API Specifications:

#### We have done POC for building the API over the model using Django Framework and Flask Frameworks of Python

#### The final API was done using Flask Framework. Django framework though it has good documentation support and wide array of utilities, we preferred Flask since it is easy to build and run the Restful Rest services

|  |  |
| --- | --- |
|  |  |
| ***HTTP Method*** | POST |
| **Query Param** | Query – pass the text as query element in the body as json |
| **Output Json element** | potentialAccidentLevel – output level |
| **App Class** | App.py |
| **Model Class** | Build\_model.py |
| **Model saved as** | Model.bin |
| **Git hub** | https://github.com/CapstoneAIML/aiml-chatbot-indsafe/tree/main/flaskapp/ind-safety-c |

#### 

#### UI Design:

To be added.

#### Implications:

* How does our solution affect the problem in the domain or business?
  + The goal is to create an AI based chatbot which can predict the potential accident level based on the description of the accident provided by the user.
  + If predicted early and correctly this bot can become a good tool to save and protect employees of the different industry sectors.
  + Hence it can help the organization’s interest by reducing the risks involve with its employees.
  + It can enhance the reputation of an organization in the views of its employees’ and also it can reduce the expenses associated with an accident.
* Recommendations:
* This chatbot can be highly effective in risky industries such as Mining and Metal Industries.
* As this chatbot has the ability to determine the potential accident level with very high precision we as a team would recommend this with the same level of confidence which we have achieved by testing it on real industry safety dataset.
* So, with more than 90% confidence we would be recommending it.

#### Limitations:

* This chatbot needs a detail text description of the incident to determine the potentiality of the accident which can be time consuming during any critical situation.
* It doesn’t consider the industry type, gender and age and different critical risk factors to determine the level of accident.
* It uses FastText model which is good in text classifications but unlike LSTM it can’t remember selective patterns.
* In Realtime there are such applications which can decide the next set of actions based on users input which ours’ can’t.
* It hasn’t learnt from a very versatile dataset.
* Future scope:

This application can be enhanced into more sophisticated software solution for industry safety by adding the some more features. E.g.

* A voice based chatbot
* Call SOS
* Trigger alarm based on the criticality.
* Recognize individuals voice and instead of text works on voice data.
* Deploy as a mobile Application.
* Train and optimize our models by using balanced datasets.

#### Reflections:

* This project is the result of teamwork and collaborations. We have worked as a unit and completed all tasks in time.
* Now we have a much better understanding about AIML with deep learning as we had to learn and apply handful of those techniques in order to reach our target. Particularly we have firm grip on NLP.
* We also had a chance to work with FastText which is widely popular.
* We have an understanding of end to end application development with python using Flask.
* What could be done differently:
* We tried Random forest but never tried other supervised machine learning algorithms. It could be tried as a next step.
* This chatbot considers only text data. Instead, it could work by recognizing and analyzing voice data.
* There are other features which aren’t considered much here as of now, we could explore those features more.
* The chatbot could be more interactive and decisive.